

1 An Empirical Evaluation of k -Means Coresets

2 **Anonymous author**

3 Anonymous affiliation

4 **Anonymous author**

5 Anonymous affiliation

6 — Abstract —

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

10 In this paper we perform such an evaluation. There currently is no algorithm known to
11 measure the distortion of a candidate coreset. We provide some evidence as to why this might be
12 computationally difficult. To complement this, we propose a benchmark data sets for which we
13 argue that computing coresets is challenging and which also allows us and easy (heuristic) evaluation
14 of coresets. Using this benchmark and real-world data sets, we conduct an exhaustive evaluation of
15 the most commonly used coreset algorithm from theory and practise.

16 **2012 ACM Subject Classification** Replace ccsgdesc macro with valid one

17 **Keywords and phrases** coresets, k -means coresets, evaluation, benchmark

18 **Digital Object Identifier** 10.4230/LIPIcs.SEA.2022.23

19 **Acknowledgements** Anonymous acknowledgements

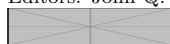


© Anonymous author(s);

licensed under Creative Commons License CC-BY 4.0

20th International Symposium on Experimental Algorithms (SEA 2022).

Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:22



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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 $f : A \times Q \rightarrow \mathbb{R}_{\geq 0}$ 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 [25, 29], and machine learning [31, 34]. 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 $w : A \rightarrow \mathbb{R}_{\geq 0}$ in some metric space with distance function dist and aim to find k centers C such that

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

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 $w : \Omega \rightarrow \mathbb{R}_{\geq 0}$ such that for any set of k centers C

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

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, 22, 20, 24, 5, 27, 37], 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, Saupic, 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 is usually difficult to assess 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

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

of a set B with respect to convex ranges is co-NP hard [19] and it is similarly co-NP hard to decide whether a point set A is a *weak coreset* of a point set B (see Proposition 5 in the appendix).

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 conflates 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.

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 [23, 36] and continue to be popular. Examples from theory include [18, 22].
2. Importance sampling: Points are sampled proportionate to their sensitivity which for a point p is defined as $\text{sens}(p) := \sup_C \frac{\min_{c \in C} \text{dist}^2(p, c)}{\text{cost}_A(C)}$ and weighted by their inverse sampling probability. In terms of theoretical performance, sensitivity sampling has largely replaced movement based constructions, see for example [14, 28].

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 [41] with the movement-based analysis from [18, 22]. 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 $O(k\varepsilon^{-d-2} \log n)$.

Sensitivity Sampling [13] The simplest implementation of sensitivity sampling first computes an $(O(1), O(1))$ bicriteria K approximation², for example by running k -means++ for $2k$ iterations [40]. 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 points proportionate to $\frac{\text{dist}^2(p, q)}{\text{cost}_{K_i}(\{q_i\})} + \frac{1}{|K_i|}$ and weighs any point by its inverse sampling probability. 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 $\tilde{O}(kd\varepsilon^{-4})$ ([13]), $\tilde{O}(k\varepsilon^{-6})$ ([24]), or $\tilde{O}(k^2\varepsilon^{-4})$ ([5]).

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 $\tilde{O}(k\varepsilon^{-4})$ ([11]).

Ray Maker [21] The algorithm computes an initial solution with k centers which is a constant factor approximation of the optimal clustering. Around each center, $O(1/\varepsilon^{d-1})$ 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 $O(k^3 \cdot \varepsilon^{-d-1})$. The algorithm has recently received some attention due to its applicability to the fair clustering problem [23].

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, 38], or by using the existence of a suitable embedding in the analysis [11, 24].

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

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 [36] and in practice [26]. Here there 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 [37]. Although its 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 $f : \mathbb{R}^d \rightarrow \mathbb{R}^m$ 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, 30, 35], with Narayanan and Nelson [35] 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, 24].

For an overview on practical aspects of dimension reduction, we refer to Venkatsubramanian and Wang [39]. In our evaluation, we focus on using PCA. The performance of the algorithms when applying a Johnson-Lindenstrauss transformation does not affect the behaviour of an algorithm that only depends on pairwise distances. We note that terminal embeddings, combined with an iterative application of the coreset construction from [5], can reduce the target dimension to a factor $\tilde{O}(\varepsilon^{-2} \log k)$. This is mainly of theoretical interest, as in practice the deciding factor wrt the target dimension is the precision, rather than dependencies on $\log n$ and $\log k$.

3 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 $t = a \cdot k + b$, $a \in \{0, \dots, \alpha - 1\}$ and $b \in \{1, \dots, k\}$, of A to be $k^{\alpha-a+1}$ stacks of v_b^{a+1} .

To get a better feel for the construction, we have given two small example benchmarks for $k = 2$ and $k = 3$ in the appendix.

3.1 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

23:6 An Empirical Evaluation of k -Means Coresets

the i th point as well as the i th 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

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

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

► **Lemma 1** (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 i th 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 [32, 33]. 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

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

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

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

- 180 1. The distance between these clustering is $1 - \frac{1}{k}$, i.e. it is maximized.
- 181 2. The clusterings have equal cost and the centers in each clustering have equal cost.
- 182 3. The clusterings are induced by a set of centers in \mathbb{R}^d .

183 A formal verification of these properties can be found in the appendix. The first and
184 second property ensure that (equally good) solutions we use for evaluation are spread out
185 through the solution space. In other words, we are less likely to only focus on a set of
186 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) coresets guarantee has to apply to.

The solutions we consider are given as follows. For the columns $a \cdot k + 1, \dots (a + 1) \cdot k$, we define the clustering $\mathcal{C}^a = \{C_1^a, \dots, C_k^a\}$ 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 .

3.2 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 points³.

Consider the clustering $\mathcal{C}^a = \{C_1^a, \dots, C_k^a\}$ for some a and let Ω with weights $w : \Omega \rightarrow \mathbb{R}_{\geq 0}$ be the coreset. Note that there are α many such clusterings, for each value of a . 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 $w(C_i^a \cap \Omega) \geq |C_i^a|(1 - \varepsilon)$, we place a center at $\mu(C_i^a)$. Conversely, if $w(C_i^a \cap \Omega) < |C_i^a|(1 - \varepsilon)$, 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.

