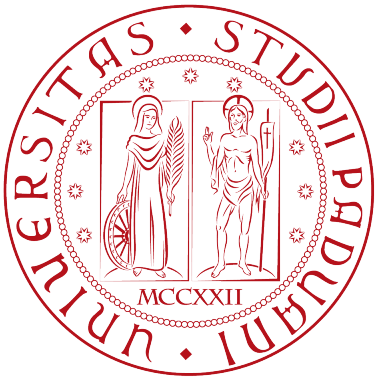


Università degli Studi di Padova
Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Magistrale
in Ingegneria Informatica

Streaming algorithms for center-based clustering in general metrics

Relatori
Ch.mo Prof. Andrea Pietracaprina
Ch.mo Prof. Geppino Pucci

Laureando
Luca Badin
1242060

---

12 Dicembre 2022
Anno Accademico 2022-2023

Abstract

Center-based clustering is an important yet computationally difficult primitive in the realm of unsupervised learning and data analysis. We specifically focus on the k-median clustering problem in general metric spaces, in which one seeks to find a set of k centers, so to minimise the sum of distances from each point in the dataset to its closest center.

In this thesis, we present and analyze efficient techniques to deal with the k-median clustering problem in the streaming setting – where the dataset is presented one point at a time and not accessible in its entirety – by leveraging the dimensionality of the dataset's underlying metric space.

Contents

Chapter 1

Introduction

Clustering is a key primitive in unsupervised learning and data analysis, with many applications throughout different fields, such as bioinformatics, medicine, digital image processing, pattern recognition, network analysis, and so on.

An important class of clustering techniques is center-based clustering, in which we seek to partition an input dataset into clusters, in such a way that similarity between items in the same cluster is defined by closeness to that cluster's center, according to some distance measure.

The related optimization problem fixes a parameter k, and looks for k such centers so to minimize the distance of each point from its center (or the squared distance, depending on the specific type of clustering). This is known to be NP-hard; as such, it is generally only feasible to look for approximate solutions.

In the realm of big data, finding efficient ways to process large datasets becomes of foremost importance, as the amount of information to be analyzed blows up.
The streaming model has often been adopted in settings where being able to store all of the dataset in local memory is not a reasonable assumption, and the only feasible way to process it is by make a few sequential passes (ideally, only a single pass) through the data, using a limited amount of space.

1.1 Previous work

Center based clustering is a well-researched problem. As mentioned, we focus on general metric k-median specifically, which has known constant approximation algorithms. Byrka, Pensyl, Rybicki et al.'s [5] offline 2.61-approximation algorithm is the current best performer, using dependent randomized rounding [9] and improving on Li and Svensson's [10]'s novel $(1 + \sqrt{3} + \varepsilon)$-approximation.

Cohen-Addad, Gupta, Hu et al. [7] recently (2022) presented a local search algorithm that yields a $(2.863 + \varepsilon)$-approximation, the first of its kind to improve on Arya et al.'s (2001) [2] local search $(3 + \varepsilon)$-approximation and offering potentially promising future

improvements.

In the realm of streaming approximation algorithms for k-median, Braverman, Lang, Levin and Rudoy [3] proposed a streaming algorithm for metric k-median clustering that returns a coreset of size $O(\varepsilon^{-3}k \log n)$. This coreset can be clustered with Byrka et al.'s [5] offline 2.61-approximation algorithm, yielding a solution that is a $(17.66 + \varepsilon)$-approximation with regards to the optimum. This ratio is shown to be essentially optimal, if sticking to space that is polylogarithmic in n.

Their contribution seeks to strike a balance between facility location based algorithms such as Charikar, O'Callaghan and Panigrahy's trend-setting PLS algorithm [6], which typically have better space complexity, and coreset based algorithms such as Feldman and Langberg's [8], which can be arbitrarily precise at the cost of significant space penalties.

1.2 Summary of contributions

This thesis explores various ways of building a coreset for an input instance of the k-median clustering problem in the streaming setting - a coreset being a weighted set representing a "summary" of the input dataset X that is much smaller than X itself, but is sufficiently informative to lead to a good approximate result for the whole instance.

We will adopt the PLS+ algorithm for streaming coreset construction due to Braverman, Meyerson, Ostrovsky et al. [4]. This algorithm builds on the parallels between k-median clustering and the facility location problem in its online variety.
PLS+ works in phases, maintaining increasingly more accurate lower bounds to the optimum clustering cost; it clusters points from the stream in facilities, until we pay too much in service cost or we open too many facilities, at which point we improve our bounds and start over, re-clustering all former facilities as weighted points before moving on to reading more unread points.

We shall define a two pass algorithm that makes use of the `StreamingCWB` primitive due to Mazzetto, Pietracaprina and Pucci [11] to refine PLS+'s output coreset, obtaining a new coreset of higher quality, paying a price through higher cardinality.
Crucially, though, this penalty is limited if the doubling dimension D of the dataset is not too big; we will formally outline the concept of doubling dimension later, but it can intuitively be described as a generalization of Euclidean dimensionality to a general metric space.
This two pass algorithm is oblivious to D, is tunable via a parameter $0 < \varepsilon < 1$ to set a space-accuracy tradeoff, and yields an ε-bounded coreset.

Lastly, we shall leverage again the notion of doubling dimension to define another algorithm which is semi-oblivious to the dataset's dimensionality, and only requires one

pass over X using PLS+. This one phase algorithm requires a suitably large parameter M representing how much space we wish the algorithm to use; from this, we extract a higher center set cardinality $k' > k$ and run PLS+ on k-median instance (X, k').

By leveraging the fact that, in metric spaces with bounded doubling dimension, we can state that $\nu^{\star}_{(X,k')} \leq \eta \cdot \nu^{\star}_{(X,k)}$ for some parameter $0 < \eta < 1$ that depends on the choice of k', this results in PLS+'s output F being an $(\alpha \cdot \eta)$-bounded coreset, improving our result by a factor η with respect to running PLS+ on the original instance.

1.3 Structure of this thesis

This thesis is organized as follows. Chapter 2 lays down some preliminary definitions and properties that we will use in the rest of the thesis.

Sections 3.1, 3.2 and 3.3 present and analyze the building blocks to our coreset construction algorithms, which are formally outlined in Sections 3.4 and 3.5.

Chapter 4 presents the implementation of our strategies, and the following experimental evaluation. Lastly, Chapter 5 closes this thesis with a summary and some final remarks.

Chapter 2

Preliminary notions

A **metric space** $\mathcal{M}$ is a set X together with a distance function $\delta(\cdot, \cdot)$ which satisfies the following properties for any pair $x, y \in X$:

- $\delta(x, y) > 0$, unless $x = y$, in which case $\delta(x, y) = 0$;
- it is symmetric: $\delta(x, y) = \delta(y, x)$
- it satisfies the triangle inequality: $\delta(x, y) + \delta(y, z) \geq \delta(x, z)$

In **general metric k-median clustering**, we are given a finite set X of points in some metric space $\mathcal{M}$, and the cost function $\delta(x, y)$ representing the distance between two points $x, y \in \mathcal{M}$.
The goal is to select a set of points $K \subset \mathcal{M}$, with $|K| = k$, so to minimize the cost function

$$\nu_X(K) = \sum_{x \in X} \min_{y \in K} \delta(x, y)$$

Throughout this thesis, we will indicate with $\nu^\star_{X,k}$ the cost of clustering X with its optimum set of centers for the k-median clustering problem - that is, the set of centers K for which the cost function takes the smallest value:

$$\nu^\star_{X,k} = \min_{K \subset \mathcal{M}: |K| = k} \sum_{x \in X} \delta(x, K)$$

(Where unambiguously clear from the context, we will drop the subscript from our notation)

The k-median clustering problem can be seen as a specific case of the uncapacitated facility location problem. In **facility location** problems, we are given a metric space, a facility cost f and a set of points X, which in general may be weighted (if so, we may equivalently consider that a point of weight $w(x)$ is represented by $w(x)$ consecutive points of unit weight).

Every point pays a service cost to be connected to the nearest open facility, which is equal to its distance δ from that facility. Alternatively, opening each new facility requires us to pay the facility cost f (which we suppose to be uniform).

The objective is to determine a set of facilities F to open so to minimize the total cost,

$$f|F| + \sum_{x \in X} \delta(x, F)$$

A crucial notion for our work is the **doubling dimension** of a dataset's underlying metric space. We can think of the doubling dimension as the generalization of Euclidean dimensionality to general metric spaces.

Definition (Doubling dimension)**.** *The doubling dimension of a metric space $\mathcal{M}$ is the smallest integer D, such that a ball of any radius r centered at any point $x \in \mathcal{M}$ can be covered by at most 2^D balls of radius $r/2$ centered at some points of $\mathcal{M}$.*

We operate in the **streaming setting**, in which set X is large enough that it is impractical to store it in its entirety and randomly access its items; the only feasible way to process X is to make a single pass through the data, or a few passes at most, using space typically polylogarithmic in $|X|$. The items arrive - and are processed - sequentially, with no prior knowledge of what comes next.

A **coreset** is a weighted set representing a "summary" of the input dataset X that is much smaller than X itself, but is sufficiently informative to lead to a good approximate result for the whole instance. We adopt the following, well-estabilished definition of coreset:

Definition (ε-bounded coreset)**.** *A set of points C is an ε-bounded coreset of a k-median instance (X, k) if there exists a map $\tau : X \to C$ such that*

$$\sum_{x \in X} \delta(x, \tau(x)) \leq \varepsilon \cdot \nu^\star$$

and, for any $c \in C$,

$$w(c) = |\{x \in X : \tau(c) = x\}|$$

We say that C is weighted according to τ.

Chapter 3

Streaming algorithms for k-median clustering

In this chapter, we will analyze and present the algorithms forming the building blocks to our coreset building strategies. Section 3.1 describes the Online Facility Location algorithm, which lays the foundations to tackling the k-median clustering problem in the streaming setting by drawing on the similarities between k-median and facility location as optimization problems. 3.2 3.2 presents PLS+, a streaming algorithm for coreset construction that builds upon OFL by maintaining increasingly accurate estimates of the optimum cost. Section 3.3 analyzes the CoverWithBalls primitive for coreset quality refinement, adapted to a streaming setting; and lastly, Sections 3.4 and 3.5 respectively present our two-pass dimensionality-oblivious coreset construction strategy and our one-pass semi-oblivious coreset construction strategy.

3.1 Online Facility Location

Before approaching PLS+ itself, it will be convenient to review how PLS+ approaches k-median in the streaming setting. Specifically, the problem is rephrased as a facility location problem, in its online (streaming) variant, resulting in the Online Facility Location (OFL) algorithm, which is used by PLS+ as a subroutine.

3.1.1 Problem definition

As defined previously, facility location problems require a metric space, a **facility cost** f and a set of points X; each point $x \in X$ can either be connected to the nearest open facility, paying a **service cost** equal to its distance $\delta(x, F)$ from that facility; or alternatively, a facility can be opened at that point, paying the specified facility cost f.

