Streaming Coreset Constructions for M-Estimators

Vladimir Braverman
Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA
vova@jhu.edu

Dan Feldman
Department of Computer Science, University of Haifa, Israel
dannyf.post@gmail.com

Harry Lang
MIT CSAIL, Cambridge, MA, USA
harry1@mit.edu

Daniela Rus
MIT CSAIL, Cambridge, MA, USA
rus@mit.edu

Abstract

We introduce a new method of maintaining a (k, ϵ)-coreset for clustering M-estimators over insertion-only streams. Let (P, w) be a weighted set (where $w : P \to [0, \infty)$ is the weight function) of points in a ρ-metric space (meaning a set $\mathcal{X}$ equipped with a positive-semidefinite symmetric function D such that $D(x, z) \leq \rho(D(x, y) + D(y, z))$ for all $x, y, z \in \mathcal{X}$). For any set of points C, we define $\mathsf{COST}(P, w, C) = \sum_{p \in P} w(p) \min_{c \in C} D(p, c)$. A (k, ϵ)-coreset for (P, w) is a weighted set (Q, v) such that for every set C of k points, $(1 - \epsilon)\mathsf{COST}(P, w, C) \leq \mathsf{COST}(Q, v, C) \leq (1 + \epsilon)\mathsf{COST}(P, w, C)$. Essentially, the coreset (Q, v) can be used in place of (P, w) for all operations concerning the COST function. Coresets, as a method of data reduction, are used to solve fundamental problems in machine learning of streaming and distributed data.

M-estimators are functions $D(x, y)$ that can be written as $\psi(d(x, y))$ where $(\mathcal{X}, d)$ is a true metric (i.e. 1-metric) space. Special cases of M-estimators include the well-known k-median ($\psi(x) = x$) and k-means ($\psi(x) = x^2$) functions. Our technique takes an existing offline construction for an M-estimator coreset and converts it into the streaming setting, where n data points arrive sequentially. To our knowledge, this is the first streaming construction for any M-estimator that does not rely on the merge-and-reduce tree. For example, our coreset for streaming metric k-means uses $O(\epsilon^{-2} k \log k \log n)$ points of storage. The previous state-of-the-art required storing at least $O(\epsilon^{-2} k \log k \log^4 n)$ points.

2012 ACM Subject Classification Theory of computation → Streaming models; Theory of computation → Facility location and clustering; Information systems → Query optimization

Keywords and phrases Streaming, Clustering, Coresets

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2019.62

Category RANDOM

Funding *Vladimir Braverman*: This research was supported in part by NSF CAREER grant 1652257, ONR Award N00014-18-1-2364, DARPA/ARO award W911NF1820267.
Harry Lang: This material is based upon work supported by the Franco-American Fulbright Commission. The author thanks INRIA (l'Institut national de recherche en informatique et en automatique) for hosting him during part of the writing of this paper.
Daniela Rus: This research was supported in part by NSF 1723943, NVIDIA, and J.P. Morgan Chase & Co.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019).
Editors: Dimitris Achlioptas and László A. Végh; Article No. 62; pp. 62:1–62:15

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

In the streaming model of computation, the input arrives sequentially. This differs from the random-access memory model (i.e. the offline setting) where the algorithm may freely and repeatedly access the entire input. The goal of a streaming algorithm is to perform the computation using a sublinear amount of memory.

A stream consists of n elements $p_1, \ldots, p_n$. Sometimes the algorithm will be allowed to pass over the stream multiple times, resulting in another parameter called the number of passes. Our algorithm uses $O(\log n)$ memory and requires only a single pass over the input stream.

Prior to the current work, the merge-and-reduce technique due to Har-Peled and Mazumdar [19] and Bentley and Sax [5] was used to maintain coresets on streams using an offline coreset construction as a blackbox. For a brief review of this technique, see Section 4.1. Suppose the offline construction's space depends on ϵ as ϵ^{-a}. In this paper we introduce an alternative technique that reduces the multiplicative overhead from $O(\log^{a+1} n)$ to $O(1)$ when moving to the streaming setting. For example, the state-of-the-art k-median offline coreset [6] has size $O(\epsilon^{-2} k \log k \log n)$. The current paper improves the space requirement from $O(\epsilon^{-2} k \log k \log^4 n)$ to $O(\epsilon^{-2} k \log k \log n)$ to maintain the coreset on a stream. While our method is not as general as merge-and-reduce (it requires a function to satisfy more than just the "merge" and "reduce" properties, defined in Section 4.1), it is general enough to apply to all M-estimators.

2 Definitions

We begin by defining a ρ-metric space. Let $\mathcal{X}$ be a set. If $D : \mathcal{X} \times \mathcal{X} \to [0, \infty)$ is a symmetric positive-semidefinite function such that for every $x, y, z \in \mathcal{X}$ we have that $D(x, z) \leq \rho(D(x, y) + D(y, z))$ then we call $(\mathcal{X}, D)$ a ρ-metric space. Note that this is a weakening of the triangle inequality, and at $\rho = 1$ we recover the definition of a metric space. Clustering M-estimators in a metric space can be re-cast as k-median in a ρ-metric space. For example, metric k-means in the space $(\mathcal{X}, d)$ is reducible to 2-metric k-median in the space $(\mathcal{X}, D)$ using $D(\cdot, \cdot) = d(\cdot, \cdot)^2$. See Table 1 for more examples. We work in this slightly abstract language since it allows a single proof to naturally generalize our results to any M-estimator.

A weighted set (P, w) is a set $P \subset \mathcal{X}$ along with a weight function $w : P \to [0, \infty)$. As usual, we define the distance between a point and a set $D(p, Z) = \min_{z \in Z} D(p, Z)$. Let

$$\mathsf{COST}(P, w, Z) = \sum_{p \in P} w(p) D(p, Z)$$

The ρ-metric k-median problem is, given input (P, w) and an integer $k \geq 1$, to find a set C of k points in $\mathcal{X}$ that minimizes $\mathsf{COST}(P, w, C)$. We use $\mathsf{OPT}_k(P)$ to denote the minimal value. Other works have shown that for small enough ϵ, it is NP-Hard to even compute a $(1 + \epsilon)$ approximation for k-means or k-median when k is part of the input (see the Related Work section of [4] for a survey of hardness results). Therefore weaker definitions of approximation have been introduced. The notion of a bicriterion approximation is well-known; we state a slightly more verbose definition that suits our needs. The difference is that usually the map f is implicit, simply mapping a point to a nearest center.