We now compare the cost as computed on the coreset and the true cost of \mathcal{S} . Due to Lemma 1 and the fact that all clusters have equal cost, 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

$$\text{cost}_{C_i^a}(\mathcal{S}) \approx \left(1 + \frac{2}{\alpha}\right) \cdot \text{cost}_{C_j^a}(\{\mu(C_j^a)\}).$$

Conversely, the cost on the coreset is

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

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. For all possible (suitably discretized) thresholds for deficiency, we can now identify the clustering \mathcal{C}^a with a maximum number of deficient clusters and use the aforementioned construction to get a lower bound on the distortion.

3.3 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.

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

222 4 Experiments

223 In this section, we present how we evaluated different algorithms. First, we propose our
 224 evaluation procedure which gauges the quality of coresets. Then, we describe the data sets
 225 used for the empirical evaluation and our experimental setup. Finally, we detail the outcome
 226 of the experiments and our interpretation of the results.

227 4.1 Evaluation Procedure

228 As mentioned before, finding a k -clustering which results in a maximal distortion is difficult.
 229 Instead, we can estimate the quality of coresets on real-world data sets as follows: we first
 230 compute a coreset Ω . Then, run k -means++ on Ω to find a set of k centers C . We repeat
 231 the k -means++ algorithm 5 times and pick the set of centers with the largest distortion.

232 While it is useful to evaluate coresets on real-world data sets, it can be tricky to gauge
 233 the general performance of coreset algorithms using only a small selection of data sets. For
 234 this reason, we used our benchmark to complement the evaluation on real-world data sets.
 235 The benchmark accomplishes two important tasks. First, the benchmark allows us to quickly
 236 find a bad solution because both good and bad clusterings are known a priori. It is unclear
 237 how to find bad clusterings for real-world data sets. Second, it is easier to make a fair
 238 comparison of different coreset constructions because the benchmark is known to generate
 239 hard instances for all known coreset algorithms. This cannot be said for real-world data sets.
 240 For the benchmark, we computed the distortion following the evaluation procedure described
 241 in Section 3.

242 Every randomized coreset construction was repeated 10 times. We aggregated the reported
 243 distortions by taking the average over all 10 evaluations. In addition, we preprocessed the
 244 data using PCA (compare Section 2.1).

245 4.2 Data sets

246 We conducted experiments on five real-world data sets and four instances of our benchmark.
 247 Benchmark instances were generated to match approximately the sizes of the real-world
 248 data sets. The sizes of the data sets are summarized in Table 1. We now provide a brief
 249 description of each of the real-world data sets.

250 The *Census*⁴ dataset is a small subset of the Public Use Microdata Samples from 1990
 251 US census. It consists of demographic information encoded as 68 categorical attributes of
 252 2,458,285 individuals.

253 *Covertypes*⁵ is comprised of cartographic descriptions and forest cover type of four wilder-
 254 ness areas in the Roosevelt National Forest of Northern Colorado in the US. It consists of
 255 581,012 records, 54 cartographic variables and one class variable. Although *Covertypes* was
 256 originally made for classification tasks, it is often used for clustering tasks by removing the
 257 class variable [1].

258 The data set with the fewest number of dimensions is *Tower*⁶. This data set consists of
 259 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
 260 pixel is represented by a RGB color value.

⁴ [https://archive.ics.uci.edu/ml/datasets/US+Census+Data+\(1990\)](https://archive.ics.uci.edu/ml/datasets/US+Census+Data+(1990))

⁵ <https://archive.ics.uci.edu/ml/datasets/covertypes>

⁶ <http://homepages.uni-paderborn.de/frahling/coremeans.html>

	Data points	Dimensions				
			k	α	Data points	Dimensions
<i>Caltech</i>	3,680,458	128	10	6	1,000,000	60
<i>Census</i>	2,458,285	68	20	5	3,200,000	100
<i>Coverttype</i>	581,012	54	30	4	810,000	120
<i>NYTimes</i>	500,000	102,660	40	4	2,560,000	160
<i>Tower</i>	4,915,200	3				

■ **Table 1** Size of real world data sets (left) and evaluated benchmarks (right).

Inspired by [17], *Caltech* was created by computing SIFT features from the images in the Caltech101⁷ 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.

4.3 Preprocessing & Experimental Setup

To understand how denoising effects the quality of the outputted coresets, we applied Principal Component Analysis (PCA) on *Caltech*, *Census* and *Coverttype* by using the k 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*, *Coverttype* 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 GitHub⁹. For BICO, we used the authors' reference implementation¹⁰. 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.

⁷ http://www.vision.caltech.edu/Image_Datasets/Caltech101/

⁸ <https://archive.ics.uci.edu/ml/datasets/bag+of+words>

⁹ Link to repository will be provided later.