The objective is to determine a set of facilities F to open so to minimize the total cost,

$$f|F| + \sum_{x \in X} \delta(x, F)$$

From now on we will denote by $c = \sum_{x \in X} \delta(x, F)$ the service cost component of the

overall cost. Indeed, observe that c is analogous to the k-median cost function ν: the sum of all distances between every point $x \in X$ and the closest facility (or center). On the other hand, k-median's constraint that $|K| = k$ is replaced by the facility cost component of the cost function, $f|F|$. Intuitively, then, c is the component of facility location's overall cost we are really interested in, in our goal to analyze a streaming algorithm for k-median.

The online variant of facility location (OFL) requires that each new point is assigned to a facility upon arrival, without knowledge of the number or value of future points, and without modifying the assignment of prior points.

3.1.2 Pseudocode and description

Meyerson's OFL algorithm works as follows: once a new point arrives, we calculate the distance $d = \delta(x, F)$ to the closest open facility. With probability $\min\{d/f, 1\}$, a new facility is opened at this point, paying facility cost f; otherwise, the point is connected to the closest open facility, paying service cost d. n other words, if $f \leq d$, we will certainly open a new facility; otherwise, the probability of opening a new facility is proportional to the ratio d/f.

The overall cost of facility location is $c + f|F|$, including both the service cost component and the facility cost component. As mentioned in the previous subsection, though, we are only interested in c, and we will call c itself the cost of OFL's returned solution F.

Algorithm 1 Online Facility Location. f is the user-supplied facility cost.

```
1: c ← 0                                              ▷ (Service) cost
2: F ← ∅                                              ▷ Set of facilities
3: while there are points still in the stream do
4:     x ← next point
5:     p ← RANDOM([0, 1])
6:     if p ≤ min{1, w(x) · δ(x, F)/f} then
7:         F ← F ∪ x                                  ▷ Open new facility
8:     else
9:         c ← c + w(x) · δ(x, F)                     ▷ Pay service cost
10:        y ← arg min_{y∈F} δ(x, y)
11:        w(y) ← w(y) + w(x)                         ▷ Increment facility weight
12:    end if
13: end while
14: return F
```

3.1.3 Analysis

We shall see that, by choosing the facility cost f appropriately, the cost c is a constant approximation of the optimum k-clustering cost on X, $\nu^\star$, and the number of facilities opened ($|F|$) is "not too much bigger" than k.

Theorem 1 (Online Facility Location)**.** *Suppose the OFL algorithm is run with*

$$f = \frac{L}{k(1+\log n)}, \qquad L < \nu^\star$$

where L is a lower bound on the cost to cluster X with its optimum k-median clustering solution, and n is the total weight of all points in the stream.

Then, with high probability, it holds that

$$c \leq \left(3 + \frac{2e}{e-1}\right)\nu^\star, \qquad |F| \leq 7k(1+\log n)\frac{\nu^\star}{L}$$

Proof. For $1 \leq i \leq k$, define:

- $k_i^\star$, an optimum center;
- $K_i^\star$, an optimum cluster (assignment of points to $k_i^\star$);
- $\delta_p^\star$, the optimum service cost for point p (its distance from its optimum center);
- $C_i^\star = \sum_{p \in K_i^\star} \delta_p^\star$, the total service cost for points in optimum cluster $K_i^\star$;

Divide cluster $K_i^\star$ into regions S_i^j, $j \geq 1$, such that the cardinality of region S_i^j is

$$|S_i^j| = |K_i^\star|/2^j$$

and so that each point in S_i^j is closer to $k_i^\star$ than any point in S_i^{j+1} - that is,

$$\forall p \in S_i^j, \quad \forall q \in S_i^{j+1}, \quad \delta_p^\star \leq \delta_q^\star$$

Lastly, let C_i^j be the total service cost of region S_i^j. It holds that $\sum_i C_i^j = C_i^\star$.

We shall now bound the service cost c of the returned solution by separately considering the service cost paid *before* (c') and *after* (c'') the first facility is opened in a region, starting with the latter.

In each region S_i^j, we may eventually open a facility at some point $q \in S_i^j$. Once we do so, subsequent points p must have service cost at most $\delta_p^\star + \delta_q^\star$, by the triangle inequality. Additionally, since $q \in S_i^j$ and all points in S_i^j have smaller optimum service cost than any point in S_i^{j+1},

$$\delta_q^\star \leq C_i^{j+1}/|S_i^{j+1}|$$

that is, q must be closer to its optimum center than the average point in the next outermost region.

By these two facts, summing over all points $p \in S_i^j$, it holds for a single region that

$$\begin{aligned}\sum_{p \in S_i^j} \left(\delta_p^\star + \delta_q^\star\right) &\leq \sum_{p \in S_i^j} \delta_p^\star + \sum_{p \in S_i^j} \frac{C_i^{j+1}}{|S_i^{j+1}|} \\ &\leq C_i^j + |S_i^j| \frac{C_i^{j+1}}{|S_i^{j+1}|} \\ &\leq C_i^j + 2C_i^{j+1}\end{aligned}$$

Which, summed over all regions, gives a deterministic guarantee that, subsequent to the first facility opened in all regions, the service cost is at most three times the optimum, $3C_i^\star$, which holds for all clusters, and therefore $c'' \leq 3\nu^\star$.

Let us now bound c', the cost paid *before* the first facility is opened in each region. If many points arrive in the same region, we will eventually open a facility there, but before that happens for each point we will pay some service cost bounded by f.
First, observe that each of the k clusters has $\log n + 1$ regions: region $j = \log n + 1$ has one point, and all further regions must be empty.

Braverman, Meyerson et al. showed that the probability that, given x regions without an open facility, the service cost due to points arriving prior to the first facility being opened is at least yf, is

$$P[x, y] \leq \exp\left(x - y\frac{e-1}{e}\right)$$

Observe it is immediate for $x = 0$ and very small values of y ($y \leq x\frac{e}{e-1}$). The general case can be proven by induction, as follows.

Suppose that x is the smallest value where the inequality can be violated, and y the smallest value where it can be violated for this x. Thus, $P[x, y] > \exp\left(x - y\frac{e-1}{e}\right)$
Let the first request in one of these x regions without a facility yields a service cost of $d > 0$. Then, the probability is

$$P[x, y] = \frac{d}{f} P\left[x - 1, y\right] + \left(1 - \frac{d}{f}\right) P\left[x, y - \frac{d}{f}\right]$$

where the first term corresponds to opening a facility (thus the number of regions without a facility decreases), and the second term corresponds to connecting to a facility in some other region, paying the service cost.

Substituting in our hypothesis the fact t-hat $P[x,y] > \exp\left(x - y\frac{e-1}{e}\right)$, we obtain

$$\begin{aligned} e^{x-y\frac{e-1}{e}} &< P[x,y] \\ &< \frac{d}{f}e^{\left(x-1-y\frac{e-1}{e}\right)} + \left(1-\frac{d}{f}\right)e^{\left(x-\left(y-\frac{d}{f}\right)\frac{e-1}{e}\right)} \end{aligned}$$

where normalising by the left hand side expression leaves us with a contradiction, $1 < \frac{d}{ef} + \left(1-\frac{d}{f}\right)e^{\frac{d}{f}\frac{e-1}{e}}$. The statement follows.

With this estabilished, setting the number of regions to be $x = k(1+\log n)$ (as just discussed) and

$$y = \frac{2e}{e-1}k(1+\log n)$$

so that the math works out, yields

$$P[c' \geq yf] \leq \exp\left(k(1+\log n) - \frac{2e}{e-1}k(1+\log n)\frac{e-1}{e}\right) = \exp\left(-k(1+\log n)\right) \leq \frac{1}{2n}$$

We therefore get the desired high probability bound on service cost before the first facility opens by substituting in f and y:

$$c' = yf = \frac{2e}{e-1}k(1+\log n)f = \frac{2e}{e-1}L \leq \frac{2e}{e-1}\nu^\star$$

Which, paired with the previous deterministic bound on service cost after the first facility has opened, c'', yields the final result:

$$c \leq c' + c'' = \left(3 + \frac{2e}{e-1}\right)\nu^\star$$

Let us now prove the bound on the facility count $|F|$. As discussed before, there's a deterministic guarantee that the first facility in each region gives us a total of $k(1+\log n)$ facilities; we must get a bound for all subsequent facilities using Chernoff bounds.

Each new point p has probability δ_p/f to open a new facility, where δ_p is the service cost on p's arrival. This event is a Bernoulli trial that we will call $\mathcal{F}_p$.
We can use the deterministic guarantee on service cost after a facility has already opened, which we found before to be $\sum_p \delta_p \leq 3\nu^\star$, to get the sum of effectively independent Bernoulli trials $\mathcal{F}$ we need: the expectation will be at most $3\nu^\star/f$, or substituting for f,

$$E[\mathcal{F}] = \mu \leq \mu' = 3k(1+\log n)\frac{\nu^\star}{L}$$

Applying Chernoff bounds,

$$Pr\left(\mathcal{F} \geq (1+\delta)\mu\right) \leq \exp(-\delta^2\mu/3) \leq \exp(-\delta^2\mu'/3) \leq \frac{1}{n} \qquad \rightarrow \quad \delta = \sqrt{\frac{3\log n}{3k(1+\log n)}\frac{L}{\nu^\star}}$$

results in the probability of having more than $6k(1+\log n)\nu^\star/L$ facilities after the first one has opened being vanishingly small:

$$Pr\left(\mathcal{F} \geq (1+\delta)3k(1+\log n)\frac{\nu^\star}{L} \geq 6k(1+\log n)\frac{\nu^\star}{L}\right) \leq \frac{1}{n}$$

We get our final result by summing this bound with the aforementioned deterministic guarantee on the first facility in each region:

$$|F| \leq \mathcal{F} + k(1+\log n) \leq 7k(1+\log n)\frac{\nu^\star}{L}$$

□

Observe that the value of f that we need to supply to the OFL algorithm depends on a "guess from below" L of the optimum cost. In what comes later, we will show how PLS+ embeds OFL into a more powerful algorithm, that makes increasingly more accurate guesses of L to converge towards an output.

3.2 Streaming k-median coreset construction

PLS+ is a streaming algorithm based on OFL that builds an $O(1)$-bounded coreset of X in one single pass over the dataset. It returns a set of facilities F of cardinality $O(k \log n)$, which is in effect a weighted set of points, each weighted point "representing" a number of points from the stream X.

PLS+ uses OFL as a subroutine, making successive guesses at the value of L, our lower bound to the optimum, in order to get a better approximation. A nice and rather significant additional benefit is that PLS+ is oblivious to the value of the optimum solution, which OFL is not.

If OFL's current solution gets too costly or includes too many facilities, we consider our guess to be no longer acceptable, improve it by a constant factor, and start over with an improved guess.

3.2.1 Pseudocode and description

Definition. *A **phase** is a single execution of the OFL algorithm on a new value of L and thus a new value of the facility cost for OFL f, corresponding to single iteration of the outer `while` loop of PLS+.*

A phase terminates once either we paid "too much" in service cost, or we opened too many facilities. Once that happens, we push all the facilities we opened back on the stream, and start over with a new, higher value of L that is closer to the optimum.

The concept of "pushing a facility on the stream" can be implemented as a stack that works in parallel to the stream. We push all facilities on the stack as points with weight corresponding to their weight as facilities, and until the stack is not empty, we stop reading from the stream, popping points from the stack as if they came from the stream. Regardless, note that this paper will make no further use of the concept of the stack - this is merely an implementation note. When facilities are pushed back on the stream, the stream will look like all previous facilities as weighted points, followed by the unread points.

