Yale University
Department of Computer Science

Pass-Efficient Algorithms for Facility Location

Kevin L. Chang
Department of Computer Science
Yale University
kchang@cs.yale.edu

YALEU/DCS/TR-1337

Supported by NSF's ITR program under grant number 0331548.

Abstract

We present streaming algorithms that use multiple passes for the facility location problem: given a metric space (X, d) consisting of n points, and a facility cost f_i for each $x_i \in X$, find a set $F \subseteq X$ that minimizes the objective function $\sum_{x_i \in F} f_i + \sum_{x_i \in X} d(x_i, F)$, where $d(x_i, C)$ denotes the distance from x_i to the closest point in the set C.

Our main result is a $3(\ell - 1)$ pass algorithm that outputs a solution to facility location with approximation ratio $O(\ell)$ using an amount of extra space equal to $\tilde{O}(k^* n^{2/\ell})$, where k^* is the number of facilities used by the approximate solution. Our multiple pass algorithm offers a tradeoff between the number of passes and the space used, such that extra space needed by the algorithm falls exponentially with the number of passes. This tradeoff allows much flexibility for fine-tuning resource usage to balance the number of passes with the space used.

This algorithm addresses the need for a small space streaming algorithm for facility location. Using communication complexity results, we show that its parameterization by k^* is necessitated by the existence of families of instances of facility location that require any ℓ pass approximation algorithm to use at least $\Omega(n/\ell)$ bits of memory to approximate the cost of the optimum solution.

1 Introduction

The importance of massive data set computation in computer science has necessitated special purpose algorithms for such problems. It is well known that I/O to secondary and tertiary storage dominates processing times, and that the throughput of these systems is optimized for sequential access to the data. Tapes and DVDs are extreme examples of this phenomenon, but also disks exhibit this behavior: rate limiting penalties for frequent random access of the disk are incurred in the form of seek times and rotational latency. For large data sets, it is especially critical to limit these penalties.

A popular theoretical model for addressing these concerns is the streaming model of computation (also called the *pass-efficient model* for multiple pass algorithms). In this model, the input is given as a read-only array that may only be accessed sequentially in a small number of passes through the entire array. During and after each pass, the algorithm is allotted a small amount of working memory, typically $o(n)$, to perform its calculations. The rationale behind this memory restriction is that the input may be much to large to store in main memory, and thus must be kept in secondary storage. The critical resources to be optimized in this model are **(a)** the number of passes used by the algorithm and **(b)** the amount of extra memory needed. The streaming model for massive data sets was first proposed by Munro and Paterson [17]; the first few papers were by Henzinger *et al* [14] and Alon *et al* [2]. Although the bulk of the streaming literature involves one pass algorithms, many papers address multiple passes for problems such as selection, sorting, graph algorithms, and matrix approximation, including [1, 5, 10, 11, 14, 17].

The facility location problem is a well-studied clustering problem in combinatorial optimization and operations research. For this problem, we are given a metric space (X, d) consisting of n points and a distance metric d, and a facility cost f_i for each $x_i \in X$. The problem is to find a set $F \subseteq X$ that minimizes the objective function $\sum_{x_i \in F} f_i + \sum_{x_i \in X} d(x_i, F)$, where $d(x_i, C)$ denotes the distance from x_i to the closest point in the set C. Many algorithms with constant factor approximation ratios have been designed for this problem, including [7, 16, 18], as well as approximation schemes for instances that lie in the Euclidean plane [3]. It is also known that the problem cannot be approximated to within a factor of 1.46, unless $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{\log \log n})$ [12]. In this paper, we consider pass-efficient algorithms for facility location.

Massive data set applications for clustering problems are especially appealing. Indeed, single pass streaming algorithms have been designed for a number of fundamental clustering problems, including k-center and k-median. However, no such results exist for the remaining fundamental problem of facility location. A primary reason for this is that there are easy-to-construct instances of facility location that cannot be approximated to within any constant factor with less than $\Omega(n)$ facilities. Intuitively, this implies that one cannot hope to design a streaming algorithm that provides a solution to the problem using less than $\Omega(n)$ space. Indeed, in this paper we prove that any ℓ pass randomized algorithm that approximates even the *cost* of the optimum solution will need at least $\Omega(n/\ell)$ space, using a simple application of the communication complexity of computing the set disjointness function.

1.1 Our Results and Techniques

We address these difficulties by parameterizing the amount of space used by our algorithm by the number of facilities opened by the approximate solution. Thus, our algorithm will use $3(\ell - 1)$ passes to compute a solution with approximation ratio $O(\ell)$ using at most $\tilde{O}(k^* n^{2/\ell})$[1] bits of memory, where k^* is the number of facilities opened by the approximate solution. Note that for many instances k^* may in fact be *much smaller* (or possibly larger) than the number of facilities opened in an optimum solution.

A compelling feature of this algorithm is that the amount of memory decreases exponentially in the number of passes. Thus, making a few more passes will substantially reduce the amount of space used by the algorithm at the cost of slightly increasing the approximation ratio. This tradeoff in the multiple pass streaming model is similar to previous work by Chang and Kannan [5] for clustering census data.

