An Empirical Evaluation of k-Means Coresets

Anonymous author
Anonymous affiliation

Anonymous author
Anonymous affiliation

Abstract

Coresets are among the most popular paradigms for summarizing data. In particular, there exist many high performance coresets for clustering problems such as k-means in both theory and practice. Curiously, there exists no work on comparing the quality of available k-means coresets.

In this paper we perform such an evaluation. We describe the difficulty in comparing the quality of different coreset algorithms. In order to perform an empirical evaluation, we therefore have to work with heuristics. To this end, we propose a new procedure for evaluating coresets on real-world datasets. To complement this, we propose a benchmark framework for generating provably hard synthetic data sets to evaluate coresets. Using this benchmark and real-world data sets, we conduct an exhaustive evaluation of the most commonly used coreset implementations.

2012 ACM Subject Classification Replace ccsdesc macro with valid one

Keywords and phrases Dummy keyword

Digital Object Identifier 10.4230/LIPIcs.SEA.2022.23

Acknowledgements Anonymous acknowledgements

1 Introduction

The design and analysis of scalable algorithms has become an important research area over the past two decades. This is particularly important in data analysis, where even polynomial running time might not be enough to handle proverbial *big data* sets. One of the main approaches to deal with the scalability issue is to compress or sketch large data sets into smaller, more manageable ones. The aim of such compression methods is to preserve the properties of the original data, up to some small error, while significantly reducing the number of data points.

Among the most popular and successful paradigms in this line of research are *coresets*. Informally, given a data set A, a coreset $S \subset A$ with respect to a given set of queries Q and query function $f : A \times Q \to \mathbb{R}_{\geq 0}$ approximates the behaviour of A for all queries up to some multiplicative distortion D via

$$\sup_{q\in Q} \max\left(\frac{f(S,q)}{f(A,q)}, \frac{f(A,q)}{f(S,q)}\right) \leq D.$$

Coresets have been applied to a number of problems such as computational geometry [2, 6], linear algebra [24, 28], and machine learning [30, 33]. But the by far most intensively studied and arguably most successful applications of the coreset framework is the k-clustering problem.

Here we are given n points A with (potential unit) weights $w : A \to \mathbb{R}_{\geq 0}$ in some metric space with distance function dist and aim to find k centers C such that

$$\text{cost}_A(C) := \frac{1}{n}\sum_{p\in A} \min_{c\in C} w(p) \cdot \text{dist}^z(p,c)$$

Omar: Shouldn't the cost be:

$$\text{cost}_A(C) := \sum_{p \in A} w(p) \cdot \min_{c \in C} \text{dist}^z(p, c)$$

is minimized. The most popular variant of this problem is probably the k-means problem in d-dimensional Euclidean space where $z = 2$ and $\text{dist}(x, y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$.

A (k, ε)-coreset is now a subset $\Omega \subset A$ with weights $w : \Omega \to \mathbb{R}_{\geq 0}$ such that for any set of k centers C

$$\sup_C \max \left(\frac{\text{cost}_A(C)}{\text{cost}_\Omega(C)}, \frac{\text{cost}_\Omega(C)}{\text{cost}_A(C)} \right) \leq 1 + \varepsilon. \tag{1}$$

The coreset definition in Equation (1) provides an upper bound for the distortion of all candidate solutions i.e., all possible k clusterings. A *weak coreset* is a relaxed guarantee that holds for optimal or nearly optimal clusterings of A instead of all clusterings.

In a long line of work spanning the last 20 years [4, 5, 7, 13, 16, 21, 19, 23, 5, 26, 36], the size of coresets has been steadily improved with the current state of the art yielding a coreset with $\tilde{O}(k\varepsilon^{-2} \cdot \min(d, \varepsilon^{-2}))$ points for a distortion $D \leq (1 + \varepsilon)$ due to Cohen-Addad, Saulpic, and Schwiegelshohn [11][1].

While we have a good grasp of the theoretical guarantees of these algorithms, our understanding of the empirical performance is somewhat lacking. There exist a number of coreset implementations, but it usually difficult to access which implementation summarizes the data best. To accurately evaluate a given coreset, we would need to come up with a k clustering C which results in a maximal distortion. Solving this problem is likely difficult: related questions such as deciding whether a 3-dimensional point set A is an ε-net of a net of a set B with respect to convex ranges is co-NP hard and it is similarly co-NP hard to decide whether a point set A is a *weak coreset* of a point set B.

Due to this difficulty, a common heuristic for evaluating coresets is as follows [1, 17]. First, compute a coreset Ω with the available algorithm(s) using some input data A. Then, run an optimization algorithm on Ω to compute a k clustering C. The *best* coreset algorithm is considered to be the one which yields a clustering with the smallest cost.

The drawback of this evaluation method is that it mixes up the two separate tasks of coreset construction and optimization. An algorithm may yield a good clustering (with small cost) yet fail to produce a high quality coreset (with small distortion). Additionally, this method is more likely to measure the performance of the underlying optimization problem, rather than evaluating the coresets themselves.

The purpose of this study is to systematically evaluate the quality of various coreset algorithms for k-means. As such, we develop a new evaluation procedure which estimates the distortion of coreset algorithms. On real-world data sets, we observe that while the evaluated coreset algorithms are generally able to find solutions with comparable costs, there is a stark difference in their distortions. This shows that differences between optimization and compression are readily observable in practice.

As a complement to our evaluation procedure on real-world data sets, we propose a benchmark framework for generating synthetic data sets. We argue why this benchmark has properties that results in hard instances for all known coreset constructions. We also show how to efficiently estimate the distortion of a candidate coreset on the benchmark.

[1] We use $\tilde{O}(x)$ to hide $\log^c x$ terms for any constant c.

2 Coreset Algorithms

Though the algorithms vary in details, coreset constructions come in one of the following two flavours:

1. Movement-based constructions: Such algorithms compute a clustering with T points such that $\text{cost}_A(T) \ll \text{OPT}$, where OPT is the cost of an optimal k-means clustering. The coreset guarantee then follows as a consequence of the triangle inequality. These algorithms all have an exponential dependency on the dimension d, and therefore have been overtaken by sampling-based methods. Nevertheless, these constructions are more robust to various constrained clustering formulations [22, 35] and continue to be popular. Examples from theory include [18, 21].
2. Importance sampling: Points are sampled proportionate to their sensitivity which for a point p is defined as $sens(p) := \sup_C \frac{\min_{c\in C} \text{dist}^2(p,c)}{\text{cost}_A(C)}$ and weighted by their inverse sampling probability. In terms of theoretical performance, sensitivity sampling has largely replaced movement based constructions, see for example [14, 27].

Of course, there exist algorithms that draw on techniques from both, see for example [11]. In what follows, we will survey implementations of various coreset constructions that we will evaluate later.

StreamKM++ [1] The popular k-means++ algorithm [3] computes a set of centers K by iteratively sampling a point p in A proportionate to $\min_{q\in K} \text{dist}^2(p,q)$ and adding it to K. The procedure terminates once the desired number of centers has been reached. The first center is typically picked uniformly at random. The StreamKM++ paper runs the k-means++ algorithms for T iterations, where T is the desired coreset size. At the end, every point q in K is weighted by the number of points in A closest to it. While the construction has elements of important sampling, the analysis is largely movement-based. The provable bound required for the algorithm to compute a coreset is $O\left(\frac{k\log n}{\delta^{d/2}\varepsilon^d} \cdot \log^{d/2} \frac{k\log n}{\delta^{d/2}\varepsilon^d}\right)$.

BICO [17] Combines the very fast, but poor quality clustering algorithm BIRCH [40] with the movement-based analysis from [18, 21]. The clustering is organized by way of a hierarchical decomposition: When adding a point p to one of the coreset points Ω at level i, it first finds the closest point q in Ω. If p is too far away from q, a new center is opened at p. Otherwise p is either added to q, or, if adding p to q increases the clustering cost of q beyond a certain threshold, the algorithm attempts to add p to the child-clusters of q. The procedure then continues recursively. The provable bound required for the algorithm to compute a coreset is $O\left(k\log n\varepsilon^{-d-2}\right)$.