¹⁰ <https://ls2-www.cs.tu-dortmund.de/grav/en/bico>

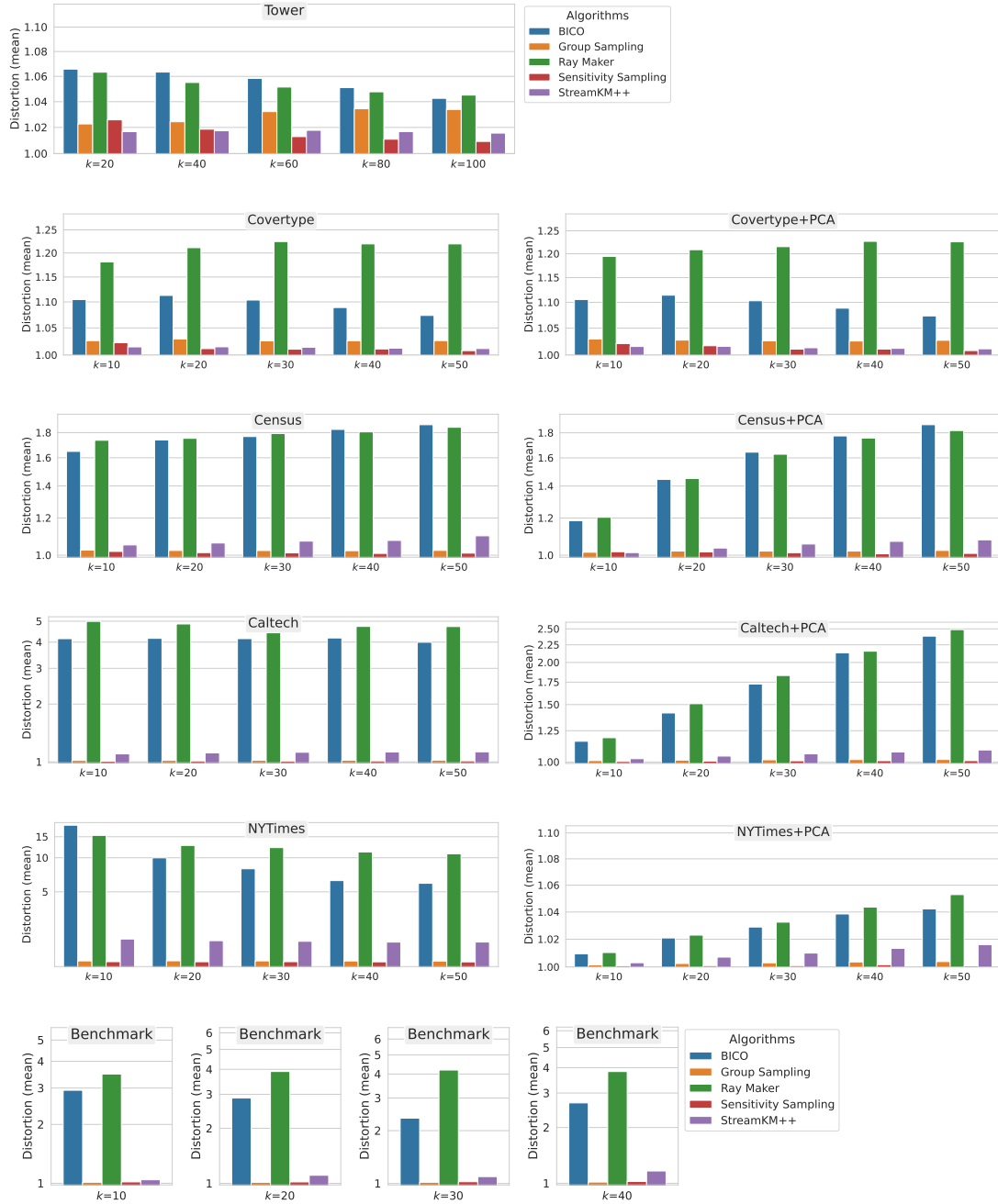


Figure 1 The distortions of the evaluated coreset algorithms on five real-world data sets and on four benchmark instances. The axis is non-linear as otherwise the bars for Sensitivity Sampling and Group Sampling would disappear on the plots. Their distortions are very close to 1.

4.4 Outcome of Experiments

We summarized the distortions in Figure 1, for numerical variance we refer to Table 2 and Table 3 in the appendix. 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 *Coverttype* with its 54 features, Ray Maker performs the worst followed by BICO and Group Sampling while Sensitivity Sampling and StreamKM++ perform the best. Differences 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 Sensitivity 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 *Coverttype*, 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.

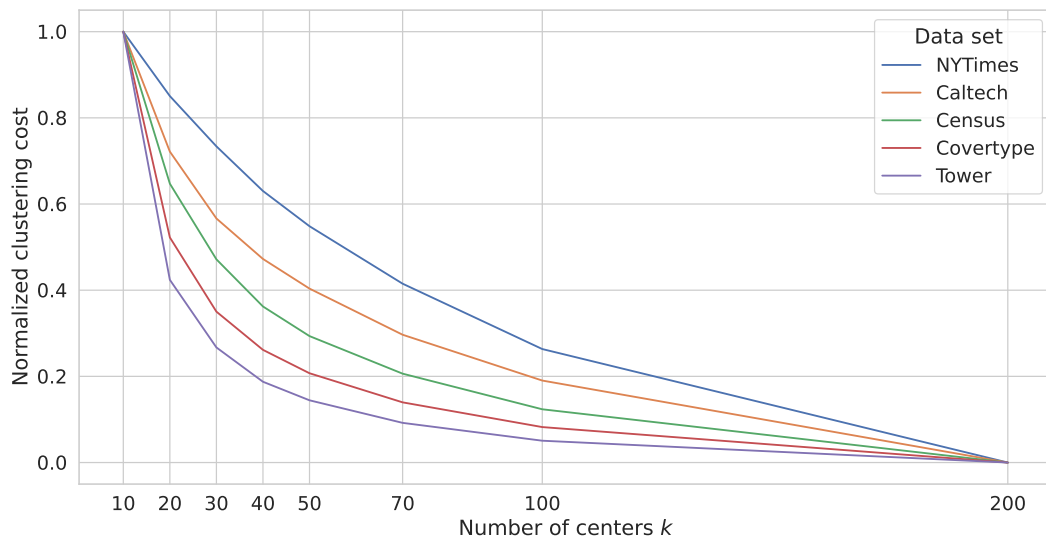
4.5 Interpretation of Experimental Results

Optimization versus Compression

While all five algorithms are equally matched when optimizing on the candidate coresets, coreset quality performance differ significantly (see Figure 1). We omit tables and plots detailing the cost due to space restrictions. For all data sets, the obtained costs differed insignificantly for all values of k .

The quality of the coreset itself can be closely tied to the change in cost with increasing number of centers. It is not uncommon for the k -means cost of real-world data sets to drop significantly for larger values of k . 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. *Tower* is a good example of a data set where optimization is very close to compression. Its cost curve bends the most which indicates that adding more centers help reduce the cost. One of the strengths of the benchmark is that there is no way of reducing the cost without capturing the right subclusters within a benchmark instance. This means that the cost does not decrease markedly beyond a certain value of k even if more centers are added, see Figure 4 in the appendix.