For facility location, our technique is a combination of the general approach from [5] and a generalization of a sublinear bicriterion approximation algorithm for k-median by Indyk [15] (we will briefly outline his algorithm in Section 1.3; note that it is not streaming). The high level approach is that we take a sample S from the datastream X and cluster the sample with a known facility location algorithm. The resulting facilities can then be shown to be a good clustering for X, except for a (hopefully small) set of outliers. In subsequent passes, we "zoom in" on the outliers by recursively calling our algorithm on those points and sampling them at a higher frequency.

More specifically, the $3(\ell-1)$ passes used by **Pass-Efficient FL** are organized into $\ell-1$ triples of passes over the datastream. In the first pass of each triple, we take a random sample S from X. We wish to compute a facility location solution on S, which then would induce a solution on X. However, in order for a solution on the sample S to be a reasonable solution on all of X, we must consider all possible facilities in X, rather than S, since perhaps there may be very attractive facilities in X (i.e. nodes with small f_i) that do not appear in the sample S. Thus, in a second pass, we choose nodes in the datastream that could potentially be attractive facilities and store them along with our sample. We then cluster the sample along with these facilities using any black box α-approximation algorithm for facility location. The crucial fact is that this clustering will cost at most $O(\alpha)\mathrm{OPT}$ for some subset of the points in X, which we identify in the third pass. We then "zoom in" on those points that are not in the subset by iterating this process on points not clustered well.

We also propose an alternative way of addressing the difficulty presented by the $\Omega(n/\ell)$ lower bound on the amount of space needed by an ℓ pass algorithm for facility location. We define the *k-facility location problem* to be a hybrid between k-median and facility location, where we must find the best facility location solution that uses at most k facilities. A situation where this might arise is if we are constrained to building at most k facilities (perhaps we only have a limited amount of labor, time, or equipment), but each will incur a different monetary cost. k-median is a special case of the problem when all facilities costs are 0. In Section 3, we adapt our multiple pass algorithm to solve this problem using a small amount of extra space.

1.1.1 Summary of Results

We present the following results:

1. In Section 2, we present our main algorithm for facility location, **Pass-Efficient FL**, which is a $3(\ell-1)$ pass algorithm with approximation ratio $O(\ell)$ that uses at most $\tilde{O}(k^* n^{2/\ell})$ bits of memory, where k^* is the number of facilities opened by the *approximate* solution.

2. In Section 3, we adapt our main algorithm to an algorithm for k-facility location with approximation ratio $O(\ell)$ using 3ℓ passes and at most $\tilde{O}(k^{(\ell-1)/\ell} n^{1/\ell})$ bits of memory.

3. In Section 4, we prove a lower bound of $\Omega(n/\ell)$ for the number of bits needed by any ℓ pass randomized algorithm with a polynomial approximation ratio that computes the *cost* of the optimum solution to facility location.

[1] The notation $\tilde{O}(\cdot)$ is asymptotic notation that suppresses polylog factors

The exponential tradeoff between the number of passes the algorithm is allotted and the amount of memory that it uses is a compelling feature; the parameters can be adjusted to meet the memory and processing constraints imposed by the system on which the program is run.

1.2 Related Work

Of related interest to facility location are the well-studied k-center and k-median problems: find a set $C \subseteq X$, $|C| = k$, that minimizes the objective functions $\max_{x_i \in X} d(x_i, C)$, and $\sum_{x_i \in X} d(x_i, C)$, respectively.

Charikar *et al* [6] designed a single pass constant factor approximation algorithm for k-center that uses at most $O(k)$ extra memory. Guha *et al* [13] designed a single pass algorithm for k-median with approximation ratio $2^{O(1/\epsilon)}$ using at most $O(n^\epsilon)$ extra space, for $\epsilon > 0$. This was later improved by Charikar *et al* in [8] to a constant factor approximation ratio using at most $O(k \log^2 n)$ extra space. Charikar [9] and Panigrahy also gave a streaming algorithm for the objective function of finding k clusters that minimize the sum of cluster diameters, using $O(k)$ space, but also using ck centers, for some constant c.

There are no algorithms for facility location in the streaming model; the closest related work is that of Meyerson [16], who gave an online algorithm with an approximation ratio (distinct from, and stronger than, the competitive ratio) of $O(\log n)$ if the demand points in the datastream are ordered adversarially, and an approximation ratio of $O(1)$ if they are ordered randomly. This is an online algorithm, and thus does not have the small-space guarantees that we would like for streaming algorithms, but of course satisfies other interesting requirements. Badoiu *et al* [4] consider massive data set algorithms for facility location. They developed a sublinear algorithm that runs in time $O(n \log^2 n)$ for approximating the *cost* of the optimum solution for the case where all nodes have uniform facility cost. Similar to our lower bound in Section 4, they prove that no algorithm can approximate the cost in $\Omega(n^2)$ time for facility location with general facility cost.