β and γ are constants supplied by the user; we will see later (Lemma 4) how they are defined. These constant govern the phase change: β is the multiplicative factor applied to the next guess for L, and γ controls the loop invariants, deciding when the set of facilities becomes "too large" and when we pay too much in service cost.

Algorithm 2 PLS+. β, γ are user-supplied constants.

1: $L_1 \leftarrow 1$ ▷ Initial guess at L
2: $i \leftarrow 1$ ▷ Phase counter
3: **while** solution not found **do**
4: $c \leftarrow 0$ ▷ Current phase's (service) cost
5: $f \leftarrow L_i/(k(1 + \log n))$ ▷ Facility cost
6: $F \leftarrow \emptyset$ ▷ Set of facilities
7: **while** there are points still in the stream **do** ▷ OFL: begin
8: $x \leftarrow$ next point
9: $p \leftarrow \text{RANDOM}([0, 1])$
10: **if** $p \leq \min\{1, w(x) \cdot \delta(x, F)/f\}$ **then**
11: $F \leftarrow F \cup x$
12: **else**
13: $c \leftarrow c + w(x) \cdot \delta(x, F)$
14: $y \leftarrow \arg\min_{y \in F} \delta(x, y)$
15: $w(y) \leftarrow w(y) + w(x)$
16: **end if** ▷ OFL: end
17: **if** $c > \gamma L_i$ or $|F| > (\gamma - 1)k(1 + \log n)$ **then** ▷ Check constraints
18: Break and raise flag ▷ Trigger new phase
19: **end if**
20: **end while**
21: **if** flag raised **then** ▷ Inner while ended because constraints broken
22: Push facilities in F on to stream
23: $L_{i+1} \leftarrow \beta L_i$
24: $i \leftarrow i + 1$
25: **else** ▷ Inner while ended because no points left
26: solution found
27: **end if**
28: **end while**
29: return F

3.2.2 Analysis

Let us first make sure that the beginning of a new phase causes previous facilities to be successfully clustered in their entirety without breaking the loop invariants (line 17), before moving on to new points from the stream. This is to make sure that PLS+ does not indefinitely create new phases without making any progress in processing new points.

To make some progress along the stream, we will need both the invariant on facility count and the invariant on cost to hold after all former facilities have been processed.
Observe that, if the facility count invariant caused the new phase to begin, then it is possible that all weighted points in front of the stream will be turned back into facilities, in case the new facility cost is still too low compared to the distances between the weighted points.
Call the first former facility f_1 and some other former facility f_2. Since the facility cost is increased by a factor β between one phase and the next, at some point it will hold that $\delta(f_1, f_2) < f$, causing f_2 to be connected to f_1 and the facility count to decrease,

which will cause the facility count invariant to be satisfied once all former facilities have been processed.

Concerning the cost invariant:

Lemma 2. *At the start of a new phase, PLS+ successfully clusters all former facilities from the previous phase, without breaking the invariant, before moving on to the unread points from the stream. That is, once all former facilities have been processed, it holds that*

$$c < \gamma L_i$$

Proof. First, observe that a similar observation to OFL applies, in that we pay at most f for a point: either we open a new facility, paying that exact price, or we pay a service cost $w(x) \cdot \delta(x, F)$ that is bounded by f.

Because of the second invariant on line 17, there are at most $(\gamma - 1)k(1 + \log n)$ "former facilities" from the previous phase, which are seen as weighted points in front of the stream. By the previous observation, these weighted points collectively have service cost at most

$$c \leq f(\gamma - 1)k(1 + \log n) = \frac{L_i}{k(1 + \log n)}(\gamma - 1)k(1 + \log n) = (\gamma - 1)L_i$$

The lemma follows by simply observing that $c \leq (\gamma - 1)L_i < \gamma L_i$.

□

Bounding the cost at the start of a phase

Second, we can show that we can bound the cost to optimally cluster any subset of points P present on the stream at the start of a phase - both unread and weighted - in terms of the cost to optimally cluster the entire stream X. This implies that P is a good approximation of the whole stream, as if we clustered P with the optimum centers for the whole stream, the resulting cost would be a constant approximation of the optimum cost over the whole stream.

Before proceeding, it will be convenient to define the following.

Definition. *Let $x \in X$ be any point of the input dataset.*
We say that a facility $y \in F$ ***represents*** *x in phase ℓ if y is the assigned facility for x, or for x's representative in phase $\ell - 1$.*
We may additionally refer to x as an ***original point****.*

Observe that once x becomes represented in any phase, it stays represented in all subsequent phases. Also, if x is created as a facility, x may represent itself.

We now proceed to the lemma.

Lemma 3. *Let P be any subset of weighted or unread points in the stream at the start of phase i. Then, the cost of the optimum k-median clustering of P is at most*

$$\nu_P^\star \leq \nu_X^\star + \frac{\gamma}{\beta - 1} L_i$$

Proof. Suppose we're at the start of phase i. At this point, the stream is composed of weighted points corresponding to the former facilities from phase $i-1$, followed by unread points.

Let us bound the service cost of any input point $x \in X$ - whether unread or read, and therefore presently represented by some other point - if we clustered it with the optimum k-median centers $K^\star$ for the whole input dataset X.

Suppose, without loss of generality, that x is not an unread point, which means it is currently represented on the stream by some former facility. Let j be the phase where x was first clustered: this means that, in the phases up to now, $j, j+1, \ldots, i-1$, x had been represented by some facilities that we will call $y_j, y_{j+1}, \ldots, y_{i-1}$, where therefore y_{i-1} is the weighted point that currently represents x at this point in time, the start of phase i.

The service cost c_x due to x in phase i will then be the service cost of its current representative y_{i-1}, so $\delta(y_{i-1}, y^\star)$, where $y^\star \in K^\star$ is the optimum center that is closest (cheapest) to y_{i-1}.

By the triangle inequality, c_x can be bound by the cost of optimally clustering the original point x, plus the distance between x and its representative y_{i-1} - the latter can in turn be bound by the sum of the distances between successive representatives, as follows:

$$\begin{aligned} c_x = \delta(y_{i-1}, y^\star) &\leq \delta(x, y^\star) + \delta(x, y_{i-1}) \\ &\leq \delta(x, y^\star) + \sum_{\ell=2}^{i-j} \delta(y_{i-\ell}, y_{i-\ell+1}) + \delta(x, y_j) \end{aligned}$$

Note that if x was an unread point instead, then it would represent itself in the current phase (y_{i-1} would be x itself), which means $\delta(x, y_i) = 0$, and the inequality still holds since $c_x = \delta(x, y^\star)$.

Consider now any subset of weighted or unread points P on the stream at the beginning of phase i, calling $p_x \in X$ the original point represented by $p \in P$ should it be a

weighted point (otherwise $p_x = p$ itself).
Let us use the result on individual points to consider the cost to optimally cluster this whole subset:

$$\nu^\star_{X'} = \sum_{p \in P} c_p \leq \sum_{p_x \in X} \delta(p_x, y^\star) + \sum_{p_x \in X} \delta(p_x, y_{i-1})$$

The first term on the right hand side of this bound is the optimum clustering cost for all original points p_x corresponding to points in P, and can simply be upperbounded by the optimum clustering cost for the whole stream, $\nu^\star_X$.

The second term represents the distance between each original point $p_x \in X$ and its current representative in phase i; we have discussed that $\delta(p_x, y_{i-1}$ is upperbounded by the sum of the distances between p_x's successive representatives between one phase and the next, from the phase where p_x was first clustered up to the previous phase $i-1$, which yielded p_x's current representative.
This means that the terms of this summation are all the service costs that were ever paid by all points $p_x \in X'$ through their representatives in all previous phases. Similarly to our argument for the first term, then, we can upperbound all the quantities corresponding to an individual phase with the cumulative service costs paid by all points that were clustered during that phase.

The cumulative service cost c_ℓ in a generic previous phase $\ell < i$ is bounded by the invariant on service costs to be:

$$c_\ell \leq \gamma L_\ell = \gamma L_i \frac{1}{\beta^{i-\ell}}$$

Summing this quantity over all previous phases ($\ell = 1, 2, \cdots, i-1$) will therefore yield the upperbound to the whole second term that we were seeking:

$$\sum_{p_x \in X} \delta(p_x, y_{i-1}) \leq \sum_{\ell=1}^{i-1} \gamma L_i \frac{1}{\beta^{i-\ell}} = \gamma \left(\frac{1}{\beta - 1} \right) L_i$$

We obtain then our claim,

$$\nu^\star_P \leq \nu^\star_X + \frac{\gamma}{\beta - 1} L_i$$

□

This lemma shows that, at any phase i, past points are well represented by their currently assigned facility, and we can obtain a good approximation of X's optimum clustering, provided that $L_i < \nu^\star$. This latter condition holds with high probability for all phases until a solution is found, as we will show next.

Termination and critical phase

If the lower approximation to the optimum that we provide grows by a factor of β every phase, it may be natural to assume it might eventually grow too big, such that it is no longer smaller than $\nu^\star$, at which point the assumptions we have made would fall apart. Luckily, we can prove that the probability of this happening is vanishingly small (in fact, the same probability as the probability that OFL fails).

Definition. *Call the last phase i where $L_i < \nu^\star$ the* ***critical phase****.*

Observe that the critical phase might not even come into existence: a solution may well be found before then, as the lemma will make clear.

Lemma 4. *Let*

$$c_{OFL} = \left(3 + \frac{2e}{e-1}\right) \simeq 6.164, \qquad k_{OFL} = 7$$

be the constant factors on the high probability bounds, respectively, on service cost c and on facility count $|F|$ resulting from the analysis of OFL. Then, by setting

$$\beta = 2c_{OFL} + 2 \simeq 14.328$$

$$\gamma = \max\{4c_{OFL}^2 + 2c_{OFL}, \beta k_{OFL} + 1\} \simeq \max\{164.3, 101.3\} = 164.3$$

the algorithm terminates at, or before, the critical phase with high probability.

Proof. Let i be the critical phase and X_i be the set of all the points, weighted and unread, that are present on the stream at the start of phase i.

Observe that, since this is the critical phase, it holds that $\nu^\star_X \leq \beta L_i$.

We can apply Lemma 3 to X_i:

$$\nu^\star_{X_i} \leq \nu^\star_X + \gamma\left(\frac{1}{\beta-1}\right) L_i \leq \left(\beta + \gamma\frac{1}{\beta-1}\right) L_i$$

The OFL algorithm that is running in this phase yields a solution so that, with high probability,

$$c \leq c_{OFL}\nu^\star_{X_i}, \qquad |F| \leq \beta k_{OFL} k(1 + \log n)$$

(where for $|F|$ we have observed that $\beta \geq \nu^\star_i / L_i$).