▶ **Definition 1** ((α,β)-approximation). *For $\alpha,\beta \geq 1$, an (α,β)-approximation for the k-median clustering of a weighted set (P,w) is a weighted set (B,v) along with a map $f : P \to B$ such that:*

1. $\sum_{p\in P} w(p)D(p,f(p)) \leq \alpha \mathsf{OPT}_k(P)$
2. $|B| \leq \beta k$
3. $v(b) = \sum_{p\in f^{-1}(b)} w(p)$ *for every* $b \in B$

Observe that a $(1,1)$-approximation is an optimal solution. We now define a coreset, the datastructure that we compute in this paper.

▶ **Definition 2** ((k,ϵ)-coreset). *For $k \in \mathbb{N}$ and $\epsilon \in (0,1)$, a (k,ϵ)-coreset for a weighted set (P,w) is a weighted set (Q,v) such that for every $Z \in \mathcal{X}^k$ we have $(1-\epsilon)\mathsf{COST}(P,w,Z) \leq \mathsf{COST}(Q,v,Z) \leq (1+\epsilon)\mathsf{COST}(P,w,Z)$.*

Coresets with possibly negative weight functions have been considered [13]. However, computing approximate solutions on these coresets in polynomial-time remains an open problem, so we restrict our definition to non-negative weight functions to ensure that an approximate solution can be quickly produced [16, 8]. This implies a PTAS for Euclidean space and a polynomial-time $2\tau(1+\epsilon)$-approximation for general metric spaces (where τ is the best polynomial-time approximation factor for the problem in the offline setting). This factor of $2\tau(1+\epsilon)$ is well-known in the literature, see [9, 7, 16] for details.

3 Our Techniques

In this work, we provide an alternative technique for constructing a ρ-metric k-median coreset in the streaming setting. Instead of using the merge-and-reduce tree (see Section 4.1) where each node of the tree uses the offline construction, we perform a single offline construction in the streaming setting. This reduces $O(\log^{a+1} n)$ multiplicative overhead[1] of merge-and-reduce to $O(1)$ overhead.

The offline coreset construction of [6] has the following structure: first, a bicriterion approximation (see Definition 1) is computed over the entire input set P. The bicriterion is used to estimate for each point p its "sensitivity" $s_P(p)$ which, intuitively speaking, measures how important p is relative to the entire set P. Then, $m = m(n,k,\epsilon)$ points are sampled i.i.d. according to the distribution of sensitivities. This suggests a two-pass streaming algorithm with no overhead: in the first pass, construct a bicriterion using an algorithm such as [7]. In the second pass, sample according to sensitivity distribution, computed using the bicriterion found in the first pass. Our contribution is showing how these two passes can be combined into a single-pass.

How do we accomplish both these tasks in parallel? If the sensitivities stayed constant, we could use weighted reservoir sampling to maintain an i.i.d. sample. However, we cannot do this for a changing distribution. The sensitivities may decrease because when a new point is added, all existing points may become less important to the overall stream.

Inductively assume that we have a coreset. Upon receiving the next point, we generate a new bicriterion which we use to update the sensitivities of the points seen so far. The first idea is that instead of sampling m points from a distribution such that point p is sampled with probability $s_P(p)$, we built a set which contains point p with probability $s_P(p)$ as follows: let $u(p)$ be a uniform random number in $[0,1)$, and store p as long as $u(p) < ms_P(p)$ (recall

[1] Recall that a is the exponent in the offline construction's dependence on ϵ^{-1}, and n is the length of the stream.

that $s_P(p)$ decreases as more points are added to P). With high probability, we return a set of $\Theta(m)$ points. The problem is that this set is not an i.i.d sample. To see this, consider the fact that unlike an i.i.d. sample, this set cannot have repeated points.

To solve this problem, we repeat the above process independently in parallel m times, where each process should sample *exactly* 1 point. The first issue is that the total probability $t = \sum_{p \in P} s_P(p)$, which must be scaled to 1, is only known up to a factor of k. Even after solving this, and assuming we have at least m trials that return exactly one point, they do not follow the original distribution. Indeed, the probability of a trial returning only one point p is the probability of drawing point p multiplied the probabilities of not drawing all other points. The rough idea to overcome this is that we distort the probabilities by computing the inverse to this transformation such that the final probabilities follow the desired distribution.

We prove our result for ρ-metric k-median. As stated before, it is well-known that most M-estimators can be recast as a ρ-metric k-median problem for a low value of ρ [14]. See Table 1 for a list of several common M-estimators along with the ρ-metric they satisfy. For many M-estimators, ours is the first coreset result over data streams besides merge-and-reduce.

Table 1 List of several examples to which our result applies. An M-estimator with ψ-function $\psi(x)$ induces a clustering problem with cost $\psi(d(p,c))$ for a point p with nearest center c. The ρ-value is calculated from the ψ-function, making the column redundant but non-trivial to compute.

Estimator	ψ-function	ρ
k-median	x	1
k-means	x^2	2
Huber	$\psi(x) = \begin{cases} \frac{x^2}{2} & \text{if } x < 1 \\ x - \frac{1}{2} & \text{if x} \geq 1 \end{cases}$	2
Cauchy	$\psi(x) = \log(1 + x^2)$	2
Tukey	$\psi(x) = \begin{cases} \frac{1}{6}(1 - (1 - x^2)^3) & \text{if } x < 1 \\ \frac{1}{6} & \text{if x} \geq 1 \end{cases}$	3

4 Related Work

Table 2 summarizes previous work along with our current results. By far, the most widely-studied problems in this class have been the k-median and k-means functions. In general, the extension to arbitrary M-estimators is non-trivial; the first such result was [14]. Our approach naturally lends itself to this extension. M-estimators are highly important for noisy data or data with outliers. As one example, Huber's estimator is widely used in the statistics community [17, 20]. It was written that "this estimator is so satisfactory that it has been recommended for almost all situations" [22]. Our results work not only for Huber's estimator but for all M-estimators, such as the Cauchy and Tukey biweight functions which are also widely-used functions.

Note that in the below table, $\tilde{O}$ notation is used to write in terms of d, ϵ, k, and $\log n$ (therefore hiding factors of $\log \log n$ but not $\log n$).

k-means

In the k-means problem we wish to compute a set k of centers (points) in some metric space, such that the sum of squared distances to the input points is minimized, where each input point is assigned to its nearest center. The corresponding coreset is a positively weighted subset of points that approximates this cost to every given set of k centers. Deterministic

Table 2 Summary of Related Work. Note that Metric M-estimators are the most general, and these results apply to all other categories.