An interesting comparison to **Pass-Efficient FL** is Chang and Kannan's multiple pass algorithm for a clustering problem related to the analysis of census data [5]. This problem, learning a mixture of k uniform distributions, involves approximating the density function of a mixture of distributions, given a datastream containing independently drawn samples from the distribution. The algorithm will output a function that is within L^1 distance ϵ of the original density. They proved a tradeoff between the number of passes and the space usage that is similar to the tradeoff in this paper: if the algorithm is allotted 2ℓ passes, then it will need at most $\tilde{O}(k^3/\epsilon^{2/\ell})$ bits of memory to compute an ϵ error approximation. Munro and Paterson's work [17] on selection and median finding in multiple passes exhibit similar tradeoffs. They give P pass algorithms with space usage $O(n^{1/P})$ and $O(n^{1/(2P)})$, respectively.

1.3 Review of Indyk's sublinear k-median algorithm

In this section, we give a sketch of a sampling-based algorithm for k-median by Indyk [15] that provides some of the intuition for our algorithm in Section 2. We assume we have a black box (α, β) bicriterion approximation algorithm for k-median (i.e. the algorithm outputs a solution with cost at most αOPT using at most βk medians).

The algorithm can be roughly stated as follows:

1. Take a sample of size $s = O(\sqrt{kn} \log k)$ from the nodes.
2. Run the black box algorithm on the sample to get a set C of βk centers.
3. Sort the points, and identify the set R that contains the $n - \tilde{O}(\sqrt{nk})$ points closest to the centers found in Step 2.
4. Run the black box algorithm on $X \setminus R$ to get an additional βk centers, and output all $2\beta k$ centers.

Indyk proved that the above algorithm would output a solution with $2\beta k$ centers that costs at most $O(1)$OPT, where OPT is the optimum objective function value for a solution with exactly k centers.

First, fix an optimum k-median solution with centers $c_1, \ldots, c_k$ and let C_i denote the set of points serviced by each center. Let $I = \left\{ i : |C_i| = \tilde{\Omega}(\sqrt{\frac{n}{k}}) \right\}$ be the set of indices that correspond to "large" clusters in the optimum solution. The rough idea of his proof of correctness of the algorithm is that, with any constant probability, the centers found in Step 2 from clustering the sample are a good approximation for all points that lie in the set $\cup_{i \in I} C_i$, in the sense that the cost of servicing those points with centers C will cost at most $O(1)$OPT. The number of points that lie outside of $\cup_{i \in I} C_i$ is at most $k \cdot \tilde{O}(\sqrt{\frac{n}{k}}) = \tilde{O}(\sqrt{kn})$; it follows that R can be serviced by C with cost at most $O(1)$OPT. The algorithm then clusters the outliers $X \setminus R$ using the black box k-median algorithm; since the optimum centers can induce a clustering of cost at most 2OPT on the nodes in $X \setminus R$, the cost of clustering the outliers with βk extra medians will be at most $O(1)$OPT.

The algorithm runs in time $\tilde{O}(kn)$, since it can run $\tilde{O}(n^2)$ time bicriterion black box algorithms for k-median on instances of size $\tilde{O}(\sqrt{kn})$.

2 The Algorithm: Pass-Efficient FL

In this section, we prove the main result of the paper:

Theorem 1 *With probability at least $1 - 1/n$, $3(\ell - 1)$ pass algorithm* **Pass-Efficient FL** *will output a solution with cost at most $O(\ell)OPT$; if the solution opens k^* facilities, then the extra space used is at most $O(k^* n^{2/\ell} \log^2 n \log(\max_i f_i))$.*

Remark The space complexity has a factor of $\log(\max_i f_i)$, which may appear strange. We note, however, that any algorithm that reads each facility cost must use at least $\log(\max_i f_i)$ bits of memory.

We first give a high-level overview of our iterative algorithm **Pass-Efficient FL**. At each iteration it takes a sample, and computes a facility location solution on the sample. The algorithm identifies and removes from consideration those points of the *entire datastream* that are serviced well by the solution on the *sample*, and iterates on points not serviced well. In subsequent iterations, the algorithm considers finer and finer samples of the remaining points (i.e. elements in the ith iteration are picked with probability roughly $n^{(i+1)/\ell}/n$), until the last iteration, where $i = \ell - 1$, when we sample with probability 1.

More specifically, in Step 1 of iteration i, **Pass-Efficient FL** takes a sample S_i of the datastream X_i. In Steps 2 and 3 it computes an approximate facility location solution on S_i that opens a set of facilities F_i. In Step 4, the algorithm tests to see if the number of facilities in F_i is small. If so, then each cluster of S_i on average must be large; intuitively this condition will be enough to show that the facilities F_i opened by clustering the sample S_i are in fact a good solution for a very large subset, $R_i \subset X_i$, of the points from X_i. In the next iteration we can recurse on the much smaller set of points $X_{i+1} = X_i \setminus R_i$ for which F_i is a poor solution. In this case, the number of samples that we take (and hence the space complexity of our algorithm) will not change from this iteration to the next: although the next iteration will take a finer sample, this will be balanced by its having many fewer points in X_{i+1} to consider. If the number of clusters in F_i is large, then we cannot make any guarantees about how well F_i will cluster X_i, and we must recurse on almost the entire datastream. In this case, the number of samples taken in the next iteration will increase by a factor of $n^{1/\ell}$.