At this point, then, we would like the invariants to be satisfied so that a new phase is not triggered: respectively,

$$c_{OFL}\nu^{\star}_{X_i} \leq \gamma L_i, \qquad \beta k_{OFL} k(1+\log n) \leq (\gamma-1)k(1+\log n)$$

must hold. Rewriting the first invariant to apply the result we found above, we obtain

$$\gamma \geq c_{OFL}\left(\beta + \gamma\frac{1}{\beta-1}\right), \qquad \gamma \geq \beta k_{OFL} + 1$$

and therefore, by setting β, γ to be

$$\beta = 2c_{OFL} + 2, \qquad \gamma = \max\{4c^2_{OFL} + 2c_{OFL}, \beta k_{OFL} + 1\} = 4c^2_{OFL} + 2c_{OFL}$$

if the OFL algorithm that is running in the critical phase succeeds with high probability, PLS+ also succeeds with the same probability.

□

Approximation factor determination

We can now state the main result of this section.

Theorem 5. *With high probability, PLS+ completes the final phase with a solution F of cost at most*

$$\sum_{x\in X}\delta(x,F) \leq \alpha\cdot\nu^{\star}_X, \qquad \alpha = \frac{\beta\gamma}{\beta-1}$$

and therefore F is an α-bounded coreset of cardinality $O(k\log n)$ for the k-median clustering problem over X.

Proof. Suppose we're in phase i, and suppose it is the final phase. Consider any point $x \in X$ of the stream, and let j be the phase where x was first clustered.

Analogously to Lemma 3, let $y_j, y_{j+1}, \ldots, y_{i-1}, y_i$ be the representatives of x in the phases, $j, j+1, \ldots, i-1, i$ - note that unlike Lemma 3, we're not at the beginning of phase i and x already has a new representative for the current phase.

If we therefore cluster X with F, the cost c_x due to x is the distance to its current representative, $\delta(x, y_i)$. By the triangle inequality, c_x can be bound by the cost of connecting x to its original representative y_j, plus the sum of the distances between successive representatives, as follows:

$$c_x = \delta(x,y_i) \leq \delta(x,y_j) + \sum_{\ell=1}^{i-j}\delta(y_{i-\ell}, y_{i-\ell+1})$$

In other words, the cost to cluster x w.r.t. F is upperbounded by all the service costs that were ever paid by its representatives through all phases.

Therefore, if we consider now the entirety of the stream X, summing c_x over all points $x \in X$ yields all of the service costs that were ever paid by every point in every phase from the first to the last one, i. We can then use again, as in Lemma 3, the invariant's bound on the collective service cost c_ℓ paid in a generic phase $\ell < i$,

$$c_\ell \leq \gamma L_\ell = \gamma L_i \frac{1}{\beta^{i-\ell}}$$

Considering this quantity over all phases $\ell = 1 \dots \ell$, and given that by Lemma 4 we terminate at a phase where $L_i \leq \nu^\star$ with high probablity, we obtain the following bound on the service cost of the solution returned by PLS+:

$$\begin{aligned} c &\leq \sum_{\ell=1}^{i} c_\ell \\ &\leq \gamma L_i + \gamma L_{i-1} + \gamma L_{i-2} + \ldots + \gamma L_1 \\ &\leq \gamma L_i \sum_{\ell=0}^{i-1} \frac{1}{\beta^\ell} \\ &\leq \frac{\beta\gamma}{\beta - 1} \nu^\star \end{aligned}$$

□

3.3 Streaming k-median coreset refinement

`CoverWithBalls` is a primitive that refines a coreset, obtaining a vastly improved bound on the optimum solution at the price of obtaining a coreset of higher cardinality.
It was defined by Mazzetto, Pietracaprina and Pucci [11].

`CoverWithBalls` aims to build an ε-bounded coreset C of a k-median instance (X, k), starting from an intermediate α-approximate solution T. The way it does that is by greedily selecting an arbitrary point $x \in X$, adding it to C, and discarding from X any points y for which

$$\delta(x, y) \leq \frac{\varepsilon}{2\alpha}\delta(x, T)$$

until all points of X are discarded.

3.3.1 Pseudocode and description

We will implement a straightforward streaming version of `CoverWithBalls` with the aim of applying it to the output of the PLS+ algorithm seen earlier. Since `CoverWithBalls` greedily selects an arbitrary point, we can make the greedy choice correspond to the next point in the stream.

Like the original version, `StreamingCWB` also allows for the constraint to be somewhat relaxed: if a point is already "close enough" to the coreset F within a threshold R, we are satisfied with a relaxed bound of $\varepsilon/(2\alpha) \cdot R$ instead of the smaller bound yielded by their actual, smaller, distance.

The importance of allowing for a relaxation will help in our analysis when we need to bound the size of the new coreset as a function of X's dimensionality, as we shall see.

Algorithm 3 `StreamingCWB`$(X, F, R, \epsilon, \alpha)$

$F' \leftarrow F$ ▷ Set of new facilities
$w(f) \leftarrow 0, \forall f \in F'$ ▷ Clear all weights
while there are points still in the stream **do**
  $x \leftarrow$ next point
  if $\delta(x, F') > \varepsilon/(2\alpha) \max\{R, \delta(x, F)\}$ **then** ▷ Does F' satisfy the new bound?
    $F' \leftarrow F' \cup x$ ▷ No: add x to F'
    $w(x) \leftarrow 1$
  else
    $z \leftarrow \arg\min_{z \in F'} \delta(x, z)$ ▷ Yes: make x's representative the closest $z \in F'$
    $w(z) \leftarrow w(z) + w(x)$
  end if
end while
return F'

Additionally, note that the map τ that connects a point to its coreset representative is not necessarily optimum, that is, each point is not necessarily linked to its nearest coreset representative. The likelihood of this happening is higher the earlier in the stream, since there subsequently might be new coreset points that are closer to a past point than its currently assigned representative. Nevertheless, the map does satisfy the bounds we impose on the quality of the coreset as we will see next.

3.3.2 Analysis

By construction, it is trivial to see that `StreamingCWB` correctly returns a weighted set that, as is our intention, reduces all distances to representatives by a factor $\varepsilon/(2\alpha)$.

We shall formalize this statement via the following lemma.

Lemma 6. *Let F' be the output of `StreamingCWB`. F' is weighted according to a map $\tau : X \to F'$ such that, for any $x \in X$,*

$$\delta(x, \tau(x)) \leq \frac{\varepsilon}{2\alpha} \max\{R, \delta(x, F)\}$$

Proof. For any $x \in X$, we define $\tau(x)$ as either:

- point x itself, if the check on line 5 succeeded;
- point z, if the check on line 5 failed.

The statement immediately follows.

□

Crucially, the result we seek from the analysis of `StreamingCWB` is - how big is the new facility set? That is, how much do we pay in terms of space in order to obtain a better approximation on the solution?

We will need a few preliminary definitions and a technical lemma.

Definition (Doubling dimension)**.** *The doubling dimension of a metric space $\mathcal{M}$ is the smallest integer D, such that a ball of any radius r centered at any point $x \in \mathcal{M}$ can be covered by at most 2^D balls of radius $r/2$ centered at some points of $\mathcal{M}$.*

We can think of the doubling dimension as the generalization of Euclidean dimensionality to general metric spaces. The proof that F' is not too large for spaces with low doubling dimension hinges on the fact that there cannot be "too many" points which are far from one another.

Definition (r-clique)**.** *A set of points P is said to be an r-clique if, for any pair of distinct points $x, y \in P$, it holds that $\delta(x, y) > r$.*

Lemma 7. *Let $0 < \varepsilon < 1$ and $\mathcal{M}$ be a metric space with doubling dimension D. Let $P \subseteq \mathcal{M}$ be an $\varepsilon \cdot r$-clique and assume that P can be covered by a ball of radius r centered at a point of $\mathcal{M}$. Then,*

$$|P| \leq \left(\frac{4}{\varepsilon}\right)^D$$

Proof. We can apply the definition of doubling dimension recursively to observe that the ball of radius r which by hypothesis covers P can be covered by 2^{jD} balls of radius $r/(2^j)$, with j any nonnegative integer.

Let i be the smallest integer for which $2^{-i} \cdot r \leq \varepsilon/2 \cdot r$ holds.

Since P is a $\varepsilon \cdot r$-clique, any of the 2^{iD} balls with radius $2^{-i} \cdot r$ can contain at most one point of P, and therefore $|P| \leq 2^{iD}$.

As $i = 1 + \lceil \log_2(1/\varepsilon) \rceil$, it follows that $|P| \leq (4/\varepsilon)^D$.

□

We then come to the crucial result of this section.

Theorem 8. *Let F' be the output of `StreamingCWB`. Suppose that the points in stream X and facility set F belong to a metric space with doubling dimension D.*
Let c be a real value such that, for any $x \in X$, $c \cdot R \geq \delta(x, F)$.
Then,

$$|F'| \leq |F| \cdot (16\alpha/\varepsilon)^D \cdot (\log_2 c + 2)$$

Proof. Let $F = \{f_1, \ldots, f_{|F|}\}$ be the set of facilities in input to `StreamingCWB`.

For any i, $1 \leq i \leq |F|$, let

$$X_i = \{x \in X : i = \arg\min_i \delta(x, f_i)\}$$

the sets of all input points that are closest to facility f_i, and

$$B_i = \{x \in X_i : \delta(x, f_i) \leq R\}$$

the sets of all points in X_i whose distance from their facility is under threshold R.

We also define, for any integer value $j \leq 0$, the sets

$$D_{i,j} = \{x \in X_i : 2^j \cdot R < \delta(x, f_i) \leq 2^{j+1} \cdot R\}$$

partitioning all points not in B_i in subsets that are "exponentially far" from their facility.

Observe that, for any $j \geq \lceil \log_2 c \rceil$, sets $D_{i,j}$ are empty since $\delta(x, F) \leq c \cdot R$.

Together, sets B_i and $D_{i,j}$ are a partition of X.

For any i, let $C_i = F' \cap B_i$: we want to show that C_i is a $\varepsilon/(2\alpha) \cdot R$-clique, and use this fact to bound its cardinality.

Let c_1, c_2 be any two different points in C_i and suppose, without loss of generality, that c_1 was added first to F' either by existing in F or by being added in F'. Since c_2 was added later, it must be farther than $\varepsilon/(2\alpha) \cdot \max\{\delta(c_2, K), R\}$ from c_1, and in fact

$$\delta(c_1, c_2) > \varepsilon/(2\alpha) \cdot \max\{\delta(c_2, K), R\} \geq \varepsilon/(2\alpha) \cdot R$$

where the last inequality stems from $c_2 \in B_i$ and thus $\delta(c_2, f_i) \leq R$.