Sensitivity Sampling [13] The simplest implementation of sensitivity sampling first computes an $(O(1), O(1))$ bicriteria K approximation[2], for example by running k-means++ for $2k$ iterations [39]. Let K be the $2k$ clustering thus computed and let K_i be an arbitrary cluster of K with center q_i. Subsequently, the algorithm picks $T-2k$ points proportionate to $\frac{\text{dist}^2(p,q)}{\text{cost}_{K_i}(\{q_i\})} + \frac{1}{|K_i|}$. Let $|\hat{K}_i|$ be the estimated number of points in the sample. Finally, the algorithm weighs each q_i by $(1+\varepsilon)\cdot|K_i| - |\hat{K}_i|$. The provable bound required for the algorithm to compute a coreset is $\tilde{O}\left(kd\varepsilon^{-4}\right)$ ([13]), $\tilde{O}\left(k\varepsilon^{-6}\right)$ ([23]), or $\tilde{O}\left(k^2\varepsilon^{-4}\right)$ ([5]).

[2] An (α, β) bicriteria approximation computes an α approximation using $\beta \cdot k$ many centers.

Group Sampling [11] First, the algorithm computes an $O(1)$ approximation (or a bicriteria approximation) K. Subsequently, the algorithm preprocesses the input into groups such that (1) for any two points $p, p' \in K_i$, their cost is identical up to constant factors and (2) for any two clusters K_i, K_j, their cost is identical up to constant factors. In every group, Group-Sampling now samples points proportionate to their cost. The authors of [11] show that there always exist a partitioning into $\log^2 1/\varepsilon$ groups. Points not contained in a group are snapped to their closest center q in K. q is weighted by the number of points snapped to it. The provable bound required for the algorithm to compute a coreset is $\tilde{O}\left(k\varepsilon^{-4}\right)$ ([11]).

Ray Maker [20] The algorithm computes an initial solution with k centers which is a constant factor approximation of the optimal clustering. Around each center, $O(1/\epsilon^{d-1})$ random rays are created which span the hyperplane. Next, each point $p \in A$ is snapped to its closest ray resulting in a set of one-dimensional points associated with each ray. Afterwards, a coreset is created for each ray by computing an optimal 1D clustering with k^2/ϵ^2 centers and weighing each center by the number of points in each cluster. The final coreset is composed of the coresets computed for all the rays. The provable bound required for the algorithm to compute a coreset is $O(n + \text{poly}(k, \log n, 1/\epsilon) + \text{func}(k, \epsilon))$ where poly is a polynomial and $\text{func}(k, \epsilon)$ is a function which depends on k and ϵ.

2.1 Dimension Reduction

Finally, we also combine coreset constructions with a variety of dimension reduction techniques. Since the seminal paper by Feldman, Schmidt, and Sohler [15], most coreset algorithms have used some form of dimension reduction to eliminate the dependency on d, either by explicitly computing a low-dimensional embedding, see for example [15, 37], or by using the existence of a suitable embedding in the analysis [11, 23].

In particular, movement-based coresets often have an exponential dependency on the dimension, which can be alleviated with some form of dimension reduction, both in theory [35] and in practice [25].

Here the are two main techniques.

Principal Component Analysis: Feldman, Schmidt, and Sohler [15] showed that projecting an input A onto the first $O(k/\varepsilon^2)$ principal components is a coreset, albeit in low dimension. The analysis was subsequently tightened by [8] and extended to other center based cost functions by [36]. Although it's target dimension is generally worse than those based on random projections and terminal embeddings, there is nevertheless reasons for using PCA regardless: It removes noise and thus may make it easier to compute a high quality coreset.

Terminal Embeddings: Given a set of points A in $\mathbb{R}^d$, a terminal embedding $f : \mathbb{R}^d \to \mathbb{R}^m$ preserves the pairwise distance between any point $p \in A$ and any point $q \in \mathbb{R}^d$ up to a $(1 \pm \varepsilon)$ factor. The statement is related to the famous Johnson-Lindenstrauss lemma but it is stronger as it does not apply to only the pairwise distances of A. Nevertheless, the same target dimension is sufficient. Terminal embeddings were studied by [12, 29, 34], with Narayanan and Nelson [34] achieving an optimal target dimension of $O(\varepsilon^{-2} \log n)$, where n is the number of points. For applications to coresets, we refer to [4, 11, 23].

For an overview on practical aspects of dimension reduction, we refer to Venkatsubramanian and Wang [38].

3 Hardness of Coreset Evaluation and a Benchmark

In this section, we first show that it is in general co-NP hard to evaluate the coreset distortion, given two point sets A and B. Thereafter we describe the benchmark and it's properties.

▶ **Proposition 1.** *Given two point sets A and B in $\mathbb{R}^d$ and a sufficiently small (constant) $\varepsilon > 0$, it is co-NP hard to decide whether A is a (k, ε)-coreset of B.*

Proof. First, we recall that for some ε_0 and candidate clustering cost V, it is NP-hard to decide whether there exists a clustering C with cost in $\text{cost}_A(C) \leq V$ and $\text{cost}_B(C) \geq (1 + \varepsilon_0) \cdot V$. Conversely, it is co-NP-hard to decide whether there exists no set of centers C such that $\text{cost}_A(C) \leq V$ and $\text{cost}_B(C) \geq (1 + \varepsilon_0) \cdot V$. ◀

We remark that the possible values for ε_0 are determined by the current APX-hardness results. Assuming NP≠P, $\varepsilon_0 \approx 1.07$ and assuming UCG, $\varepsilon_0 \approx 1.17$ [10, 9] for k-means in Euclidean spaces.

3.1 Benchmark Construction

In this section, we describe our benchmark. The benchmark has a parameter α which controls the number of points and dimensions. For a given value of k the benchmark consists of $n = k^\alpha$ points and $d = \alpha \cdot k$ dimensions. It is recursively constructed as follows.

Denote by $\mathbb{1}_k$ the k-dimensional all 1 vector and by v_i^1 the k dimensional vector with entries $(v_i^1)_j = \begin{cases} -\frac{1}{k} & \text{if } i \neq j \\ \frac{k-1}{k} & \text{if } i = j \end{cases}$. For $\ell \leq \alpha$, recursively define the k^ℓ dimensional vector $v_i^\ell = \begin{bmatrix} (v_i^{\ell-1})_1 \cdot \mathbb{1}_k \\ (v_i^{\ell-1})_2 \cdot \mathbb{1}_k \\ \vdots \\ (v_i^{\ell-1})_1 \cdot \mathbb{1}_k \end{bmatrix}$. Finally, set the column $t = a \cdot k + b$, $a \in \{0, \ldots \alpha - 1\}$ and $b \in \{1, \ldots k\}$, of A to be $k^{\alpha-a}$ Omar: $k^{\alpha-a+1}$? stacks of v_b^{a+1}.

To get a better feel for the construction, we have given two example inputs in Figure 1.

$$\begin{bmatrix}
\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\
-\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\
\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\
-\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\
\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\
-\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\
\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\
-\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2}
\end{bmatrix}
\qquad
\begin{bmatrix}
\frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\
-\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\
-\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} & \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\
\frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} \\
-\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} \\
-\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} \\
\frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} \\
-\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} \\
-\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3}
\end{bmatrix}$$

Figure 1 Benchmark construction for $k = 2$ and $\alpha = 3$ (left) and $k = 3$ and $\alpha = 2$ (right).