Our algorithm uses a black-box facility location algorithm with approximation ratio α that uses a linear amount of space (for example, [7, 16]). It also uses subroutine $\mathbf{One-PassSelect}(Y, k)$, where Y is a datastream of numbers and k an integer. $\mathbf{One-PassSelect}(Y, k)$ will return an element of Y of rank between $|Y| - k$ and $|Y| - 2k$ with probability $1 - 1/(3\ell n)$, using at most $O((|Y|/k) \log n)$ bits of memory. We discuss this algorithm in Section 2.1

As a preprocessing step, we round the facility cost of each node down to the nearest power of 2 and solve this modified instance, as in [16]. By rounding each facility cost, we have ensured that there are at most $\log \max_i f_i$ different values of facility costs; if we run an algorithm with approximation ratio β with these facility costs, the resulting solution will have an approximation ratio of at most 2β for the original facility costs.

Algorithm **Pass-Efficient FL**:
Initialize: $X_1 \leftarrow X$, $n_1 \leftarrow n$, $F_{\text{approx}} \leftarrow \emptyset$.
Main Loop. For $p = 1, \ldots, \ell - 1$, do:

1. In a single pass, draw a sample of X_p uniformly at random with replacement of size

$$s_p \geq c \frac{n^{(p+1)/\ell}}{n} \cdot n_p \log n,$$

   where c is the constant from Lemma 3. Call this sample S_p. If $p = \ell - 1$, just set $S_{\ell-1} \leftarrow X_{\ell-1}$.

2. In a second pass, for each node $x \in S_p$, find the closest node in X with facility cost f_i, for all distinct facility costs. Let F'_p be the set containing all these nodes.

3. Construct the following instance of facility location on a subset of X: metric space $(S_p \cup F'_p, d')$ where $d'(x, y) = (n_p/s_p) \cdot d(x, y)$, with facility costs the same as in X. Let the demand for each point of S_p be 1, and the demand of each point in $F'_p \setminus S_p$ be 0.

   Solve the instance using a black box algorithm with approximation ratio α. Let F_p be the output.

4. (a) If $|F_p| < s_p/(cn^{1/\ell} \log n)$ and $p \neq \ell - 1$, run algorithm **One-Pass Select** to find an element $e \in X_p$ such that $d(e, F_p)$ is between the $n_p - n_p \cdot n^{-1/\ell}$ and $n_p - \frac{1}{2} n_p \cdot n^{-1/\ell}$ elements of $x \in X_p$ with largest distance $d(x, F_p)$. Set $R_p \leftarrow \{x : x \in X_p, d(x, F_p) \leq d(e, F_p)\}$.

   (b) If $|F_p| \geq s_p/(cn^{1/\ell} \log n)$ and $p \neq \ell$ set $R_p \leftarrow \{x : x \in S_p\}$.

   (c) If $p = \ell - 1$, set $R_{\ell-1} \leftarrow X_{\ell-1}$.

5. Set $X_{p+1} \leftarrow X_p \setminus R_p$, $n_{p+1} \leftarrow |X_{p+1}|$, $F_{\text{approx}} \leftarrow F_{\text{approx}} \cup F_p$, and $p \leftarrow p + 1$.

Output the facilities F_{approx}.

Remarks For the purposes of our arguments, we will assume that the points in R_p get serviced by the facilities in F_p, although there may be better facilities found in other iterations.

At the $\ell - 1$th iteration of the algorithm, we are at the special bottom case where our "sample" is just the entire remainder of the datastream $X_{\ell-1}$.

Note that in Step 1, we assume we know the value of n_p. We don't know the value exactly, but know it to within a factor of 2, which is enough to implement Step 1.

In order to prove Theorem 1, we first bound the cost of F_{approx} on X in Lemma 4, then bound the amount of memory used by **Pass-Efficient FL** in Lemma 5.

Definition 1 *Let service*$(S, F) = \sum_{x_i \in S} d(x_i, F)$ *denote the* service cost *of servicing points S with facilities F. Define cost*$(S, F, \eta) = \sum_{x_i \in F} f_i + \eta$*service*$(S, F)$ *to be the total cost of servicing the set of points S with the facilities F, with distance scaled up by a factor of η.*

In order to bound the cost of the solution, $\text{cost}(X, F_{\text{approx}}, 1)$, we first present the following lemma, which is partially adapted from an argument from [15]:

Lemma 2 *For fixed p, with probability at least $1 - 1/(10\ell n^2)$, cost*$(S_p, F_p, n_p/s_p) \leq 12\alpha\, OPT$.

Proof: Fix an optimum solution on X_p that opens a set of facilities $F^* \subseteq X$, with $\text{cost}(X_p, F^*, 1) = \text{OPT}_p \leq \text{OPT}$. The cost of these facilities on the sample S_p with distances scaled by n_p/s_p as in Step 2 is: $\text{cost}(S_p, F^*, n_p/s_p) = \sum_{x_i \in F^*} f_i + \frac{n_p}{s_p} \sum_{x_i \in S_p} d(x_i, F^*)$. By the Markov inequality, we know that with probability at most 1/3,

$$\frac{n_p}{s_p} \sum_{x_i \in S_p} d(x_i, F^*) \geq 3 \sum_{x_i \in X} d(x_i, F^*).$$

Thus, with probability at least 2/3, we have $\text{cost}(S_p, F^*, n_p/s_p) \leq \sum_{x_i \in F^*} f_i + 3 \sum_{x_i \in X} d(x_i, F^*) \leq 3\text{OPT}$.

