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

2012 ACM Subject Classification Replace ccsdesc macro with valid one

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

Digital Object Identifier 10.4230/LIPIcs.ESA.2022.23

The European Symposium on Algorithms (ESA).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:35

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 $\Omega \subset A$ 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 $\sup_{q \in Q} \max\left(\frac{f(\Omega,q)}{f(A,q)}, \frac{f(A,q)}{f(\Omega,q)}\right) \leq D$. Coresets have been applied to a number of problems such as computational geometry [2, 6], linear algebra [26, 30], and machine learning [32, 35]. 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 a set of 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 $w : \Omega \to \mathbb{R}_{\geq 0}$ 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, 12, 14, 17, 23, 21, 25, 5, 28, 38], the size of coresets has been steadily improved with the current state of the art yielding a coreset with $\tilde{O}(k\varepsilon^{-2} \cdot \min(d, k, \varepsilon^{-2}))$ points for a distortion $D \leq (1 + \varepsilon)$ due to [9][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 set B with respect to convex ranges is co-NP hard [20]. 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, 18]. 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. The *best* coreset algorithm is considered to be the one which yields a clustering with the smallest cost.

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

This practice has substantial drawbacks. The first is that evaluation method conflates the two separate tasks of coreset construction and optimization. It is important to note that the first step of virtually all coreset algorithms is a low-cost (bicriteria) constant factor approximation, i.e. a solution with $\beta \cdot k$ clusters that costs at most $\alpha \cdot \mathrm{OPT}$, where OPT is the cost of an optimal k clustering. Given that this initial solution has an α approximation to the cost, a routine calculation shows that the additive error of the coreset, i.e. the maximum difference $|\mathrm{cost}_A(C) - \mathrm{cost}_B(C)|$ over all solutions C is at most $O(\alpha) \cdot \mathrm{cost}_A(C)$. In particular, in the case that the initial bicriteria approximation has $\alpha \ll 2$, which is not too difficult to achieve with more than k centers, any γ approximation algorithm will find solutions with approximation factor $O(\gamma + \alpha) \cdot \mathrm{OPT}$. In particular, the distortion may be unbounded, for example if B only consists of the k centers, while simply returning B itself yields a low cost clustering. Thus, it is difficult to measure coreset quality in this way.

The second drawback is that this practice will mainly measure the performance of the optimization algorithm, rather than the performance of the coreset algorithms. During its execution it might simply not consider any solution with high distortion. For example, if the approximation factor γ of the solution returned by the algorithm is large then this solution (as well as any even higher cost solution considered during the algorithm's execution) will have a low distortion.

The third drawback of this evaluation method is that is does not consider the main use cases of coresets, nor the full power of their guarantee. Indeed, if speeding up the computation of an optimization algorithm, one would hardly need a strong coreset; approximating the cost of every candidate solution, as weaker coreset definitions (or indeed a bicriteria approximation) would be suitable as well. A coreset's main and most powerful feature is *composability*, i.e. given two disjoint point sets X and Y, the union of a coreset of X and a coreset of Y is a coreset. Composability is what enables coresets to scale to massively parallel computation models and enables simple streaming algorithms via the merge and reduce technique. To which degree a coreset is composable is generally not a property of an optimal clustering of the point set, as optimal solutions C_X of X or C_Y of Y may have little in common with an optimal solution of $X \cup Y$.

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:

Movement-based constructions: Such algorithms compute a coreset Ω with T points given some input point set A such that $\mathrm{cost}_\Omega(C) \ll \mathrm{OPT}$, where OPT is the cost of an optimal k-means clustering of A. 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 [24, 37] and continue to be popular. Examples from theory include [19, 23].

Importance sampling: Points are sampled proportionate to their sensitivity which for a point p is defined as $sens(p) := \sup_C \frac{\min_{c \in C} \operatorname{dist}^2(p,c)}{\operatorname{cost}_A(C)}$ and weighted by their inverse sampling probability. In terms of theoretical performance, sensitivity sampling has largely replaced movement-based constructions, see for example [15, 29].

Of course, there exist algorithms that draw on techniques from both, see for example [12]. 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 $\min_{q \in K} \operatorname{dist}^2(p,q)$ 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 $O\left(\frac{k \log n}{\delta^{d/2} \varepsilon^d} \cdot \log^{d/2} \frac{k \log n}{\delta^{d/2} \varepsilon^d}\right)$. Despite its simplicity, its running time compares unfavourably to all other constructions.

BICO [18]: BICO combines the very fast, but poor quality clustering algorithm BIRCH [42] with the movement-based analysis from [19, 23]. 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 cluster is opened with center at p. Otherwise p is either added to the same cluster as q, or, if adding p to q's cluster increases the clustering cost 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 $O\left(k\varepsilon^{-d-2} \log n\right)$.

Sensitivity Sampling [14]: The simplest implementation of sensitivity sampling first computes an $(O(1), O(1))$ bicriteria approximation[2], for example by running k-means++ for $2k$ iterations [41]. 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 $\frac{\operatorname{dist}^2(p,q)}{\operatorname{cost}_{K_i}(\{q_i\})} + \frac{1}{|K_i|}$ 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 $\tilde{O}\left(kd\varepsilon^{-4}\right)$ ([14]), $\tilde{O}\left(k\varepsilon^{-6}\right)$ ([25]), or $\tilde{O}\left(k^2\varepsilon^{-4}\right)$ ([5]).

Group Sampling [12]: 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 [12] 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 $\tilde{O}\left(k\varepsilon^{-2} \min(d, k, \varepsilon^{-2})\right)$ ([9]). While this improves over sensitivity sampling, it is generally slower and not as easy to implement.

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

Ray Maker [22]: 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 $O(k^3 \cdot \varepsilon^{-d-1})$. The algorithm has recently received some attention due to its applicability to the fair clustering problem [24].

Dimension Reduction

Finally, we also combine coreset constructions with a variety of dimension reduction techniques. Since the seminal paper by Feldman, Schmidt, and Sohler [16], 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 [16, 39], or by using the existence of a suitable embedding in the analysis [12, 25]. 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 [37] and in practice [27]. In this paper, we mainly focus on principal component analysis. A more detailed discussion on aspects of dimension reduction is found in the appendix.

3 Benchmark Construction

In this section, we describe our benchmark. We start by describing the aims of the benchmark, followed by giving the construction. Our aim is to generate a data set containing many clusterings with the following properties.

1. The benchmark has many clusterings that, in a well defined sense, are highly dissimilar. Specifically, we want the overlap between any two clusters of different clusterings to be small.
2. The different clusterings have very similar and low cost. This ensures that despite the solutions being different in terms of composition and center placement, a good coreset has to consider them equally regarding distortion.
3. The clusterings are induced by a minimal cost assignment of input points to a set of centers in $\mathbb{R}^d$. This final property ensures that the coreset guarantee has to apply to these clusterings.

To generate the benchmark, we now use the following construction. The benchmark has a parameter α which controls the number of points and dimensions of the generated data instance. For a given value of k, the benchmark instance consists of $n = k^\alpha$ points and $d = \alpha \cdot k$ dimensions, i.e. we will construct and $n \times d$ matrix A where every row corresponds to an input point and every column corresponds to one of the dimensions.

Let $\mathbb{1}_k$ be the k-dimensional all-one vector and v_i^1 be 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 $v_i^\ell = v_i^{\ell-1} \otimes \mathbb{1}_k$, where $\otimes$ denotes the Kronecker product, i.e. $v_i^{\ell-1} \otimes \mathbb{1}_k = \begin{bmatrix} (v_i^{\ell-1})_1 \cdot \mathbb{1}_k \\ (v_i^{\ell-1})_2 \cdot \mathbb{1}_k \\ \vdots \\ (v_i^{\ell-1})_1 \cdot \mathbb{1}_k \end{bmatrix}$.

Finally, set the t-th column of A, for $t = a \cdot k + b$, $a \in \{0, \dots \alpha - 1\}$ and $b \in \{1, \dots k\}$, to be $\mathbb{1}_{k^{\alpha-a+1}} \otimes v_b^{a+1}$.

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

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

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

We also require the following distance measure on clusterings as proposed by Meila [33, 34]. 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 $\frac{k-1}{k}$.

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 $\mathcal{C}^a$. These clusterings satisfy the properties we stated at the beginning of this section, that is:

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

A formal verification of these properties can be found in the appendix.

Benchmark Evaluation

We now describe how we use the benchmark to measure the distortion of a coreset. Assume that the coresets are subsets of the original input points[3].

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

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 and let $\delta > 0$ be a parameter. 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 $w(C_i^a \cap \Omega) \geq |C_i|(1-\delta)$, we place a center at $\mu(C_i^a)$. Conversely, if $w(C_i^a \cap \Omega) < |C_i^a|(1-\delta)$, we do not place a center at $\mu(C_i^a)$. We call such clusters *deficient*. Let $\mathcal{S}$ be the centers of the these deficient clusters.

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, i.e. all values of δ, 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.

4 Experiments

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

Evaluation Procedure

Accurately evaluating a k-means coreset of a real-world data set requires constructing a solution (a set of k centers) which results in a maximal distortion. Finding such a solution, however, is difficult. Instead, we can estimate the quality of a given coreset by finding meaningful candidate solutions.

A first attempt can be to randomly generate candidate solutions. It is not readily apparent how to define a distribution of meaningful solutions from which to sample. One could, for instance, generate k random points inside the convex hull or the minimum enclosing ball (MEB) of a coreset Ω. Convex hulls in high dimensions are infeasible to compute, so we sample a center by choosing random convex combination of the centers of the initial bicriteria approximation computed for every coreset. A better way to generate candidate solutions turns out to be use k-means++. The main advantage of this approach is that k-means++ can uncover natural cluster structures in the data, which uniform sampling generally does not. For all variants we generated 5 candidate solution, with the candidate solution with the largest distortion being a lower bound for the true distortion of the coreset.

Given the usefulness of evaluating coresets on real-world data sets, it can be tricky to gauge the general performance of coreset algorithms using only a small selection of data sets. For this reason, we used our benchmark to complement the evaluation 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

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

	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

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

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

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. 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 maximum distortions for every run by taking the average over all 10 evaluations. It is important to not aggregate the distortions here by taking that maximum over all runs: If one run of the coreset algorithm fails but the others succeed, then such an aggregation predicts in a far worse distortion than what we could typically expect.

Data sets

We conducted experiments on five real-world data sets *Census*[4], *Covertype*[5], *Tower*[6], Caltech101[7], *NYTimes*[8] and four instances of our benchmark. Benchmark instances were generated to match approximately the sizes of the real-world data sets. The sizes of the data sets are summarized in Table 1. The size of the considered data sets are given in Table 1. We provide a brief description of the data sets in the appendix.

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* and *NYTimes* by using the k singular vectors corresponding to the largest singular values. 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, 18]. For the target coreset size T, we experimented with $T = mk$ for $m = \{50, 100, 200, 500\}$. On *Caltech*, *Census*, *Covertype* and *NYTimes*, we used values k in $\{10, 20, 30, 40, 50\}$, while for *Tower* we used larger cluster sizes $k \in \{20, 40, 60, 80, 100\}$. On the benchmark, we generated four instances ($k \in \{10, 20, 30, 40\}$) to match roughly the sizes of the real-world data sets.

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'

[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`
[9] Link to repository will be provided later.

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.

Outcome of Experiments

We observed that in the majority of our experiments, varying the coreset sizes does not significantly change the performance profiles of individual algorithms when comparing them against each other. Therefore, in the following sections, we focus on a cross-section of the experiments where $m = 200$ i.e., $T = 200k$. For numerical results including variances of all the experiments and tables containing distortion, cost and running times, we refer to Appendix E in the appendix.

In Figure 1, we summarized the distortions of the experiments with coreset sizes $T = 200k$. 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 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.

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). For all data sets, the obtained costs differed insignificantly for all values of k (see Appendix F and Figure 6 in the appendix), irrespective of the coreset algorithm used, while distortions varied strongly, depending on the coreset algorithm.