3.2 Properties of the Benchmark

We now summarize the key properties of the benchmark. To this end, we require a few notions. Let A be the input matrix. We slightly abuse notation and refer to A_i as both the ith point as well as the ith row of the matrix A. For a clustering $\mathcal{C} = \{C_1, \dots, C_k\}$, we define that the $n \times k$ indicator matrix $\tilde{X}$ induced by $\mathcal{C}$ via

$$\tilde{X}_{i,j} = \begin{cases} 1 & \text{if } A_i \in C_j \\ 0 & \text{else.} \end{cases}$$

Furthermore, we will also use the $n \times k$ normalized clustering matrix X defined as

$$X_{i,j} = \begin{cases} \frac{1}{\sqrt{|C_i|}} & \text{if } A_i \in C_j \\ 0 & \text{else.} \end{cases}$$

We also recall the following lemma which will allow us to express the k-means cost of a clustering $\mathcal{C}$ with optimally chosen centers in terms of the cost of X and A.

▶ **Lemma 2** (Folklore). *Let A be an arbitrary set of points and let $\mu(A) = \frac{1}{|A|} \sum_{p \in A} p$ be the mean. Then for any point c*

$$\sum_{p \in A} \|p - c\|^2 = \sum_{p \in A} \|p - \mu(A)\|^2 + |A| \cdot \|\mu(A) - c\|^2.$$

This lemma proves that for any given cluster C_j, the mean is the optimal choice of center. We also note that any two distinct columns of X are orthogonal. Furthermore $\frac{1}{n}\mathbf{1}\mathbf{1}^T A$ copies the mean into every entry of A. Combining these two observations, we see that the matrix $XX^T A$ maps the ith row of A onto the mean of the cluster it is assigned to. Finally, define the Frobenius norm of an $n \times d$ A by $\|A\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^d A_{i,j}^2}$. Then the k-means cost of the clustering $\mathcal{C}$ is precisely

$$\|A - XX^T A\|_F^2.$$

We also require the following distance measure on clusterings as proposed by Meila [31, 32]. Given two clusterings $\mathcal{C}$ and $\mathcal{C}'$, the $k \times k$ confusion matrix M is defined as

$$M_{i,j} = |C_i \cap C'_j|.$$

Furthermore for the indicator matrices $\tilde{X}$ and $\tilde{X}'$ induced by $\mathcal{C}$ and $\mathcal{C}'$ we have the identity $M = \tilde{X}^T \tilde{X}'$. Denote by Π_k the set of all permutations over k elements. Then the distance between $\mathcal{C}$ and $\mathcal{C}'$ is defined as

$$d(\mathcal{C}, \mathcal{C}') = 1 - \frac{1}{n} \max_{\pi \in \Pi_k} \sum_{i=1}^k M_{i,\pi(i)}.$$

Observe that for clusters that are identical, their distance is 0. The maximum distance between any two k clusterings is always $\frac{k-1}{k}$.

We are now ready to state the desired properties of our benchmark. The benchmark was designed to generate many clusterings such that

1. The distance between these clustering is maximized.
2. The clusterings have equal cost.
3. The clusterings are induced by a set of centers in $\mathbb{R}^d$.

The first and second property ensure that (equally good) solutions we use for evaluation are spread out through the solution space. In other words, we are less likely to only focus on a set of solutions $\mathcal{S}$ for which a low distortion on one $S \in \mathcal{S}$ implies a low distortion for all elements of $\mathcal{S}$. The third property is important as these are the only clusterings the (standard) coreset guarantee has to apply to.

The solutions we consider are given as follows. For the columns $a \cdot k + 1, \ldots (a+1) \cdot k$, we define the clustering $\mathcal{C}^a = \{C_1^a, \ldots C_k^a\}$ with $A_i \in C_j^a$ if and only if $A_{i,j} > 0$. Let $\tilde{X}^a$ and X^a denote the indicator matrix and clustering matrix, respectively, as induced by $\mathcal{C}^a$.

▶ **Fact 3.** *For $a \neq a'$, we have $d(\mathcal{C}^a, \mathcal{C}^{a'}) = 1 - 1/k$.*

Proof. Consider an arbitrary vector v_i^ℓ. By construction, the positive entries of v_i^ℓ range from $k^{\ell-1} \cdot i + 1$ to $k^{\ell-1} \cdot (i+1)$. Similarly, the positive entries for the vector $v_j^{\ell-1}$ range from range from $k^{\ell-2} \cdot j + 1$ to $k^{\ell-2} \cdot (j+1)$. Therefore, concatenating $v_j^{\ell-1}$ k times into a vector v', v' and v_i^ℓ can share at most one positive coordinate. Inductively, the same holds true for any concatenation of vectors $v_j^{\ell-h}$. Thus, the two clusters induced by the columns formed by concatenating the vectors v can share only a $1/k$ fraction of the points. Since each cluster consists of exactly $k^\alpha/k = k^{\alpha-1}$ points, the confusion matrix M only has entries $\frac{n}{k^2}$ and for any permutation π, we have $d(\mathcal{C}^a, \mathcal{C}^{a'}) = 1 - 1/k$. ◀

▶ **Fact 4.** *For any C_j^a, we have $cost_{C_j^a}(\{\mu(C_j^a)\}) = (\alpha - 1) \cdot k^{\alpha-2} \cdot (k-1)$.*

Proof. Without loss of generality, we consider C_1^0; the proof is analogous for the other choices of j and a. We first note that for any point $A_i \in C_1^0$, the coordinates $A_{i,\ell}$ are identical for $\ell < k$. Furthermore for the column $\ell \geq k$, we have by construction $\sum_{A_i \in C_j} A_{i,\ell} = k^{\alpha-1} \cdot \frac{k-1}{k} + (k^\alpha - k^{\alpha-1})\frac{1}{k} = k^{\alpha-1} \cdot (\frac{k-1}{k} - (k-1)\frac{1}{k}) = 0$. Therefore, the mean of C_1^0 satisfies $\mu(C_1^0)_\ell = \begin{cases} A_{i,\ell} & \text{if } \ell < k \\ 0 & \text{else.} \end{cases}$. Thus, the cost is precisely $(\alpha - 1) \cdot k^{\alpha-1} \cdot \left(\left(\frac{k-1}{k}\right)^2 + \left(\frac{1}{k}\right)^2 \right) = (\alpha - 1) \cdot k^{\alpha-2} \cdot (k-1)$. ◀

Finally, we show that the means for the clustering $\mathcal{C}^a$ also induce $\mathcal{C}^a$.

▶ **Fact 5.** *For a clustering $\mathcal{C}^a$, let $\mu(C_j^a)$ denote the mean of cluster C_j^a. Then every point of Omar: A_i of C_j^a is assigned to its closest center. Moreover, every point A_i of C_j^a has equal distance to any center $\mu(C_h^a)$ with $h \neq j$.*

Proof. Again, we assume without loss of generality $a = 0$. Let A_i be an arbitrary point of cluster C_h^a and consider the mean $\mu(C_j^a)_\ell = \begin{cases} A_{i,\ell} & \text{if } \ell < k \\ 0 & \text{else.} \end{cases}$ of cluster C_j^a. By definition, the positive coordinates of A_i are not equal to the positive coordinates of $\mu(C_j^a)$. The only difference in coordinates between the means of $\mu(C_j^a)$ and $\mu(C_h^a)$ are the first k coordinates, as the rest are 0. But here the coordinates of $\mu(C_h^a)$ and A_i are identical, hence $\mu(C_j^a)$ cannot be closer to A_i.