So C_i is indeed a $\varepsilon/(2\alpha) \cdot R$-clique: we can now apply Lemma 7 to it, observing that $C_i \subseteq B_i$ is contained in a ball of radius R centered in f_i, obtaining that

$$|C_i| \leq \left(\frac{8\alpha}{\varepsilon}\right)^D$$

Let us now define $C_{i,j} = F' \cap D_{i,j}$, for any i, j, and use a similar reasoning to bound their size.
Let c_1, c_2 be any two different points in $C_{i,j}$ and suppose again that c_1 was added first to F'. It therefore still holds that $\varepsilon/(2\alpha) \cdot \max\{\delta(c_2, K), R\}$ from c_1, and in fact

$$\delta(c_1, c_2) > \varepsilon/(2\alpha) \cdot \max\{\delta(c_2, K), R\} \geq \varepsilon/(4\alpha) \cdot 2^{j+1} R$$

where the last inequality stems from $c_2 \in D_{i,j}$ and thus $\delta(c_2, f_i) > 2^j \cdot R$.
Since the set $C_{i,j} \subseteq D_{i,j}$ is contained in a ball of radius $2^{j+1}R$ centered in f_i, we can similarly apply Lemma 7 to obtain

$$|C_{i,j}| \leq \left(\frac{16\alpha}{\varepsilon}\right)^D$$

Since sets C_i and $C_{i,j}$ collectively form a partition of F', we can bound its cardinality as the sum of the bounds we just found for C_i and $C_{i,j}$:

$$|F'| \leq \sum_{i=1}^{|F|} |C_i| + \sum_{i=1}^{|F|} \sum_{j=0}^{\lceil \log_2 c - 1 \rceil} |C_{i,j}| \leq |F| \cdot \left(\frac{16\alpha}{\varepsilon}\right)^D \cdot (\log_2 c + 2)$$

which concludes the proof. □

3.4 Two-pass dimensionality-oblivious coreset construction

Combining the results from Sections 3.2 and 3.3 allows us to define a two pass streaming algorithm for building an ε-bounded coreset, for $0 < \varepsilon < 1$, as follows.

Theorem 9. *Let $\alpha = \beta\gamma/(\beta - 1)$, with β, γ defined as in the analysis of PLS+ (Lemma 4); ε be such that $0 < \varepsilon < 1$; and c be defined as in Theorem 8.*
Running PLS+ on the stream X, with $|X| = n$, and then applying the `StreamingCWB` algorithm to the weighted set F that is output by PLS+, yields a weighted set F' such that

$$|F'| \leq |F| \cdot \left(\frac{16\alpha}{\varepsilon}\right)^D \cdot (\log_2 c + 2), \qquad \sum_{x \in X} \delta(x, \tau(x)) \leq \varepsilon \cdot \nu^\star$$

representing an ε-bounded coreset of cardinality $O(k \log n)$ for the k-median clustering problem over X.

Proof. The cardinality claim results directly from Theorem 8 by noting that `StreamingCWB` is ran over a coreset F, output by PLS+, which by Theorem 5 is such that

$$c = \sum_{x \in X} \delta(x, \tau(x)) \leq \frac{\beta\gamma}{\beta - 1} \nu^\star$$

Therefore, setting $\alpha = (\beta\gamma/\beta - 1)$ suffices to obtain this result.

The approximation claim is due to the fact `StreamingCWB` outputs a ε-bounded coreset, as discussed in Lemma 6. In particular, PLS+ outputs a weighted set such that

$$c = \delta(x, F) \leq \alpha \cdot \nu^\star$$

where α is defined as above, while by Lemma 6 `StreamingCWB` reduces the distances by a factor $\varepsilon/(2\alpha)$, and therefore

$$\sum_{x \in X} \delta(x, \tau(x)) \leq \alpha \frac{\varepsilon}{2\alpha} \cdot \nu^\star = \varepsilon \cdot \nu^\star$$

where we rescale ε by a factor 2.

□

Observe that, as stated before, it holds that

$$\sum_{x \in X} \delta(x, F') \leq \sum_{x \in X} \delta(x, \tau(x)) \leq \varepsilon \cdot \nu^\star$$

since τ is not necessarily the lowest cost map (linking each $x \in X$ to its actual nearest representative in F').

3.5 One-pass semi-oblivious coreset construction

The advantage of the two pass algorithm outlined in the previous section is that it is oblivious to the doubling dimension of the dataset. The disadvantage, clearly, is that it requires two passes over the dataset, as the algorithms are ran sequentially, `StreamingCWB` being applied to PLS+'s output and to the dataset.

In this section, we will explore a semi-oblivious way of constructing a coreset, by introducing a parameter representing how much actual physical space we can work with, and inferring from that the approximation ratio of our solution. This is well suited to real world applications, since we typically have some amount of system memory available to run our algorithm on, which we ideally would like to use in its entirety.

The semi-oblivious moniker stems from the fact that the doubling dimension is not a parameter that we need to supply to the algorithm, but it does factor in in the analysis.

In this section, we will use $\nu^{\star}_{(X,k)}$ to denote the value of the optimum cost for k-median on the dataset X, thus explicitly including the desired number of centers in the notation.

3.5.1 Description

Suppose we are given an instance (X, k) of the k-median clustering problem and a parameter M, representing the number of points we are allowed to physically store in local memory.

Suppose this M is adequately large: since PLS+ cannot store more than $(\gamma - 1)k(1 + \log n)$ points without triggering a new phase, this logically means $M > (\gamma - 1)k(1 + \log n)$.

We can then set a new parameter $k' > k$ to be

$$k' = \frac{M}{(\gamma - 1)(1 + \log n)}$$

and run PLS+ on a new instance (X, k') of the k-median clustering problem, using this larger center set size. No modifications to the code or behaviour of PLS+ itself are necessary: note that the stopping condition for a phase of the algorithm, based on the count of the number of facilities, becomes

$$|F| > (\gamma - 1)k'(1 + \log n) = M$$

which is in effect an "out of memory" stopping condition.

3.5.2 Analysis

The result hinges on further exploitation of the properties of the doubling dimension of a metric space.

Lemma 10. *Let $\mathcal{M}$ be a metric space of doubling dimension D and (X, k) be an instance of the k-median clustering problem, with $X \subseteq \mathcal{M}$. By picking a $k' > k$ such that*

$$k' = k \cdot \log n \cdot \left(\frac{4}{\eta}\right)^D, \qquad 0 < \eta < 1$$

it holds that

$$\nu^\star_{(X,k')} \leq \eta \cdot \nu^\star_{(X,k)}$$

Proof. Let $K^\star = \{c_1, \ldots, c_k\}$ be the set of optimum centers for (X, k), and $X_1, \ldots, X_k$ be the corresponding optimum clusters.

For ease of notation, fix an arbitrary i, $1 \leq i \leq k$, and call $C = X_i$ and $c = c_i$. Furthermore, let

$$R = \frac{\sum_{x \in C} \delta(x, c)}{|C|}$$

We show that, for every optimum cluster, we can build a set S of appropriate cardinality, such that $\nu_C(S) \leq \eta \cdot \nu^\star_{C,1}$; we can compose such sets across all clusters to get a set of centers K' of the desired cardinality k' that yields $\nu_X(K') \leq \eta \cdot \nu^\star_{C,k'}$.

Analogously to Theorem 8, partition C by defining "concentric" partitions C_j,

$$\begin{aligned} C_0 &= \{x \in C : \delta(x, c) \leq R\} \\ C_j &= \{x \in C : \delta(x, c) \leq 2^j \cdot R\} \setminus C_{j-1} \qquad\qquad j > 0 \end{aligned}$$

Observe that, for any $j > \lceil \log_2 |C| \rceil$, sets C_j are empty since, by definition of R, $\delta(x, c) \leq |C| \cdot R$, for any $x \in C$.

Every concentric partition C_j, $0 \leq j \leq \log_2 |C|$, belongs to a metric space of doubling dimension D, and therefore can be covered by 2^D balls of radius $(2^j \cdot R)/2$. In general, by repeatedly applying the notion of doubling dimension, C_j can be covered by 2^{aD} balls of radius $2^j \cdot R/2^a$; by setting $2^a = 4/\eta$, we obtain therefore that C_j can be covered by $(4/\eta)^D$ balls of radius $2^j \cdot R \cdot \eta/4$.

Let S_j be the set of points constructed by arbitrarily picking one point from each of these balls. Observe that, if we clustered C_j with the set of centres S_j, the distance of any point of C_j to the nearest centre is at most $2^j \cdot R \cdot \eta/2$, by the triangle inequality.

Let $S = \cup S_j$. Then, the cost of clustering optimum cluster C with the set of centres S is upperbounded by the following quantity:

$$\begin{aligned}
\nu_C(S) &\leq \sum_{j=0}^{\log|C|} \nu_{C_j}(S_j) \\
&\leq \sum_{j=0}^{\log|C|} |C_j| \cdot \frac{\eta}{2} \cdot 2^j R = \eta \sum_{j=0}^{\log|C|} |C_j| \cdot 2^{j-1} R \\
&\leq \eta \sum_{j=0}^{\log|C|} \nu_{C_j}(\{c\}) \\
&\leq \eta \cdot \nu_C(\{c\}) = \eta \cdot \nu^\star_{C,1}
\end{aligned}$$

By construction, $|S| = \log|C| \cdot \left(\frac{4}{\eta}\right)^D$, and since this holds for all optimum clusters, this means there exists a set K' of cardinality $k' = k \cdot \log n \cdot \left(\frac{4}{\eta}\right)^D$, formed by the union of all k sets S so described, such that

$$\nu_X(K') \leq \eta \cdot \nu^\star_{C,k}$$

and therefore it must additionally hold that

$$\nu^\star_{X,k'} \leq \eta \cdot \nu^\star_{C,k}$$

□

Having estabilished this, the result is straightforward:

Theorem 11. *Let $M > (\gamma - 1)k(1 + \log n)$ and (X, k) be an instance of the k-median clustering problem. When running PLS+ with β, γ as in Theorem 4, and with*

$$k' = \frac{M}{(\gamma - 1)(1 + \log n)}$$

we have that, with high probability, the facility set F output by PLS+ is an $\alpha \cdot \eta$-bounded coreset for (X, k):

$$\sum_{x \in X} \delta(x, F) \leq \alpha\eta \cdot \nu^{\star}_{(X,k)}$$

where

$$\alpha = \frac{\beta\gamma}{\beta - 1}, \qquad \eta = c \cdot \sqrt[D]{(\gamma - 1)\frac{k(1 + \log n)}{M}(\log n)}$$

Proof. According to Lemma 10, by picking

$$k' = k \cdot \log n \cdot \left(\frac{4}{\eta}\right)^{D}$$

where D is the doubling dimension of the dataset and $0 < \eta < 1$, it holds that

$$\nu^{\star}_{(X,k')} \leq \eta \cdot \nu^{\star}_{(X,k)}$$

Setting $k \log n(8/\eta)^D = M/((\gamma - 1)(1 + \log n))$ and solving for η, we get

$$\eta = c \cdot \sqrt[D]{(\gamma - 1)\frac{k(1 + \log n)}{M}(\log n)}$$

Therefore, by Theorem 5 we have that, with high probability, the coreset F obtained by running PLS+ on (X, k') is such that

$$\sum_{x \in X} \delta(x, F) \leq \alpha \cdot \nu^{\star}_{(X,k')}$$

where $\alpha = (\beta\gamma)(\beta - 1)$, which implies, by the choice of k' and the above derivation for η, that

$$\sum_{x \in X} \delta(x, F) \leq \alpha \cdot \eta \cdot \nu^{\star}_{(X,k)}$$

□

Chapter 4

Experimental evaluation

This chapter describes the implementation and evaluation of the techniques presented in this thesis. We will first set out our objectives in Section 4.1, describe the testing setup in terms of the hardware and software used in Section 4.2, and present the datasets we used in our evaluation in Section 4.3.

