Consistent k-Clustering

Silvio Lattanzi [1] **Sergei Vassilvitskii** [2]

Abstract

The study of online algorithms and competitive analysis provides a solid foundation for studying the quality of irrevocable decision making when the data arrives in an online manner. While in some scenarios the decisions are indeed irrevocable, there are many practical situations when changing a previous decision is not impossible, but simply expensive. In this work we formalize this notion and introduce the *consistent k-clustering* problem. With points arriving online, the goal is to maintain a constant approximate solution, while minimizing the number of reclusterings necessary. We prove a lower bound, showing that $\Omega(k \log n)$ changes are necessary in the worst case for a wide range of objective functions. On the positive side, we give an algorithm that needs only $O(k^2 \log^4 n)$ changes to maintain a constant competitive solution, an exponential improvement on the naive solution of reclustering at every time step. Finally, we show experimentally that our approach performs much better than the theoretical bound, with the number of changes growing approximately as $O(\log n)$.

1. Introduction

Competitive analysis of online algorithms has been an area of spirited research with beautiful results over the past two decades. At its heart, this area is about decision making under uncertainty about the future—the input is revealed in an online manner, and at every point in time the algorithm must make an irrevocable choice. A standard example is that of caching algorithms—at every time step the algorithm must make a choice about which elements to keep in the cache, and which elements to evict (Fiat et al., 1991). The generalization of caching to metric spaces is encapsulated in the k-server problem, which has been the subject of intense study (Bansal et al., 2015; Manasse et al., 1990).

The key metric in online algorithms is the *competitive ratio*. It measures the quality of the solution obtained by an online algorithm versus an offline optimum, which has the luxury of seeing the whole input before making any decisions. In situations where the competitive ratio is relatively small, for example, the list update problem (Sleator & Tarjan, 1985), this is a great measure by which we can compare different algorithms. However, in some scenarios strong lower bounds on the competitive ratio imply that any algorithm that makes irrevocable choices will necessarily perform poorly when compared to an offline optimum.

Online clustering is one such example. In this setting points $x_1, x_2, \ldots$ arrive one at a time, and must be instantly given one of k cluster labels. As is typical, the goal is to have the highest quality clustering (under some pre-specified objective function, like k-CENTER or k-MEDIAN) at every point in time. As Liberty et al. (2016) showed, not only do online clustering algorithms have an unbounded competitive ratio, but one must use bi-criteria approximations to have any hope of a constant approximate solution.

Another approach to evade strong lower bounds is to make additional assumptions about the input to the problem. For example, one may assume that the input comes in a random (or partially random) order. This assumption has been a fruitful avenue when studying online problems in different contexts, as the classic secretary problem (Ferguson, 1989; Kesselheim et al., 2015; Kleinberg, 2005) or matching (Karp et al., 1990; Mahdian & Yan, 2011). Another alternative is to assume some additional structure on the distribution that points are coming from (Feldman et al., 2009). A big downside of both of these assumptions is that they are hard to test and validate in practice, which is why we take a different approach in this work.

1.1. Consistency

While the irrevocability of past choices makes sense from a theoretical standpoint, for some practical problems this requirement is unrealistically draconian. For example, consider a load balancer, which, when faced with requests arriving online, assigns them to different machines. Better cache performance dictates that similar requests should be

*Equal contribution [1]Google, Zurich, Switzerland [2]Google, New York, New York, USA. Correspondence to: Silvio Lattanzi <silviol@google.com>, Sergei Vassilvitskii <sergeiv@google.com>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

assigned to the same machine, thus the load balancer is essentially performing online clustering. However, fundamentally, nothing is preventing the load balancer from reassigning some of the past jobs to other machines. In this situation, a re-clustering—a reassignment of jobs to machines to increase performance—is not an impossible operation.

Another common example of a costly, but not prohibitive recomputation comes from standard applications of unsupervised clustering: feature engineering for large scale machine learned systems. In this setting a feature vector x, is augmented with the id of a cluster it falls in, x', and the full vector (x, x') is given as input to the learner. This is mainly done to introduce expressiveness and non-linearity to simple systems. In this situation, changing the clustering would entail changing the set of features passed to the learner, and retraining the whole system; thus one certainly does not want to do it at every time step, but it can be done if the gains are worthwhile.