To prove that the distances between A_i and any $\mu(C_h^a)$ with $h \neq j$ are equal, again consider that any difference can only exist among the first k coordinates. Here, we have $\mu(C_h^a)_h = \frac{k-1}{k}$, and the remaining columns are $-\frac{1}{k}$. Since $A_{i,h} = -\frac{1}{k}$ for any $h \neq j$, the claim follows. ◀

3.3 Benchmark Evaluation

We now describe how we use the benchmark to measure the distortion of a coreset. Assume that the coresets only consists of input point[3]

[3] It is not necessary for coreset constructions in general to consists of input points. One can adjust the evaluation in to account for this. But since all algorithms considered in this paper have the property

Consider the clustering $\mathcal{C}^a = \{C_1^a, \dots C_k^a\}$ for some a and let Ω with weights $w : \Omega \to \mathbb{R}_{\geq 0}$ be the coreset. We use $w(C_i^a \cap \Omega) := \sum_{p \in C_i^a \cap \Omega} w(p)$ to denote the mass of points of C_i^a in Ω. For every cluster C_i^a with $w(C_i^a \cap \Omega) \geq |C_i|(1-\varepsilon)$, we place a center at $\mu(C_i^a)$. Conversely, if $w(C_i^a \cap \Omega) \leq |C_i|(1-\varepsilon)$ Omar: Replace $\leq$ with $<$?, we do not place a center at $\mu(C_i^a)$. We call such clusters *deficient*. Let $\mathcal{S}$ be the total number of thus placed centers. Omar: Replace all C_i with C_i^a?

We now compare the cost as computed on the coreset and the true cost of $\mathcal{S}$. Due to Lemma 2 and Fact 5, we may write for any deficient cluster C_i^a $\text{cost}_{C_i^a}(\mathcal{S}) = \text{cost}_{C_j^a}(\{\mu(C_j^a)\}) + k^{\alpha-1}\|\mu(C_j^a) - \mu(C_h^a)\|_2^2$, where C_h^a is a non-deficient cluster. Thus, the cost is due to Fact 4

$$\begin{aligned} \text{cost}_{C_i^a}(\mathcal{S}) &= (\alpha-1)\cdot k^{\alpha-2}\cdot(k-1) + k^{\alpha-1}\cdot 2 \\ &\approx (1+\frac{2}{\alpha})\cdot \text{cost}_{C_j^a}(\{\mu(C_j^a)\}). \end{aligned}$$

Conversely, the cost on the coreset is

$$\begin{aligned} &\text{cost}_{\Omega\cap C_i^a}(\mathcal{S}) \\ =\ & w(C_i^a \cap \Omega)\left((\alpha-1)\cdot\left(\left(\frac{k-1}{k}\right)^2 + (k-1)\left(\frac{1}{k}\right)^2\right) + 2\right) \\ \approx\ & \frac{w(C_i^a\cap\Omega)}{\text{cost}_{C_j^a}(\{\mu(C_j^a)\})}(1+\frac{2}{\alpha})\cdot \text{cost}_{C_j^a}(\{\mu(C_j^a)\}). \end{aligned}$$

Thus for each deficient clustering individually, the distortion will be close to $\frac{k^{\alpha-1}}{w(C_i^a\cap\Omega)} \geq \frac{1}{1-\varepsilon}$. If there are many deficient clusters, then this will also be the overall distortion.

3.4 Further Extensions

On the benchmark we considered here, both Sensitivity Sampling, as well as Group Sampling are similar to uniform sampling and, indeed, uniform sampling could be used to construct a good coreset. We can eliminate uniform sampling as a viable algorithm for this instance by combining multiple benchmarks $B_1, \dots B_t$ with $\sum_{i=1}^t k_i = k$. Each benchmark then has size $\sum_{i=1}^t k_i^\alpha$. We then add an additive offset to the coordinates of each benchmark such that they do not interfere. In this case, uniform sampling does not work if the values of the k_i are different enough. Since it is well known that uniform sampling is not a viable coreset algorithm in both theory and practice, we only used the basic benchmark for our evaluations.

4 Experiments

In this section, we present a new procedure for evaluating k-means coresets. We begin our exposition by motivating why there is a need for a new evaluation procedure. Then we describe the proposed procedure, the data sets used for the empirical evaluation and the experimental setup. Finally, we detail the outcome of the experiments and our observations on the results.

and it makes the evaluation simpler, we will only consider coresets that are subsets of the original point set.

4.1 Motivation for Introducing New Evaluation Procedure

Chris: I think we mentioned most of this before? I am currently in favour of removing it. We need to shorten the paper anyway for space reasons. Previous works evaluated coreset algorithms on real-world data by comparing the cost of applying a coreset algorithm followed by an optimization procedure [1, 17]. Although this approach seems like a reasonable way of measuring coreset performance, it is more likely to capture the performance of the underlying optimization rather than the coreset construction.

Determining the quality of a k-means coreset in isolation requires generating a candidate solution (a k clustering) which results in a maximal distortion. This follows from the definition of coreset in Equation (1). Finding such a worst-case solution is conjectured to be computationally hard. One obvious alternative approach would be to randomly generate a worst-case solution with high probability. The challenge, however, is that it is not clear how to define a distribution of meaningful solutions from which to sample from. Moreover, a randomly drawn solution, which does not exploit the behavior of a coreset construction, is less likely to yield a worst-case solution in terms of high distortion.

A practical approach for finding meaningful solutions is to use k-means++ as it can uncover cluster structures in the data. Running multiple iterations of k-means++ allows us to pick a solution with high distortion. The obtained solution can then be used to estimate the quality of any given coreset computed on a real-world data set.

Still, it can be tricky to gauge the general performance of coreset algorithms using only a selection of real-world data sets. For this reason, we proposed a synthetic benchmark to complement the evaluation procedure on real-world data sets. The benchmark accomplishes two important tasks. First, the benchmark allows us to quickly find a bad solution because both good and bad clusterings are known a priori. It is unclear how to find bad clusterings for real-world data sets. Second, it is easier to make a fair comparison of different coreset constructions because the benchmark is known to generate hard instances for all known coreset algorithms. This cannot be said for real-world data sets.

Chris: I think the following paragraph has value, but it should go to the complement interpretation/outcome section. For example, it is not uncommon for the k-means cost to drop significantly for larger values of k in real-world data sets. Figure 2 illustrates this behavior for several real-world data sets. The more the curve bends, the less of a difference there is between computing a coreset and a clustering with low cost. For data sets with a L-shaped cost curve, a coreset algorithm adding more centers to the coreset will seem to be performing well when evaluating it based on the outcome of the optimization. However, in the benchmark dataset, there is no way of reducing the cost without capturing the right subclusters within the benchmark instance. This means that the cost does not decrease markedly beyond a certain value of k even if more centers are added as depicted in Figure 3.

Chris: I think the first figures should be distortion plots. Again, should be moved to the outcome section.

4.2 Evaluation Procedure

For estimating the quality of the coresets produced by various coreset algorithms, we evaluated the algorithms on both real-world data sets and the proposed benchmark.

On each real-world data set A, we first computed a coreset Ω and then ran k-means++ on Ω to find a set of k centers C. The k-means++ algorithm was repeated 5 times in order to pick the set of centers with the largest distortion. Recall that distortion of a given solution

Figure 2 Depicts how clustering costs of five real-world data sets decrease as the number of centers increase. The most widely used data sets for evaluating coresets are *Tower*, *Covertype*, and *Census*, while *Caltech* is rarely used and *NYTimes* has never been used before. Plotting the cost curve allows us to study whether we can observe a difference between coreset construction and optimization in a data set when evaluating a coreset based on cost.