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. On 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++. 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.



■ **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*, *Covertypes*, 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.

332 Movement-based versus Sampling-based Approaches

333 In general, movement-based constructions perform the worst in terms of coreset quality. We
 334 observed that BICO and Ray Maker have the highest distortions across all data sets including
 335 on the benchmark instances. Among the sampling-based algorithms, Sensitive Sampling
 336 performs well with Group Sampling generally being competitive. This runs contrary to theory
 337 where Group Sampling has the better (currently known) theoretical bounds. StreamKM++
 338 is an interesting case. Like the movement-based methods, its distortion increases with the
 339 dimension. Nevertheless, it generally performs significantly better than BICO and Ray
 340 Maker. This can be attributed to the fact that the coreset produced by StreamKM++
 341 consists entirely of k -means++ centers weighted by the number of points of a minimal cost
 342 assignment. This is similar to movement-based algorithms such as BICO. Nevertheless, it
 343 also retains some of the performance from pure importance schemes.

344 In practice as well as in theory, the distortion of movement-based algorithms are affected
 345 by the dimension. By comparison, sampling-based algorithms are affected very little. Theor-
 346 etically, there should not exist a difference, as the sampling bounds are independent of the
 347 dimension. What little effect can be observed is likely due to PCA making it easier to find
 348 low cost solutions that form the backbone of all coreset constructions. StreamKM++ is an
 349 interesting case, as it is still affected by the dimension, though less than the other movement
 350 based methods, due its performance without dimension reduction being significantly better
 351 than the worst-case bounds would suggest.

352 Impact of PCA

353 On almost all our data sets, the performance improves when input data is preprocessed with
 354 PCA, especially for the movement-based algorithms. Empirically, the more noise is removed
 355 (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 *Coverttype*, 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 5). 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, importance sampling methods are still superior when it comes to the quality of the compression. Summarizing, all methods benefit from PCA, and in case of movement based constructions, we consider PCA a necessary preprocessing step. For the sampling based methods, the computational expense of using PCA in preprocessing does not seem justify the comparatively meagre gains in coreset distortion.

5 Conclusion

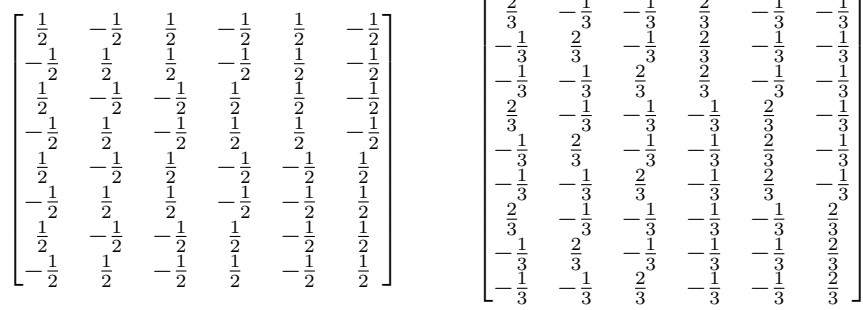
In this work, we studied how to assess the quality of k -means coresets computed by state-of-the-art algorithms. Previous work generally measured the quality of optimization algorithms run on the coreset, which we empirically observed to be poor indicator of coreset quality. For real data sets, we sampled candidate clusterings and evaluated the worst case distortion on them. Complementing this, we also proposed a benchmark framework which generates hard instances for known k -means coreset algorithms. Our experiments indicate a general advantage for algorithms based on importance sampling over movement-based methods aiming towards computing low-cost clusterings. Despite movement based methods running on very efficient code, it is necessary to compliment them with rather expensive dimension reduction methods, rendering what efficiency they might have over importance sampling somewhat moot.

Two results bear further investigation. First, the currently known provable coreset sizes for Sensitivity Sampling are worse than those provable via Group Sampling. Empirically, we observed the opposite: While Group Sampling is competitive, Sensitivity Sampling always outperforms it. Since Group Sampling requires somewhat cumbersome computational overhead, practical applications should prefer Sensitivity Sampling. In light of these results, a theoretical analysis for Sensitivity Sampling matching the performance of Group Sampling would be welcome.

The second point of interest focusses on the performance of StreamKM++. The distortion of this algorithm is significantly better than what one would expect from its theoretical analysis. Empirically, StreamKM++ is notably better than the other movement based constructions across all data sets, and especially on high dimensional data. While it is not competitive to the pure importance sampling algorithms, there are several reasons for investigating it further. It essentially only requires running k -means++ for additional iterations, which is already a nearly ubiquitous algorithm for the k -means problem. Although the other sampling-based coreset algorithms can also be readily implemented, doing so might be cumbersome. In particular, the theoretically (but not empirically) best algorithm Group Sampling requires extensive preprocessing steps. This begs the question whether there exist a better theoretical analysis for StreamKM++.

In addition, StreamKM++ currently weighs each point by the number of points assigned to it. It may also be possible to improve the performance of the algorithm in both theory and practise by using a different weighting scheme. We leave this as an open problem for future research.

A Appendix: Properties of the Benchmark and Missing Plots



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

403 ► **Fact 2.** For $a \neq a'$, we have $d(\mathcal{C}^a, \mathcal{C}^{a'}) = 1 - 1/k$.

404 **Proof.** Consider an arbitrary vector v_i^ℓ . By construction, the positive entries of v_i^ℓ range
 405 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
 406 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
 407 vector v' , v' and v_i^ℓ can share at most one positive coordinate. Inductively, the same holds
 408 true for any concatenation of vectors $v_j^{\ell-h}$. Thus, the two clusters induced by the columns
 409 formed by concatenating the vectors v can share only a $1/k$ fraction of the points. Since
 410 each cluster consists of exactly $k^\alpha/k = k^{\alpha-1}$ points, the confusion matrix M only has entries
 411 $\frac{n}{k^2}$ and for any permutation π , we have $d(\mathcal{C}^a, \mathcal{C}^{a'}) = 1 - 1/k$. ◀

412 ► **Fact 3.** For any C_j^a , we have $\text{cost}_{C_j^a}(\{\mu(C_j^a)\}) = (\alpha - 1) \cdot k^{\alpha-2} \cdot (k - 1)$.