Nevertheless, the cost drop with increasing values of k is a predictor for the quality of certain coresets. 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 an 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 5 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

[10] `https://ls2-www.cs.tu-dortmund.de/grav/en/bico`

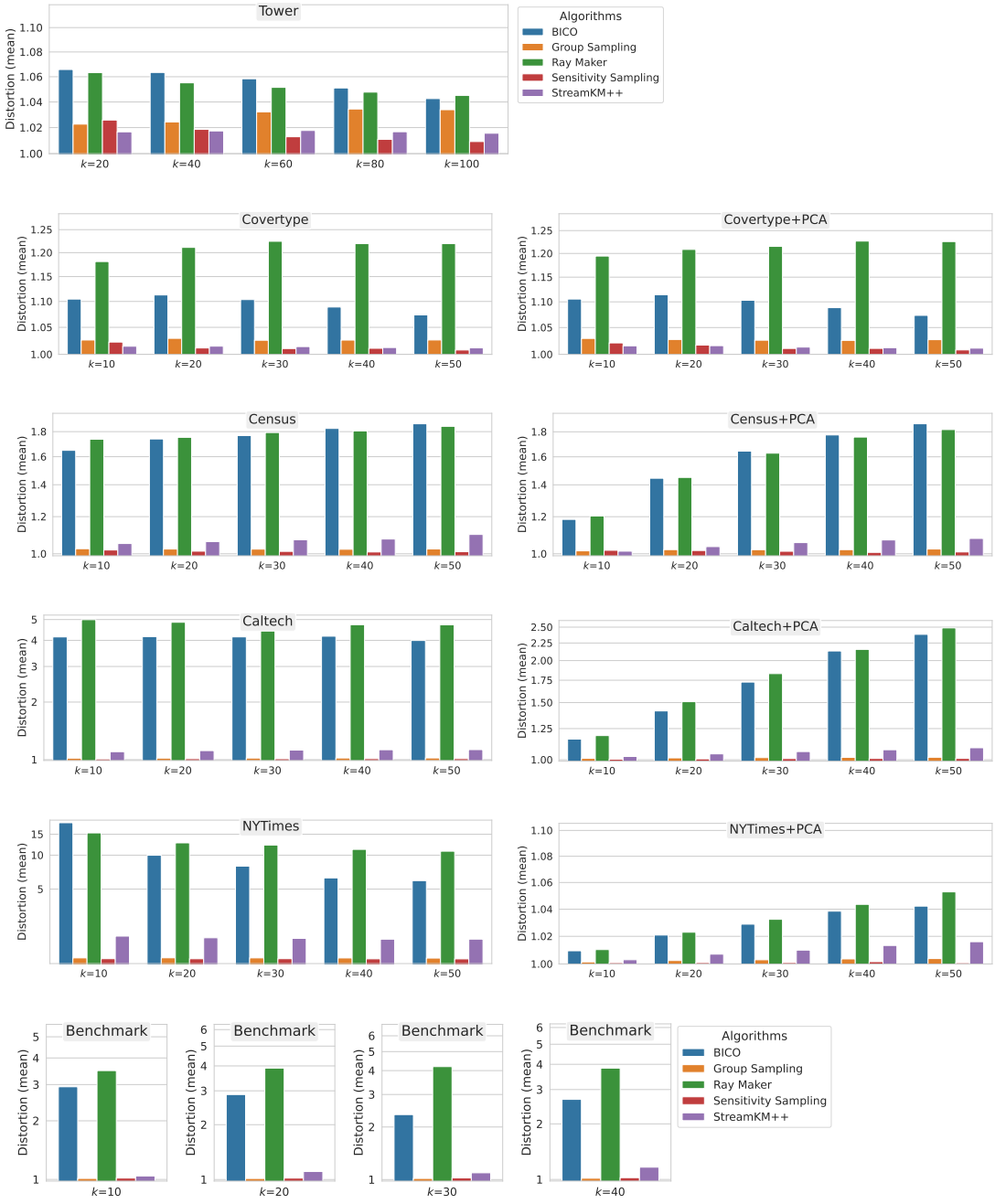

Figure 1 The distortions of the evaluated coreset algorithms with coreset size $T = 200k$ on five real-world data sets and on four benchmark instances. Notice that the axis is non-linear as otherwise the bars for Sensitivity Sampling and Group Sampling would disappear on the plots as their distortions are close to 1.

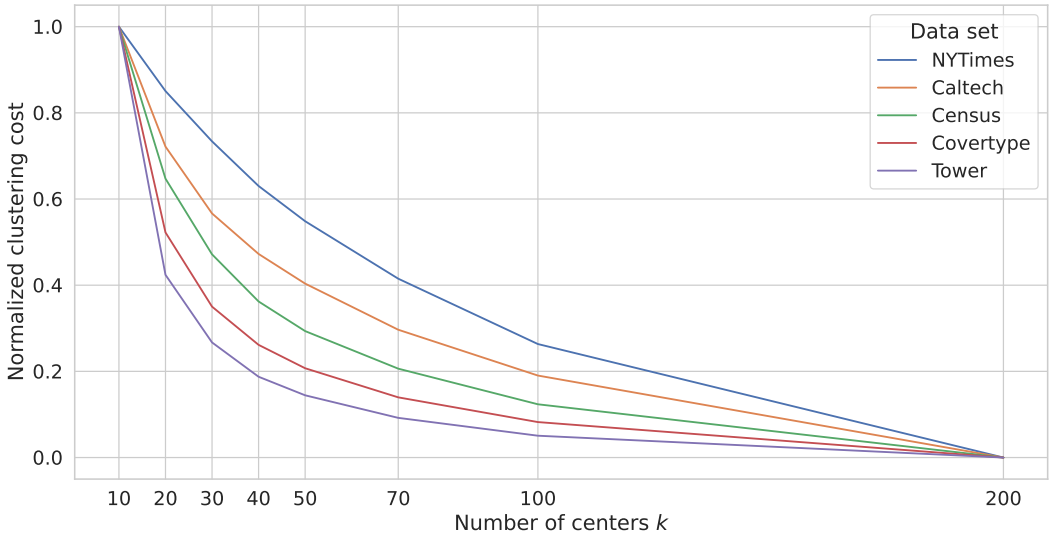

Figure 2 Depicts how clustering costs of five real-world data sets decrease as the number of centers increase. Plotting the cost curve allows us to study whether we can observe a difference between coreset construction and optimization in a data set when evaluating a coreset based on cost.

for BICO and Ray Maker than for StreamKM++. Importance sampling 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.

Movement-based versus Sampling-based Approaches: 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 well with Group Sampling generally being competitive. This runs contrary to theory where Group Sampling has the better (currently known) theoretical bounds. 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.

In practice as well as in theory, the distortion of movement-based algorithms are affected by the dimension. By comparison, sampling-based algorithms are affected very little. Theoretically, there should not exist a difference, as the sampling bounds are independent of the dimension. What little effect can be observed is likely due to PCA making it easier to find low cost solutions that form the backbone of all coreset constructions. StreamKM++ is an interesting case, as it is still affected by the dimension, though less than the other movement based methods.

At notable exception is the benchmark. Here, sensitivity sampling generally found the lowest cost clustering, with BICO finding the second lowest cost clustering. This happens *despite* BICO generally having a worse distortion than for example Group Sampling or StreamKM++.

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 4). 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 meager 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 a 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. Despite movement based methods running on very efficient code, it is necessary to complement 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 focuses 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 practice by using a different weighting scheme. We leave this as an open problem for future research.

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, Kasper Green Larsen, David Saulpic, and Chris Schwiegelshohn. Towards optimal lower bounds for k-median and k-means coresets. *CoRR*, abs/2202.12793, 2022. URL: https://arxiv.org/abs/2202.12793, arXiv:2202.12793.
10. 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.
11. 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.
12. 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.
13. Michael Elkin, Arnold Filtser, and Ofer Neiman. Terminal embeddings. *Theor. Comput. Sci.*, 697:1–36, 2017. doi:10.1016/j.tcs.2017.06.021.
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 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.

16 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.

17 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.

18 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.

19 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.

20 Panos Giannopoulos, Christian Knauer, Magnus Wahlström, and Daniel Werner. Hardness of discrepancy computation and ϵ-net verification in high dimension. *J. Complex.*, 28(2):162–176, 2012. doi:10.1016/j.jco.2011.09.001.

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

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

23 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.

24 Lingxiao Huang, Shaofeng H.-C. Jiang, and Nisheeth K. Vishnoi. Coresets for clustering with fairness constraints. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 7587–7598, 2019. URL: https://proceedings.neurips.cc/paper/2019/hash/810dfbbebb17302018ae903e9cb7a483-Abstract.html.

25 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.

26 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.

27 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.

28 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.

29 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.

30 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.

31 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.

32 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.

33 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.

34 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.

35 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.

36 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.

37 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.

38 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.

39 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.

40 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.

41 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.

571 **42** Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: A new data clustering algorithm
572 and its applications. *Data Min. Knowl. Discov.*, 1(2):141–182, 1997. URL: `http://dx.doi.`
573 `org/10.1023/A:1009783824328`, `doi:10.1023/A:1009783824328`.

A Properties of the Benchmark

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

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

▶ **Fact 2.** *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 3.** *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 4.** *For a clustering $\mathcal{C}^a$, let $\mu(C_j^a)$ denote the mean of cluster C_j^a. Then every point 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. ◀

B Hardness of Weak Coreset Evaluation

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

▶ **Proposition 5.** *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 weak 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, $\varepsilon_0 \approx 1.17$ [11, 10] for k-means in Euclidean spaces.

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.

C Details on the Real World Data Sets

The *Census*[11] 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[12] 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].

[11] https://archive.ics.uci.edu/ml/datasets/US+Census+Data+(1990)
[12] https://archive.ics.uci.edu/ml/datasets/covertype

The data set with the fewest number of dimensions is *Tower*[13]. 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 [18], *Caltech* was created by computing SIFT features from the images in the Caltech101[14] 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[15] 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.

D A Brief Introduction to Dimension Reduction for Coresets

There are essentially two main dimension reduction techniques for coresets.

Principal Component Analysis: Feldman, Schmidt, and Sohler [16] 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 [38]. 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 $f : \mathbb{R}^D \to \mathbb{R}^d$ 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 [13, 31, 36], with Narayanan and Nelson [36] 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, 12, 25].

For an overview on practical aspects of dimension reduction, we refer to Venkatsubramanian and Wang [40]. 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. Thus terminal embeddings are mainly used in the analysis of an algorithm, rather than as an explicit preprocessing step. 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 $\log k$.

E Distortions

The tables (Table 2, Table 3, Table 4, Table 5, Table 6, Table 7, Table 8, Table 9, Table 10, Table 11) show the distortions of the 5 evaluated algorithms on the different data sets. We vary the coreset size T for different k values using the formula: $T = mk$ where $m =$

[13] http://homepages.uni-paderborn.de/frahling/coremeans.html
[14] http://www.vision.caltech.edu/Image_Datasets/Caltech101/
[15] https://archive.ics.uci.edu/ml/datasets/bag+of+words

676 $\{50, 100, 200, 500\}$. The running time for StreamKM++ with coreset size $T = 500k$ exceeds
677 the allocated time budget of 12 hours on almost all data sets. For this reason, the distortions
678 for StreamKM++ with $m = 500$ not present in the tables.

Distortions on the *Benchmark* data set

k	m	BICO	Group Sampling	Ray Maker	Sensitivity Sampling	StreamKM++
10	50	3.40 (0.440)	1.02 (0.010)	5.05 (0.157)	1.02 (0.005)	1.07 (0.005)
	100	3.24 (0.729)	1.01 (0.004)	3.84 (0.081)	1.01 (0.003)	1.05 (0.004)
	200	2.90 (0.153)	1.01 (0.002)	3.48 (0.052)	1.01 (0.002)	1.04 (0.002)
	500	2.62 (0.095)	1.01 (0.001)	3.40 (0.058)	1.00 (0.001)	
20	50	3.22 (0.160)	1.04 (0.004)	5.52 (0.266)	1.02 (0.003)	1.08 (0.006)
	100	3.09 (0.122)	1.02 (0.004)	4.31 (0.130)	1.01 (0.002)	1.08 (0.003)
	200	2.88 (0.078)	1.01 (0.002)	3.93 (0.120)	1.01 (0.001)	1.11 (0.002)
	500	2.36 (0.051)	1.01 (0.001)	3.81 (0.116)	1.01 (0.001)	
30	50	2.81 (0.244)	1.03 (0.006)	6.43 (0.470)	1.02 (0.002)	1.13 (0.004)
	100	2.58 (0.105)	1.02 (0.003)	4.65 (0.256)	1.02 (0.002)	1.13 (0.004)
	200	2.38 (0.098)	1.01 (0.002)	4.12 (0.234)	1.01 (0.002)	1.10 (0.002)
	500	2.01 (0.065)	1.01 (0.001)	4.00 (0.138)	1.01 (0.001)	
40	50	3.16 (0.500)	1.03 (0.004)	5.62 (0.287)	1.02 (0.004)	1.10 (0.004)
	100	2.78 (0.258)	1.02 (0.001)	4.42 (0.200)	1.02 (0.002)	1.14 (0.003)
	200	2.79 (0.217)	1.02 (0.001)	3.82 (0.111)	1.01 (0.001)	1.17 (0.004)
	500	2.57 (0.113)	1.01 (0.001)	3.86 (0.122)	1.01 (0.001)	

Table 2 Distortions of the evaluated algorithms on the *Benchmark* data set. Each cell specify the mean distortion along with the standard deviation in parenthesis of 10 repetitions of the experiment.