Note that possibly F^* contains points that are not in $S_p \cup F'_p$. Thus, we construct a set of facilities $F^{*'} \subset S_p \cup F'_p$. For each node $x_i \in F^*$, find the closest node in S_p, and add the closest facility of cost $f_i \in F'_p$ to $F^{*\prime}$. A standard argument will show that service$(S_p, F^{*\prime}) \leq 4$service$(S_p, F^*)$. Thus, cost$(S_p, F^{*\prime}, n_p/s_p) \leq 4$cost$(S_p, F^*, n_p/s_p) \leq 12$OPT. The black box algorithm will find a solution with cost at most 12αOPT.

We can boost the probability to $1 - 1/(10\ell n^2)$ by repeating Steps 1, 2 and 3 in parallel $O(\log n)$ times (note that one would never have the situation $\ell > n$), and choosing the solution with the smallest value of cost$(S_p, F, n_p/s_p)$.□

We will also use a less general version of Lemma 9 by Charikar *et al*, used for proving the correctness of a one pass algorithm for k-median clustering with outliers:

Lemma 3 *[8] Let S be a sample of s points taken uniformly at random from X, for $s \geq ck \log n/\epsilon$ for some constant c. Let C be a set of k centers on S. Then with probability at least $1 - \frac{1}{n^2}$, we can service at least a $1 - \epsilon/2$ fraction of the points of X with centers C at cost at most $2\frac{|X|}{|S|}$service(S, C).*

Lemma 4 *With probability at least $1 - 1/n$, cost$(X, F_{approx}, 1) \leq 48\alpha(\ell - 1)OPT$.*

PROOF: Since the R_ps form a partition of X, and $F_{\text{approx}} = \cup_p F_p$, we know cost$(X, F_{\text{approx}}, 1) \leq \sum_p$ cost$(R_p, F_p, 1)$.

Claim: cost$(R_p, F_p, 1) \leq 24\alpha$OPT.

PROOF OF CLAIM: Recall that S_p is a sample of size s_p from data stream X_p of size n_p. We have two cases. If in Step 4 of **Pass-Efficient FL** , we found that $|F_p| \geq s_p/(cn^{1/\ell} \log n)$, then the claim follows immediately from Lemma 2.

Otherwise, in Step 4 we found that $|F_p| < s_p/(cn^{1/\ell} \log n)$. In this case, we can view F_p as a set of $k = s_p/(cn^{1/\ell} \log n)$ centers. Since S_p is a sample of X_p of size at least $ckn^{1/\ell} \log n$ points, it follows from Lemma 3, that at least a $1-1/(2n^{1/\ell})$ fraction of X can be serviced with cost at most $2(n_p/s_p)\cdot$service(S_p, F_p). Since R_p is a set containing at most a $1 - 1/(2n^{1/\ell})$ fraction of the points of X_p closest to facilities in F_p, we have that service$(R_p, F_p, 1) \leq 2(n_p/s_p)$service$(S_p, F_p)$. Thus,

$$\text{cost}(R_p, F_p, 1) \leq 2\left(\sum_{x_i \in F_p} f_i + \frac{n_p}{s_p}\text{service}(S_p, F)\right) \leq 24\alpha\text{OPT},$$

where the last inequality follows from Lemma 2. □

With the claim, we have:

$$\text{cost}(X, F_{\text{approx}}, 1) \quad \leq \quad \sum_{p=1}^{\ell-1} \text{cost}(R_p, F_p, 1) \leq 24\alpha(\ell - 1)\text{OPT}.$$

Recall that we rounded the facility cost down to the nearest power of 2. This will increase the cost of the solution by at most a factor of two. Thus, F_{approx} is a $48\alpha(\ell - 1)$OPT approximation. □

Lemma 5 *The amount of memory used by* **Pass-Efficient FL** *is at most $O(k^* n^{2/\ell} \log^2 n \log(\max_i f_i))$, where k^* is the number of facilities output by the algorithm.*

PROOF: By induction, we will prove that $s_p \leq c(\max_{i<p} |F_i|) \cdot n^{2/\ell} \log n$ for $p \geq 1$. We will artificially set $|F_0| = 1$ (this is just for the base case analysis; there is no F_0 constructed by our algorithm). Since the final set of facilities output by the algorithm, F_{approx}, satisfies $|F_{\text{approx}}| \geq \max_{p \geq 0} |F_p|$, the lemma will follow (note the extra $(\log n \log(\max_i f))$ factor in the lemma, which is derived from the boosting in Lemma 2, and the fact that we must store the set F'_p from Step 2).

Assume the inductive hypothesis for $p \leftarrow m$: $s_m \leq c(\max_{i<m} |F_i|) \cdot n^{2/\ell} \log n$.