413 **Proof.** Without loss of generality, we consider C_1^0 ; the proof is analogous for the other choices
 414 of j and a . We first note that for any point $A_i \in C_1^0$, the coordinates $A_{i,\ell}$ are identical
 415 for $\ell < k$. Furthermore for the column $\ell \geq k$, we have by construction $\sum_{A_i \in C_j} A_{i,\ell} =$
 416 $k^{\alpha-1} \cdot \frac{k-1}{k} + (k^\alpha - k^{\alpha-1}) \frac{1}{k} = k^{\alpha-1} \cdot (\frac{k-1}{k} - (k-1) \frac{1}{k}) = 0$. Therefore, the mean of C_1^0 satisfies
 417 $\mu(C_1^0)_\ell = \begin{cases} A_{i,\ell} & \text{if } \ell < k \\ 0 & \text{else.} \end{cases}$. Thus, the cost is precisely $(\alpha - 1) \cdot k^{\alpha-1} \cdot \left(\left(\frac{k-1}{k} \right)^2 + \left(\frac{1}{k} \right)^2 \right) =$
 418 $(\alpha - 1) \cdot k^{\alpha-2} \cdot (k - 1)$. ◀

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

420 ► **Fact 4.** For a clustering \mathcal{C}^a , let $\mu(C_j^a)$ denote the mean of cluster C_j^a . Then every point
 421 is assigned to its closest center. Moreover, every point A_i of C_j^a has equal distance to any
 422 center $\mu(C_h^a)$ with $h \neq j$.

423 **Proof.** Again, we assume without loss of generality $a = 0$. Let A_i be an arbitrary point of
 424 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,
 425 the positive coordinates of A_i are not equal to the positive coordinates of $\mu(C_j^a)$. The only
 426 difference in coordinates between the means of $\mu(C_j^a)$ and $\mu(C_h^a)$ are the first k coordinates,
 427 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
 428 be closer to A_i .

429 To prove that the distances between A_i and any $\mu(C_h^a)$ with $h \neq j$ are equal, again
 430 consider that any difference can only exist among the first k coordinates. Here, we have
 431 $\mu(C_h^a)_h = \frac{k-1}{k}$, and the remaining columns are $-\frac{1}{k}$. Since $A_{i,h} = -\frac{1}{k}$ for any $h \neq j$, the
 432 claim follows. ◀

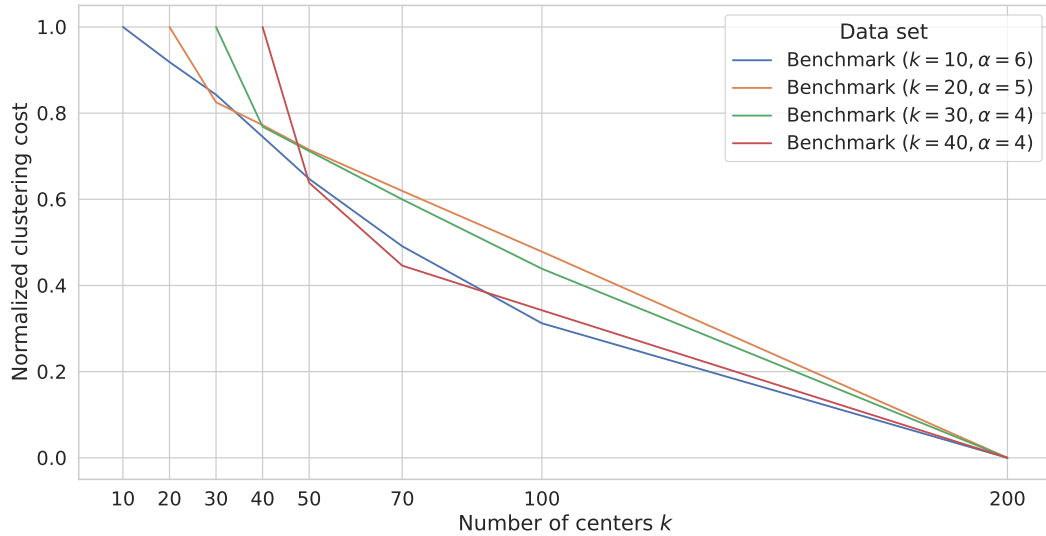
433 B Hardness of Weak Coreset Evaluation

434 Here, we show that it is in general co-NP hard to evaluate whether two point sets A and B
 435 are weak coresets of each other. A weak coreset only requires that a $(1 + \varepsilon)$ approximation
 436 for one point set is a $(1 + O(\varepsilon))$ for the other.

437 ▶ **Proposition 5.** *Given two point sets A and B in \mathbb{R}^d and a sufficiently small (constant)*
 438 *$\varepsilon > 0$, it is co-NP hard to decide whether A is a weak coreset of B .*

439 **Proof.** First, we recall that for some ε_0 and candidate clustering cost V , it is NP-hard
 440 to decide whether there exists a clustering C with cost in $\text{cost}_A(C) \leq V$ and $\text{cost}_B(C) \geq$
 441 $(1 + \varepsilon_0) \cdot V$. Conversely, it is co-NP-hard to decide whether there exists no set of centers C
 442 such that $\text{cost}_A(C) \leq V$ and $\text{cost}_B(C) \geq (1 + \varepsilon_0) \cdot V$. ◀

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



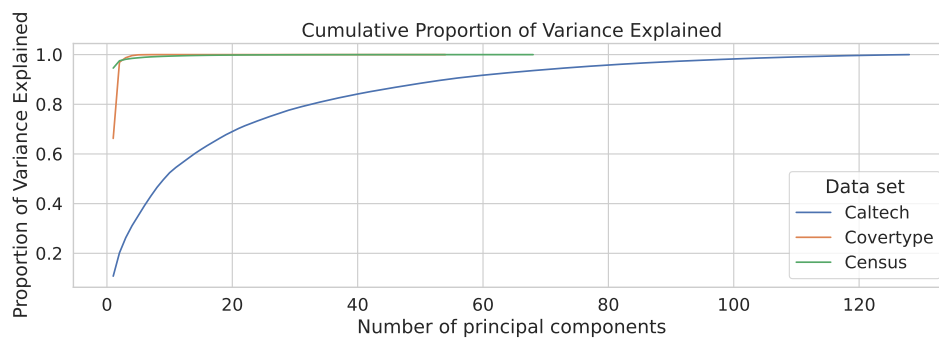
■ **Figure 4** Shows the clustering costs of four instances of the benchmark framework as a function of the number of centers. In contrast to real-world data sets, the costs do not decrease rapidly as more cluster centers are added.