Distortions on the *Caltech* data set

k	m	BICO	Group Sampling	Ray Maker	Sensitivity Sampling	StreamKM++
10	50	5.12 (0.307)	1.07 (0.009)	7.05 (0.250)	1.05 (0.013)	1.16 (0.008)
	100	4.48 (0.284)	1.04 (0.006)	5.46 (0.254)	1.03 (0.008)	1.13 (0.005)
	200	4.08 (0.319)	1.02 (0.004)	5.00 (0.196)	1.01 (0.005)	1.10 (0.005)
	500	3.41 (0.215)	1.01 (0.003)	4.75 (0.188)	1.00 (0.003)	
20	50	6.35 (1.173)	1.06 (0.008)	6.92 (0.209)	1.05 (0.007)	1.18 (0.008)
	100	4.65 (0.283)	1.04 (0.005)	5.29 (0.146)	1.03 (0.005)	1.14 (0.005)
	200	4.19 (0.384)	1.02 (0.002)	4.80 (0.112)	1.01 (0.004)	1.12 (0.004)
	500	3.50 (0.404)	1.01 (0.002)	4.61 (0.096)	1.00 (0.001)	
30	50	6.01 (0.335)	1.06 (0.005)	6.79 (0.149)	1.04 (0.004)	1.19 (0.005)
	100	5.10 (0.628)	1.04 (0.005)	5.30 (0.123)	1.02 (0.005)	1.15 (0.002)
	200	4.29 (0.659)	1.02 (0.003)	4.70 (0.075)	1.01 (0.003)	1.13 (0.002)
	500	3.09 (0.138)	1.01 (0.002)	4.62 (0.090)	1.00 (0.002)	
40	50	6.24 (0.524)	1.07 (0.004)	6.85 (0.145)	1.05 (0.004)	1.20 (0.004)
	100	5.23 (0.874)	1.04 (0.003)	5.22 (0.092)	1.02 (0.002)	1.16 (0.003)
	200	4.50 (1.085)	1.02 (0.002)	4.72 (0.086)	1.01 (0.001)	1.13 (0.002)

	500	3.38 (0.398)	1.01 (0.001)	4.55 (0.094)	1.00 (0.001)	
50	50	7.50 (1.013)	1.07 (0.004)	6.80 (0.092)	1.05 (0.004)	1.20 (0.006)
	100	5.21 (0.968)	1.04 (0.003)	5.18 (0.084)	1.03 (0.002)	1.16 (0.005)
	200	4.21 (0.296)	1.02 (0.002)	4.69 (0.065)	1.01 (0.003)	1.13 (0.002)
	500	3.36 (0.429)	1.01 (0.001)	4.52 (0.058)	1.00 (0.001)	

Table 3 Distortions of the evaluated algorithms on the *Caltech* data set. Each cell specify the mean distortion along with the standard deviation in parenthesis of 10 repetitions of the experiment.

Distortions on the *Caltech* data set (with PCA preprocessing)

k	m	BICO	Group Sampling	Ray Maker	Sensitivity Sampling	StreamKM++
10	50	1.21 (0.009)	1.03 (0.004)	1.25 (0.005)	1.02 (0.007)	1.04 (0.005)
	100	1.19 (0.006)	1.02 (0.006)	1.21 (0.006)	1.01 (0.004)	1.03 (0.004)
	200	1.16 (0.003)	1.01 (0.002)	1.19 (0.003)	1.01 (0.003)	1.02 (0.002)
	500	1.13 (0.007)	1.00 (0.002)	1.19 (0.006)	1.00 (0.001)	
20	50	1.54 (0.024)	1.04 (0.006)	1.63 (0.008)	1.03 (0.006)	1.07 (0.003)
	100	1.48 (0.020)	1.02 (0.002)	1.54 (0.010)	1.01 (0.004)	1.06 (0.003)
	200	1.41 (0.006)	1.01 (0.002)	1.50 (0.011)	1.01 (0.003)	1.04 (0.003)
	500	1.35 (0.008)	1.00 (0.001)	1.49 (0.008)	1.00 (0.001)	
30	50	1.99 (0.032)	1.05 (0.006)	2.05 (0.020)	1.03 (0.003)	1.10 (0.004)
	100	1.82 (0.099)	1.03 (0.004)	1.90 (0.014)	1.02 (0.003)	1.08 (0.002)
	200	1.72 (0.046)	1.01 (0.002)	1.84 (0.009)	1.01 (0.002)	1.06 (0.002)
	500	1.62 (0.037)	1.01 (0.001)	1.81 (0.011)	1.00 (0.002)	
40	50	2.34 (0.112)	1.05 (0.003)	2.48 (0.020)	1.04 (0.005)	1.12 (0.004)
	100	2.25 (0.066)	1.03 (0.003)	2.23 (0.011)	1.02 (0.001)	1.09 (0.003)
	200	2.11 (0.072)	1.02 (0.002)	2.15 (0.012)	1.01 (0.003)	1.07 (0.002)
	500	1.90 (0.070)	1.01 (0.002)	2.12 (0.011)	1.00 (0.001)	
50	50	2.86 (0.214)	1.06 (0.003)	2.94 (0.026)	1.04 (0.005)	1.14 (0.005)
	100	2.58 (0.124)	1.03 (0.001)	2.59 (0.019)	1.02 (0.001)	1.11 (0.002)
	200	2.35 (0.139)	1.02 (0.001)	2.49 (0.021)	1.01 (0.002)	1.09 (0.002)
	500	2.15 (0.075)	1.01 (0.001)	2.44 (0.015)	1.00 (0.001)	

Table 4 Distortions of the evaluated algorithms on the *Caltech* data set (with PCA preprocessing). Each cell specify the mean distortion along with the standard deviation in parenthesis of 10 repetitions of the experiment.

Distortions on the *Census* data set

k	m	BICO	Group Sampling	Ray Maker	Sensitivity Sampling	StreamKM++
10	50	2.08 (0.106)	1.08 (0.019)	2.30 (0.162)	1.03 (0.031)	1.10 (0.012)
	100	1.83 (0.124)	1.04 (0.021)	1.92 (0.094)	1.03 (0.012)	1.07 (0.006)
	200	1.65 (0.056)	1.02 (0.005)	1.72 (0.050)	1.02 (0.012)	1.05 (0.005)
	500	1.48 (0.053)	1.02 (0.004)	1.69 (0.029)	1.02 (0.010)	
20	50	2.18 (0.164)	1.07 (0.022)	2.31 (0.078)	1.03 (0.007)	1.10 (0.008)

	100	1.93 (0.080)	1.04 (0.011)	1.91 (0.048)	1.02 (0.011)	1.08 (0.005)
	200	1.71 (0.031)	1.03 (0.006)	1.75 (0.027)	1.01 (0.006)	1.06 (0.004)
	500	1.51 (0.048)	1.02 (0.005)	1.66 (0.019)	1.01 (0.004)	
30	50	2.28 (0.177)	1.07 (0.011)	2.38 (0.077)	1.03 (0.011)	1.11 (0.007)
	100	2.05 (0.096)	1.04 (0.006)	1.95 (0.040)	1.02 (0.007)	1.09 (0.004)
	200	1.76 (0.055)	1.03 (0.005)	1.79 (0.034)	1.01 (0.007)	1.07 (0.003)
	500	1.56 (0.016)	1.01 (0.003)	1.71 (0.038)	1.01 (0.003)	
40	50	2.47 (0.101)	1.08 (0.009)	2.43 (0.069)	1.02 (0.010)	1.12 (0.006)
	100	2.11 (0.104)	1.04 (0.006)	1.99 (0.034)	1.01 (0.004)	1.09 (0.003)
	200	1.83 (0.038)	1.02 (0.007)	1.81 (0.026)	1.01 (0.005)	1.08 (0.002)
	500	1.56 (0.039)	1.01 (0.005)	1.73 (0.026)	1.01 (0.003)	
50	50	2.50 (0.138)	1.07 (0.008)	2.48 (0.054)	1.02 (0.011)	1.13 (0.004)
	100	2.12 (0.109)	1.04 (0.007)	2.02 (0.055)	1.01 (0.004)	1.10 (0.003)
	200	1.84 (0.059)	1.03 (0.004)	1.82 (0.022)	1.01 (0.005)	1.08 (0.004)
	500	1.57 (0.019)	1.02 (0.004)	1.75 (0.022)	1.01 (0.003)	

Table 5 Distortions of the evaluated algorithms on the *Census* data set. Each cell specify the mean distortion along with the standard deviation in parenthesis of 10 repetitions of the experiment.

Distortions on the *Census* data set (with PCA preprocessing)

k	m	BICO	Group Sampling	Ray Maker	Sensitivity Sampling	StreamKM++
10	50	1.34 (0.048)	1.06 (0.018)	1.35 (0.030)	1.04 (0.021)	1.03 (0.007)
	100	1.24 (0.015)	1.03 (0.007)	1.25 (0.017)	1.03 (0.016)	1.02 (0.005)
	200	1.19 (0.020)	1.02 (0.009)	1.19 (0.013)	1.02 (0.018)	1.01 (0.002)
	500	1.11 (0.003)	1.01 (0.005)	1.18 (0.017)	1.01 (0.003)	
20	50	1.84 (0.118)	1.07 (0.018)	1.78 (0.052)	1.02 (0.012)	1.07 (0.005)
	100	1.57 (0.073)	1.03 (0.010)	1.56 (0.020)	1.02 (0.008)	1.05 (0.005)
	200	1.44 (0.037)	1.02 (0.007)	1.44 (0.030)	1.01 (0.008)	1.03 (0.002)
	500	1.30 (0.014)	1.01 (0.005)	1.39 (0.017)	1.01 (0.005)	
30	50	2.13 (0.153)	1.07 (0.010)	2.10 (0.045)	1.02 (0.011)	1.10 (0.005)
	100	1.81 (0.050)	1.04 (0.009)	1.79 (0.029)	1.01 (0.006)	1.07 (0.003)
	200	1.63 (0.044)	1.02 (0.006)	1.65 (0.029)	1.01 (0.005)	1.06 (0.002)
	500	1.47 (0.021)	1.01 (0.004)	1.58 (0.014)	1.01 (0.004)	
40	50	2.37 (0.102)	1.07 (0.007)	2.31 (0.039)	1.03 (0.008)	1.11 (0.006)
	100	2.01 (0.118)	1.04 (0.007)	1.90 (0.027)	1.01 (0.004)	1.09 (0.004)
	200	1.76 (0.027)	1.03 (0.005)	1.75 (0.024)	1.01 (0.002)	1.07 (0.002)
	500	1.52 (0.044)	1.02 (0.003)	1.67 (0.022)	1.01 (0.003)	
50	50	2.45 (0.139)	1.07 (0.008)	2.46 (0.048)	1.02 (0.012)	1.13 (0.004)
	100	2.14 (0.135)	1.04 (0.007)	1.99 (0.028)	1.01 (0.004)	1.10 (0.003)
	200	1.87 (0.055)	1.03 (0.005)	1.82 (0.018)	1.01 (0.004)	1.08 (0.002)
	500	1.56 (0.008)	1.02 (0.004)	1.73 (0.036)	1.01 (0.004)	

Table 6 Distortions of the evaluated algorithms on the *Census* data set (with PCA preprocessing). Each cell specify the mean distortion along with the standard deviation in parenthesis of 10 repetitions of the experiment.

Distortions on the *Covertype* data set

k	m	BICO	Group Sampling	Ray Maker	Sensitivity Sampling	StreamKM++
10	50	1.25 (0.018)	1.08 (0.022)	1.31 (0.027)	1.04 (0.025)	1.05 (0.008)
	100	1.17 (0.015)	1.04 (0.016)	1.22 (0.017)	1.03 (0.014)	1.02 (0.003)
	200	1.11 (0.004)	1.03 (0.012)	1.18 (0.014)	1.02 (0.008)	1.02 (0.002)
	500	1.05 (0.001)	1.01 (0.004)	1.16 (0.009)	1.01 (0.006)	
20	50	1.27 (0.025)	1.09 (0.022)	1.35 (0.031)	1.04 (0.008)	1.05 (0.006)
	100	1.18 (0.004)	1.05 (0.016)	1.24 (0.010)	1.02 (0.006)	1.03 (0.003)
	200	1.11 (0.003)	1.02 (0.008)	1.21 (0.017)	1.01 (0.007)	1.01 (0.002)
	500	1.04 (0.001)	1.01 (0.004)	1.19 (0.009)	1.01 (0.006)	
30	50	1.29 (0.008)	1.08 (0.017)	1.37 (0.013)	1.04 (0.013)	1.05 (0.005)
	100	1.17 (0.003)	1.05 (0.007)	1.26 (0.013)	1.01 (0.007)	1.03 (0.003)
	200	1.10 (0.001)	1.03 (0.009)	1.21 (0.008)	1.01 (0.005)	1.01 (0.001)
	500	1.04 (0.000)	1.01 (0.003)	1.20 (0.006)	1.01 (0.003)	
40	50	1.28 (0.009)	1.09 (0.014)	1.39 (0.012)	1.03 (0.013)	1.05 (0.004)
	100	1.18 (0.005)	1.05 (0.011)	1.26 (0.009)	1.02 (0.007)	1.03 (0.002)
	200	1.09 (0.002)	1.02 (0.003)	1.22 (0.008)	1.01 (0.005)	1.01 (0.001)
	500	1.03 (0.000)	1.01 (0.002)	1.21 (0.008)	1.01 (0.003)	
50	50	1.28 (0.009)	1.08 (0.007)	1.38 (0.008)	1.03 (0.010)	1.05 (0.004)
	100	1.15 (0.003)	1.05 (0.005)	1.27 (0.011)	1.02 (0.005)	1.02 (0.001)
	200	1.07 (0.001)	1.02 (0.004)	1.23 (0.009)	1.01 (0.002)	1.01 (0.001)
	500	1.03 (0.000)	1.01 (0.004)	1.23 (0.009)	1.01 (0.004)	

Table 7 Distortions of the evaluated algorithms on the *Covertype* data set. Each cell specify the mean distortion along with the standard deviation in parenthesis of 10 repetitions of the experiment.