From a theoretical perspective, the ability to correct for past mistakes offers the ability for much better solutions. In particular for clustering problems, it avoids the lower bounds introduced by Liberty et al. (2016). As we will show, the option to recluster dramatically improves the quality of the solution, even if it is taken rarely. More formally, we will introduce a parameter β which controls the number of times the solution changes. Setting $\beta = 0$ is equivalent to online algorithms, whereas a large value of β is equivalent to recomputing the answer from scratch at every time step.

1.2. Our Contributions

In this paper we focus on exploring the trade-off between the approximation ratio of clustering algorithms, and the number of times we must recompute the results.

We begin by formally defining the notion of (α, β)-consistent clustering in Section 3. Then we prove a lower bound, showing that any constant competitive algorithm must change its cluster centers at least $\Omega(k \log n)$ times (Section 3.1). Then we show that a known algorithm by Charikar et al. (2004) achieves this bound for the k-CENTER problem, and we develop a new algorithm for other clustering objectives, and show that it requires at most $O(k^2 \log^4 n)$ reclusterings, an exponential improvement over the naive solution (Section 5). Finally, we show that the proposed algorithms perform well on real world datasets (Section 7).

1.3. Related Work

There are two avenues for related work that we build on in this paper. The first is clustering algorithms, particularly the online clustering variants. In their seminal work Charikar et al. (2004) gave algorithms for the k-CENTER problem. The case of k-MEDIAN and k-MEANS proved more complex. For the former, Meyerson (2001) gave an $O(\log n)$ competitive ration for closely related online facility location problem. This result was further improved by Fotakis (2008) and Anagnostopoulos et al. (2004). The latter was recently studied by Liberty et al. (2016) who gave bicriteria approximations and showed that these are necessary in an online setting. For the soft partition version of the k-clustering problem, an Expectation Maximization algorithm was suggested by Liang & Klein (2009).

The second, closely related area, is that of streaming algorithms. The literature of clustering in the streaming model is very rich, we highlight the most relevant results. The first paper to study clustering problem is by Charikar et al. (2004) studying the k-CENTER problem. Guha et al. (2000) give the first single pass constant approximation algorithm to the k-MEDIAN variant. Subsequently their result has been is improved by Charikar et al. (2003). Finally, the best algorithm for the closely related variant of facility location is due to Czumaj et al. (2013), who gave a $(1+\epsilon)$-approximation for the problem.

2. Preliminaries

Let X be a set of n points, and $d : X \times X \to \mathbb{R}$ a distance function. We assume that d is symmetric and that (X, d) form a metric space, that is $d(x, x) = 0$ for any $x \in X$; $d(x, y) = d(y, x) \geq 0$ for any $x, y \in X$; and, for any $x, y, z \in X$, $d(x, y) \leq d(x, z) + d(z, x)$. Finally, by scaling d, let $\min_{x,y} d(x, y) = 1$ and denote by Δ the maximum pairwise distance, $\max_{x,y} d(x, y)$. We will assume that Δ is bounded by a polynomial in n, therefore $\log \Delta = O(\log n)$.

Consider a set of k points $\mathsf{c} = \{\mathsf{c}_1, \mathsf{c}_2, \ldots, \mathsf{c}_k\} \subseteq X$, which we will refer to as *centers*. For each c_i, let $C_i \subseteq X$ be the set of points in X closer to c_i than to any other center $\mathsf{c} \in C$. [1] Formally, $C_i = \{x \in X \mid d(x, \mathsf{c}_i) \leq \min_{\mathsf{c} \in \mathsf{c}} d(x, \mathsf{c})\}$.

Given a $p > 0$, in the rest of the paper we refer to the cost of a point x with to respect to a set of centers as: $\mathsf{cost}_p(x, \mathsf{c}) = \min_{\mathsf{c}_i} d(x, \mathsf{c}_i)^p$. And cost of a cluster C_i as: $\mathsf{cost}_p(X, C_i) = \sum_{x \in C_i} d(x, \mathsf{c}_i)^p$.

Now we are ready to define our problem. For any $p > 0$ we can define the cost of clustering of points X with respect to the centers $\mathsf{c} \subseteq X$ as: $\mathsf{cost}_p(X, \mathsf{c}) = \sum_{x \in X} \mathsf{cost}_p(x, \mathsf{c}) = \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mathsf{c}_i)^p$.