Data set	k	BICO	Group Sampling	Ray Maker	Sensitivity Sampling	StreamKM++
Benchmark	10	2.93 (0.144)	1.01 (0.002)	3.49 (0.055)	1.02 (0.003)	1.04 (0.002)
	20	2.87 (0.137)	1.01 (0.002)	3.90 (0.112)	1.02 (0.003)	1.11 (0.002)
	30	2.34 (0.070)	1.02 (0.001)	4.19 (0.137)	1.02 (0.002)	1.10 (0.002)
	40	2.68 (0.230)	1.02 (0.001)	3.83 (0.172)	1.02 (0.002)	1.17 (0.004)
Caltech	10	4.15 (0.485)	1.02 (0.003)	4.99 (0.124)	1.01 (0.004)	1.10 (0.004)
	20	4.16 (0.708)	1.02 (0.003)	4.86 (0.134)	1.01 (0.003)	1.12 (0.004)
	30	4.15 (0.416)	1.02 (0.001)	4.78 (0.117)	1.01 (0.002)	1.13 (0.003)
	40	4.18 (0.486)	1.02 (0.002)	4.73 (0.068)	1.01 (0.002)	1.13 (0.001)
	50	3.99 (0.560)	1.02 (0.002)	4.72 (0.090)	1.01 (0.002)	1.13 (0.002)
Census	10	1.65 (0.082)	1.03 (0.011)	1.74 (0.044)	1.02 (0.006)	1.05 (0.005)
	20	1.74 (0.054)	1.02 (0.004)	1.75 (0.038)	1.01 (0.005)	1.06 (0.003)
	30	1.77 (0.063)	1.02 (0.003)	1.79 (0.032)	1.01 (0.005)	1.07 (0.002)
	40	1.83 (0.038)	1.02 (0.002)	1.81 (0.021)	1.01 (0.005)	1.08 (0.003)
	50	1.87 (0.088)	1.02 (0.002)	1.85 (0.044)	1.01 (0.003)	1.10 (0.059)
Coverttype	10	1.10 (0.002)	1.03 (0.011)	1.18 (0.016)	1.02 (0.007)	1.01 (0.002)
	20	1.11 (0.004)	1.03 (0.010)	1.21 (0.015)	1.01 (0.007)	1.01 (0.002)
	30	1.10 (0.003)	1.03 (0.005)	1.22 (0.008)	1.01 (0.005)	1.01 (0.002)
	40	1.09 (0.001)	1.03 (0.005)	1.22 (0.012)	1.01 (0.005)	1.01 (0.001)
	50	1.07 (0.001)	1.03 (0.003)	1.22 (0.008)	1.01 (0.002)	1.01 (0.001)
NYTimes	10	18.59 (5.356)	1.04 (0.003)	15.37 (0.740)	1.02 (0.004)	1.75 (0.105)
	20	9.98 (1.559)	1.04 (0.003)	12.71 (0.448)	1.01 (0.004)	1.69 (0.024)
	30	8.03 (2.156)	1.04 (0.002)	12.18 (0.974)	1.02 (0.003)	1.66 (0.030)
	40	6.31 (0.436)	1.03 (0.003)	11.20 (0.742)	1.01 (0.002)	1.63 (0.024)
	50	5.97 (0.381)	1.03 (0.002)	10.81 (0.571)	1.01 (0.002)	1.63 (0.026)
Tower	20	1.07 (0.003)	1.02 (0.007)	1.06 (0.004)	1.03 (0.009)	1.02 (0.001)
	40	1.06 (0.004)	1.02 (0.006)	1.06 (0.002)	1.02 (0.007)	1.02 (0.001)
	60	1.06 (0.001)	1.03 (0.004)	1.05 (0.004)	1.01 (0.003)	1.02 (0.001)
	80	1.05 (0.001)	1.03 (0.005)	1.05 (0.002)	1.01 (0.006)	1.02 (0.001)
	100	1.04 (0.002)	1.03 (0.004)	1.05 (0.001)	1.01 (0.004)	1.02 (0.001)

■ **Table 2** Distortions of the evaluation algorithms on all data sets without PCA preprocessing. Each cell specify the mean distortion along with the standard deviation in brackets of 10 repetition of the experiment.