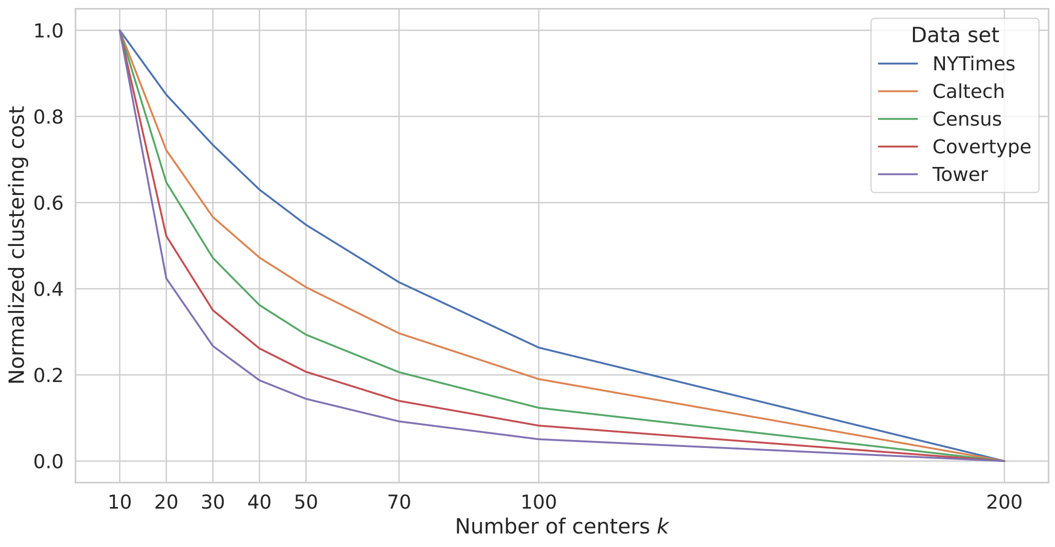
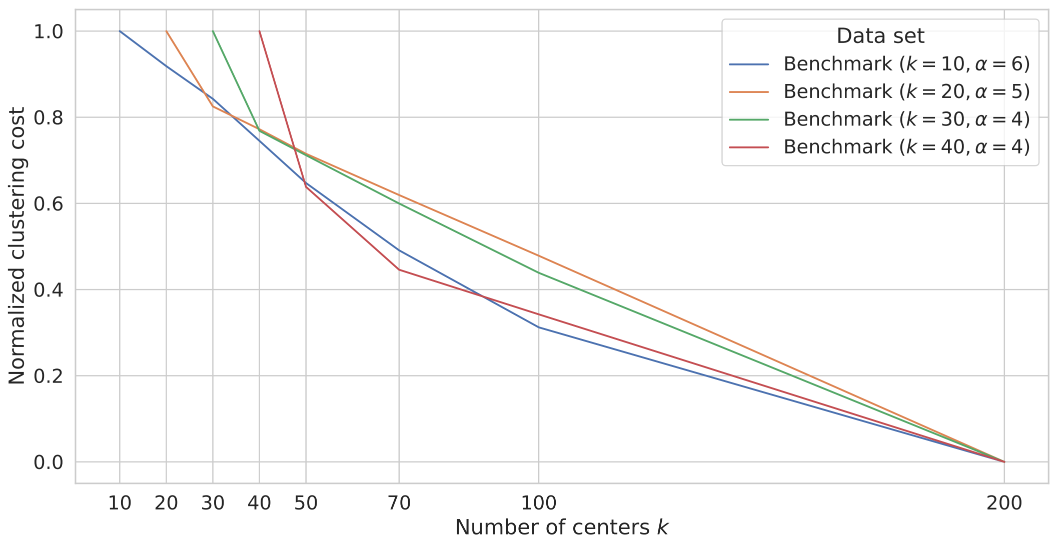


Figure 3 Shows the clustering costs of four instances of the benchmark framework as a function of the number of centers. In contrast to real-world data sets, the costs do not decrease rapidly as more cluster centers are added.



C is defined as:

$$\max\left(\frac{\mathrm{cost}_A(C)}{\mathrm{cost}_\Omega(C)}, \frac{\mathrm{cost}_\Omega(C)}{\mathrm{cost}_A(C)}\right)$$

For the benchmark, we computed the distortion following the evaluation procedure described in Section 3. Every randomized coreset construction was repeated 10 times. We aggregated the reported distortions by taking the maximum over all 10 evaluations. Chris: I don't get this. Did you take $D = \frac{1}{10}\sum D_i$, where D_i is the maximum distortion in the ith evaluation or $D = \max D_i$? Because I would understand using the mean (or median), but not so much using the max both times. We repeat the experiment to filter out outliers and bad runs; using the max means we always take the biggest outlier. In addition, we preprocessed the data using the dimension reduction techniques described in Section 2.

4.3 Data sets

We conducted experiments on five real-world data sets and four instances of our benchmark. The sizes of the real-world data sets are summarized in Table 1. Benchmark instances were generated to match approximately the sizes of the real-world data sets. The chosen parameter values and the corresponding instance sizes are shown in Table 2. We now provide a brief description of each of the real-world data sets.

The *Census*[4] dataset is a small subset of the Public Use Microdata Samples from 1990 US census. It consists of demographic information encoded as 68 categorical attributes of 2,458,285 individuals.

Covertype[5] is comprised of cartographic descriptions and forest cover type of four wilderness areas in the Roosevelt National Forest of Northern Colorado in the US. It consists of 581,012 records, 54 cartographic variables and one class variable. Although *Covertype* was originally made for classification tasks, it is often used for clustering tasks by removing the class variable [1].

The data set with the fewest number of dimensions is *Tower*[6]. This data set consists of 4,915,200 rows and 3 features as it is a 2,560 by 1,920 picture of a tower on a hill where each pixel is represented by a RGB color value.

Inspired by [17], *Caltech* was created by computing SIFT features from the images in the Caltech101[7] image database. This database contains pictures of objects partitioned into 101 categories. Disregarding the categories, we concatenated the 128-dimensional SIFT vectors from each image into one large data matrix with 3,680,458 rows and 128 columns.

NYTimes[8] is a dataset composed of the bag-of-words (BOW) representations of 300,000 news articles from The New York Times. The vocabulary size of the text collection is 102,660. Due to the BOW encoding, *NYTimes* has a very large number of dimensions and is highly sparse. To make processing feasible, we reduced the number of dimensions to 100 using terminal embeddings.

4.4 Preprocessing & Experimental Setup

To understand how denoising effects the quality of the outputted coresets, we applied Principal Component Analysis (PCA) on *Caltech*, *Census* and *Covertype* by using the k

[4] `https://archive.ics.uci.edu/ml/datasets/US+Census+Data+(1990)`
[5] `https://archive.ics.uci.edu/ml/datasets/covertype`
[6] `http://homepages.uni-paderborn.de/frahling/coremeans.html`
[7] `http://www.vision.caltech.edu/Image_Datasets/Caltech101/`
[8] `https://archive.ics.uci.edu/ml/datasets/bag+of+words`

Table 1 The sizes of the real-world data sets used for the experimental evaluation

	Data points	Dimensions
Caltech	3,680,458	128
Census	2,458,285	68
Covertype	581,012	54
NYTimes	500,000	102,660
Tower	4,915,200	3

Table 2 The parameter values and the sizes of the benchmark instances used for the experimental evaluation.

k	α	Data points	Dimensions
10	6	1,000,000	60
20	5	3,200,000	100
30	4	810,000	120
40	4	2,560,000	160

singular vectors corresponding to the largest singular values. For these three data sets, we preserved the dimensions of the original data. The *NYTimes* dataset did not permit the preservation of dimensions as the number of dimensions is very large. In this case, we used PCA to reduce the dimensions to k. We did not perform any preprocessing on *Tower* due to its low dimensionality.

We followed the same experimental procedure with respect to the choice of parameter values for the algorithms as prior works [1, 17]. For the target coreset size, we used $200k$ for all our experiments. On *Caltech*, *Census*, *Covertype* and *NYTimes*, we used k values in $\{10, 20, 30, 40, 50\}$, while for *Tower* we used larger cluster sizes $k \in \{20, 40, 60, 80, 100\}$. On the benchmark instances, we settled on $k \in \{10, 20, 30, 40\}$ as a reasonable trade-off between running time and data set size.

We implemented Sensitivity Sampling, Group Sampling, Ray Maker, and StreamKM++ in C++. The source code can be found on GitHub[9]. For BICO, we used the authors' reference implementation[10]. The source code was compiled with gcc 9.3.0. The experiments were performed on a machine with 14 cores (3.3 GHz) and 256 GB of memory.