4.1 Objectives

The objective of our experiments is both to evaluate how the techniques we outlined behave in practice, as well as testing specific aspects of their behaviour. Specifically, we first seek to estabilish what kind of results PLS+ and `StreamingCWB` yield in practice, when run as per the analysis. We will then delve deeper into their behaviour, noting how the theoretical analysis, as it often happens, leads to very conservative bounds and resulting poor performance in practice - especially where, as in our case, these conservative approximation guarantees actively influence the runtime as parameters.

We will explore how the bounds resulting from the analysis of PLS+ are not at all tight, and dramatically lower values of γ yield coresets with better approximation guarantees and lower cardinality. To do so, we will make use of a strategy analogous to the well-known non-streaming `k-means++` algorithm [1], typically used to yield a starting set of centers for Lloyd's algorithm for the k-means clustering problem; we will modify it to work with k-median, and use the cost of its solution when run on the entire stream as an estimation of the optimum cost of a clustering, to be compared with the "guesses from below" (the values L_i) maintained by PLS+.

We will elaborate on these results by running our full two-step strategy with the lowest value of γ that yields a valid solution for each dataset, hence previewing what kind of results might be obtained if only we had an idea of the cost of the optimum clustering. Lastly, we will return to our original value of γ suggested by the analysis, and apply our modified version of `k-means++` to both the coreset resulting by running PLS+ alone and the coreset output by our two pass strategy. This allows us to compare how the quality and size of the input coresets affect the cost of the returned k-clustering.

4.2 Hardware/Software platform

The code was implemented in C++, because of its inbuilt support of streams, which neatly matches the streaming nature of our algorithms. Additionally, C++ supports multithreading via OpenMP in a way that is both simple to implement on existing code, compared with bespoke libraries such as MPI, and does not require the executable to be run with any additional parameters or requirements, and that can be disabled by simply compiling without the `-fopenmp` flag.

Although trivially parallelised distance computation was used as a way to speed up operations, had this code been intended for extended usage on even bigger datasets, other ways of achieving speedups would have been preferable, such as using spatial data structures or approximate computations.

The code was implemented, debugged, compiled and tested on a variety of platforms. The results presented in this section were obtained on a platform consisting of an Intel(R) Core(TM) i5-4570S CPU running at 2.90GHz, with four cores without hyperthreading and 12 GB of system memory, running Ubuntu Server 22.04.1 LTS.

The code was hosted on the Department of Information Engineering's GitLab instance and was compiled using the `-O3` and `-fopenmp` flags.

4.3 Datasets

We tested our code and ran our experiments on synthetic datasets, generated in the following way. Given $n \in \mathbb{N}^+$ points to be generated, $k > 1$ centers, $D \in \mathbb{N}^+$ integer dimensions, $r \in \mathbb{R}^+$ radius and $s \in \mathbb{N}^+$ integer spread factor:

- k centers were generated uniformly and independently within a radius r from the origin;
- for each of the n points to be generated, one of the centers was randomly and uniformly chosen, and each of the point's dimensions was generated by shifting the center by a random, normally distributed amount with zero mean and unit variance, multiplied by r/s - in effect, placing the point within a radius r/s of its center, with Gaussian noise being applied to each dimension indipendently.

Datasets with the following parameters were generated and used in the experiments:

Name	n	k	D	r	s
`100K`	100,000	4	2	100	4
`200K`	200,000	4	2	100	4
`500K`	500,000	4	2	100	4
`1M`	1,000,000	4	2	100	4

In all cases, the Euclidean distance was used as a distance metric. We did not implement or test other metrics, though as stated before our analysis extends to general metric spaces.

4.4 Results

We present time statistics on each run as a matter of completeness. We shall not comment on them, as the actual time performance of our implementation is not a factor in our experiments, and it is undoubtedly possible to get better results by changing the way distance is computed, e.g. via approximation or spatial data structures, or even trivially by examining metric spaces that use distance functions that are quicker to evaluate such as the L1 norm.

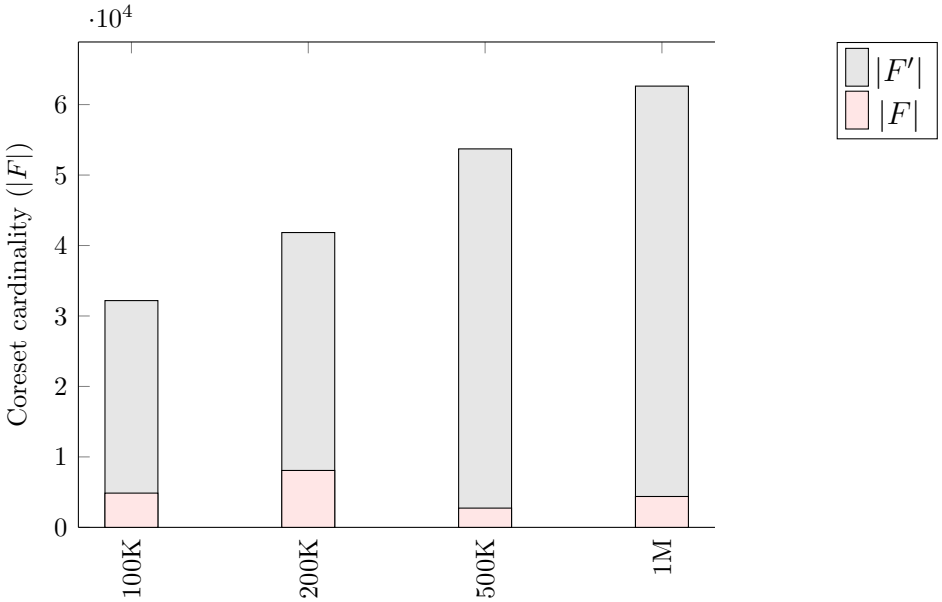
4.4.1 General performance of PLS+ and `StreamingCWB`

First, we present the results of running the two pass, dimensionality oblivious algorithm described in Section 3.4, on different sized datasets.

For this test, `StreamingCWB` was run with $R = 250$, $\varepsilon = 1$.

Dataset	PLS+			`StreamingCWB`	
	$\lvert F\rvert$	Phases	Time (mm:ss)	$\lvert F'\rvert$	Time (mm:ss)
`100K`	4862	4	0:53	32183	3:48
`200K`	8077	4	2:01	41841	11:22
`500K`	2733	5	5:53	53713	33:24
`1M`	4382	5	9:16	62624	97:04

Note how the size of the coreset returned by PLS+ is heavily dependent on how close to a phase change we were when the stream ends. Specifically, the size that is returned will be limited by the invariant to be at most $(\gamma - 1)k(1 + \log n)$, so we may encounter particularly unlucky instances that run out of points while working on a guess L that is big enough not to trigger a new phase, but too small to lead to a small size coreset, as many points will be turned into facilities rather than being connected to existing ones.



4.4.2 Impact of ε on `StreamingCWB` output cardinality

In this test, we investigated the practical impact of running different values of the space-accuracy tradeoff parameter ε on the cardinality of the coreset returned by `StreamingCWB`.

For the purpose of this test, we re-used the coresets returned by PLS+ in the previous section. In all instances, we again set parameter $R = 250$.

Dataset	ε	$\lvert F' \rvert$	Time (mm:ss)
`200K`	0.5	92729	23:30
	0.75	59945	14:39
	1	41841	11:22
`500K`	0.5	140181	106:00
	0.75	81930	53:09
	1	53713	33:24

Observe how the behaviour of ε is not at all linear, as the analysis already suggested, as smaller values will precipitate a blowup in coreset size, while it might be possible to get meaningful quality improvements for only a small penalty in space, if we limit ourselves to "reasonable" values of ε that are not too far from 1.

4.4.3 PLS+: Varying values of gamma

We investigated the tightness of PLS+'s analysis by running tests on the impact of varying values of γ. It is well known that bounds on theoretical analysis of randomized algorithms are often very conservative, and their performance is in practice much better; in our case, though, parameters derived from the analysis directly factor in the runtime of the algorithm. We are interested in keeping track of the cost and cardinality of the solutions returned by PLS+ by using values of γ lower than that mandated by the analysis, and checking how the corresponding final values of L compare to an estimate of the optimum cost of a k-median clustering on the entire dataset. We seek then to show that the lowest value of γ corresponding to a solution returned with L being lower than the estimated optimum cost is much lower than the analysis's suggested value.

We implemented a variant of the well-known `k-means++` algorithm, which we dubbed `k-median++`. The purpose of this was to obtain an estimate of the optimum cost of a k-median clustering on the entire dataset, which we can use as a comparison with the behaviour of PLS+. In the following, we will use the notation $\widetilde{\nu}^{\star}$ to indicate such estimates.

A brief sketch of `k-median++`'s strategy is as follows:

1. Select the first center randomly and uniformly among the input dataset;

2. Compute the distance between each input point and the center set;

3. Select the next center according to a weighted distribution, proportional to the distance of a point to the center set;

4. Repeat steps 2-3 until k centers have been selected.

To further improve on the cost estimate returned by `k-median++`, a final recentering step has been added to the algorithm, in which we calculate the centroids of the "provisional" clusters resulting from regular `k-median++`, and pick the closest input point to each of the k centroids as a center.

For each dataset, we ran PLS+ using values of γ corresponding to all powers of 2 from 2 to 128, in addition to the value suggested by the analysis ($\gamma \simeq 164.3$); `k-median++` was run three times, taking the lowest cost estimate.

Dataset	γ	Phases	Final L	$\|F\|$	Cost	Time (mm:ss)
`100K`	2	7	8.6516e+06	20	2.07133e+06	0:01
	4	6	603829	134	1.01518e+06	0:01
	8	6	603829	130	825553	0:02
	16	5	42143.5	810	435049	0:05
	32	5	42143.5	833	393774	0:07
	64	5	42143.5	833	296954	0:24
	128	4	2941.36	4804	169733	0:41
	164.3	4	2941.36	4905	162694	0:54

For dataset `100K`, `k-median++` returns $\widetilde{\nu}^\star = 3.11 \times 10^6$, which makes $\gamma = 4$ the smallest parameter value for which the final guess for L is smaller than the optimum.

Dataset	γ	Phases	Final L	$\|F\|$	Cost	Time (mm:ss)
`200K`	2	7	8.6516e+06	38	3.78833e+06	0:02
	4	7	8.6516e+06	33	2.68224e+06	0:05
	8	6	603829	220	1.5594e+06	0:06
	16	6	603829	221	1.20486e+06	0:18
	32	5	42143.5	1390	649342	0:32
	64	5	42143.5	1362	565734	1:04
	128	4	2941.36	7930	277396	3:01
	164.3	4	2941.36	8091	268216	3:28

For dataset `200K`, $\widetilde{\nu}^\star = 6.17 \times 10^6$, making $\gamma = 8$ the smallest valid choice.