If at Step 5 of the algorithm, we had found that $|F_m| < s_m/(cn^{1/\ell}\log n)$, then $n_{m+1} \le n_m/n^{1/\ell}$. This implies that

$$s_{m+1} \le c\frac{n^{(p+1)/\ell}}{n} \cdot n_{m+1}\log n \le c\frac{n^{p/\ell}}{n} \cdot n_m \log n = s_m \le c\left(\max_{i\le m}|F_i|\right) \cdot n^{2/\ell}\log n.$$

If on the other hand at Step 5, we had found that $|F_m| \ge s_m/(cn^{1/\ell}\log n)$ then

$$s_{m+1} \le n^{1/\ell}s_m \le c\,|F_m|\,n^{2/\ell}\log n.$$

□

Combining Lemmas 4 and 5 proves Theorem 1.

Remark A technical condition that we have not addressed is how the metric space is presented to us in the datastream. Possibly distances can be given by a function of the nodes (for instance, Hamming or Euclidean distance), in which case our algorithm does not need to be adjusted. A general approach would be that each point is represented by an $(n-1)$-dimensional vector giving its distance to every other point in X; in this case, when we sample the points in Step 1, we do not want to save the entire $(n-1)$ dimensional vector for each sample point. Instead, in the first pass we would draw our sample and in the second pass we could then store an $(|S_p|-1)$-dimensional vector for each node that contains the distance between the node and the $|S_p|$ nodes in the sample. Note this would then require on the order of $(k^*)^2 n^{4/\ell}$ integers of working memory, but correspondingly X would contain n^2 integers.

2.1 Approximate selection in one pass

In this section we describe the one pass subroutine **One-Pass Select** used by **Pass-Efficient FL** in Step 4. The input to the algorithm is a datastream of numbers Y and an integer k, and it outputs an element of Y of rank between $|Y|-2k$ and $|Y|-k$.

Of related interest is an algorithm of Munro and Paterson [17], who give a P pass algorithm for selecting the kth largest element of a datastream with n elements, using extra space at most $O(n^{1/P}\log^{2-2/P} n)$, and prove a nearly matching lower bound for deterministic algorithms. **One-Pass Select** avoids making more than one pass by using randomization and guaranteeing only approximate selection.

Algorithm **One-Pass Select**.
Input: Datastream Y of m numbers, and an integer k.

1. In a single pass, take a sample S uniformly at random with replacement of size $s = c(m/k)\log(1/\delta)$ from the datastream.
2. Sort S and output the element with rank $s - \frac{3}{2}\frac{k}{m}s$.

Lemma 6 One-Pass Select *will find an element with rank between $m-k$ and $m-2k$ with probability at least $1-\delta$ using at most $c(m/k)\log(1/\delta)$ bits of extra memory.*

PROOF: Let $y_1, \ldots, y_m$ be the sorted elements of Y, breaking ties arbitrarily. Define index$(\cdot)$, to be the function that takes an element $z \in Y$ and maps it to an index such that $z = y_{\text{index}(z)}$. Let e be the element output by the algorithm, and $q = index(e)$.

We first upper bound $\Pr[q \ge m-k]$. Call s_i the ith element chosen uniformly at random for the sample. Let X_i be a random variable that is 1 if index$(s_i) \ge m-k$ and 0 otherwise. Since $\mathrm{E}[X_i] = k/m$, an application of the Chernoff bound yields:

$$\Pr\left[\sum_i X_i \ge \left(1+\frac{1}{2}\right)\frac{k}{m}s\right] \le \frac{\delta}{2}.$$

Note that $q \geq m-k$ implies that at least $3/2(k/m)s$ of the elements of S have indices that are greater than $m-k$; this corresponds to the event that $\sum X_i \geq \frac{3}{2}\frac{k}{m}s$ Thus, $\Pr[q \geq m-k] \leq \delta/2$.

We next lower bound $\Pr[q \leq m-2k]$. Let Y_i be a random variable that is 1 if $\text{index}(s_i) \geq m-2k$ and 0 otherwise. Since $\mathrm{E}[Y_i] = 2k/m$, an application of the Chernoff bound yields:

$$\Pr\left[\sum_i Y_i \leq \left(1-\frac{1}{10}\right)\frac{2k}{m}s\right] \leq \frac{\delta}{2}.$$

Note that $q \leq m-2k$ implies that $\sum Y_i \leq \frac{9}{10}\frac{2k}{m}s$. Thus, $\Pr[q \leq m-2k] \leq \delta/2$.

It follows that $\Pr[m-2k \leq q \leq m-k] \geq 1-\delta$. □

3 k-Facility Location

The above algorithm can be adapted to solving the following hybrid between k-median and facility location:

Problem 2 (k-Facility Location) *Find a set $F \subset X$, such that $|F| \leq k$, that minimizes the objective function: $\sum_{x_i \in F} f_i + \sum_{x_i \in X} d(x_i, F)$.*

Note that k-median is a special case of k-facility location where $f_i = 0$ for all i.

Theorem 7 *There exists a 3ℓ pass algorithm with approximation ratio $O(\ell)$ for k-facility location using extra space at most $O(\ell k^{(\ell-1)/\ell} n^{1/\ell} \log^2 n \log(\max_i f_i))$.*

The algorithm is simpler than **Pass-Efficient FL**, and can be considered a generalization of Indyk's algorithm that we outlined in Section 1.3 to multiple passes for k facility location.