4.5 Outcome of Experiments

We summarized the distortions in Figure 4. All five algorithms are matched on the *Tower* dataset. The worst distortions across the algorithms are close to 1, and performance between the algorithms is negligible. The performance difference between sampling-based and movement-based methods become more pronounced as the number of dimensions increase. On *Covertype* with its 54 features, Ray Maker performs the worst followed by BICO and Group Sampling while Sensitivity Sampling and StreamKM++ perform the best. Differences

[9] Link to repository will be provided later.
[10] `https://ls2-www.cs.tu-dortmund.de/grav/en/bico`

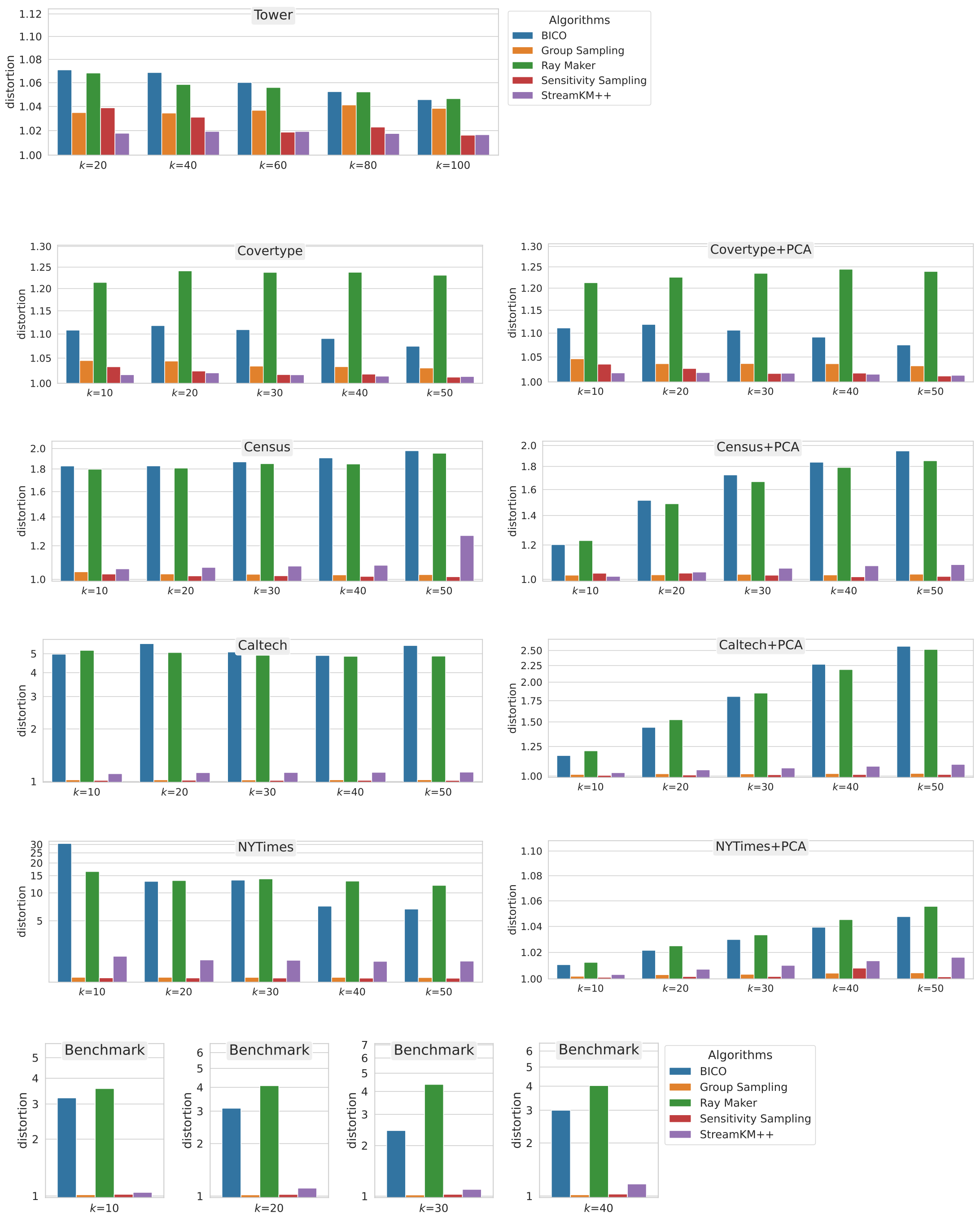


Figure 4 The distortions of the evaluated coreset algorithms on five real-world data sets and on four benchmark instances.

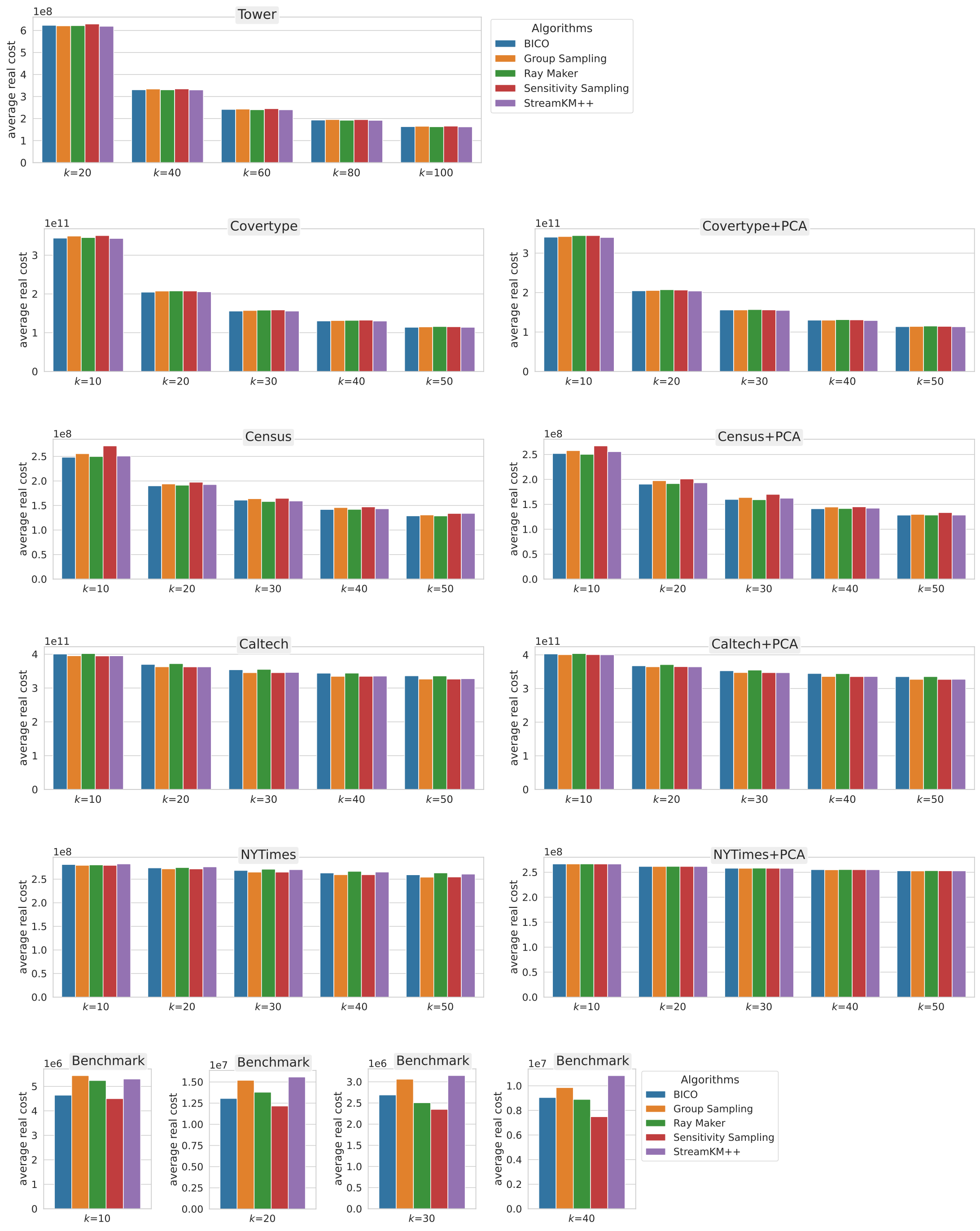


Figure 5 The average costs of running the evaluated coreset algorithms multiple times on different data sets. In general, the five coreset algorithms are able to compute coresets which result in solutions with comparable costs on the different real-world data sets. The differences in cost is more noticeable on the benchmark instances. Here, Senstivity Sampling is the winner because it seems to be better at capturing the correct "clusters" inherent in the benchmark instances.

in performance are more noticeable on *Census*, *Caltech*, and *NYTimes* where methods based on importance sampling perform much better. Sensitivity Sampling and Group Sampling perform the best, StreamKM++ come in second while BICO and Ray Maker perform the worst across these data sets. On the *Benchmark*, Ray Maker is the worst while Sensitivty Sampling and Group Sampling are the best. StreamKM++ performs also very well compared to BICO.