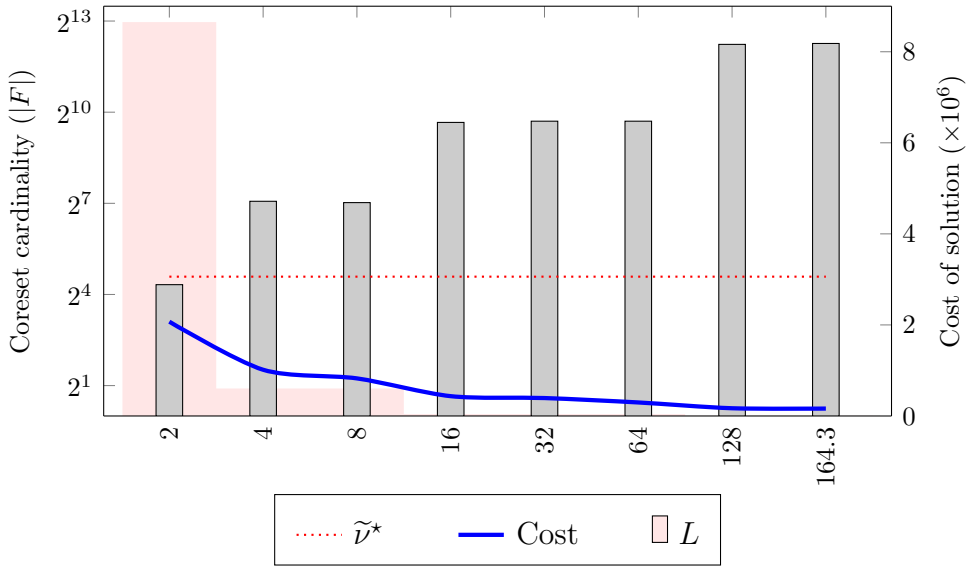
Dataset	γ	Phases	Final L	$\|F\|$	Cost	Time (mm:ss)
`500K`	2	7	8.6516e+06	74	7.374e+06	0:06
	4	7	8.6516e+06	76	5.44998e+06	0:09
	8	6	603829	458	2.94355e+06	0:21
	16	6	603829	457	2.58399e+06	0:40
	32	6	603829	440	2.07615e+06	1:35
	64	5	42143.5	2610	1.17138e+06	2:11
	128	5	42143.5	2718	958128	4:53
	164.3	5	42143.5	2697	858252	7:51

For dataset `500K`, `k-median++` returns $\widetilde{\nu}^\star = 1.57 \times 10^7$, meaning that already for $\gamma = 2$ the algorithm's final guess for the optimum cost, L_7, is valid.

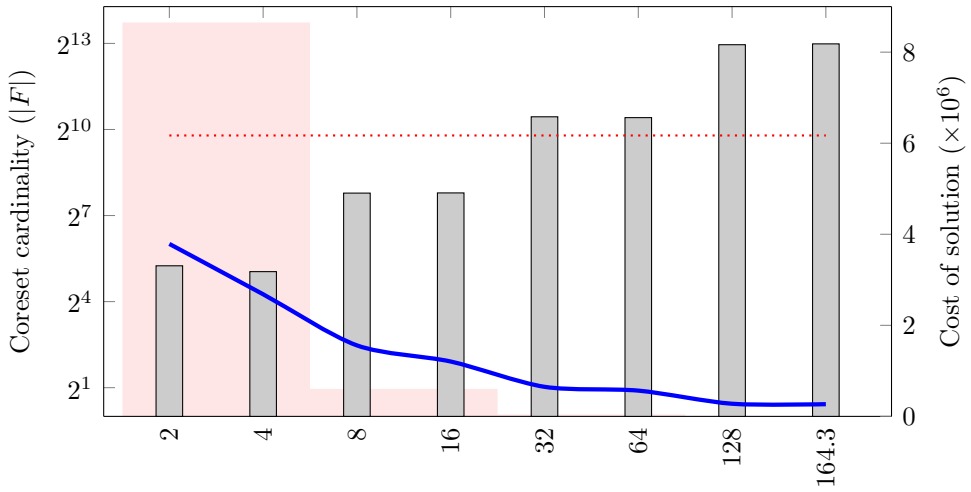
Dataset	γ	Phases	Final L	$\lvert F\rvert$	Cost	Time (mm:ss)
`1M`	2	8	1.23959e+08	18	1.95264e+07	0:06
	4	7	8.6516e+06	130	1.0009e+07	0:11
	8	7	8.6516e+06	128	7.6619e+06	0:27
	16	6	603829	723	4.44169e+06	0:46
	32	6	603829	752	3.81085e+06	1:31
	64	5	42143.5	4195	1.94863e+06	4:14
	128	5	42143.5	4273	1.72807e+06	6:46
	164.3	5	42143.5	4361	1.62708e+06	9:13

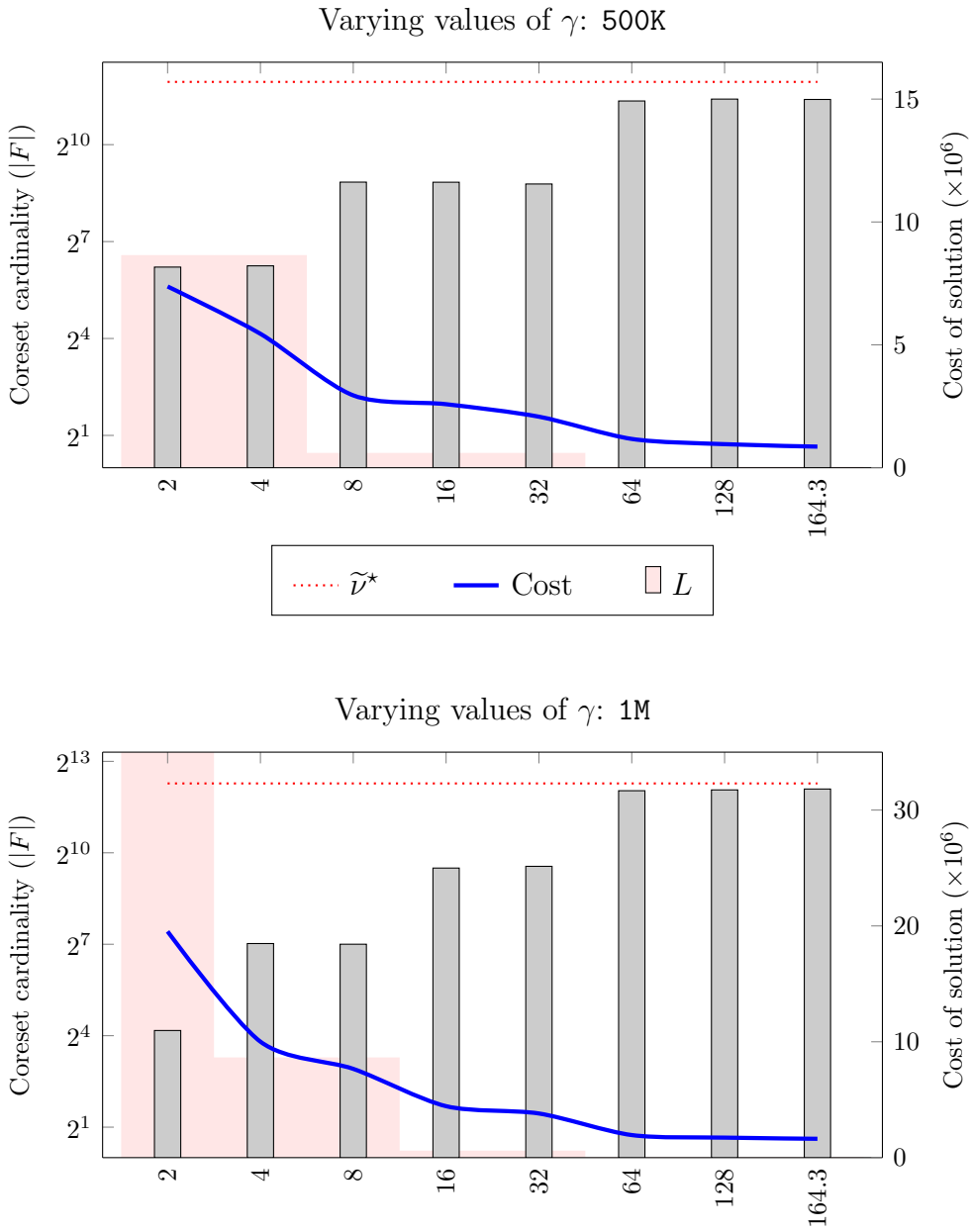
Finally, for `1M`, $\widetilde{\nu}^\star = 3.23 \times 10^7$, and $\gamma = 4$ is the smallest valid choice.

Varying values of γ: `100K`



Varying values of γ: `200K`





With this evidence at hand, it seems clear that the choice of γ suggested by the analysis is very conservative, and leads to a blowup both in the size of the returned coreset and in the time needed to return it. Additionally, γ strongly influences the approximation factor α of the returned solution, since $\alpha = \gamma\beta/(\beta - 1)$.

This is not at all uncommon in theoretical analysis of approximation algorithms, especially those that rely on randomization techniques. It seems clear that PLS+ would behave much better in practice if only we could tune its parameters to a specific dataset's optimum cost estimation. Of course, the way we obtained our values of $\widetilde{\nu}^{\star}$ are not applicable to general usage of our techniques, as we used an offline algorithm. It would be exceedingly difficult to get such an estimation via a streaming algorithm, not to mention that we would need to obtain it before PLS+ even began running, adding at least one more (potentially onerous) pass over the dataset to the overall coreset construction strategy.

4.4.4 Running the two pass coreset construction for the best values of γ

A natural question that might now arise is how might `StreamingCWB` behave when used on the much lower approximation results on the coreset that arise from using PLS+ with appropriately lower values of γ.

For each of the four datasets we used in the previous experiment, we ran our two pass coreset construction by first applying PLS+ with the best value of γ, and then applying `StreamingCWB` with a suitably resized α parameter.
We ran tests on different values of ε and $R = 250$.