Algorithm: **k-FL**
Initialize $X_1 \leftarrow X$, $p \leftarrow 1$, $F \leftarrow \emptyset$.
For $p = 1, \ldots, \ell$, do:

1. In a single pass, draw a sample S_p of size $s = ck^{(\ell-1)/\ell} n^{1/\ell} \log n$ from the datastream X_p. For $p = \ell$, just set $S_\ell \leftarrow X_\ell$.
2. Using a second pass, run an α approximation algorithm on S_p taking into consideration all facilities in X exactly as in **Pass-Efficient FL**. Call the output F_p.
3. Call **One-Pass Select** to find a point $e \in X_p$ such that e is within the $k^{p/\ell} n^{1-p/\ell}$ and $(1/2)k^{p/\ell} n^{1-p/\ell}$ points of X_p that are farthest away from F_p. Set $R_p \leftarrow \{x : x \in X_p, d(x, F_p) < d(e, F_p)\}$.
4. Set $X_{p+1} \leftarrow X_p \setminus R_p$, $F \leftarrow F \cup F_p$, and $p \leftarrow p+1$.

Run a k-median algorithm on the set F, with each $x_i \in F$ weighted by the number of points in X that it services.
Output the resulting set of k facilities as F_{approx}.

Proof of Theorem 7: If the black box k-median and facility location algorithms use only $\tilde{O}(n)$ space, then it is clear that the above algorithm will use at most $\tilde{O}(\ell k^{(\ell-1)/\ell} n^{1/\ell})$ extra space. Thus, in order to prove Theorem 7, we just need to bound $\text{cost}(X, F_{\text{approx}}, 1)$. We first bound the cost of the set of facilities F prior to running the k-median approximation algorithm as before:

$$\text{cost}(X, F, 1) \leq \sum_p \text{cost}(R_p, F_p, 1). \tag{1}$$

Claim $\text{cost}(R_p, F_p, 1) \leq O(\alpha)\text{OPT}$.

PROOF OF CLAIM: Since Steps 1 and 2 are identical to Steps 1, 2, and 3 in **Pass-Efficient FL**, Lemma 2 applies to the situation and we have: $\text{cost}(S_p, F_p, n_p/s) \leq 12\alpha\text{OPT}$.

Fix an optimum k-facility location solution on X_p with facilities $C = \{c_1, \ldots, c_k\}$. Let C_i be the set of points in X_p that are serviced by c_i and let $I = \left\{i : |C_i| \geq \frac{1}{2}k^{-(\ell-p)/\ell}n^{1-p/\ell}\right\}$ be the set of indices corresponding to large clusters in C. Note that $|X_p| \leq k^{(p-1)/\ell}n^{1-(p-1)/\ell}$. Since we are drawing a sample of size $s = ck^{(\ell-1)/\ell}n^{1/\ell}\log n$ points from X_p, we will have $|X_p|/s < (1/c)k^{-(\ell-p)/\ell}n^{1-p/\ell}\log n$; thus with an appropriately chosen constant c, $S_p \cap C_i$ for $i \in I$ will be "large" with high probability, and intuitively if we cluster S_p, a large C_i should be serviced well. Indeed, we can prove with an argument that is exactly the same as the argument in [15] (but with the addition of facility costs) that

$$\text{cost}(\cup_{i\in I}C_i, F_p, 1) \leq O(\alpha)\text{OPT}.$$

Note that the number of points of X_p that do not fall in $\cup_{i\in I}C_i$ is at most $k \cdot \frac{1}{2}k^{-(\ell-p)/\ell}n^{1-p/\ell} = \frac{1}{2}k^{p/\ell}n^{1-p/\ell}$. Since the set R_p excludes at least the $\frac{1}{2}k^{p/\ell}n^{1-p/\ell}$ points of X_p that are farthest away from facilities in F_p, it follows that $\text{cost}(R_p, F_p, 1) \leq \text{cost}(\cup_{i\in I}C_i, F_p, 1) \leq O(1)\text{OPT}$. □

The claim, along with Equation (1), implies that $\text{cost}(X, F, 1) \leq O(\ell)\text{OPT}$.

Recall that F_{approx} is the set of k facilities derived from running a constant factor approximation algorithm for k-median on F, with each point in F weighted by the number of points in X that it services. Reasoning similar to the proof of Theorem 2.3 from [13] will prove that $\text{service}(X, F_{\text{approx}}) = O(1)\text{OPT}$. Lastly, since $F_{\text{approx}} \subseteq F$, it follows that $\sum_{x_i\in F_{\text{approx}}} f_i \leq \sum_{x_i\in F} f_i \leq O(1)\text{OPT}$.

Thus, $\text{cost}(X, F_{\text{approx}}, 1) = O(1)\text{OPT}$. □

4 Lower bounds for multiple pass clustering

Using the communication complexity of the set disjointness problem, we can prove the following lower bound:

Theorem 8 *Any ℓ pass randomized algorithm that, with probability at least $2/3$, computes an $O(n^m)$ approximation to the cost of the optimum solution to facility location must use at least $\Omega(n/\ell)$ bits of memory.*

PROOF: Suppose Alice and Bob have subsets $A, B \subseteq [n]$ and want to determine whether $|A \cap B| \geq 1$ or $|A \cap B| = 0$. Let $a, b \in \{0,1\}^n$ be the characteristic vectors of the sets A and B, i.e. $a_i = 1$ iff $i \in A$, $b_i = 1$ iff $i \in B$.