Distortions on the *Covertype* data set (with PCA preprocessing)

k	m	BICO	Group Sampling	Ray Maker	Sensitivity Sampling	StreamKM++
10	50	1.26 (0.024)	1.09 (0.025)	1.30 (0.019)	1.04 (0.017)	1.04 (0.008)
	100	1.16 (0.010)	1.05 (0.017)	1.22 (0.015)	1.03 (0.017)	1.02 (0.003)
	200	1.10 (0.003)	1.03 (0.017)	1.17 (0.014)	1.02 (0.011)	1.01 (0.002)
	500	1.06 (0.001)	1.01 (0.006)	1.16 (0.008)	1.01 (0.007)	
20	50	1.27 (0.018)	1.10 (0.021)	1.34 (0.021)	1.04 (0.018)	1.05 (0.004)
	100	1.17 (0.005)	1.05 (0.011)	1.25 (0.013)	1.02 (0.010)	1.03 (0.002)
	200	1.11 (0.003)	1.02 (0.007)	1.20 (0.012)	1.02 (0.005)	1.01 (0.002)
	500	1.04 (0.001)	1.01 (0.004)	1.19 (0.011)	1.01 (0.008)	
30	50	1.29 (0.010)	1.09 (0.013)	1.38 (0.013)	1.04 (0.011)	1.05 (0.003)
	100	1.17 (0.004)	1.05 (0.011)	1.26 (0.009)	1.01 (0.007)	1.03 (0.003)
	200	1.10 (0.002)	1.03 (0.005)	1.22 (0.011)	1.01 (0.007)	1.01 (0.002)
	500	1.04 (0.000)	1.01 (0.003)	1.21 (0.010)	1.01 (0.005)	
40	50	1.28 (0.006)	1.09 (0.013)	1.38 (0.019)	1.04 (0.014)	1.05 (0.005)
	100	1.18 (0.003)	1.05 (0.008)	1.27 (0.016)	1.02 (0.008)	1.02 (0.002)
	200	1.09 (0.001)	1.02 (0.004)	1.22 (0.010)	1.01 (0.004)	1.01 (0.002)

	500	1.03 (0.000)	1.01 (0.003)	1.21 (0.009)	1.01 (0.003)	
50	50	1.29 (0.017)	1.09 (0.014)	1.39 (0.009)	1.03 (0.010)	1.05 (0.003)
	100	1.15 (0.003)	1.05 (0.007)	1.27 (0.009)	1.02 (0.008)	1.02 (0.001)
	200	1.07 (0.001)	1.02 (0.003)	1.22 (0.005)	1.01 (0.002)	1.01 (0.001)
	500	1.03 (0.000)	1.01 (0.002)	1.23 (0.006)	1.01 (0.003)	

Table 8 Distortions of the evaluated algorithms on the *Covertype* data set (with PCA preprocessing). Each cell specify the mean distortion along with the standard deviation in parenthesis of 10 repetitions of the experiment.

Distortions on the *NYTimes* data set

k	m	BICO	Group Sampling	Ray Maker	Sensitivity Sampling	StreamKM++
10	50	35.32 (8.376)	1.08 (0.009)	28.37 (2.761)	1.05 (0.012)	2.04 (0.244)
	100	24.50 (5.355)	1.06 (0.008)	17.64 (0.561)	1.03 (0.007)	1.80 (0.069)
	200	13.97 (2.143)	1.04 (0.004)	15.17 (0.512)	1.02 (0.008)	1.75 (0.110)
	500	8.09 (1.241)	1.03 (0.004)	14.16 (1.031)	1.01 (0.004)	
20	50	21.70 (3.965)	1.09 (0.008)	23.94 (1.466)	1.05 (0.010)	1.97 (0.083)
	100	16.00 (3.906)	1.05 (0.005)	14.56 (0.846)	1.03 (0.006)	1.79 (0.046)
	200	8.64 (1.429)	1.04 (0.003)	12.71 (0.786)	1.02 (0.003)	1.68 (0.025)
	500	5.39 (0.441)	1.02 (0.001)	11.73 (0.551)	1.01 (0.003)	
30	50	21.63 (3.611)	1.09 (0.007)	20.44 (0.862)	1.05 (0.008)	1.97 (0.106)
	100	13.36 (3.778)	1.05 (0.005)	13.00 (0.763)	1.03 (0.005)	1.74 (0.035)
	200	7.76 (0.884)	1.03 (0.003)	11.68 (0.754)	1.01 (0.004)	1.66 (0.028)
	500	4.70 (0.418)	1.02 (0.001)	11.17 (0.682)	1.01 (0.002)	
40	50	22.03 (7.691)	1.08 (0.006)	18.56 (0.955)	1.05 (0.006)	1.92 (0.045)
	100	10.49 (1.801)	1.05 (0.006)	12.27 (0.688)	1.03 (0.006)	1.75 (0.057)
	200	6.39 (0.339)	1.03 (0.002)	11.08 (0.461)	1.01 (0.005)	1.63 (0.024)
	500	4.19 (0.151)	1.02 (0.001)	10.68 (0.736)	1.01 (0.002)	
50	50	15.78 (3.123)	1.09 (0.005)	17.61 (0.940)	1.05 (0.007)	1.89 (0.053)
	100	9.85 (2.053)	1.05 (0.004)	11.71 (0.292)	1.03 (0.003)	1.71 (0.036)
	200	6.03 (0.505)	1.03 (0.002)	10.76 (0.454)	1.01 (0.004)	1.63 (0.020)
	500	4.07 (0.182)	1.02 (0.001)	10.58 (1.080)	1.00 (0.002)	

Table 9 Distortions of the evaluated algorithms on the *NYTimes* data set. Each cell specify the mean distortion along with the standard deviation in parenthesis of 10 repetitions of the experiment.

Distortions on the *NYTimes* data set (with PCA preprocessing)

k	m	BICO	Group Sampling	Ray Maker	Sensitivity Sampling	StreamKM++
10	50	1.02 (0.002)	1.01 (0.002)	1.02 (0.001)	1.00 (0.002)	1.01 (0.000)
	100	1.01 (0.001)	1.00 (0.001)	1.01 (0.001)	1.00 (0.002)	1.00 (0.000)
	200	1.01 (0.000)	1.00 (0.001)	1.01 (0.000)	1.00 (0.001)	1.00 (0.000)
	500	1.01 (0.000)	1.00 (0.001)	1.01 (0.001)	1.00 (0.001)	
20	50	1.03 (0.002)	1.01 (0.001)	1.03 (0.001)	1.00 (0.002)	1.01 (0.001)

	100	1.03 (0.001)	1.00 (0.001)	1.03 (0.001)	1.00 (0.000)	1.01 (0.000)
	200	1.02 (0.001)	1.00 (0.001)	1.02 (0.001)	1.00 (0.001)	1.01 (0.000)
	500	1.01 (0.001)	1.00 (0.000)	1.02 (0.001)	1.00 (0.000)	
30	50	1.04 (0.003)	1.01 (0.001)	1.04 (0.001)	1.00 (0.002)	1.02 (0.001)
	100	1.04 (0.002)	1.01 (0.001)	1.03 (0.001)	1.00 (0.001)	1.01 (0.000)
	200	1.03 (0.001)	1.00 (0.001)	1.03 (0.000)	1.00 (0.001)	1.01 (0.000)
	500	1.02 (0.000)	1.00 (0.000)	1.03 (0.001)	1.00 (0.001)	
40	50	1.06 (0.001)	1.01 (0.001)	1.05 (0.001)	1.00 (0.001)	1.02 (0.000)
	100	1.05 (0.002)	1.01 (0.001)	1.05 (0.001)	1.00 (0.001)	1.02 (0.000)
	200	1.04 (0.001)	1.00 (0.001)	1.04 (0.001)	1.00 (0.001)	1.01 (0.000)
	500	1.03 (0.000)	1.00 (0.000)	1.05 (0.001)	1.00 (0.000)	
50	50	1.07 (0.004)	1.01 (0.001)	1.06 (0.002)	1.00 (0.001)	1.03 (0.000)
	100	1.06 (0.001)	1.01 (0.001)	1.05 (0.001)	1.00 (0.001)	1.02 (0.000)
	200	1.04 (0.003)	1.00 (0.001)	1.05 (0.001)	1.00 (0.001)	1.02 (0.000)
	500	1.03 (0.001)	1.00 (0.000)	1.06 (0.001)	1.00 (0.000)	

Table 10 Distortions of the evaluated algorithms on the *NYTimes* data set (with PCA preprocessing). Each cell specify the mean distortion along with the standard deviation in parenthesis of 10 repetitions of the experiment.

Distortions on the *Tower* data set

k	m	BICO	Group Sampling	Ray Maker	Sensitivity Sampling	StreamKM++
20	50	1.18 (0.018)	1.09 (0.025)	1.44 (0.045)	1.04 (0.025)	1.04 (0.004)
	100	1.11 (0.005)	1.05 (0.016)	1.16 (0.013)	1.03 (0.007)	1.03 (0.003)
	200	1.06 (0.002)	1.03 (0.007)	1.06 (0.007)	1.02 (0.009)	1.02 (0.001)
	500	1.03 (0.001)	1.01 (0.006)	1.03 (0.002)	1.01 (0.005)	
40	50	1.20 (0.012)	1.11 (0.014)	1.48 (0.020)	1.04 (0.019)	1.05 (0.002)
	100	1.11 (0.006)	1.05 (0.013)	1.15 (0.008)	1.02 (0.010)	1.03 (0.002)
	200	1.07 (0.005)	1.03 (0.007)	1.05 (0.004)	1.01 (0.006)	1.02 (0.001)
	500	1.02 (0.001)	1.01 (0.003)	1.03 (0.001)	1.01 (0.003)	
60	50	1.20 (0.008)	1.11 (0.011)	1.49 (0.019)	1.04 (0.009)	1.05 (0.002)
	100	1.10 (0.002)	1.06 (0.007)	1.13 (0.005)	1.01 (0.006)	1.03 (0.002)
	200	1.06 (0.002)	1.03 (0.007)	1.05 (0.002)	1.01 (0.004)	1.02 (0.001)
	500	1.02 (0.001)	1.02 (0.005)	1.03 (0.001)	1.01 (0.003)	
80	50	1.18 (0.005)	1.11 (0.010)	1.47 (0.013)	1.03 (0.012)	1.05 (0.004)
	100	1.10 (0.009)	1.06 (0.008)	1.13 (0.004)	1.01 (0.005)	1.03 (0.001)
	200	1.05 (0.001)	1.03 (0.003)	1.05 (0.002)	1.01 (0.004)	1.02 (0.000)
	500	1.02 (0.001)	1.02 (0.003)	1.02 (0.001)	1.01 (0.004)	
100	50	1.19 (0.007)	1.10 (0.009)	1.45 (0.010)	1.03 (0.008)	1.05 (0.002)
	100	1.10 (0.008)	1.06 (0.007)	1.12 (0.005)	1.02 (0.007)	1.03 (0.001)
	200	1.04 (0.001)	1.03 (0.005)	1.05 (0.002)	1.01 (0.003)	1.02 (0.000)
	500	1.01 (0.001)	1.02 (0.003)	1.03 (0.002)	1.01 (0.003)	

Table 11 Distortions of the evaluated algorithms on the *Tower* data set. Each cell specify the mean distortion along with the standard deviation in parenthesis of 10 repetitions of the experiment.