Reducing noise with PCA helped boost the performance on real-world data sets with large number of dimensions. On *Covertype*, PCA does not change the performance numbers by much. On *Census* ($d = 68$), the preprocessing step with PCA improves the performance of BICO and Ray Maker slightly for lower values of k. The distortions of BICO and Ray Maker are reduced markedly on *Caltech* ($d = 128$) after applying PCA.

4.6 Interpretation of Experimental Results

Chris: Can we have one central take-away for each? I wrote a suggestion, let me know how you feel about them

Optimization versus Compression

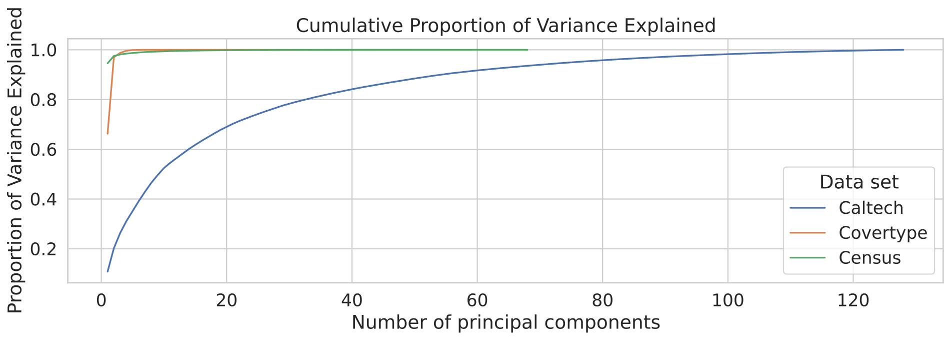
For some data sets, optimization is very close to compression. Clearly, this is the case for *Tower*. The cost curve in Figure 2 for this data set seem to support this claim as the "elbow" for *Tower* bends the most which indicates that adding more centers help reduce the cost. For BICO, Ray Maker, and StreamKM++, there is a correlation between the steepness of the cost curve for a data set and the distortion of the generated coreset. Chris: excellent. please add the aforementioned stuff here (plots and the parts from the paragraph that we will probably erase). For data sets where the curve is less steep, we observed higher distortions. The effect is more pronounced for the movement-based algorithms (BICO and Ray Maker) than for StreamKM++ which performs sampling. The other two sampling-based approaches (Group Sampling and Sensitivity Sampling) seem to be free from this behavior as they consistently generate high quality coresets irrespective of the shape of cost curve.

Chris: The aforementioned correlation between cost and distortion for movement and the fact that sampling seems to be rather oblivious to this.

Movement-based versus Sampling-based Approaches

While all five algorithms are equally matched when comparing clustering costs (see Figure 5), coreset quality performance differ significantly (see Figure 4). In general, movement-based constructions perform the worst in terms of coreset quality. We observed that BICO and Ray Maker have the highest distortions across all data sets including on the benchmark instances. Among the sampling-based algorithms, Sensitive Sampling performs with Group Sampling generally being competitive. StreamKM++ is an interesting case. Like the movement-based methods, its distortion increases with the dimension. Nevertheless, it generally performs significantly better than BICO and Ray Maker. This can be attributed to the fact that the coreset produced by StreamKM++ consists entirely of k-means++ centers weighted by the number of points of a minimal cost assignment. This is similar to movement-based algorithms such as BICO. Nevertheless, it also retains some of the performance from pure importance schemes. Chris: Can we say something about the impact of dimension? I am not sure I read the plot right (the non-linearity of the axis makes it a bit difficult), but is the distortion of the sampling based stuff affected by the dimension, or more or less independent? Also, I

Figure 6 The cumulative propotion of explained variance by principal components on *Caltech*, *Covertype*, and *Census*.



think we want to highlight more one key take away message: Dimension affects distortion for movement based in practise, not just in theory.

BICO versus StreamKM++

Despite having similar worst case bounds, the quality of coresets produced by BICO and StreamKM++ differ significantly. The theoretical analysis of the worst case bounds for BICO and StreamKM++ are derived from a movement-based construction. Both methods feature similar bounds as described in Section 2. Surprisingly, the quality of the coresets produced by StreamKM++ is significantly better than what one would expect from the analysis. In terms of distortion StreamKM++ is significantly better than BICO across all data sets, and especially on high dimensionality data. This begs the question whether there exist a better theoretical analysis for StreamKM++. We leave this as an open problem for future research. Chris: Put this into conclusion. I think most of the points were raised before, but the question whether there exists a purely k-means++-based coreset is certainly an interesting one.

Impact of PCA

On almost all our data sets, the performance improves when input data is preprocessed with PCA, especially for the movement-based algorithms. Empirically, the more noise is removed (i.e., small k value), the lower the distortion. Notice that k is the number of principal components that the input data is projected on to. The rest of the low variance components are treated as noise and removed. Method utilizing sampling (Group Sampling, Sensitivity Sampling and StreamKM++) are less effected by the preprocessing step. On *Covertype*, PCA does not change the distortions by much because almost all the variance in the data is explained by the first five principal components (see Figure 6). On *Caltech* and *NYTimes*, the quality of the coresets by BICO and Ray Maker improves greatly because the noise removal is more aggressive. Even if the quality is much better for movement-based coreset constructions due to PCA transformation, importance sampling methods are still superior when it comes to the quality of the compression. Chris: Take away message at the end: All coreset constructions are affected by the dimension (even sampling, though markedly less so). We can say, PCA is computationally very expensive, but may be nevertheless be worth it.

5 Conclusion

In this work, we studied how to assess the quality of k-means coresets computed by state-of-the-art algorithms. It is generally hard to measure the quality of a coreset because it requires computing the worst-case distortion. Due to this difficulty, earlier works evaluated coresets by the outcome of an optimization algorithm. This method of comparison has the drawback that it is more likely to measure the performance of the underlying optimization problem, rather than evaluating coresets. As a alternative, we proposed a new evaluation procedure which can estimate the quality of coresets on real-world data sets. To complement this, we also proposed a benchmark framework which provably generates hard instances for all known k-means coreset algorithms. We evaluated the quality of five k-means coreset algorithms on five real-world datasets and four instances of the proposed benchmark. We experimented with both movement-based and sampling-based coreset algorithms. We found that while all algorithms produce coresets which yield similar low cost clusterings, sampling-based methods are superior to movement-based algorithms. Chris: Add the k-means++ question here. I'm not sure if I ever mentioned that before. If you came up with it by yourself, I am very impressed. It also flowed naturally in what you were writing.

References

1 Marcel R. Ackermann, Marcus Märtens, Christoph Raupach, Kamil Swierkot, Christiane Lammersen, and Christian Sohler. Streamkm++: A clustering algorithm for data streams. *ACM Journal of Experimental Algorithmics*, 17(1), 2012. URL: `http://doi.acm.org/10.1145/2133803.2184450`, `doi:10.1145/2133803.2184450`.