The k-clustering family of problems asks to find the set of centers c that minimize cost_p for a specific p. When $p = 1$, $\mathsf{cost}_1(X, \mathsf{c})$ is precisely the k-MEDIAN clustering

[1] For clarity of the exposition we will assume that all of the pairwise distances are unique. The results still hold when ties are broken lexicographically.

objective. Setting $p = 2$ is equivalent to the k-MEDOIDS problem[2]. Finally, with $p = \infty$, we recover the k-CENTER problem, which asks to minimize the maximum distance of any point to its nearest cluster center.

Observe that although $d(\cdot, \cdot)$ satisfies the triangle inequality, when raised to p-th power we need to relax the condition. In particular we have that for any $x, y, z \in X$: $d(x, y)^p \leq 2^{p-1}(d(x, z)^p + d(z, y)^p)$.

When p is clear from the context, we will refer to $\mathsf{cost}_p(X, \mathfrak{c})$ as the cost of the clustering and denote it $\mathsf{cost}(X, \mathfrak{c})$. We will us $\mathsf{opt}_p(X)$ to denote the optimum cost for the metric space (X, d). We will use $\mathfrak{c}^* = \{\mathsf{c}_1^*, \mathsf{c}_2^*, \ldots, \mathsf{c}_k^*\}$ to denote the optimal solution.

The k clustering problem is NP-hard to solve exactly, thus we consider approximate solutions. We say that a clustering generated from a set of centers $\mathfrak{c}$ is α-approximate if $\mathsf{cost}_p(X, \mathfrak{c}) \leq \alpha \cdot \mathsf{opt}_p(X)$. The best known approximation factors are 2 for the k-CENTER problem (Gonzalez, 1985), $1 + \sqrt{3} + \epsilon$ for the k-MEDIAN problem (Li & Svensson, 2016), and $9 + \epsilon$ for the k-MEDOIDS problem (Kanungo et al., 2004).

3. Consistency

As noted in the introduction, in many online clustering applications the choices made by the online algorithm are not irrevocable, but simply expensive to change. Moreover, by allowing a small number of full recomputations, we can circumvent the stringent lower bounds on competitive ratio for online clustering.

To this end, our goal in this work is to better understand the trade-off between the approximation ratio of online clustering algorithms, and the number of times the representative centers change.

We focus on a dynamic setting where the points arrive sequentially. Let x_t denote the point that arrives at time t, and denote by X_t the set of points that has arrived from the beginning. Thus $X_0 = \emptyset$, and $X_{i+1} = X_i \cup \{x_{i+1}\} = \{x_1, x_2, \ldots, x_{i+1}\}$.

For any two sets of centers $\mathfrak{c}, \mathfrak{c}'$ let $|\mathfrak{c} - \mathfrak{c}'|$ denote the number of elements present in $\mathfrak{c}$, but not in $\mathfrak{c}'$: $|\mathfrak{c} - \mathfrak{c}'| = |\mathfrak{c} \setminus (\mathfrak{c} \cap \mathfrak{c}')|$. Observe that when $\mathfrak{c}$ and $\mathfrak{c}'$ have the same cardinality, $|\mathfrak{c} - \mathfrak{c}'| = |\mathfrak{c}' - \mathfrak{c}|$.

Definition 3.1. *Given a sequence of sets of centers, $\mathfrak{c}_0$, $\mathfrak{c}_1$, ..., $\mathfrak{c}_t$ and a positive monotone non-decreasing function $\beta : \mathbb{Z} \to \mathbb{R}$, we say that the sequence is β-consistent if for all T, $\sum_{t=1}^{T} |\mathfrak{c}_t - \mathfrak{c}_{t-1}| \leq \beta(T)$.*

In other words, a sequence is β-consistent, if at time T at most $\beta(T)$ centers have changed between successive sets.