Problem	Streaming Size	Paper
Euclidean k-means	$O(k\epsilon^{-d}\log^{d+2} n)$	[19]
Euclidean k-means	$O(k^3\epsilon^{-(d+1)}\log^{d+2} n)$	[18]
Euclidean k-means	$O(dk^2\epsilon^{-2}\log^8 n)$	[10]
Euclidean k-means	$O(dk\log k\epsilon^{-4}\log^5 n)$	[13]
Euclidean k-means	$\tilde{O}((d/\epsilon)^{O(d)}k\log^{O(d)} n)$	[1]
Metric k-means	$O(\epsilon^{-2}k^2\log^8 n)$	[11]
Metric k-means	$O(\epsilon^{-4}k\log k\log^6 n)$	[13]
Euclidean k-median	$O(dk^2\epsilon^{-2}\log^8 n)$	[10]
Euclidean k-median	$O(k\epsilon^{-d}\log^{d+2} n)$	[19]
Euclidean k-median	$O(k^2\epsilon^{-O(d)}\log^{d+1} n)$	[18]
Euclidean k-median	$O(d\epsilon^{-2}k\log k\log^3 n)$	[13]
Metric k-median	$O(k^2\epsilon^{-2}\log^8 n)$	[10]
Metric k-median	$O(\epsilon^{-2}k\log k\log^4 n)$	[13]
Euclidean M-estimators	$O(\epsilon^{-2}k^{O(k)}d^2\log^5 n)$	[14]
Metric M-estimators	$O(\epsilon^{-2}k^{O(k)}\log^7 n)$	[14]
Metric M-estimators	$O(\epsilon^{-2}k\log k\log n)$	This paper

coresets of size exponential in d were first suggested by Har-Peled and Mazumdar in [19]. The first coreset construction of size polynomial in d was suggested by Ke-Chen in [10] using several sets of uniform sampling. Other high-dimensional results (e.g. [2]) are also known in the streaming setting.

Streaming

The metric results of [10, 13] and Euclidean results of [10, 19, 18, 13] that rely on merge-and-reduce appear in Table 2. In Euclidean space, a more diverse set of stronger results is known. In particular, coreset constructions are known that do not begin with a bicriterion solution, and whose streaming variant does not rely on merge-and-reduce [1]. Sketches have been given in [12] for M-estimators in Euclidean space. With the additional assumption that points lie on a discrete Euclidean grid $\{1,\ldots,\Delta\}^d$, alternative techniques are known for k-means and other problems, even when the stream allows the deletion of points [15].

4.1 Merge and Reduce Tree

We briefly summarize the previous technique for maintaining coresets in the streaming setting due to Har-Peled and Mazumdar [19] and Bentley and Sax [5]. In this method, a merge-and-reduce tree is built by using an offline coreset construction as a blackbox. Previously, this was the only known technique for building a streaming coreset for many metric problems. It relies solely on the following two properties which can be easily verified:

1. Merge: The union of (k,ϵ)-coresets is a (k,ϵ)-coreset.
2. Reduce: A (k,ϵ)-coreset of a (k,ϵ')-coreset is a $(k,\epsilon+\epsilon'+\epsilon\epsilon')$-coreset.

The merge-and-reduce tree works as follows. There are buckets T_i for $i\geq 0$. In each step, the bucket T_0 takes in a segment of $O(1)$ points from the stream. Then the tree works like counting in binary: whenever buckets T_0 to T_{i-1} are full, these i buckets are merged and then reduced by taking a $(k,\frac{\epsilon}{\log n})$-coreset of their union and storing the result in T_i.

Let s be the space of offline construction, which depends on ϵ as ϵ^{-a}. At the end of the stream, $O(\log n)$ buckets have been used and each bucket uses $O(s \log^a n)$ space; this incurs a multiplicative overhead of $\Theta(\log^{a+1} n)$ in the storage requirement. The second factor comes from using the accuracy parameter $\frac{\epsilon}{\log n}$, which is necessary by Property 2 since the construction will be compounded $O(\log n)$ times. Due to this compounding, the runtime is multiplied by a factor of $O(\log n)$.

5 Streaming Algorithm

We present a streaming algorithm to construct a coreset for ρ-metric k-median. Our method combines a streaming bicriterion algorithm [7, 21] and a batch coreset construction [13] to create a streaming coreset algorithm. The space requirements are combined addivitely, therefore ensuring no overhead.

We now state our main theorem. The probability of success $1-\delta$ typically has one of two meanings: that the construction succeeds at the end of the stream (a weaker result), or that the construction succeeds at every intermediate point of the stream (a stronger result). Our theorem gives the stronger result, maintaining a valid coreset at every point of the stream. Our space and time bounds follow the convention that a point can be stored in $O(1)$ space.

▶ **Theorem 3** (Main Theorem). *Let $\epsilon, \delta \in (0,1)$. Given the problem of k-median clustering in a ρ-metric space for $\rho = O(1)$, there exists an insertion-only streaming algorithm that maintains a (k,ϵ)-coreset on a stream of n points while requiring $O(\epsilon^{-2}k(\log k \log n + \log \frac{1}{\delta}))$ space and worst-case update time, and succeeds at every point of the stream with probability at least $1-\delta$.*

In Section 5.1 we introduce the streaming bicriterion algorithm. Then in Section 5.2 we review the offline coreset construction we will be adapting to the streaming setting. We prove in Section 5.3 how to use the bicriterion to bound the importance of points. Finally we present the streaming algorithm in Section 5.4.

5.1 Streaming Bicriterion Algorithm

Let P_i denote the prefix of the stream $\{p_1, \ldots, p_i\}$. The entire stream P is then P_n. Recall that in the streaming setting, we receive each point sequentially in the order $(p_1, p_2, \ldots)$. We use the function $\mathbb{1} : P \to \{1\}$ to map every point to unit weight. For ease of exposition we assume the input set is weighted as $(P, \mathbb{1})$ and that all points of P are distinct. Consider the moment when the first i points have arrived, meaning that the prefix P_i is the current set of arrived points. The algorithm $\mathcal{A}$ of [7] provides an $(O(1), O(\log n))$-approximation of P_i. We now restate their general result, adding several details (such as the outputs L_i and π_i) that were merely internal details for them but will be crucial for us.