Data set	k	BICO	Group Sampling	Ray Maker	Sensitivity Sampling	StreamKM++
Caltech+PCA	10	1.16 (0.004)	1.01 (0.002)	1.19 (0.009)	1.00 (0.001)	1.02 (0.002)
	20	1.42 (0.016)	1.01 (0.002)	1.51 (0.008)	1.01 (0.001)	1.04 (0.002)
	30	1.73 (0.045)	1.02 (0.001)	1.83 (0.014)	1.01 (0.001)	1.06 (0.002)
	40	2.13 (0.083)	1.02 (0.002)	2.16 (0.016)	1.01 (0.002)	1.07 (0.002)
	50	2.38 (0.139)	1.02 (0.001)	2.49 (0.017)	1.01 (0.002)	1.09 (0.002)
Census+PCA	10	1.18 (0.012)	1.01 (0.005)	1.20 (0.016)	1.02 (0.009)	1.01 (0.002)
	20	1.44 (0.046)	1.02 (0.003)	1.45 (0.019)	1.02 (0.010)	1.04 (0.002)
	30	1.64 (0.044)	1.02 (0.004)	1.63 (0.021)	1.01 (0.007)	1.06 (0.003)
	40	1.77 (0.035)	1.02 (0.002)	1.76 (0.028)	1.01 (0.004)	1.07 (0.002)
	50	1.87 (0.068)	1.02 (0.002)	1.82 (0.017)	1.01 (0.004)	1.08 (0.002)
Covertypes+PCA	10	1.11 (0.003)	1.03 (0.012)	1.19 (0.011)	1.02 (0.010)	1.02 (0.002)
	20	1.11 (0.003)	1.03 (0.006)	1.21 (0.010)	1.02 (0.007)	1.02 (0.002)
	30	1.10 (0.002)	1.03 (0.007)	1.22 (0.011)	1.01 (0.005)	1.01 (0.002)
	40	1.09 (0.002)	1.03 (0.006)	1.23 (0.010)	1.01 (0.005)	1.01 (0.001)
	50	1.07 (0.001)	1.03 (0.003)	1.23 (0.008)	1.01 (0.003)	1.01 (0.001)
NYTimes+PCA	10	1.01 (0.000)	1.00 (0.000)	1.01 (0.001)	1.00 (0.000)	1.00 (0.000)
	20	1.02 (0.000)	1.00 (0.000)	1.02 (0.001)	1.00 (0.000)	1.01 (0.000)
	30	1.03 (0.000)	1.00 (0.000)	1.03 (0.001)	1.00 (0.000)	1.01 (0.000)
	40	1.04 (0.001)	1.00 (0.000)	1.04 (0.001)	1.00 (0.002)	1.01 (0.000)
	50	1.04 (0.002)	1.00 (0.000)	1.05 (0.001)	1.00 (0.000)	1.02 (0.000)

■ **Table 3** Distortions of the evaluation algorithms on the data sets preprocessed with PCA. Each cell specify the mean distortion along with the standard deviation in brackets of 10 repetitions.



■ **Figure 5** The cumulative proportion of explained variance by principal components on *Caltech*, *Covertypes*, and *Census*.

References

- 1 Marcel R. Ackermann, Marcus Mörtens, 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, Sarel 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 Daniel 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 l_p 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 Daniel 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.

- 483 11 Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. A new coresets framework
484 for clustering. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd*
485 *Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25,*
486 *2021*, pages 169–182. ACM, 2021.
- 487 12 Michael Elkin, Arnold Filtser, and Ofer Neiman. Terminal embeddings. *Theor. Comput. Sci.*,
488 697:1–36, 2017. doi:10.1016/j.tcs.2017.06.021.
- 489 13 Dan Feldman and Michael Langberg. A unified framework for approximating and clustering
490 data. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San*
491 *Jose, CA, USA, 6-8 June 2011*, pages 569–578, 2011.
- 492 14 Dan Feldman and Michael Langberg. A unified framework for approximating and clustering
493 data. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San*
494 *Jose, CA, USA, 6-8 June 2011*, pages 569–578, 2011.
- 495 15 Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data:
496 Constant-size coresets for k -means, PCA and projective clustering. In *Proceedings of the*
497 *Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New*
498 *Orleans, Louisiana, USA, January 6-8, 2013*, pages 1434–1453, 2013.
- 499 16 Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data:
500 Constant-size coresets for k -means, pca, and projective clustering. *SIAM J. Comput.*, 49(3):601–
501 657, 2020. doi:10.1137/18M1209854.
- 502 17 Hendrik Fichtenberger, Marc Gillé, Melanie Schmidt, Chris Schwiegelshohn, and Christian
503 Sohler. BICO: BIRCH meets coresets for k -means clustering. In *Algorithms - ESA 2013 -*
504 *21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings,*
505 pages 481–492, 2013.
- 506 18 Gereon Frahling and Christian Sohler. Coresets in dynamic geometric data streams. In
507 *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages
508 209–217, 2005.
- 509 19 Panos Giannopoulos, Christian Knauer, Magnus Wahlström, and Daniel Werner. Hardness of
510 discrepancy computation and ϵ -net verification in high dimension. *J. Complex.*, 28(2):162–176,
511 2012. doi:10.1016/j.jco.2011.09.001.
- 512 20 Sariel Har-Peled and Akash Kushal. Smaller coresets for k -median and k -means clustering.
513 *Discrete & Computational Geometry*, 37(1):3–19, 2007.
- 514 21 Sariel Har-Peled and Akash Kushal. Smaller coresets for k -median and k -means clustering.
515 *Discrete & Computational Geometry*, 37(1):3–19, 2007.
- 516 22 Sariel Har-Peled and Soham Mazumdar. On coresets for k -means and k -median clustering. In
517 *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA,*
518 *June 13-16, 2004*, pages 291–300, 2004.
- 519 23 Lingxiao Huang, Shaofeng H.-C. Jiang, and Nisheeth K. Vishnoi. Coresets for clus-
520 tering with fairness constraints. In Hanna M. Wallach, Hugo Larochelle, Alina Bey-
521 gelzimer, Florence d’Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances*
522 *in Neural Information Processing Systems 32: Annual Conference on Neural Inform-*
523 *ation Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC,*
524 *Canada*, pages 7587–7598, 2019. URL: [https://proceedings.neurips.cc/paper/2019/hash/](https://proceedings.neurips.cc/paper/2019/hash/810dfbbebb17302018ae903e9cb7a483-Abstract.html)
525 [810dfbbebb17302018ae903e9cb7a483-Abstract.html](https://proceedings.neurips.cc/paper/2019/hash/810dfbbebb17302018ae903e9cb7a483-Abstract.html).
- 526 24 Lingxiao Huang and Nisheeth K. Vishnoi. Coresets for clustering in euclidean spaces: im-
527 portance sampling is nearly optimal. In Konstantin Makarychev, Yury Makarychev, Madhur
528 Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proceedings of the 52nd Annual ACM*
529 *SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26,*
530 *2020*, pages 1416–1429. ACM, 2020. doi:10.1145/3357713.3384296.
- 531 25 Piotr Indyk, Sepideh Mahabadi, Shayan Oveis Gharan, and Alireza Rezaei. Composable
532 core-sets for determinant maximization problems via spectral spanners. In Shuchi Chawla,
533 editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020*,