2 Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. Geometric approximation via coresets. In *Combinatorial and computational geometry, MSRI*, pages 1–30. University Press, 2005.

3 David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 1027–1035, 2007. URL: `http://dl.acm.org/citation.cfm?id=1283383.1283494`.

4 Luca Becchetti, Marc Bury, Vincent Cohen-Addad, Fabrizio Grandoni, and Chris Schwiegelshohn. Oblivious dimension reduction for k-means: beyond subspaces and the johnson-lindenstrauss lemma. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1039–1050, 2019. `doi:10.1145/3313276.3316318`.

5 Vladimir Braverman, Shaofeng H.-C. Jiang, Robert Krauthgamer, and Xuan Wu. Coresets for clustering in excluded-minor graphs and beyond. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2679–2696. SIAM, 2021. `doi:10.1137/1.9781611976465.159`.

6 Timothy M. Chan. Dynamic coresets. *Discret. Comput. Geom.*, 42(3):469–488, 2009. `doi:10.1007/s00454-009-9165-3`.

7 Ke Chen. On coresets for k-median and k-means clustering in metric and Euclidean spaces and their applications. *SIAM J. Comput.*, 39(3):923–947, 2009.

8 Michael B. Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Madalina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 163–172, 2015.

9 Vincent Cohen-Addad and Karthik C. S. Inapproximability of clustering in lp metrics. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science,*

FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019, pages 519–539. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00040.

10 Vincent Cohen-Addad, Karthik C. S., and Euiwoong Lee. On approximability of clustering problems without candidate centers. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2635–2648. SIAM, 2021. doi:10.1137/1.9781611976465.156.

11 Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. A new coreset framework for clustering. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 169–182. ACM, 2021.

12 Michael Elkin, Arnold Filtser, and Ofer Neiman. Terminal embeddings. *Theor. Comput. Sci.*, 697:1–36, 2017. doi:10.1016/j.tcs.2017.06.021.

13 Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 569–578, 2011.

14 Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 569–578, 2011.

15 Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, PCA and projective clustering. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1434–1453, 2013.

16 Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca, and projective clustering. *SIAM J. Comput.*, 49(3):601–657, 2020. doi:10.1137/18M1209854.

17 Hendrik Fichtenberger, Marc Gillé, Melanie Schmidt, Chris Schwiegelshohn, and Christian Sohler. BICO: BIRCH meets coresets for k-means clustering. In *Algorithms - ESA 2013 - 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, pages 481–492, 2013.

18 Gereon Frahling and Christian Sohler. Coresets in dynamic geometric data streams. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 209–217, 2005.

19 Sariel Har-Peled and Akash Kushal. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry*, 37(1):3–19, 2007.

20 Sariel Har-Peled and Akash Kushal. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry*, 37(1):3–19, 2007.

21 Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 291–300, 2004.

22 Lingxiao Huang, Shaofeng H.-C. Jiang, and Nisheeth K. Vishnoi. Coresets for clustering with fairness constraints. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 7587–7598, 2019. URL: http://papers.nips.cc/paper/8976-coresets-for-clustering-with-fairness-constraints.

23 Lingxiao Huang and Nisheeth K. Vishnoi. Coresets for clustering in euclidean spaces: importance sampling is nearly optimal. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 1416–1429. ACM, 2020. doi:10.1145/3357713.3384296.

24 Piotr Indyk, Sepideh Mahabadi, Shayan Oveis Gharan, and Alireza Rezaei. Composable core-sets for determinant maximization problems via spectral spanners. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020,* *Salt Lake City, UT, USA, January 5-8, 2020*, pages 1675–1694. SIAM, 2020. doi:10.1137/1.9781611975994.103.

25 Jan-Philipp W. Kappmeier, Daniel R. Schmidt, and Melanie Schmidt. Solving k-means on high-dimensional big data. In *Experimental Algorithms - 14th International Symposium, SEA 2015, Paris, France, June 29 - July 1, 2015, Proceedings*, pages 259–270, 2015. doi:10.1007/978-3-319-20086-6_20.

26 Michael Langberg and Leonard J. Schulman. Universal ε-approximators for integrals. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 598–607, 2010.

27 Michael Langberg and Leonard J. Schulman. Universal ε-approximators for integrals. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 598–607, 2010. URL: http://dx.doi.org/10.1137/1.9781611973075.50, doi:10.1137/1.9781611973075.50.

28 Alaa Maalouf, Ibrahim Jubran, and Dan Feldman. Fast and accurate least-mean-squares solvers. In *Advances in Neural Information Processing Systems*, pages 8307–8318, 2019.

29 Sepideh Mahabadi, Konstantin Makarychev, Yury Makarychev, and Ilya P. Razenshteyn. Nonlinear dimension reduction via outer bi-lipschitz extensions. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 1088–1101, 2018. URL: http://doi.acm.org/10.1145/3188745.3188828, doi:10.1145/3188745.3188828.

30 Tung Mai, Anup B. Rao, and Cameron Musco. Coresets for classification - simplified and strengthened. *CoRR*, abs/2106.04254, 2021. URL: https://arxiv.org/abs/2106.04254, arXiv:2106.04254.

31 Marina Meila. Comparing clusterings: an axiomatic view. In Luc De Raedt and Stefan Wrobel, editors, *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*, volume 119 of *ACM International Conference Proceeding Series*, pages 577–584. ACM, 2005. doi:10.1145/1102351.1102424.

32 Marina Meila. The uniqueness of a good optimum for k-means. In William W. Cohen and Andrew W. Moore, editors, *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, volume 148 of *ACM International Conference Proceeding Series*, pages 625–632. ACM, 2006. doi:10.1145/1143844.1143923.

33 Alexander Munteanu, Chris Schwiegelshohn, Christian Sohler, and David P. Woodruff. On coresets for logistic regression. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6562–6571, 2018. URL: https://proceedings.neurips.cc/paper/2018/hash/63bfd6e8f26d1d3537f4c5038264ef36-Abstract.html.

34 Shyam Narayanan and Jelani Nelson. Optimal terminal dimensionality reduction in euclidean space. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1064–1069. ACM, 2019. doi:10.1145/3313276.3316307.

35 Melanie Schmidt, Chris Schwiegelshohn, and Christian Sohler. Fair coresets and streaming algorithms for fair k-means. In *Approximation and Online Algorithms - 17th International Workshop, WAOA 2019, Munich, Germany, September 12-13, 2019, Revised Selected Papers*, pages 232–251, 2019. doi:10.1007/978-3-030-39479-0_16.

36 Christian Sohler and David P. Woodruff. Strong coresets for k-median and subspace approximation: Goodbye dimension. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 802–813, 2018. doi:10.1109/FOCS.2018.00081.

37 Christian Sohler and David P. Woodruff. Strong coresets for k-median and subspace approximation: Goodbye dimension. *CoRR*, abs/1809.02961, 2018. URL: `http://arxiv.org/abs/1809.02961`, `arXiv:1809.02961`.

38 Suresh Venkatasubramanian and Qiushi Wang. The johnson-lindenstrauss transform: An empirical study. In Matthias Müller-Hannemann and Renato Fonseca F. Werneck, editors, *Proceedings of the Thirteenth Workshop on Algorithm Engineering and Experiments, ALENEX 2011, Holiday Inn San Francisco Golden Gateway, San Francisco, California, USA, January 22, 2011*, pages 164–173. SIAM, 2011.

39 Dennis Wei. A constant-factor bi-criteria approximation guarantee for k-means++. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 604–612, 2016.

40 Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: A new data clustering algorithm and its applications. *Data Min. Knowl. Discov.*, 1(2):141–182, 1997. URL: `http://dx.doi.org/10.1023/A:1009783824328`, `doi:10.1023/A:1009783824328`.