▶ **Theorem 4** ([7], restated). *Let $\alpha, \gamma > 1$ be absolute constants. Define $B_0 = \emptyset$. Let $(p_1, \ldots, p_n)$ be a stream of at most n points in a ρ-metric space for $\rho = O(1)$. Let $n, k \geq 1$, $\delta \in (0,1)$ be input parameters. Upon receiving point p_i, algorithm $\mathcal{A}(k, n, \delta)$ returns a weighted set (B_i, w_i), a value $L_i > 0$, and a map $\pi_i : B_{i-1} \cup \{p_i\} \to B_i$. Define $f_i(p_j) = \pi_i(\pi_{i-1}(\ldots \pi_j(p_j)\ldots))$. For any integer $i \in [n]$ we have with probability at least $1-\delta$ that the following three statements hold.*

1. $L_i \leq \mathsf{OPT}_k(P_i)$
2. $\sum_{p \in P_i} D(p, f_i(p)) \leq \alpha L_i$
3. *(B_i, w_i) along with the map $f_i : P_i \to B_i$ is an $(\alpha, \gamma(\log n + \log \frac{1}{\delta}))$-approximation for $(P, \mathbb{1})$.*

The algorithm requires $O(\gamma(\log n + \log \frac{1}{\delta}))$ space and update time.

Algorithm $\mathcal{A}$ reduces the number of distinct points by combining nearby points into a single point of higher weight.

5.2 Offline Coreset Construction

In the offline coreset construction of [6], the sensitivity of a point $p \in P$ in a ρ-metric space $(\mathcal{X}, D)$ is defined as:

$$s_P(p) = \max_{Z \in \mathcal{X}^k} \frac{D(p,Z)}{\sum_{q \in P} D(q,Z)}$$

Notice that $0 \leq s_P(p) \leq 1$ for every point $p \in P$, and that the sensitivity of a point p is relative to the set P it belongs to. When context is clear, we omit the subscript and write $s(p)$. Computing $s(p)$ may be difficult, but we can give an upper bound $s'(p) \in [s(p), 1]$. Define the total sensitivity $t' = \sum_{p \in P} s'(p)$. We will apply the following theorem:

▶ **Theorem 5** (proven in [6]). *Let P be a set of n points, and define $s' : P \to [0,1]$ and t' as above. Let $\delta, \epsilon \in (0,1)$ be input parameters. Consider a distribution $\mathcal{S}$ supported on P where p has weight $s'(p)/t'$. Define $m' = \lceil 3t'\epsilon^{-2}(\log n \log t' + \log(1/\delta))\rceil$. Let Q be an i.i.d. sample of at least m' points from $\mathcal{S}$. Define a weight function $v : Q \to [0,\infty)$ as $v(q) = (|Q|s'(q))^{-1}$. With probability at least $1-\delta$, (Q,v) is a (k,ϵ)-coreset for P.*

One may be skeptical why only an upper bound is necessary, wondering why not simply set $s'(p) = 1$. This does indeed work, but has the undesirable effect of setting $t' = n$ and therefore results in a coreset of $\Omega(n)$ points. More generally, the coreset is useless if t' is large since the size of Q may be comparable to the size of P. In the next subsection, we show how to bound $t' = O(\rho^2 k)$. Observe that neither k nor ρ appear explicitly in the sample size, but they both appear implicitly through the value of t'.

5.3 Bounding the Sensitivity

Let the map $p \mapsto p'$ be an (σ,λ)-approximation of P for some constants σ and λ. Define $P(p) = \{q \in P : q' = p'\}$ to be the cluster containing p.

▶ **Lemma 6.** *Let the map $p \mapsto p'$ define an (σ,λ)-approximation for the k-median clustering of P. For every point $p \in P$:*

$$s(p) \leq \frac{\rho\sigma D(p,p')}{\sum_{q \in P} D(q,q')} + \frac{\rho^2(\sigma+1)}{|P(p)|}$$

Proof. For an arbitrary $Z \in \mathcal{X}^k$ we need to provide a uniform bound for

$$\begin{aligned} \frac{D(p,Z)}{\sum_{q \in P} D(q,Z)} &\leq \frac{\rho D(p,p')}{\sum_{q \in P} D(q,Z)} + \frac{\rho D(p',Z)}{\sum_{q \in P} D(q,Z)} \\ &\leq \frac{\sigma\rho D(p,p')}{\sum_{q \in P} D(q,q')} + \frac{\rho D(p',Z)}{\sum_{q \in P} D(q,Z)} \end{aligned} \tag{1}$$

where the second inequality holds because $\sum_{q \in P} D(q,q') \leq \sigma \mathsf{OPT}(P) \leq \sigma \sum_{q \in P} D(q,Z)$. To

bound the last term, recall that $q' = p'$ for all $q \in P(p)$ so:

$$\begin{aligned} D(p', Z)|P(p)| &= \sum_{q \in P(p)} D(p', Z) = \sum_{q \in P(p)} D(q', Z) \\ &\leq \rho \sum_{q \in P(p)} (D(q', q) + D(q, Z)) \\ &\leq \rho \sum_{q \in P} D(q', q) + \rho \sum_{q \in P(p)} D(q, Z) \\ &\leq \rho\sigma \sum_{q \in P} D(q, Z) + \rho \sum_{q \in P(p)} D(q, Z) \\ &\leq \rho(\sigma + 1) \sum_{q \in P} D(q, Z) \end{aligned}$$

Dividing by $|P(p)| \sum_{q \in P} D(q, Z)$ gives

$$\frac{D(p', Z)}{\sum_{q \in P} D(q, Z)} \leq \frac{\rho(\sigma + 1)}{|P(p)|}$$

Substituting this in (1) yields the desired result. ◀

We will use Lemma 6 to define our upper bound $s'(p)$. An immediate but extremely important consequence of Lemma 6 is that $t' = \sum_{p \in P} s'(p) = \rho\sigma + \rho^2(\sigma + 1)\lambda k$. This can be seen by directly summing the formula given by the lemma.

5.4 Streaming Algorithm

Consider the prefix P_i which is the input after the first i points have arrived. For ease of notation, we write $s'_i(p)$ to refer to $s'_{P_i}(p)$, an upper bound on the sensitivity of p with respect to the first i points. After processing first i points of the stream, Algorithm 1 constructs a set (Q_i, v_i). With probability at least $1 - \delta$, (Q_i, v_i) is a (k, ϵ)-coreset for P_i for every $i \in [n]$. This algorithm will satisfy the claims of Theorem 3.