Given sets A and B, construct the following instance of facility location with $2n$ nodes x_i^a, x_i^b for $i = 1, \ldots, n$: $d(x_i^a, x_i^b) = 1$ for all i, and all other distances are arbitrarily large. If $a_i = 0$, set $f_{x_i^a} = 1$. If $a_i = 1$, set $f_{x_i^a} = n^{m+1}$. Similarly, If $b_i = 0$, set $f_{x_i^b} = 1$. If $b_i = 1$, set $f_{x_i^b} = n^{m+1}$.

It follows that if A and B are disjoint, then the instance has cost $2n$. Otherwise, the instance has cost at least n^m.

Suppose there exists an ℓ pass facility location algorithm FL that uses at most $M(FL)$ bits of memory. We now give a standard argument that shows how Alice and Bob can compute the disjointness function using at most $(2\ell-1)M(FL)$ bits of communication. Alice and Bob each have enough information to construct a different half of the datastream that would result from the facility location instance associated with a and b. Alice simulates the first pass of FL on her half of the data stream, and sends the contents of her $M(FL)$ bits of memory to Bob. Bob then continues the pass on his data stream, and sends his $M(FL)$ bits of memory back to Alice. They simulate all ℓ passes in this fashion, sending a total of $2\ell - 1$ messages of size at most $M(FL)$ to each other. At the end of the communication, Bob will have an approximation to the cost of the facility location instance and will thus be able to compute the disjointness.

Since the randomized communication complexity of the disjointness problem is known to be $\Omega(n)$, it follows that $M(FL) = \Omega(n/\ell)$. □

5 Conclusion

We have presented pass-efficient algorithms for the facility location problem that use multiple passes and small space. Furthermore, the algorithm exhibits a tradeoff in the amount of space used and the number of passes taken, such that the amount of space falls, in some sense exponentially, with the number of passes taken.

Open questions arising from this work include: Can we design streaming or pass-efficient algorithms for facility location that use space less than $\tilde{O}(k^* n^{2/\ell})$? Can we change the dependence from k^* to perhaps k_{OPT}, where k_{OPT} is the number of facilities opened by an optimum solution? (Since k_{OPT} need not be smaller than k^*, this is not necessarily better in terms of resource usage; rather it is in some sense more aesthetically pleasing). Lastly, our lower bound used instances of facility location that required $\omega(n)$ facilities for any approximation. Is it possible to prove lower bounds on instances that require only k facilities but that require at least $\Omega(k)$ space to solve by a pass-efficient algorithm?

References

[1] G. Aggarwal, M. Datar, S. Rajagopalan, and M. Ruhl. On the streaming model augmented with a sorting primitive. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*, pages 540–549, 2004.

[2] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing*, pages 20–29, 1996.

[3] S. Arora, P. Raghavan, and S. Rao. Approximation schemes for euclidean k-medians and related problems. In *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing*, pages 106–113, 1998.

[4] M. Badoiu, A. Czumaj, P. Indyk, and C. Sohler. Facility location in sublinear time. In *32nd International Colloquium on Automata, Languages, and Programming*, pages 866–877, 2005.

[5] K. Chang and R. Kannan. The space complexity of pass-efficient algorithms for clustering. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2006.

[6] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing*, pages 626–635, 1997.

[7] M. Charikar and S. Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, pages 378–388, 1999.

[8] M. Charikar, L. O'Callaghan, and R. Panigrahy. Better streaming algorithms for clustering problems. In *Proceedings of the 35th Annual ACM Symposium on the Theory of Computing*, pages 30–39, 2003.

[9] M. Charikar and R. Panigrahy. Clustering to minimize the sum of cluster diameters. In *Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing*, pages 1–10, 2001.

[10] P. Drineas and R. Kannan. Pass efficient algorithm for large matrices. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 223–232, 2003.

[11] J. Feigenbaum, S. Kannan, A. McGregor, and S. Suri. Graph distances in the streaming model: The value of space. In *Proceedings of the 16th Symposium on Discrete Algorithms*, pages 745–754, 2005.

[12] S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 649–657, 1998.

[13] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *Proceedings of the 32nd Annual ACM Symposium on the Theory of Computing*, pages 359–366, 2000.

[14] M. R. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. *External memory algorithms*, pages 107–118, 1999.

[15] P. Indyk. Sublinear time algorithms for metric space problems. In *Proceedings of the 31st Annual ACM Symposium on the Theory of Computing*, pages 428–434, 1999.

[16] A. Meyerson. Online facility location. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 426–431, 2001.

[17] J. I. Munro and M. Paterson. Selection and sorting with limited storage. *Theoretical Computer Science*, 12:315–323, 1980.

[18] D. B. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems (extended abstract). In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing*, pages 265–274, 1997.