- 534 Salt Lake City, UT, USA, January 5-8, 2020, pages 1675–1694. SIAM, 2020. doi:10.1137/1.
535 9781611975994.103.
- 536 **26** Jan-Philipp W. Kappmeier, Daniel R. Schmidt, and Melanie Schmidt. Solving k -means on
537 high-dimensional big data. In *Experimental Algorithms - 14th International Symposium,*
538 *SEA 2015, Paris, France, June 29 - July 1, 2015, Proceedings*, pages 259–270, 2015. doi:
539 10.1007/978-3-319-20086-6_20.
- 540 **27** Michael Langberg and Leonard J. Schulman. Universal ε -approximators for integrals. In
541 *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms,*
542 *SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 598–607, 2010.
- 543 **28** Michael Langberg and Leonard J. Schulman. Universal ε -approximators for integrals. In
544 *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms,*
545 *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.
- 546 **29** Alaa Maalouf, Ibrahim Jubran, and Dan Feldman. Fast and accurate least-mean-squares
547 solvers. In *Advances in Neural Information Processing Systems*, pages 8307–8318, 2019.
- 548 **30** Sepideh Mahabadi, Konstantin Makarychev, Yury Makarychev, and Ilya P. Razenshteyn.
549 Nonlinear dimension reduction via outer bi-lipschitz extensions. In *Proceedings of the 50th*
550 *Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA,*
551 *USA, June 25-29, 2018*, pages 1088–1101, 2018. URL: [http://doi.acm.org/10.1145/3188745.](http://doi.acm.org/10.1145/3188745.3188828)
552 3188828, doi:10.1145/3188745.3188828.
- 553 **31** Tung Mai, Anup B. Rao, and Cameron Musco. Coresets for classification - simplified and
554 strengthened. *CoRR*, abs/2106.04254, 2021. URL: <https://arxiv.org/abs/2106.04254>,
555 arXiv:2106.04254.
- 556 **32** Marina Meila. Comparing clusterings: an axiomatic view. In Luc De Raedt and Stefan Wrobel,
557 editors, *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML*
558 *2005), Bonn, Germany, August 7-11, 2005*, volume 119 of *ACM International Conference*
559 *Proceeding Series*, pages 577–584. ACM, 2005. doi:10.1145/1102351.1102424.
- 560 **33** Marina Meila. The uniqueness of a good optimum for k -means. In William W. Cohen and
561 Andrew W. Moore, editors, *Machine Learning, Proceedings of the Twenty-Third International*
562 *Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, volume 148 of
563 *ACM International Conference Proceeding Series*, pages 625–632. ACM, 2006. doi:10.1145/
564 1143844.1143923.
- 565 **34** Alexander Munteanu, Chris Schwiegelshohn, Christian Sohler, and David P. Wood-
566 ruff. On coresets for logistic regression. In Samy Bengio, Hanna M. Wallach, Hugo
567 Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Ad-*
568 *vances in Neural Information Processing Systems 31: Annual Conference on Neural*
569 *Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal,*
570 *Canada*, pages 6562–6571, 2018. URL: [https://proceedings.neurips.cc/paper/2018/hash/](https://proceedings.neurips.cc/paper/2018/hash/63bfd6e8f26d1d3537f4c5038264ef36-Abstract.html)
571 63bfd6e8f26d1d3537f4c5038264ef36-Abstract.html.
- 572 **35** Shyam Narayanan and Jelani Nelson. Optimal terminal dimensionality reduction in euclidean
573 space. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM*
574 *SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26,*
575 *2019*, pages 1064–1069. ACM, 2019. doi:10.1145/3313276.3316307.
- 576 **36** Melanie Schmidt, Chris Schwiegelshohn, and Christian Sohler. Fair coresets and streaming
577 algorithms for fair k -means. In *Approximation and Online Algorithms - 17th International*
578 *Workshop, WAOA 2019, Munich, Germany, September 12-13, 2019, Revised Selected Papers,*
579 pages 232–251, 2019. doi:10.1007/978-3-030-39479-0_16.
- 580 **37** Christian Sohler and David P. Woodruff. Strong coresets for k -median and subspace ap-
581 proximation: Goodbye dimension. In *59th IEEE Annual Symposium on Foundations of*
582 *Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 802–813, 2018.
583 doi:10.1109/FOCS.2018.00081.

- 585 38 Christian Sohler and David P. Woodruff. Strong coresets for k-median and subspace approx-
586 imation: Goodbye dimension. *CoRR*, abs/1809.02961, 2018. URL: [http://arxiv.org/abs/](http://arxiv.org/abs/1809.02961)
587 1809.02961, [arXiv:1809.02961](https://arxiv.org/abs/1809.02961).
- 588 39 Suresh Venkatasubramanian and Qiushi Wang. The johnson-lindenstrauss transform: An
589 empirical study. In Matthias Müller-Hannemann and Renato Fonseca F. Werneck, editors,
590 *Proceedings of the Thirteenth Workshop on Algorithm Engineering and Experiments, ALENEX*
591 *2011, Holiday Inn San Francisco Golden Gateway, San Francisco, California, USA, January*
592 *22, 2011*, pages 164–173. SIAM, 2011.
- 593 40 Dennis Wei. A constant-factor bi-criteria approximation guarantee for k-means++. In
594 Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett,
595 editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural*
596 *Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 604–612,
597 2016.
- 598 41 Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: A new data clustering algorithm
599 and its applications. *Data Min. Knowl. Discov.*, 1(2):141–182, 1997. URL: [http://dx.doi.](http://dx.doi.org/10.1023/A:1009783824328)
600 [org/10.1023/A:1009783824328](http://dx.doi.org/10.1023/A:1009783824328), [doi:10.1023/A:1009783824328](https://doi.org/10.1023/A:1009783824328).