5.4.1 Overview of the algorithm

The algorithm initializes on Lines 1-9. The main loop of Lines 10-38 processes the stream, accepting one point per iteration. For each iteration, the first step is to process the point with algorithm $\mathcal{A}$ (Line 11) then compute a $(O(1), O(1))$-bicriterion approximation on its output (Line 18). On Line 23 we use this to maintain an upper bound $s'_i(p)$ on the sensitivity of a point p with respect to P_i. In the analysis, let $t'_i = \sum_{\ell=1}^{i} s'_i(p_\ell)$ be the upper bound on the total sensitivity of P_i. We now define $\mathcal{S}_i$, the distribution from which we will draw our sample. Note that $\mathcal{S}_i$ is non-deterministic, since the values of $s'_i(p)$ (and therefore t'_i) depend on the randomness of Algorithm $\mathcal{A}$ as well as any possible randomness used in the bicriterion approximation.

▶ **Definition 7.** *The probability distribution $\mathcal{S}_i$ is supported on P_i and assigns probability $s'_i(p)/t'_i$ to point p.*

By Theorem 5, it suffices to sample $m'_i = \lceil 3t'_i\epsilon^{-2}(\log n \log t'_i + \log(n/\delta)) \rceil$ points i.i.d. from $\mathcal{S}_i$ to construct a coreset for $(P_i, \mathbb{1})$ with probability at least $1 - \delta/n$. On Line 2 we define $t_\circ$ to upper bound the maximal value of t'_i over any set of points in $\mathcal{X}$. Likewise on Line 3 we set $m_\circ = \lceil 3t_\circ\epsilon^{-2}(\log n \log t_\circ + \log(n/\delta)) \rceil$ which is an upper bound on the required sample size.

We maintain $\Theta(k(\log n \log k + \log \frac{n}{\delta}))$ sets M_y, each containing a sample of the stream. We hope that many of these samples contain only a single point, because if so then by Lemma 6 the point follows the distribution $\mathcal{S}_i$. With high probability, at least m'_i sets will be singletons, and so we take their union to construct the coreset (Q_i, v_i).

Memory considerations

Due to memory constraints, we only store the maps g_i, f_i, u_y, z_i, and s'_i for points in $\cup_{y \in Y} M_y$. In the analysis only, we use f_i, z_i, and $s'_i(p)$ for points that have been deleted. This refers to the value that would have been set if we continued executing Lines 21-23 for p.

5.4.2 Proof of correctness

Using Algorithm $\mathcal{A}$ we obtain an $(\alpha, \gamma(\log n + \log \frac{n}{\delta}))$-approximation (B_i, w_i) with the map $\pi_i : B_{i-1} \cup \{p_i\} \to B_i$ which we use to obtain the map $f_i : P_i \to B_i$. Line 23 runs an offline (γ, λ)-approximation algorithm on B_i, and we obtain weighted set (C_i, v_i). The following lemma shows that (C_i, v_i) is a (σ, λ)-approximation for P_i, where $\sigma = \rho\alpha + 2\rho^2\gamma(\alpha + 1)$ as defined on Line 1. The clustering map will be $g_i \circ f_i : P_i \to B_i \to C_i$.

▶ **Lemma 8.** *Assume that algorithm $\mathcal{A}$ has not failed. Then (C_i, w_i^C) with $g_i \circ f_i : P_i \to C_i$ is a (σ, λ)-approximation of P_i.*

Proof. As the context is clear, we drop the subscript i. By Theorem 4, (B, w^B) with $f : P \to B$ is an (α, β)-approximation of P where $\beta = \gamma(\log n + \log \frac{n}{\delta})$. Also, (C, w^c) with $g : B \to C$ is the (γ, λ)-approximation of B. In the following, all sums will be taken over all $p \in P$. The hypotheses state that $\sum D(p, f(p)) \leq \alpha \mathsf{OPT}_k(P)$ and $\sum D(f(p), g(f(p))) \leq \gamma \mathsf{OPT}_k(B)$. Let P^* be an optimal clustering of P, that is $\sum D(p, P^*) = \mathsf{OPT}_k(P)$. Then $\frac{1}{2}\mathsf{OPT}_k(B) \leq \sum D(f(p), P^*) \leq \rho \sum (D(f(p), p) + D(p, P^*)) \leq \rho(\alpha+1)\mathsf{OPT}(P)$. The factor of $\frac{1}{2}$ comes from the fact that $\mathsf{OPT}(B)$ is defined using centers restricted to B (see [16] for details). We now write $\sum D(p, g(f(p))) \leq \rho \sum (D(p, f(p)) + D(f(p), g(f(p)))) \leq (\rho\alpha + 2\rho^2\gamma(\alpha + 1))\mathsf{OPT}_k(P)$ as desired. ◀

To use Lemma 6 to determine $s'_i(p)$, we will compute the cluster sizes $|P_i(p)|$ and estimate the clustering cost $\sum_{q \in P} D(q, q')$ by L_i. We must bound the clustering cost from below because we require $s'_i(p)$ to be an upper-bound of $s_i(p)$.

▶ **Lemma 9.** *Assume that algorithm $\mathcal{A}$ has not failed and that (Q_{i-1}, v_{i-1}) is a (k, ϵ)-coreset for P_{i-1}. Then $z_i(p) \geq s_i(p)$ for every $p \in P_i$.*

Proof. For the first claim, consider Lemma 6 applied with the clustering map $g_i \circ f_i : P_i \to C_i$. We write $p' = g_i \circ f_i(p)$. By Lemma 8, this map is a (σ, λ)-approximation of P_i. Observe that $|P_i(p)|$ (from Lemma 6) is precisely $w_i^C(g_i \circ f_i(p))$ since the weight of a point c is determined by how many points in P_i were clustered to c. By Theorem 4, $L_i \leq \mathsf{OPT}_k(P_i) \leq \mathsf{COST}(P_i, \mathbb{1}, C_i)$. We may then write:

$$\begin{aligned} z_i(r) &= \frac{\rho\sigma D(r, g_i \circ f_i(r))}{L_i} + \frac{\rho^2(\sigma+1)}{w_i^C(g_i \circ f_i(r))} \\ &\geq \frac{\rho\sigma D(r, r')}{\sum_{p \in P_i} D(p, p')} + \frac{\rho^2(\sigma+1)}{|P_i(r)|} \\ &\geq s_i(r) \end{aligned}$$

where the last inequality follows by Lemma 6. ◀

Algorithm 1 Input: parameters $\epsilon, \delta \in (0,1)$ and $n, k \in \mathbb{N}$. A stream of n points in a ρ-metric space. Notes: $\mathcal{A}(\cdot,\cdot,\cdot)$ denotes the blackbox algorithm from Theorem 4 along with its universal constants α and γ. Line 18 uses any RAM-model $(O(1), O(1))$-approximation such as [3].