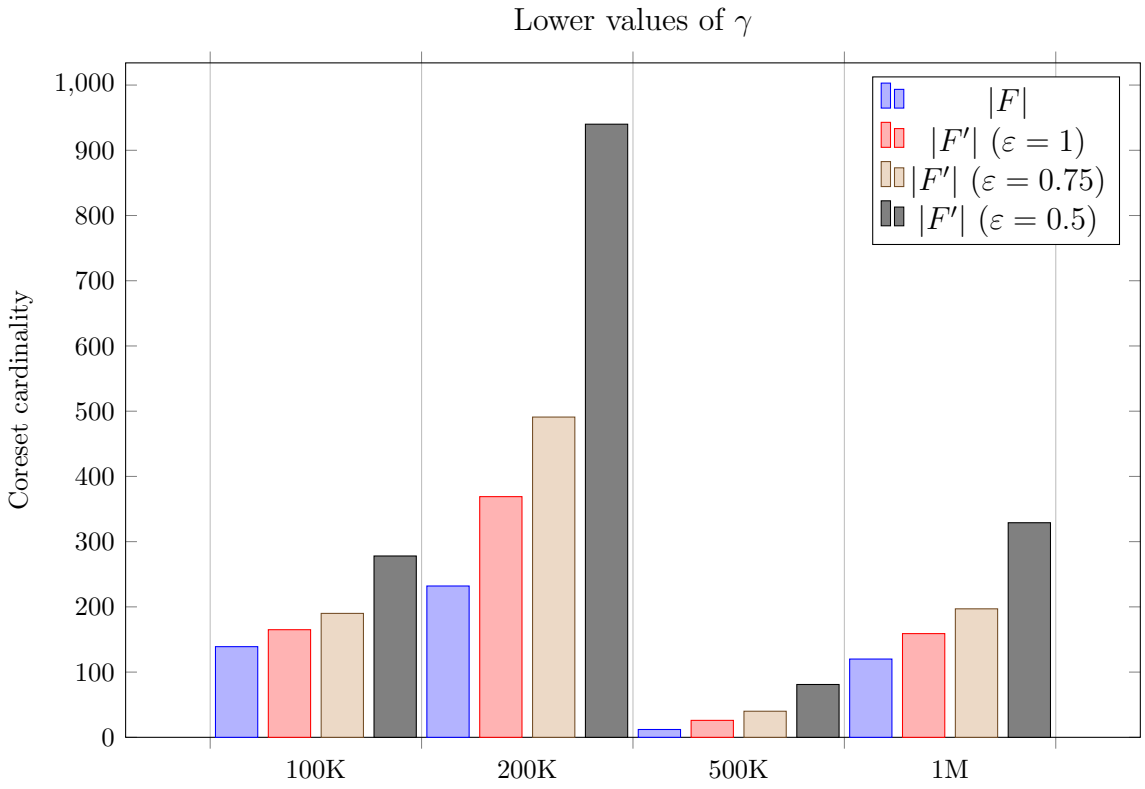
Data-set	Approx		PLS+				`StreamingCWB`			
	γ	α	$\lvert F\rvert$	Cost	Ph.	Time (s)	ε	$\lvert F'\rvert$	Cost	Time (s)
`100K`	4	4.3	139	756676	6	1.098	0.5	278	613529	3.609
							0.75	190	713121	2.972
							1	165	736054	2.773
`200K`	8	8.6	232	1.19e+06	6	4.046	0.5	940	684483	17.591
							0.75	491	949170	11.409
							1	369	1.07e+06	10.304
`500K`	2	2.15	12	1.26e+07	8	3.219	0.5	81	6.7e+06	5.547
							0.75	40	9.31e+06	4.001
							1	26	1.08e+07	3.429
`1M`	4	4.3	120	8.30e+06	7	10.454	0.5	329	6.35e+06	37.658
							0.75	197	7.72e+06	28.895
							1	159	8.06e+06	26.044

Unsurprisingly, the returned solutions are all-around much better; fewer facilities are created, leading to smaller coresets, since smaller values of γ lead to breaking the facility count invariant much more often, triggering new phases that have higher and higher facility costs. This in turn will make it much more expensive to open a new facility, with most of the new points being connected to existing facilities.

Also, recall that the approximation guarantees for these smaller values of γ in turn lead to a much more precise solution; lastly, working with smaller sets of facilities leads to clear time improvements as distance computations get much less costly, even with our straightforward parallel linear scan implementation. This incidentally allowed us to do some more meaningful experimentation with the ε space-accuracy tradeoff parameter, and may enable a hypothetical real user to select more aggressive values of ε.

4.4.5 Applying weighted `k-median++` on output coresets

Finally, we investigate the result of applying a weighted variant of `k-median++` to the coresets obtained via PLS+ alone, and both PLS+ and `StreamingCWB` in our two pass strategy. This can help us gauge the practical impact of `StreamingCWB` in refining PLS+'s solution, and serves as an example of how to connect our two pass construction



with a final k-clustering of the coreset.

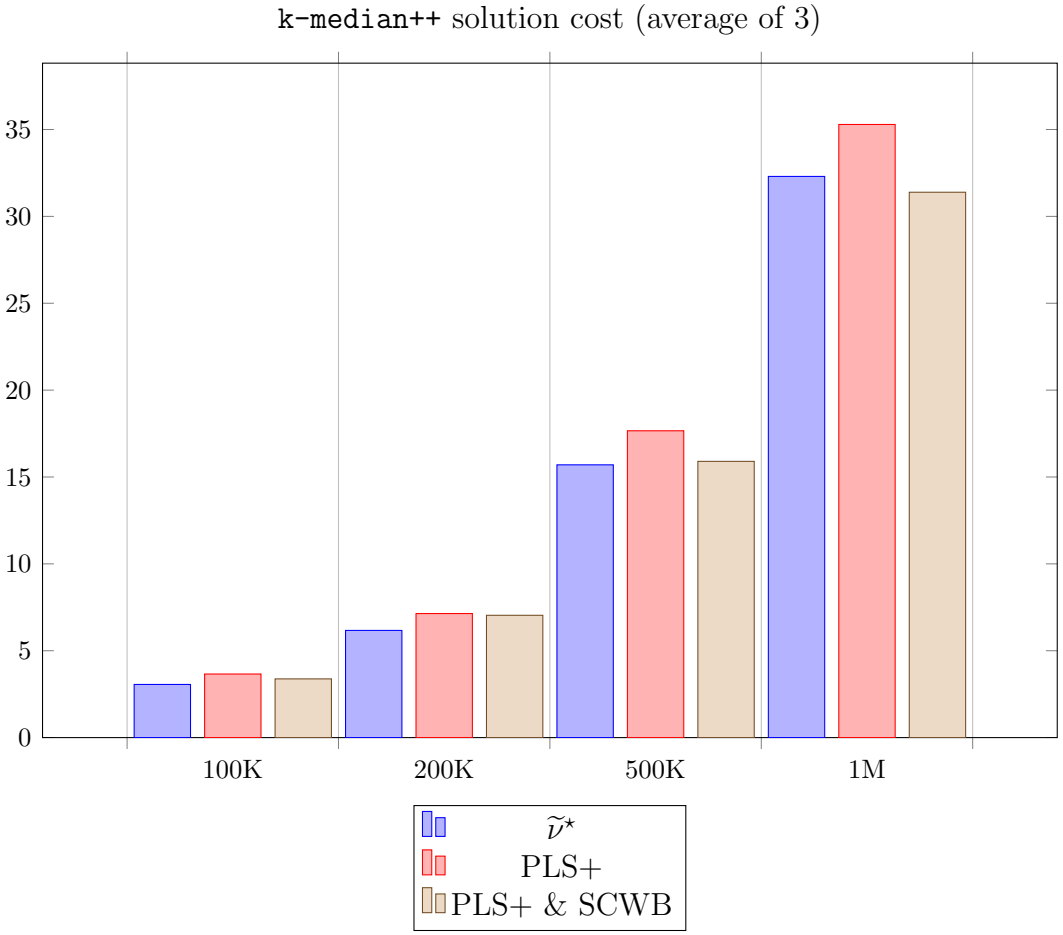
The extension of `k-median++` to handle weighted inputs is trivial and only requires multiplying each point-nearest center distance by the point's weight in the probability computation. We continue applying a recentering step, which is also adapted to handle weighted points by taking weights into account in the centroid computation.

We shall use as a starting point the results of Subsection 4.4.1. Thus, `StreamingCWB` is ran with $R = 250, \varepsilon = 1$, and we return to using $\gamma = 164.3$ as suggested by the analysis. Our results were assembled by averaging together three runs of `k-median++` on each input coreset.

Recall that we denote by $\widetilde{\nu}^\star$ the estimate on the optimum k-median clustering cost that we obtained by running `k-median++` directly on the dataset.

Dataset	$\widetilde{\nu}^\star$	`k-median++` on PLS+ alone		`k-median++` on PLS+ and `StreamingCWB`	
		Input size	Output cost	Input size	Output cost
`100K`	3.06e+06	4928	3.66214e+06	32092	3.38258e+06
`200K`	6.17e+06	8040	7.14514e+06	41777	7.04497e+06
`500K`	1.57e+07	2740	1.76646e+07	53702	1.59015e+07
`1M`	3.23e+07	4303	3.52897e+07	62682	3.13905e+07

We do not see big improvements in the output cost due to the application of `StreamingCWB`, likely because of our choice of ε, which is somewhat forced by the big input sizes. It is encouraging to see that the costs over the coresets are not too far off the estimations over the whole input dataset, though again this is somewhat dampened by the large cardinalities of the coresets as a result of the theoretical analysis of PLS+.



Chapter 5

Conclusions

We presented and analyzed streaming coreset-building techniques for the k-median clustering problem in general metric spaces. Our techniques leverage intrinsic properties of a dataset's underlying metric space - its doubling dimension - in oblivious and semi-oblivious ways, defining one and two pass construction strategies.

Our experimental evaluation of both the PLS+ algorithm on its own and of our two pass coreset construction strategy proved that the theoretical analysis outlined is in practice very conservative, and better results could be obtained if we had approximate knowledge of the cost of an optimum k-clustering on the dataset. As mentioned before, this is likely impractical to do with a streaming strategy, though the dramatic improvements we saw by running lower values of γ might justify the effort - besides, a very rough estimate could suffice.

Another open question is the possibility of somehow improving our two pass strategy so that it only requires one pass over the input dataset. It is trivial to extend our one pass semi-oblivious technique should the dataset have known dimensionality, but we did not ultimately devise a one-pass streaming algorithm that is fully oblivious to it.
Ideas that were raised include fusing the two phases into one, and proceeding to cull the resulting coreset in some way; or running two instances of PLS+ in parallel, with a simplified instance of PLS+, running on a modified guess $L' = \varepsilon L$, alongside the main instance, ultimately yielding a coreset of size augmented by a factor $(c/\varepsilon)^D$. Neither strategy should be of trivial analysis.

Bibliography

[1] 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 '07, page 1027–1035, USA, 2007. Society for Industrial and Applied Mathematics.

[2] 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, page 21–29, New York, NY, USA, 2001. Association for Computing Machinery.

[3] Vladimir Braverman, Harry Lang, Keith Levin, and Yevgeniy Rudoy. Metric k-median clustering in insertion-only streams. *Discrete Applied Mathematics*, 304:164–180, 12 2021.

[4] Vladimir Braverman, Adam Meyerson, Rafail Ostrovsky, Alan Roytman, Michael Shindler, and Brian Tagiku. Streaming k-means on well-clusterable data. In Dana Randall, editor, *Proceedings of the 2011 ACM-SIAM Symposium on Discrete Algorithms, SODA 2011*, pages 26–40. SIAM, 2011.

[5] Jarosław Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median and positive correlation in budgeted optimization. *ACM Trans. Algorithms*, 13(2), mar 2017.

[6] 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, page 30–39, New York, NY, USA, 2003. Association for Computing Machinery.

[7] Vincent Cohen-Addad, Anupam Gupta, Lunjia Hu, Hoon Oh, and David Saulpic. An improved local search algorithm for k-median. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, pages 1556–1612. SIAM, 2022.

[8] 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, page 569–578, New York, NY, USA, 2011. Association for Computing Machinery.

[9] Rajiv Gandhi, Samir Khuller, Srinivasan Parthasarathy, and Aravind Srinivasan. Dependent rounding and its applications to approximation algorithms. *J. ACM*, 53(3):324–360, may 2006.

[10] Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, page 901–910, New York, NY, USA, 2013. Association for Computing Machinery.

[11] Alessio Mazzetto, Andrea Pietracaprina, and Geppino Pucci. Accurate mapreduce algorithms for k-median and k-means in general metric spaces. In Pinyan Lu and Guochuan Zhang, editors, *ISAAC*, volume 149 of *LIPIcs*, pages 34:1–34:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.