F Costs

Costs on the *Benchmark* data set

k	m	BICO	Group Sampling	Ray Maker	Sensitivity Sampling	StreamKM++
10	50	4.8e+06 (2.3e+05)	5.2e+06 (4.1e+05)	5.0e+06 (2.9e+05)	4.7e+06 (3.8e+05)	5.3e+06 (0.0e+00)
	100	4.6e+06 (1.7e+05)	5.3e+06 (1.1e+05)	4.9e+06 (2.8e+05)	4.5e+06 (1.3e+05)	5.3e+06 (0.0e+00)
	200	4.7e+06 (3.1e+05)	5.4e+06 (2.4e+05)	5.1e+06 (3.6e+05)	4.6e+06 (2.5e+05)	5.3e+06 (0.0e+00)
	500	4.6e+06 (1.0e+05)	5.4e+06 (1.4e+05)	5.1e+06 (2.1e+05)	4.7e+06 (3.6e+05)	
20	50	1.4e+07 (8.8e+05)	1.5e+07 (2.6e+05)	1.3e+07 (6.2e+05)	1.2e+07 (2.9e-09)	1.6e+07 (2.9e+05)
	100	1.3e+07 (9.8e+05)	1.5e+07 (4.6e+05)	1.4e+07 (5.9e+05)	1.3e+07 (1.1e+06)	1.6e+07 (1.7e+05)
	200	1.2e+07 (2.2e+05)	1.5e+07 (4.0e+05)	1.3e+07 (3.5e+05)	1.2e+07 (6.1e+05)	1.6e+07 (1.0e+05)
	500	1.3e+07 (6.2e+05)	1.5e+07 (3.4e+05)	1.3e+07 (3.2e+05)	1.2e+07 (8.1e+05)	
30	50	2.7e+06 (2.6e+05)	3.0e+06 (4.0e+04)	2.5e+06 (6.3e+04)	2.3e+06 (0.0e+00)	3.2e+06 (3.1e+04)
	100	2.8e+06 (9.5e+04)	3.0e+06 (1.0e+05)	2.4e+06 (1.3e+05)	2.3e+06 (0.0e+00)	3.1e+06 (3.8e+04)
	200	2.6e+06 (2.1e+05)	3.1e+06 (7.6e+04)	2.6e+06 (8.6e+04)	2.3e+06 (0.0e+00)	3.1e+06 (5.0e+04)
	500	2.6e+06 (8.4e+04)	3.0e+06 (5.6e+04)	2.5e+06 (1.2e+05)	2.3e+06 (0.0e+00)	
40	50	9.2e+06 (6.6e+05)	9.8e+06 (1.4e+05)	9.1e+06 (2.6e+05)	7.5e+06 (0.0e+00)	1.0e+07 (1.5e+05)
	100	8.5e+06 (3.7e+05)	9.8e+06 (2.9e+05)	8.5e+06 (6.8e+05)	7.5e+06 (2.3e-09)	1.0e+07 (1.2e+05)
	200	8.7e+06 (4.8e+05)	9.7e+06 (1.4e+05)	9.2e+06 (3.1e+05)	7.5e+06 (3.1e-09)	1.1e+07 (5.4e+04)
	500	8.4e+06 (1.8e+05)	9.4e+06 (1.8e+05)	8.8e+06 (5.4e+05)	7.5e+06 (0.0e+00)	

Table 12 Costs of the evaluated algorithms on the *Benchmark* data set. Each cell specify the mean along with the standard deviation in parenthesis of 10 repetitions of the experiment.

Costs on the *Caltech* data set

k	m	BICO	Group Sampling	Ray Maker	Sensitivity Sampling	StreamKM++
10	50	4.1e+11 (3.2e+09)	4.1e+11 (2.9e+09)	4.0e+11 (1.7e+09)	4.0e+11 (2.5e+09)	4.1e+11 (2.7e+09)
	100	4.0e+11 (1.6e+09)	4.0e+11 (3.1e+09)	4.0e+11 (1.7e+09)	4.0e+11 (3.0e+09)	4.0e+11 (8.4e+08)
	200	4.0e+11 (3.9e+09)	4.0e+11 (1.5e+09)	4.0e+11 (2.2e+09)	4.0e+11 (1.5e+09)	3.9e+11 (1.1e+09)
	500	4.0e+11 (1.8e+09)	3.9e+11 (1.7e+09)	4.0e+11 (1.9e+09)	3.9e+11 (6.3e+08)	
20	50	3.8e+11 (3.1e+09)	3.7e+11 (1.3e+09)	3.7e+11 (2.8e+09)	3.7e+11 (2.4e+09)	3.7e+11 (1.4e+09)
	100	3.7e+11 (2.9e+09)	3.7e+11 (1.2e+09)	3.7e+11 (1.9e+09)	3.7e+11 (7.8e+08)	3.7e+11 (1.2e+09)
	200	3.7e+11 (2.0e+09)	3.6e+11 (1.1e+09)	3.7e+11 (1.4e+09)	3.6e+11 (6.8e+08)	3.6e+11 (5.4e+08)
	500	3.7e+11 (3.1e+09)	3.6e+11 (8.8e+08)	3.7e+11 (2.1e+09)	3.6e+11 (7.5e+08)	
30	50	3.6e+11 (2.4e+09)	3.6e+11 (1.3e+09)	3.6e+11 (1.7e+09)	3.6e+11 (1.1e+09)	3.5e+11 (1.0e+09)
	100	3.6e+11 (2.5e+09)	3.5e+11 (9.0e+08)	3.6e+11 (1.4e+09)	3.5e+11 (1.1e+09)	3.5e+11 (1.0e+09)
	200	3.6e+11 (3.4e+09)	3.5e+11 (8.0e+08)	3.6e+11 (9.6e+08)	3.5e+11 (4.5e+08)	3.5e+11 (4.6e+08)
	500	3.5e+11 (1.5e+09)	3.4e+11 (6.2e+08)	3.5e+11 (1.1e+09)	3.4e+11 (9.0e+08)	
40	50	3.5e+11 (2.3e+09)	3.4e+11 (8.2e+08)	3.4e+11 (5.4e+08)	3.4e+11 (1.0e+09)	3.4e+11 (7.9e+08)
	100	3.5e+11 (2.8e+09)	3.4e+11 (1.1e+09)	3.4e+11 (1.4e+09)	3.4e+11 (6.0e+08)	3.4e+11 (3.1e+08)
	200	3.4e+11 (3.9e+09)	3.3e+11 (5.2e+08)	3.4e+11 (7.2e+08)	3.4e+11 (4.2e+08)	3.4e+11 (3.6e+08)
	500	3.4e+11 (1.9e+09)	3.3e+11 (3.1e+08)	3.4e+11 (9.3e+08)	3.3e+11 (2.8e+08)	

50	50	3.4e+11 (1.4e+09)	3.4e+11 (9.2e+08)	3.4e+11 (8.1e+08)	3.4e+11 (1.0e+09)	3.4e+11 (9.9e+08)
	100	3.4e+11 (2.8e+09)	3.3e+11 (3.0e+08)	3.4e+11 (1.1e+09)	3.3e+11 (5.7e+08)	3.3e+11 (5.5e+08)
	200	3.4e+11 (1.5e+09)	3.3e+11 (4.7e+08)	3.4e+11 (7.7e+08)	3.3e+11 (3.3e+08)	3.3e+11 (2.6e+08)
	500	3.3e+11 (2.6e+09)	3.2e+11 (3.7e+08)	3.4e+11 (7.7e+08)	3.2e+11 (1.7e+08)	

Table 13 Costs of the evaluated algorithms on the *Caltech* data set. Each cell specify the mean along with the standard deviation in parenthesis of 10 repetitions of the experiment.

Costs on the *Caltech* data set (with PCA preprocessing)

k	m	BICO	Group Sampling	Ray Maker	Sensitivity Sampling	StreamKM++
10	50	4.1e+11 (1.3e+09)	4.1e+11 (1.1e+09)	4.1e+11 (1.9e+09)	4.1e+11 (1.9e+09)	4.0e+11 (1.5e+09)
	100	4.0e+11 (1.0e+09)	4.0e+11 (1.4e+09)	4.0e+11 (1.9e+09)	4.0e+11 (1.2e+09)	4.0e+11 (1.7e+09)
	200	4.0e+11 (2.1e+09)	4.0e+11 (1.3e+09)	4.0e+11 (1.8e+09)	4.0e+11 (1.2e+09)	4.0e+11 (8.4e+08)
	500	4.0e+11 (1.4e+09)	4.0e+11 (9.7e+08)	4.0e+11 (1.5e+09)	4.0e+11 (9.2e+08)	
20	50	3.7e+11 (1.3e+09)	3.7e+11 (1.3e+09)	3.7e+11 (1.1e+09)	3.7e+11 (1.2e+09)	3.7e+11 (1.6e+09)
	100	3.7e+11 (1.3e+09)	3.7e+11 (9.1e+08)	3.7e+11 (9.5e+08)	3.7e+11 (1.0e+09)	3.7e+11 (9.9e+08)
	200	3.7e+11 (1.1e+09)	3.7e+11 (5.1e+08)	3.7e+11 (1.6e+09)	3.7e+11 (7.8e+08)	3.6e+11 (3.4e+08)
	500	3.7e+11 (6.4e+08)	3.6e+11 (5.2e+08)	3.7e+11 (8.8e+08)	3.6e+11 (4.7e+08)	
30	50	3.6e+11 (1.2e+09)	3.5e+11 (1.3e+09)	3.6e+11 (1.1e+09)	3.5e+11 (9.0e+08)	3.5e+11 (5.5e+08)
	100	3.6e+11 (2.2e+09)	3.5e+11 (9.5e+08)	3.6e+11 (1.1e+09)	3.5e+11 (1.4e+09)	3.5e+11 (7.2e+08)
	200	3.5e+11 (1.7e+09)	3.5e+11 (4.9e+08)	3.6e+11 (6.5e+08)	3.5e+11 (8.1e+08)	3.5e+11 (4.0e+08)
	500	3.5e+11 (1.7e+09)	3.5e+11 (5.7e+08)	3.6e+11 (1.6e+09)	3.5e+11 (4.8e+08)	
40	50	3.5e+11 (1.7e+09)	3.4e+11 (1.1e+09)	3.4e+11 (1.0e+09)	3.4e+11 (8.0e+08)	3.4e+11 (5.3e+08)
	100	3.5e+11 (2.0e+09)	3.4e+11 (3.6e+08)	3.4e+11 (8.5e+08)	3.4e+11 (6.9e+08)	3.4e+11 (5.1e+08)
	200	3.4e+11 (1.5e+09)	3.4e+11 (4.7e+08)	3.4e+11 (1.2e+09)	3.4e+11 (6.2e+08)	3.4e+11 (3.9e+08)
	500	3.4e+11 (1.6e+09)	3.3e+11 (1.9e+08)	3.4e+11 (1.1e+09)	3.3e+11 (3.0e+08)	
50	50	3.4e+11 (1.2e+09)	3.4e+11 (6.1e+08)	3.4e+11 (5.7e+08)	3.3e+11 (1.1e+09)	3.3e+11 (1.0e+09)
	100	3.4e+11 (1.6e+09)	3.3e+11 (6.2e+08)	3.4e+11 (6.2e+08)	3.3e+11 (4.5e+08)	3.3e+11 (2.8e+08)
	200	3.3e+11 (2.1e+09)	3.3e+11 (5.1e+08)	3.4e+11 (8.1e+08)	3.3e+11 (3.3e+08)	3.3e+11 (2.9e+08)
	500	3.3e+11 (1.4e+09)	3.3e+11 (2.3e+08)	3.4e+11 (1.1e+09)	3.3e+11 (2.8e+08)	

Table 14 Costs of the evaluated algorithms on the *Caltech* data set (with PCA preprocessing). Each cell specify the mean along with the standard deviation in parenthesis of 10 repetitions of the experiment.

Costs on the *Census* data set

k	m	BICO	Group Sampling	Ray Maker	Sensitivity Sampling	StreamKM++
10	50	2.5e+08 (4.7e+06)	2.6e+08 (9.7e+06)	2.5e+08 (7.5e+06)	2.7e+08 (1.5e+07)	2.5e+08 (8.9e+06)
	100	2.5e+08 (6.1e+06)	2.6e+08 (1.2e+07)	2.5e+08 (5.1e+06)	2.6e+08 (8.3e+06)	2.6e+08 (7.9e+06)
	200	2.5e+08 (5.8e+06)	2.6e+08 (1.2e+07)	2.5e+08 (5.0e+06)	2.7e+08 (1.0e+07)	2.5e+08 (5.1e+06)
	500	2.5e+08 (4.3e+06)	2.6e+08 (1.5e+07)	2.4e+08 (2.6e+06)	2.6e+08 (1.2e+07)	
20	50	1.9e+08 (4.3e+06)	2.1e+08 (1.1e+07)	1.9e+08 (2.0e+06)	2.0e+08 (1.1e+07)	1.9e+08 (4.3e+06)
	100	1.9e+08 (4.9e+06)	2.0e+08 (7.2e+06)	1.9e+08 (2.1e+06)	2.0e+08 (9.7e+06)	1.9e+08 (4.0e+06)

	200	1.9e+08 (3.1e+06)	2.0e+08 (6.4e+06)	1.9e+08 (2.1e+06)	2.0e+08 (6.0e+06)	1.9e+08 (2.5e+06)
	500	1.9e+08 (3.4e+06)	1.9e+08 (6.2e+06)	1.9e+08 (2.2e+06)	2.0e+08 (8.3e+06)	
30	50	1.6e+08 (2.4e+06)	1.7e+08 (4.0e+06)	1.6e+08 (3.1e+06)	1.7e+08 (4.9e+06)	1.6e+08 (1.8e+06)
	100	1.6e+08 (3.2e+06)	1.7e+08 (3.4e+06)	1.6e+08 (2.4e+06)	1.7e+08 (4.9e+06)	1.6e+08 (2.9e+06)
	200	1.6e+08 (2.5e+06)	1.6e+08 (5.3e+06)	1.6e+08 (2.5e+06)	1.7e+08 (3.2e+06)	1.6e+08 (2.7e+06)
	500	1.6e+08 (1.5e+06)	1.6e+08 (5.8e+06)	1.6e+08 (2.4e+06)	1.7e+08 (4.5e+06)	
40	50	1.4e+08 (2.5e+06)	1.5e+08 (7.1e+06)	1.4e+08 (2.2e+06)	1.5e+08 (3.7e+06)	1.4e+08 (1.6e+06)
	100	1.4e+08 (1.7e+06)	1.5e+08 (3.4e+06)	1.4e+08 (1.6e+06)	1.5e+08 (2.2e+06)	1.4e+08 (2.0e+06)
	200	1.4e+08 (1.7e+06)	1.5e+08 (4.6e+06)	1.4e+08 (8.6e+05)	1.4e+08 (3.9e+06)	1.4e+08 (1.9e+06)
	500	1.4e+08 (1.6e+06)	1.4e+08 (2.4e+06)	1.4e+08 (1.8e+06)	1.4e+08 (3.2e+06)	
50	50	1.3e+08 (2.1e+06)	1.3e+08 (1.5e+06)	1.3e+08 (1.2e+06)	1.3e+08 (3.3e+06)	1.3e+08 (2.9e+06)
	100	1.3e+08 (1.5e+06)	1.3e+08 (2.2e+06)	1.3e+08 (2.2e+06)	1.4e+08 (3.4e+06)	1.3e+08 (2.3e+06)
	200	1.3e+08 (1.0e+06)	1.3e+08 (1.6e+06)	1.3e+08 (1.8e+06)	1.3e+08 (3.3e+06)	1.3e+08 (1.9e+06)
	500	1.3e+08 (2.0e+06)	1.3e+08 (1.6e+06)	1.3e+08 (1.7e+06)	1.3e+08 (2.4e+06)	

Table 15 Costs of the evaluated algorithms on the *Census* data set. Each cell specify the mean along with the standard deviation in parenthesis of 10 repetitions of the experiment.