1: $\sigma \leftarrow \rho\alpha + 2\rho^2\gamma(\alpha+1)$
2: $t_\circ \leftarrow \rho\sigma\alpha + \rho^2(\sigma+1)\lambda k$
3: $m_\circ \leftarrow \lceil 3t_\circ\epsilon^{-2}(\log n \log t_\circ + \log(n/\delta))\rceil$
4: $Q_0 \leftarrow \emptyset$
5: Initialize $\mathcal{A}(k, n, \delta/n)$
6: $Y \leftarrow \{1, \ldots, 8m_\circ\}$
7: **for** each $y \in Y$ **do**
8: $\quad M_y \leftarrow \emptyset$
9: **end for**
10: **for** the next point p_i from the stream **do**
11: $\quad (B_i, w_i^B, \pi_i, L_i) \leftarrow$ update $\mathcal{A}$ with point p_i
12: $\quad$ **for** each $y \in Y$ **do**
13: $\quad\quad u_y(p_i) \leftarrow$ uniform random number from $[0,1)$
14: $\quad\quad M_y \leftarrow M_y \cup \{p_i\}$
15: $\quad$ **end for**
16: $\quad f_{i-1}(p_i) \leftarrow p_i$
17: $\quad s'_{i-1}(p_i) \leftarrow 1$
18: $\quad (C_i, w_i^C, g_i) \leftarrow (\gamma,\lambda)$-approximation of (B_i, w_i^B)
19: $\quad R \leftarrow \cup_y M_y$
20: $\quad$ **for** each $r \in R$ **do**
21: $\quad\quad f_i(r) \leftarrow \pi_i \circ f_{i-1}(r)$
22: $\quad\quad z_i(r) \leftarrow \frac{\rho\sigma D(r, g_i \circ f_i(r))}{L_i} + \frac{\rho^2(\sigma+1)}{w_i^C(g_i \circ f_i(r))}$
23: $\quad\quad s'_i(r) \leftarrow \min(s'_{i-1}(r), z_i(r))$
24: $\quad$ **end for**
25: $\quad$ **for** each $y \in Y$ **do**
26: $\quad\quad$ **for** each $q \in M_y$ **do**
27: $\quad\quad\quad$ **if** $u_y(q) > \frac{s'_i(q)}{s'_i(q)+t_\circ}$ **then**
28: $\quad\quad\quad\quad$ Delete q from M_y
29: $\quad\quad\quad$ **end if**
30: $\quad\quad$ **end for**
31: $\quad$ **end for**
32: $\quad \Gamma_i \leftarrow \{y \in Y : |M_y| = 1\}$
33: $\quad Q_i \leftarrow \cup_{y \in \Gamma_i} M_y$
34: $\quad$ **for** each $q \in Q_i$ **do**
35: $\quad\quad v_i(q) \leftarrow (|\Gamma_i| s'_i(q))^{-1}$
36: $\quad$ **end for**
37: $\quad$ **return** (Q_i, v_i)
38: **end for**

The implication is that our upper bound on sensitivity is valid, as we now prove formally:

▶ **Lemma 10.** *Assume that algorithm $\mathcal{A}$ has not failed and that (Q_{i-1}, v_{i-1}) is a (k,ϵ)-coreset for P_{i-1}. Then $s'_i(p) \geq s_i(p)$ for every $p \in P_i$.*

Proof. Fix a point $p \in P_i$. We see from Line 23 that $s'_i(p) = z_j(p)$ for some $j \leq i$. Lemma 11 shows that $z_j(p) \geq s_j(p)$. Observe directly from the definition of sensitivity that $s_i(p) \leq s_j(p)$ for any $j \leq i$. Combining these shows that $s'_i(p) = z_j(p) \geq s_j(p) \geq s_i(p)$. We conclude that $s'_i(p) \geq s_i(p)$ for all $p \in P_i$. ◀

For each point p, the value of $s'_i(p)$ is non-increasing in i. This is because $s'_i(p)$ is defined as the minimum of itself and a new value on Line 23. It follows from the monotonicity of $f(x) = \frac{x}{x+1}$ that $\frac{s'_i(r)}{s'_i(r)+t_\circ}$ is also non-increasing in i. Therefore once the deletion condition on Line 27 becomes satisfied, it remains satisfied forever. This is essential because after deleting a point from memory, it can never be retrieved again. We can characterize $M_y^{(i)}$ without reference to the streaming setting: $M_y^{(i)} = \{p \in P_i : u_y(p) \leq \frac{s'_i(p)}{s'_i(p)+t_\circ}\}$. This has the important implication that $Pr(p \in M_y^{(i)}) = \frac{s'_i(p)}{s'_i(p)+t_\circ}$.

▶ **Lemma 11.** *Assume that algorithm $\mathcal{A}$ has not failed and that (Q_{i-1}, v_{i-1}) is a (k,ϵ)-coreset for P_{i-1}. Then $\sum_{p\in P_i} z_i(p) < t_\circ$.*

Proof. The value of $t_\circ$ is defined on Line 2 as $\rho\sigma\alpha + \rho^2(\sigma+1)\lambda k$.

$$\begin{aligned}\sum_{p\in P_i} z_i(p) &= \sum_{p\in P_i} \frac{\rho\sigma D(p, g_i \circ f_i(p))}{L_i} + \frac{\rho^2(\sigma+1)}{w_i^C(g_i \circ f_i(p))} \\ &\leq \frac{\rho\sigma\alpha L_i}{L_i} + \rho^2(\sigma+1)\lambda k \\ &\leq \rho\sigma\alpha + \rho^2(\sigma+1)\lambda k \\ &= t_\circ\end{aligned}$$

where the first inequality comes from Lemma 8 and Theorem 4. Note that we have summed the second term using the fact that a center $c \in C_i$ with weight $w_i^C(c)$ has exactly $w_i^C(c)$ points of P_i clustered to it. Therefore we may re-write the sum:

$$\sum_{p\in P_i} \frac{1}{w_i^C(g_i \circ f_i(p))} = \sum_{c\in C_i} 1 = \lambda k$$ ◀

To construct a coreset by sampling from $\mathcal{S}_i$, the algorithm take the union of those M_y for $y \in Y$ that are singletons. Any set M_y that is either empty or contains more than one point will be ignored, but still kept track of since it may later become a singleton. We now show that if a sample M_y contains a single point, then it follows the distribution $\mathcal{S}_i$. Let $M_y^{(i)}$ denote the state of M_y after the prefix P_i has been processed, and let $Pr(A : B)$ denote the probability of event A conditioned on event B.