Costs on the *Census* data set (with PCA preprocessing)

k	m	BICO	Group Sampling	Ray Maker	Sensitivity Sampling	StreamKM++
10	50	2.5e+08 (6.2e+06)	2.6e+08 (9.8e+06)	2.5e+08 (4.7e+06)	2.7e+08 (8.1e+06)	2.6e+08 (6.9e+06)
	100	2.5e+08 (5.8e+06)	2.7e+08 (8.8e+06)	2.5e+08 (6.8e+06)	2.7e+08 (1.5e+07)	2.6e+08 (1.1e+07)
	200	2.5e+08 (4.2e+06)	2.5e+08 (8.2e+06)	2.5e+08 (4.9e+06)	2.7e+08 (1.1e+07)	2.5e+08 (7.2e+06)
	500	2.5e+08 (5.9e+06)	2.6e+08 (1.1e+07)	2.5e+08 (7.2e+06)	2.6e+08 (1.2e+07)	
20	50	2.0e+08 (5.2e+06)	2.0e+08 (7.1e+06)	1.9e+08 (5.0e+06)	2.0e+08 (7.6e+06)	2.0e+08 (6.3e+06)
	100	1.9e+08 (1.9e+06)	2.0e+08 (5.3e+06)	1.9e+08 (4.1e+06)	2.1e+08 (1.4e+07)	1.9e+08 (3.4e+06)
	200	1.9e+08 (2.7e+06)	2.0e+08 (6.9e+06)	1.9e+08 (3.7e+06)	2.0e+08 (8.4e+06)	1.9e+08 (3.3e+06)
	500	1.9e+08 (2.8e+06)	2.0e+08 (4.9e+06)	1.9e+08 (4.1e+06)	2.0e+08 (6.2e+06)	
30	50	1.6e+08 (2.9e+06)	1.7e+08 (7.9e+06)	1.6e+08 (2.5e+06)	1.7e+08 (5.3e+06)	1.6e+08 (3.1e+06)
	100	1.6e+08 (2.2e+06)	1.6e+08 (4.2e+06)	1.6e+08 (2.1e+06)	1.7e+08 (2.2e+06)	1.6e+08 (2.9e+06)
	200	1.6e+08 (2.6e+06)	1.6e+08 (3.4e+06)	1.6e+08 (2.1e+06)	1.7e+08 (5.0e+06)	1.6e+08 (2.7e+06)
	500	1.6e+08 (1.6e+06)	1.6e+08 (3.4e+06)	1.6e+08 (2.9e+06)	1.7e+08 (6.8e+06)	
40	50	1.5e+08 (2.8e+06)	1.5e+08 (2.5e+06)	1.4e+08 (1.4e+06)	1.5e+08 (3.5e+06)	1.4e+08 (2.6e+06)
	100	1.4e+08 (2.6e+06)	1.4e+08 (2.8e+06)	1.4e+08 (1.9e+06)	1.5e+08 (6.9e+06)	1.4e+08 (2.4e+06)
	200	1.4e+08 (2.4e+06)	1.4e+08 (2.4e+06)	1.4e+08 (2.2e+06)	1.5e+08 (5.0e+06)	1.4e+08 (2.0e+06)
	500	1.4e+08 (1.9e+06)	1.4e+08 (2.5e+06)	1.4e+08 (2.0e+06)	1.4e+08 (4.4e+06)	
50	50	1.3e+08 (2.0e+06)	1.4e+08 (3.0e+06)	1.3e+08 (2.3e+06)	1.3e+08 (3.3e+06)	1.3e+08 (1.5e+06)
	100	1.3e+08 (1.3e+06)	1.3e+08 (2.5e+06)	1.3e+08 (7.1e+05)	1.3e+08 (4.2e+06)	1.3e+08 (1.8e+06)
	200	1.3e+08 (1.8e+06)	1.3e+08 (2.0e+06)	1.3e+08 (1.5e+06)	1.3e+08 (6.3e+06)	1.3e+08 (1.5e+06)
	500	1.3e+08 (1.3e+06)	1.3e+08 (4.3e+06)	1.3e+08 (2.0e+06)	1.3e+08 (3.8e+06)	

Table 16 Costs of the evaluated algorithms on the *Census* data set (with PCA preprocessing). Each cell specify the mean along with the standard deviation in parenthesis of 10 repetitions of the experiment.

Costs on the *Covertype* data set

k	m	BICO	Group Sampling	Ray Maker	Sensitivity Sampling	StreamKM++
10	50	3.5e+11 (5.4e+09)	3.6e+11 (4.8e+09)	3.5e+11 (5.5e+09)	3.6e+11 (9.1e+09)	3.5e+11 (8.5e+09)
	100	3.5e+11 (4.4e+09)	3.5e+11 (4.8e+09)	3.5e+11 (6.2e+09)	3.6e+11 (4.9e+09)	3.5e+11 (5.6e+09)
	200	3.4e+11 (3.2e+09)	3.5e+11 (5.2e+09)	3.5e+11 (5.1e+09)	3.5e+11 (5.7e+09)	3.4e+11 (1.8e+09)
	500	3.4e+11 (3.3e+09)	3.4e+11 (5.1e+09)	3.5e+11 (6.7e+09)	3.5e+11 (3.2e+09)	
20	50	2.1e+11 (2.5e+09)	2.1e+11 (2.3e+09)	2.1e+11 (1.9e+09)	2.2e+11 (4.1e+09)	2.1e+11 (2.8e+09)
	100	2.1e+11 (1.9e+09)	2.1e+11 (3.0e+09)	2.1e+11 (2.4e+09)	2.1e+11 (4.2e+09)	2.1e+11 (1.5e+09)
	200	2.0e+11 (9.0e+08)	2.1e+11 (2.0e+09)	2.1e+11 (1.4e+09)	2.1e+11 (2.5e+09)	2.0e+11 (1.3e+09)
	500	2.0e+11 (8.4e+08)	2.1e+11 (2.0e+09)	2.1e+11 (1.9e+09)	2.1e+11 (1.6e+09)	
30	50	1.6e+11 (1.9e+09)	1.6e+11 (1.4e+09)	1.6e+11 (1.8e+09)	1.6e+11 (2.7e+09)	1.6e+11 (1.5e+09)
	100	1.6e+11 (1.4e+09)	1.6e+11 (1.4e+09)	1.6e+11 (1.6e+09)	1.6e+11 (1.4e+09)	1.6e+11 (1.4e+09)
	200	1.6e+11 (1.3e+09)	1.6e+11 (1.4e+09)	1.6e+11 (1.5e+09)	1.6e+11 (1.4e+09)	1.6e+11 (1.2e+09)
	500	1.5e+11 (1.1e+09)	1.6e+11 (9.3e+08)	1.6e+11 (2.1e+09)	1.6e+11 (1.4e+09)	
40	50	1.3e+11 (1.6e+09)	1.4e+11 (8.7e+08)	1.3e+11 (7.5e+08)	1.4e+11 (1.3e+09)	1.3e+11 (1.9e+09)
	100	1.3e+11 (7.9e+08)	1.3e+11 (1.2e+09)	1.3e+11 (1.2e+09)	1.3e+11 (1.5e+09)	1.3e+11 (1.2e+09)
	200	1.3e+11 (8.5e+08)	1.3e+11 (1.3e+09)	1.3e+11 (1.8e+09)	1.3e+11 (9.3e+08)	1.3e+11 (8.8e+08)
	500	1.3e+11 (3.8e+08)	1.3e+11 (9.9e+08)	1.3e+11 (7.0e+08)	1.3e+11 (9.1e+08)	
50	50	1.2e+11 (8.8e+08)	1.2e+11 (9.4e+08)	1.2e+11 (7.2e+08)	1.2e+11 (1.2e+09)	1.2e+11 (1.1e+09)
	100	1.2e+11 (7.7e+08)	1.2e+11 (9.6e+08)	1.2e+11 (6.4e+08)	1.2e+11 (1.1e+09)	1.2e+11 (1.0e+09)
	200	1.1e+11 (7.2e+08)	1.2e+11 (1.5e+09)	1.2e+11 (4.9e+08)	1.2e+11 (7.0e+08)	1.1e+11 (7.6e+08)
	500	1.1e+11 (6.0e+08)	1.1e+11 (7.4e+08)	1.2e+11 (9.1e+08)	1.1e+11 (7.0e+08)	

Table 17 Costs of the evaluated algorithms on the *Covertype* data set. Each cell specify the mean along with the standard deviation in parenthesis of 10 repetitions of the experiment.

Costs on the *Covertype* data set (with PCA preprocessing)

k	m	BICO	Group Sampling	Ray Maker	Sensitivity Sampling	StreamKM++
10	50	3.5e+11 (5.4e+09)	3.6e+11 (1.0e+10)	3.5e+11 (7.4e+09)	3.6e+11 (9.2e+09)	3.5e+11 (5.9e+09)
	100	3.5e+11 (4.6e+09)	3.5e+11 (6.8e+09)	3.5e+11 (3.2e+09)	3.6e+11 (3.5e+09)	3.5e+11 (3.0e+09)
	200	3.4e+11 (3.1e+09)	3.5e+11 (6.9e+09)	3.5e+11 (6.7e+09)	3.5e+11 (3.6e+09)	3.4e+11 (3.8e+08)
	500	3.4e+11 (2.2e+09)	3.4e+11 (3.5e+09)	3.5e+11 (3.1e+09)	3.5e+11 (3.7e+09)	
20	50	2.1e+11 (1.7e+09)	2.1e+11 (2.6e+09)	2.1e+11 (2.7e+09)	2.2e+11 (4.7e+09)	2.1e+11 (4.3e+09)
	100	2.1e+11 (2.6e+09)	2.1e+11 (3.7e+09)	2.1e+11 (2.5e+09)	2.1e+11 (1.6e+09)	2.1e+11 (1.6e+09)
	200	2.1e+11 (1.2e+09)	2.1e+11 (1.7e+09)	2.1e+11 (1.7e+09)	2.1e+11 (2.1e+09)	2.0e+11 (1.3e+09)
	500	2.0e+11 (6.3e+08)	2.1e+11 (1.2e+09)	2.1e+11 (1.8e+09)	2.1e+11 (2.0e+09)	
30	50	1.6e+11 (1.6e+09)	1.6e+11 (2.2e+09)	1.6e+11 (1.6e+09)	1.7e+11 (4.8e+09)	1.6e+11 (1.7e+09)
	100	1.6e+11 (1.2e+09)	1.6e+11 (1.7e+09)	1.6e+11 (1.4e+09)	1.6e+11 (1.5e+09)	1.6e+11 (1.0e+09)
	200	1.6e+11 (5.7e+08)	1.6e+11 (2.5e+09)	1.6e+11 (1.6e+09)	1.6e+11 (1.5e+09)	1.5e+11 (6.7e+08)
	500	1.5e+11 (6.0e+08)	1.6e+11 (1.5e+09)	1.6e+11 (1.0e+09)	1.6e+11 (1.2e+09)	
40	50	1.3e+11 (1.6e+09)	1.4e+11 (1.6e+09)	1.3e+11 (1.2e+09)	1.4e+11 (1.6e+09)	1.3e+11 (9.7e+08)
	100	1.3e+11 (1.3e+09)	1.3e+11 (1.4e+09)	1.3e+11 (1.4e+09)	1.3e+11 (1.8e+09)	1.3e+11 (1.3e+09)
	200	1.3e+11 (8.2e+08)	1.3e+11 (9.9e+08)	1.3e+11 (1.2e+09)	1.3e+11 (1.8e+09)	1.3e+11 (3.1e+08)

	500	1.3e+11 (5.4e+08)	1.3e+11 (7.2e+08)	1.3e+11 (8.2e+08)	1.3e+11 (1.3e+09)	
50	50	1.2e+11 (8.4e+08)	1.2e+11 (1.4e+09)	1.2e+11 (9.7e+08)	1.2e+11 (1.7e+09)	1.2e+11 (1.4e+09)
	100	1.2e+11 (1.4e+09)	1.2e+11 (1.0e+09)	1.2e+11 (4.7e+08)	1.2e+11 (1.1e+09)	1.2e+11 (1.1e+09)
	200	1.1e+11 (9.7e+08)	1.2e+11 (1.0e+09)	1.2e+11 (9.4e+08)	1.2e+11 (6.0e+08)	1.1e+11 (8.8e+08)
	500	1.1e+11 (6.2e+08)	1.1e+11 (6.1e+08)	1.2e+11 (9.0e+08)	1.1e+11 (7.0e+08)	

Table 18 Costs of the evaluated algorithms on the *Covertype* data set (with PCA preprocessing). Each cell specify the mean along with the standard deviation in parenthesis of 10 repetitions of the experiment.

Costs on the *NYTimes* data set

k	m	BICO	Group Sampling	Ray Maker	Sensitivity Sampling	StreamKM++
10	50	2.8e+08 (7.7e+05)	2.8e+08 (4.6e+05)	2.8e+08 (3.6e+05)	2.8e+08 (5.2e+05)	2.9e+08 (1.5e+06)
	100	2.8e+08 (7.7e+05)	2.8e+08 (4.4e+05)	2.8e+08 (4.9e+05)	2.8e+08 (5.1e+05)	2.8e+08 (7.2e+05)
	200	2.8e+08 (9.3e+05)	2.8e+08 (8.6e+05)	2.8e+08 (7.7e+05)	2.8e+08 (5.2e+05)	2.8e+08 (7.4e+05)
	500	2.8e+08 (4.5e+05)	2.8e+08 (3.0e+05)	2.8e+08 (5.6e+05)	2.8e+08 (3.9e+05)	
20	50	2.8e+08 (9.7e+05)	2.8e+08 (1.2e+06)	2.8e+08 (5.5e+05)	2.8e+08 (9.1e+05)	2.8e+08 (5.3e+05)
	100	2.7e+08 (9.1e+05)	2.8e+08 (6.8e+05)	2.7e+08 (5.1e+05)	2.8e+08 (8.8e+05)	2.8e+08 (7.5e+05)
	200	2.7e+08 (5.0e+05)	2.7e+08 (6.6e+05)	2.7e+08 (5.9e+05)	2.7e+08 (6.8e+05)	2.8e+08 (1.0e+06)
	500	2.7e+08 (6.5e+05)	2.7e+08 (2.4e+05)	2.7e+08 (1.0e+06)	2.7e+08 (4.6e+05)	
30	50	2.7e+08 (6.6e+05)	2.7e+08 (1.1e+06)	2.7e+08 (7.7e+05)	2.7e+08 (1.7e+06)	2.8e+08 (1.1e+06)
	100	2.7e+08 (7.7e+05)	2.7e+08 (1.1e+06)	2.7e+08 (1.1e+06)	2.7e+08 (1.0e+06)	2.7e+08 (1.0e+06)
	200	2.7e+08 (7.4e+05)	2.7e+08 (6.6e+05)	2.7e+08 (8.2e+05)	2.7e+08 (9.0e+05)	2.7e+08 (1.2e+06)
	500	2.7e+08 (7.9e+05)	2.6e+08 (5.2e+05)	2.7e+08 (1.0e+06)	2.6e+08 (4.5e+05)	
40	50	2.7e+08 (1.1e+06)	2.7e+08 (9.1e+05)	2.7e+08 (1.0e+06)	2.7e+08 (9.9e+05)	2.8e+08 (5.2e+05)
	100	2.6e+08 (7.2e+05)	2.6e+08 (8.4e+05)	2.7e+08 (8.2e+05)	2.6e+08 (1.0e+06)	2.7e+08 (8.3e+05)
	200	2.6e+08 (8.1e+05)	2.6e+08 (4.6e+05)	2.7e+08 (7.6e+05)	2.6e+08 (5.0e+05)	2.7e+08 (8.7e+05)
	500	2.6e+08 (9.5e+05)	2.6e+08 (7.4e+05)	2.7e+08 (1.1e+06)	2.6e+08 (4.1e+05)	
50	50	2.6e+08 (1.0e+06)	2.7e+08 (8.0e+05)	2.6e+08 (5.5e+05)	2.6e+08 (1.5e+06)	2.7e+08 (8.9e+05)
	100	2.6e+08 (9.9e+05)	2.6e+08 (9.0e+05)	2.6e+08 (5.8e+05)	2.6e+08 (1.5e+06)	2.7e+08 (9.0e+05)
	200	2.6e+08 (9.4e+05)	2.6e+08 (4.3e+05)	2.6e+08 (8.0e+05)	2.6e+08 (5.5e+05)	2.6e+08 (1.1e+06)
	500	2.6e+08 (6.5e+05)	2.5e+08 (5.9e+05)	2.6e+08 (1.1e+06)	2.5e+08 (4.6e+05)	

Table 19 Costs of the evaluated algorithms on the *NYTimes* data set. Each cell specify the mean along with the standard deviation in parenthesis of 10 repetitions of the experiment.

Costs on the *NYTimes* data set (with PCA preprocessing)

k	m	BICO	Group Sampling	Ray Maker	Sensitivity Sampling	StreamKM++
10	50	2.7e+08 (5.5e+05)	2.7e+08 (5.2e+05)	2.7e+08 (4.7e+05)	2.7e+08 (5.3e+05)	2.7e+08 (5.4e+05)
	100	2.7e+08 (4.5e+05)	2.7e+08 (4.5e+05)	2.7e+08 (6.9e+05)	2.7e+08 (7.4e+05)	2.7e+08 (3.4e+05)
	200	2.7e+08 (4.9e+05)	2.7e+08 (3.4e+05)	2.7e+08 (5.0e+05)	2.7e+08 (5.9e+05)	2.7e+08 (8.4e+04)
	500	2.7e+08 (2.2e+05)	2.7e+08 (2.5e+05)	2.7e+08 (3.6e+05)	2.7e+08 (5.1e+05)	
20	50	2.6e+08 (4.2e+05)	2.6e+08 (4.3e+05)	2.6e+08 (2.6e+05)	2.6e+08 (4.4e+05)	2.6e+08 (2.5e+05)

	100	2.6e+08 (3.8e+05)	2.6e+08 (5.9e+05)	2.6e+08 (3.2e+05)	2.6e+08 (5.1e+05)	2.6e+08 (2.5e+05)
	200	2.6e+08 (2.7e+05)	2.6e+08 (2.5e+05)	2.6e+08 (3.9e+05)	2.6e+08 (2.5e+05)	2.6e+08 (2.0e+05)
	500	2.6e+08 (1.4e+05)	2.6e+08 (1.8e+05)	2.6e+08 (2.2e+05)	2.6e+08 (1.5e+05)	
30	50	2.6e+08 (5.7e+05)	2.6e+08 (3.7e+05)	2.6e+08 (3.0e+05)	2.6e+08 (4.2e+05)	2.6e+08 (3.4e+05)
	100	2.6e+08 (4.3e+05)	2.6e+08 (3.4e+05)	2.6e+08 (3.0e+05)	2.6e+08 (3.1e+05)	2.6e+08 (3.7e+05)
	200	2.6e+08 (4.2e+05)	2.6e+08 (2.2e+05)	2.6e+08 (3.8e+05)	2.6e+08 (5.1e+05)	2.6e+08 (3.0e+05)
	500	2.6e+08 (2.5e+05)	2.6e+08 (1.8e+05)	2.6e+08 (2.8e+05)	2.6e+08 (3.6e+05)	
40	50	2.6e+08 (2.3e+05)	2.6e+08 (4.3e+05)	2.6e+08 (4.0e+05)	2.6e+08 (4.4e+05)	2.6e+08 (4.5e+05)
	100	2.6e+08 (6.4e+05)	2.6e+08 (2.7e+05)	2.6e+08 (4.6e+05)	2.6e+08 (5.0e+05)	2.6e+08 (1.1e+05)
	200	2.6e+08 (4.7e+05)	2.6e+08 (2.9e+05)	2.6e+08 (4.0e+05)	2.6e+08 (4.1e+05)	2.6e+08 (1.8e+05)
	500	2.6e+08 (3.5e+05)	2.6e+08 (2.0e+05)	2.6e+08 (3.3e+05)	2.6e+08 (2.4e+05)	
50	50	2.5e+08 (3.8e+05)	2.5e+08 (4.6e+05)	2.5e+08 (3.6e+05)	2.5e+08 (5.5e+05)	2.5e+08 (4.3e+05)
	100	2.5e+08 (4.4e+05)	2.5e+08 (3.2e+05)	2.5e+08 (5.0e+05)	2.5e+08 (5.2e+05)	2.5e+08 (3.9e+05)
	200	2.5e+08 (3.6e+05)	2.5e+08 (3.6e+05)	2.5e+08 (5.0e+05)	2.5e+08 (3.8e+05)	2.5e+08 (2.8e+05)
	500	2.5e+08 (3.7e+05)	2.5e+08 (3.3e+05)	2.5e+08 (3.8e+05)	2.5e+08 (3.2e+05)	

Table 20 Costs of the evaluated algorithms on the *NYTimes* data set (with PCA preprocessing). Each cell specify the mean along with the standard deviation in parenthesis of 10 repetitions of the experiment.

Costs on the *Tower* data set

k	m	BICO	Group Sampling	Ray Maker	Sensitivity Sampling	StreamKM++
20	50	6.5e+08 (1.3e+07)	6.6e+08 (3.4e+07)	6.6e+08 (1.8e+07)	6.6e+08 (1.4e+07)	6.5e+08 (2.3e+07)
	100	6.4e+08 (1.8e+07)	6.5e+08 (2.3e+07)	6.3e+08 (1.3e+07)	6.8e+08 (4.2e+07)	6.4e+08 (1.4e+07)
	200	6.3e+08 (4.8e+06)	6.4e+08 (2.0e+07)	6.2e+08 (7.0e+06)	6.5e+08 (1.9e+07)	6.3e+08 (1.4e+07)
	500	6.2e+08 (8.4e+06)	6.5e+08 (4.8e+07)	6.2e+08 (1.2e+07)	6.4e+08 (2.0e+07)	
40	50	3.4e+08 (7.3e+06)	3.6e+08 (6.3e+06)	3.4e+08 (5.2e+06)	3.6e+08 (8.4e+06)	3.4e+08 (4.9e+06)
	100	3.4e+08 (9.1e+06)	3.5e+08 (5.9e+06)	3.4e+08 (6.0e+06)	3.5e+08 (7.9e+06)	3.4e+08 (1.6e+06)
	200	3.3e+08 (2.2e+06)	3.4e+08 (5.9e+06)	3.3e+08 (2.5e+06)	3.5e+08 (5.5e+06)	3.3e+08 (4.4e+06)
	500	3.3e+08 (3.2e+06)	3.3e+08 (5.9e+06)	3.3e+08 (5.2e+06)	3.4e+08 (5.6e+06)	
60	50	2.5e+08 (3.1e+06)	2.5e+08 (3.4e+06)	2.5e+08 (3.8e+06)	2.6e+08 (5.3e+06)	2.5e+08 (3.4e+06)
	100	2.5e+08 (3.7e+06)	2.5e+08 (2.3e+06)	2.4e+08 (1.4e+06)	2.5e+08 (5.2e+06)	2.4e+08 (2.9e+06)
	200	2.4e+08 (1.7e+06)	2.5e+08 (2.9e+06)	2.4e+08 (2.3e+06)	2.5e+08 (3.5e+06)	2.4e+08 (2.2e+06)
	500	2.4e+08 (2.1e+06)	2.4e+08 (1.8e+06)	2.4e+08 (2.2e+06)	2.5e+08 (3.3e+06)	
80	50	2.0e+08 (1.5e+06)	2.0e+08 (3.3e+06)	2.0e+08 (1.6e+06)	2.0e+08 (3.4e+06)	2.0e+08 (1.7e+06)
	100	2.0e+08 (1.9e+06)	2.0e+08 (2.7e+06)	2.0e+08 (1.7e+06)	2.0e+08 (2.6e+06)	2.0e+08 (2.6e+06)
	200	1.9e+08 (1.4e+06)	2.0e+08 (2.4e+06)	1.9e+08 (1.4e+06)	2.0e+08 (2.0e+06)	1.9e+08 (1.4e+06)
	500	1.9e+08 (1.2e+06)	2.0e+08 (2.4e+06)	1.9e+08 (1.7e+06)	1.9e+08 (2.2e+06)	
100	50	1.7e+08 (2.1e+06)	1.7e+08 (2.6e+06)	1.7e+08 (1.5e+06)	1.7e+08 (1.5e+06)	1.7e+08 (2.0e+06)
	100	1.7e+08 (1.5e+06)	1.7e+08 (2.0e+06)	1.7e+08 (1.4e+06)	1.7e+08 (1.7e+06)	1.7e+08 (9.2e+05)
	200	1.6e+08 (1.2e+06)	1.7e+08 (1.3e+06)	1.6e+08 (1.2e+06)	1.7e+08 (1.8e+06)	1.6e+08 (7.8e+05)
	500	1.6e+08 (7.7e+05)	1.7e+08 (9.1e+05)	1.6e+08 (1.3e+06)	1.7e+08 (1.3e+06)	

Table 21 Costs of the evaluated algorithms on the *Tower* data set. Each cell specify the mean along with the standard deviation in parenthesis of 10 repetitions of the experiment.