▶ **Lemma 12.** *For any $p \in P_i$, $Pr(M_y^{(i)} = \{p\} : |M_y^{(i)}| = 1) = s'_i(p)/\sum_{\ell=1}^{i} s'_i(p_\ell)$.*

Proof. Define $\alpha_\ell = \frac{s'_i(p_\ell)}{s'_i(p_\ell)+t_\circ}$ and $\Psi = \prod_{\ell=1}^{i}(1-\alpha_\ell)$. For $z \in [i]$ let E_z denote the event that $M_y^{(i)} = \{p_z\}$. The probability that the sampler contains only p_z means that it failed to sample all p_ℓ for $\ell \neq z$, meaning that $Pr(E_z) = \alpha_z\Psi/(1-\alpha_z) = s'_i(p_z)\Psi/t_\circ$. The result is obtained since:

$$\begin{aligned} Pr(E_z : |M_y^{(i)}| = 1) &= Pr(E_z)/Pr(|M_y^{(i)}| = 1) \\ &= Pr(E_z)/Pr(\cup_{\ell=1}^{i} E_\ell) \\ &= Pr(E_z)/\sum_{\ell=1}^{i} Pr(E_\ell) \\ &= s_i'(p_z)/\sum_{\ell=1}^{i} s_i'(p_\ell) \end{aligned}$$ ◀

The significance of Lemma 12 is tantamount. If a sample M_y contains a singleton, then it is equivalent to a random draw from $\mathcal{S}_i$. Therefore if at least m_i' samplers contain a singleton, taking their union gives us an i.i.d. sample of at least m_i' points from $\mathcal{S}_i$. This is precisely what we need to construct a coreset using Theorem 5. However, it remains to show that we will have enough singleton samplers with high probability. The next lemma begins to establish this fact.

▶ **Lemma 13.** *Fix any $y \in Y$ and $i \in [n]$. Then $P(|M_y^{(i)}| = 1) \geq \frac{t'}{4t_\circ}$.*

Proof. Let p_i be the most recently arrived point, and define $\alpha_\ell = \frac{s_i'(p_\ell)}{s_i'(p_\ell)+t_\circ}$. Observe that $s_i'(p_\ell) \geq 0$ implies $\alpha_\ell \leq s_i'(p_\ell)/t_\circ$. It follows from Line 27 that $Pr(p_\ell \in M_y) = \alpha_\ell \leq s_i'(p_\ell)/t_\circ$. The expected value of $|M_y|$ is therefore at most $\sum_{\ell=1}^{i} \frac{1}{t_\circ} s_i'(p_\ell) = \frac{t'}{t_\circ}$. Markov's inequality yields $Pr(|M_y| \geq 2) = Pr(|M_y| \geq \frac{2t_\circ}{t'}\frac{t'}{t_\circ}) \leq \frac{t'}{2t_\circ} \leq \frac{1}{2}$.

$$\begin{aligned} P(|M_y| = 1) &= \sum_{\ell=1}^{i} P(|M_y| = \{p_\ell\}) \\ &= \sum_{\ell=1}^{i} \alpha_\ell \prod_{z \neq \ell} (1 - \alpha_z) \\ &= \sum_{\ell=1}^{i} \frac{\alpha_\ell}{1-\alpha_\ell} \prod_{z=1}^{i} (1 - \alpha_z) \\ &= \frac{1}{t_\circ} \left(\sum_{\ell=1}^{i} s_i'(p_\ell) \right) \left(\prod_{z=1}^{i} (1 - \alpha_z) \right) \\ &= \frac{t'}{t_\circ} Pr(|M_y| = 0) \end{aligned}$$

We know that $Pr(|M_y| = 0) + Pr(|M_y| = 1) + Pr(|M_y| \geq 2) = 1$, so substitution gives $(\frac{t_\circ}{t'} + 1)Pr(|M_y| = 1) + \frac{1}{2} \geq 1$. Rearranging this, we obtain $Pr(|M_y| = 1) \geq \frac{1}{2}\frac{1}{t_\circ/t'+1} \geq \frac{t'}{4t_\circ}$ as desired. ◀

Now that we have provided a lower bound on the probability of any sample M_y being a singleton, we move on to lower bound the probability of at least m_i' of the samples $\{M_y\}_{y \in Y}$ being singletons. Observe that the $\{M_y\}$ are entirely independent. We use a Chernoff bound to lower bound the size of Γ_i, defined on Line 32, which is the set of all singleton samples.

▶ **Lemma 14.** *$|\Gamma_i| \geq m_i'$ with probability at least $1 - \delta/n$.*

Proof. We see directly from Line 32 that $|\Gamma_i|$ is a sum of $|Y|$ independent Bernoulli trials, each which succeeds with probability at least $\frac{t'}{4t_\circ}$ by Lemma 13. By a Chernoff bound, $Pr(|S| \leq \frac{1}{2}|Y|\frac{t'}{4t_\circ}) \leq e^{-|Y|t'/32t_\circ}$. We note that $|Y|t'/32t_\circ = 8m_\circ t'/32t_\circ \geq \ln(n/\delta)$. Plugging this into the Chernoff bound yields that $|S| \geq m_\circ t'/t_\circ$ with probability at least $1 - \delta/n$. We conclude by noting that $m_\circ t'/t_\circ = 3t'(\log n \log t_\circ + \log \frac{\delta}{n}) \geq m'$. ◀

We now have all the tools to proceed with the proof of Theorem 1. We begin with the space requirement, then prove correctness.

▶ **Lemma 15.** *After processing P_i, Algorithm 1 stores $O(\epsilon^{-2}k(\log k \log n + \log \frac{n}{\delta}))$ points with probability at least $1 - \delta/n$.*

Proof. First note that $O(\log n + \log \frac{n}{\delta}) = O(\log n + \log \frac{1}{\delta})$. By Theorem 4 we know that $\mathcal{A}$ stores $O(k(\log n + \log \frac{n}{\delta}))$ points deterministically. In addition to the blackbox $\mathcal{A}$, the algorithm stores the sets M_y along with a constant amount of satellite data per point.

We showed that $E[|M_y|] \leq t'/t_\circ$ in the proof of Lemma 13. Directly from Lemma 13, we lower bound $E[|M_y|] \geq Pr(|M_y| = 1) \geq t'/4t_\circ$. Combining these upper and lower bounds permits us to write $2m_\circ t'/t_\circ \leq E[\sum_{y \in Y} |M_y|] \leq 8m_\circ t'/t_\circ$.