G Generating Candidate Solutions

As mentioned in Section 4, it is difficult to construct a candidate solution consisting of k centers which yields a maximal distortion. We experimented with three methods for candidate solution generation; find k centers via k-means++, randomly generate k points inside the convex hull (Random CH), and randomly generate k points inside the minimum enclosing ball (Random MEB). Our experiments show that the k-means++ method generates candidate solutions which consistently result in large distortions across all algorithms and data sets. The numerical results are presented in Table 22. Notice that the the numbers in Table 22 are the maximum distortions across k and m because we are only interesting in determining which of the three methods is superior. Table 23 shows all the numbers for BICO on the *Caltech* data set. For all the other combinations of algorithms and data sets, the takeaway message is virtually the same; k-means++ is the best method for generating candidate solutions.

Data set	Algorithm	k-means++	Random CH	Random MEB
Caltech	BICO	7.499068	4.800188	2.629218
	Group Sampling	1.068441	1.035997	1.006828
	Ray Maker	7.053856	5.442665	3.453930
	Sensitivity Sampling	1.048977	1.016564	1.029646
	StreamKM++	1.202426	1.111698	1.012216
Caltech+PCA	BICO	2.862242	2.411732	1.825558
	Group Sampling	1.058707	1.021386	1.007735
	Ray Maker	2.941248	2.563131	2.053860
	Sensitivity Sampling	1.040106	1.012539	1.032113
	StreamKM++	1.139141	1.073384	1.013089
Census	BICO	2.502564	1.441889	1.040356
	Group Sampling	1.077525	1.032349	1.028025
	Ray Maker	2.483973	1.633945	1.236878
	Sensitivity Sampling	1.032978	1.088607	1.079925
	StreamKM++	1.127443	1.044656	1.003459
Census+PCA	BICO	2.454087	1.472404	1.030620
	Group Sampling	1.070332	1.036615	1.031606
	Ray Maker	2.456844	1.617594	1.234714
	Sensitivity Sampling	1.036155	1.086034	1.083994
	StreamKM++	1.126928	1.042801	1.003570
Covertype	BICO	1.294620	1.146309	1.037080
	Group Sampling	1.094779	1.041165	1.023978
	Ray Maker	1.388141	1.227148	1.121157
	Sensitivity Sampling	1.044575	1.115355	1.103073
	StreamKM++	1.048912	1.019965	1.004857
Covertype+PCA	BICO	1.292637	1.147499	1.037594
	Group Sampling	1.097391	1.038144	1.023664
	Ray Maker	1.385070	1.229436	1.122238
	Sensitivity Sampling	1.039479	1.097699	1.092900
	StreamKM++	1.049769	1.022796	1.006932

NYTimes	BICO	35.319657	31.015943	28.932553
	Group Sampling	1.089067	1.061821	1.033067
	Ray Maker	28.373678	24.930933	19.520125
	Sensitivity Sampling	1.050988	1.027549	1.012441
	StreamKM++	2.039047	1.861497	1.674643
Tower	BICO	1.199658	1.040606	1.005894
	Group Sampling	1.113305	1.043449	1.030855
	Ray Maker	1.490978	1.286890	1.234467
	Sensitivity Sampling	1.044464	1.103949	1.075281
	StreamKM++	1.051491	1.009577	1.000630

Table 22 Distortions across experiments on different algorithms and data sets.

Distortions of BICO on the *Caltech* data set

k	m	k-means++	Random CH	Random MEB
10	50	5.12 (0.307)	4.30 (0.270)	2.63 (0.166)
	100	4.48 (0.284)	3.91 (0.240)	2.54 (0.125)
	200	4.08 (0.319)	3.54 (0.236)	2.46 (0.134)
	500	3.41 (0.215)	3.06 (0.155)	2.18 (0.080)
20	50	6.35 (1.173)	4.73 (0.632)	2.63 (0.147)
	100	4.65 (0.283)	3.82 (0.245)	2.41 (0.104)
	200	4.19 (0.384)	3.49 (0.271)	2.26 (0.141)
	500	3.50 (0.404)	3.07 (0.301)	2.10 (0.128)
30	50	6.01 (0.335)	4.32 (0.204)	2.57 (0.096)
	100	5.10 (0.628)	3.89 (0.340)	2.35 (0.125)
	200	4.29 (0.659)	3.47 (0.440)	2.21 (0.170)
	500	3.09 (0.138)	2.69 (0.122)	1.87 (0.064)
40	50	6.24 (0.524)	4.33 (0.228)	2.44 (0.076)
	100	5.23 (0.874)	3.86 (0.434)	2.29 (0.191)
	200	4.50 (1.085)	3.49 (0.595)	2.16 (0.183)
	500	3.38 (0.398)	2.85 (0.258)	1.92 (0.111)
50	50	7.50 (1.013)	4.80 (0.449)	2.47 (0.190)
	100	5.21 (0.968)	3.76 (0.475)	2.20 (0.149)
	200	4.21 (0.296)	3.35 (0.195)	2.10 (0.065)
	500	3.36 (0.429)	2.81 (0.288)	1.86 (0.105)

Table 23 The effect of different solution generation approaches on the distortions (see Appendix G). The distortions are obtained by running BICO on the *Caltech* data set.

H Auxiliary Plots

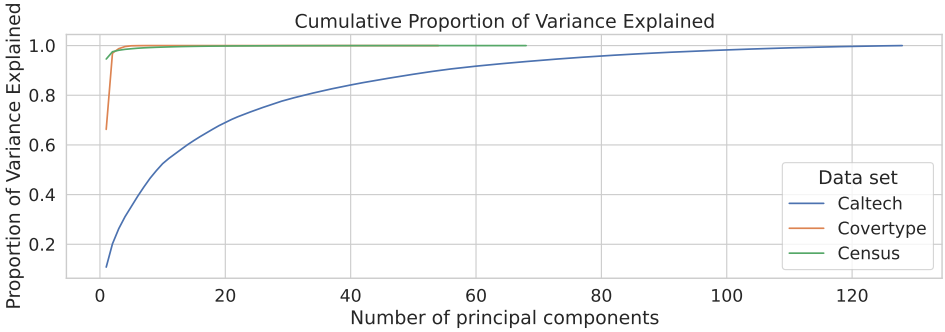

Figure 4 The cumulative proportion of explained variance by principal components on *Caltech*, *Covertype*, and *Census*.

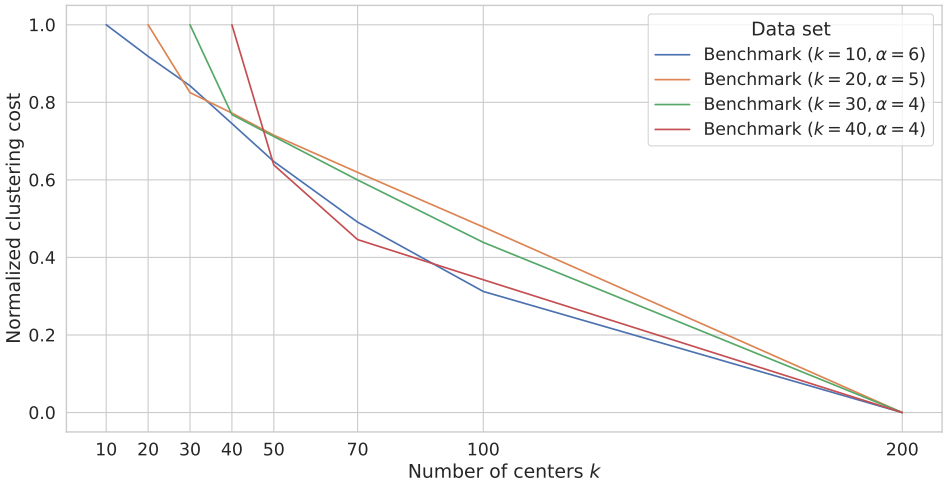

Figure 5 Shows the clustering costs of four instances of the benchmark framework as a function of the number of centers. In contrast to real-world data sets, the costs do not decrease rapidly as more cluster centers are added.

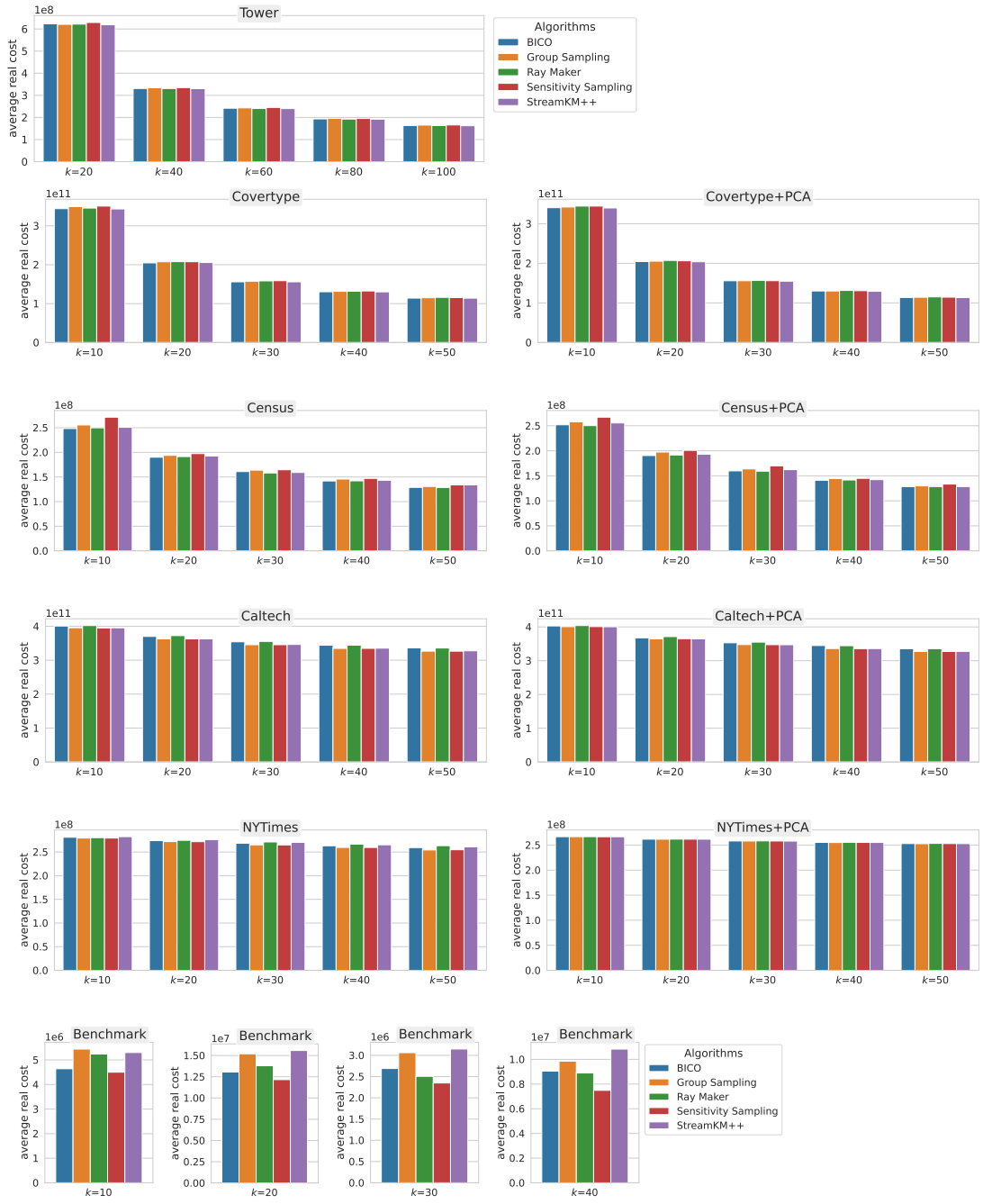

Figure 6 The average costs of running the evaluated coreset algorithms multiple times on different data sets. In general, the five coreset algorithms are able to compute coresets which result in solutions with comparable costs on the different real-world data sets. The differences in cost is more noticeable 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.