A Chernoff bound can be applied for a high-probability guarantee. The random variable $X = \sum_{y \in Y} |M_y|$ is a sum of $|Y|i$ independent Bernoulli trials, each event being $p_\ell \in M_y$ for some $1 \leq \ell \leq i$ and $y \in Y$. We have the Chernoff bound that $\text{Prob}[X \geq (1+\eta)\mu] \leq e^{-\eta^2\mu/(2+\eta)}$ for any $\eta \geq 1$ where $\mu = E[X]$. Using $\eta = 1$, this yields that $X < 16m_\circ$ with probability at least $1 - e^{-2m_\circ t'/3t_\circ} < e^{-2m'/3} < \delta/n$. ◀

We now prove correctness, using the following lemma as a tool for our final claim.

▶ **Lemma 16.** *Assume that algorithm $\mathcal{A}$ has not failed and that (Q_{i-1}, v_{i-1}) is a (k, ϵ)-coreset for P_{i-1}. Then (Q_i, v_i) is a (k, ϵ) coreset of $(P_i, \mathbb{1})$ with probability at least $1 - 3\delta/n$.*

Proof. Lemma 10 shows that $\mathcal{S}_i$ meets the criteria of Theorem 5 with probability at least $1 - \delta/n$. Lemmas 12 and 14 show that Q_i is an i.i.d. sample of at least m'_i points from $\mathcal{S}_i$ with probability at least $1 - \delta/n$. By Theorem 5, conditioning on the success of the previous two statements, (Q_i, v_i) is a (k, ϵ)-coreset for $(P_i, \mathbb{1})$ with probability at least $1 - \delta/n$. We arrive at the desired result by applying the union bound. ◀

We complete the proof of Theorem 3 by induction over each prefix P_i on the following events: (1) the success of $\mathcal{A}$; (2) that (Q_i, v_i) is a (k, ϵ) coreset of $(P_i, \mathbb{1})$; and (3) the storage requirement holding from Lemma 15. The base cases hold trivially.

By Theorem 4, Algorithm $\mathcal{A}(n, k, \delta/n)$ will succeed on P_i with probability at least $1 - \delta/n$. By Lemma 15, the space requirement is maintained with probability at least $1 - \delta/n$. By Lemma 16, (Q_i, v_i) is a (k, ϵ) coreset of $(P_i, \mathbb{1})$ with probability at least $1 - 3\delta/n$. Combining these pieces, we succeed inductively after processing a single point with probability at least $1 - 5\delta/n$. Therefore we succeed at every step of the entire stream with probability at least $1 - 5\delta$. Theorem 3 follows by scaling δ appropriately.

References

1 Marcel R. Ackermann, Marcus Märtens, Christoph Raupach, Kamil Swierkot, Christiane Lammersen, and Christian Sohler. StreamKM++: A Clustering Algorithm for Data Streams. *J. Exp. Algorithmics*, 17:2.4:2.1–2.4:2.30, May 2012. doi:10.1145/2133803.2184450.

2 Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A Framework for Projected Clustering of High Dimensional Data Streams. In Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer, editors, *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*, pages 852–863. Morgan Kaufmann, 2004. URL: http://www.vldb.org/conf/2004/RS21P7.PDF, doi:10.1016/B978-012088469-8.50075-9.

3 Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local Search Heuristic for K-median and Facility Location Problems. In *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing*, STOC '01, pages 21–29, New York, NY, USA, 2001. ACM. doi:10.1145/380752.380755.

4 Pranjal Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali Kemal Sinop. The Hardness of Approximation of Euclidean k-Means. In *31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands*, pages 754–767, 2015.

5 Jon Louis Bentley and James B Saxe. Decomposable searching problems I. Static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980.

6 Vladimir Braverman, Dan Feldman, and Harry Lang. New Frameworks for Offline and Streaming Coreset Constructions. *CoRR*, abs/1612.00889, 2016. arXiv:1612.00889.

7 Vladimir Braverman, Adam Meyerson, Rafail Ostrovsky, Alan Roytman, Michael Shindler, and Brian Tagiku. Streaming K-means on Well-clusterable Data. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 26–40. SIAM, 2011. URL: http://dl.acm.org/citation.cfm?id=2133036.2133039.

8 Jaroslaw Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An Improved Approximation for K-median, and Positive Correlation in Budgeted Optimization. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 737–756. SIAM, 2015. URL: http://dl.acm.org/citation.cfm?id=2722129.2722179.

9 Moses Charikar, Liadan O'Callaghan, and Rina Panigrahy. Better Streaming Algorithms for Clustering Problems. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pages 30–39, New York, NY, USA, 2003. ACM. doi:10.1145/780542.780548.

10 Ke Chen. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM Journal on Computing*, 39(3):923–947, 2009.

11 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, August 2009. doi:10.1137/070699007.

12 Kenneth L. Clarkson and David P. Woodruff. Sketching for M-estimators: A Unified Approach to Robust Regression. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 921–939, Philadelphia, PA, USA, 2015. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=2722129.2722192.

13 Dan Feldman and Michael Langberg. A Unified Framework for Approximating and Clustering Data. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC '11, pages 569–578, New York, NY, USA, 2011. ACM. doi:10.1145/1993636.1993712.

14 Dan Feldman and Leonard J Schulman. Data reduction for weighted and outlier-resistant clustering. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1343–1354. SIAM, 2012.

15 G. Frahling and C. Sohler. Coresets in dynamic geometric data streams. In *Proc. 37th Annu. ACM Symp. on Theory of Computing (STOC)*, pages 209–217, 2005.

16 Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering Data Streams: Theory and Practice. *IEEE Trans. on Knowl. and Data Eng.*, 15(3):515–528, March 2003. doi:10.1109/TKDE.2003.1198387.

17 Frank Hampel, Christian Hennig, and Elvezio Ronchetti. A smoothing principle for the Huber and other location M-estimators. *Computational Statistics & Data Analysis*, 55(1):324–337, 2011. doi:10.1016/j.csda.2010.05.001.

18 S. Har-Peled and A. Kushal. Smaller coresets for k-median and k-means clustering. *Discrete Comput. Geom.*, 37(1):3–19, 2007. doi:10.1007/s00454-006-1271-x.

19 S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In *STOC*, 2004.

20 P. J. Huber. Robust Statistics. *Wiley*, 1981.

21 Harry Lang. Online Facility Location Against a t-bounded Adversary. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 1002–1014, Philadelphia, PA, USA, 2018. Society for Industrial and Applied Mathematics.

22 Z. Zhang. M-estimators. *http://research.microsoft.com/en-us/um/people/zhang/INRIA/Publis/Tutorial-Estim/node20.html*, [accessed July 2011].