Definition 3.2. *Given a sequence of points $x_1, x_2, \ldots, x_T$, and a parameter p, a sequence of centers $\mathfrak{c}_1, \mathfrak{c}_2, \ldots, \mathfrak{c}_T$ is (α, β)-consistent if:*
(i) **Approximation.** *At every time t, the centers $\mathfrak{c}_t$ form an α approximate solution to the optimum solution at that time: $\mathsf{cost}_p(X_t, \mathfrak{c}_t) \leq \alpha \cdot \mathsf{opt}_p(X_t)$ for all $t \leq T$.*
(ii) **Consistency.** *The sets of centers form a β-consistent sequence.*

3.1. A lower bound

Before we look for (α, β) consistent algorithms it is useful to understand what values are possible. We show that it is impossible to get a constant approximation and achieve consistency of $o(\log n)$ for any of the k clustering problems. Later, in Section 6 we will give a non-constructive result that shows that there is always a sequence of clusterings that is simultaneously constant-approximate and $O(k \log^2 n)$ consistent.[3]

Lemma 3.3. *There exists a sequence of points such that for any constant $\alpha > 0$, any algorithm that returns an α-approximate solution while processing n points must be $\Omega(k \log n)$-consistent.*

Proof. For ease of exposition, assume that $p = 1$, and consider points lying in $(k-1)$-dimensional Euclidean space, $\mathbb{R}^{k-1}$. We begin by adding a point x_0 at the origin, and points $x_1, \ldots, x_{k-1}$ in positions $e_1, e_2, \ldots, e_{k-1}$, where e_j is the standard basis vector that is 1 in the j-th dimension, and 0 everywhere else.

We then proceed in phases, where in phase $1 \leq i < \log n$ we add points at position $(\gamma)^i \cdot e_j$ for each $j \in [1, k-1]$, for some $\gamma > 0$ that we will set later. In phase $\log n$ we add the remaining $n - (k-1)\log n - 1$ points at arbitrary positions within the convex hull of already added points.

Let P_i be the set of points at the end of phase i. Consider any algorithm that returns an α-approximate solution on P_i. Let $p_1, p_2, \ldots, p_{k-1}$ be the points added to the input during phase i, $p_j = \gamma^i \cdot e_j$. Then $P_i = P_{i-1} \cup \{p_1, \ldots, p_{k-1}\}$. One feasible solution choses as centers the points added in phase i as well as the origin, $C = \{p_1, p_2, \ldots, p_{k-1}, 0\}$.

For every point in P_{i-1} the origin is closer than any of the other centers, therefore the total cost is: $\mathsf{opt}(P_i) \leq \mathsf{cost}(P_i, C) = (k-1)\sum_{z=1}^{i-1} \gamma^z \leq (k-1)\frac{\gamma^i - 1}{\gamma - 1}$. On the other hand, consider a set of centers $\mathfrak{c}'$ that does not include some $p_j = \gamma^i e_j$. The closest point to p_j is at $\gamma^{i-1} e_j$, which is at distance $\gamma^{i-1}(\gamma - 1)$ away. There-

[2] In the Euclidean space if the centers do not need to be part of the input, setting $p = 2$ recovers the k-MEANS problem.

[3] Note that we assume throughout the paper that the maximum distance between any two points, Δ, is polynomial in n. Alternatively we can restate the lower bound in this section as a $\Omega(k \log \Delta)$ upper bound in section 6 as a $O(k \log^2 \Delta)$.

fore, $\mathsf{cost}(P_i, \mathfrak{c}') \geq \mathsf{cost}(\{p_j\}, \mathfrak{c}') = \gamma^{i-1}(\gamma - 1)$. If $\gamma \geq (2 + k\alpha)$ then we can bound the approximation ratio as: $\frac{\mathsf{cost}(P_i, \mathfrak{c}')}{\mathsf{opt}(P_i)} \geq \frac{\gamma^{i-1}(\gamma-1)}{(k-1)\frac{\gamma^i-1}{\gamma-1}} \geq \frac{\gamma^{i-1}(\gamma-1)^2}{(k-1)\gamma^i} \geq \frac{\gamma-2}{k-1} > \alpha$ so $\mathfrak{c}'$ cannot be an α-approximate solution. Therefore at the end of phase $1 \leq i < \log n$, any α-approximate set of centers, must include all points added in phase i. Thus any sequence of sets of centers must be $\Omega(k \log n)$-consistent.

Note that considering any $p > 1$ only makes any omission of point p_j even more costly, as compared to the optimum solution. □

4. Warm up: k-CENTER Clustering

To gain some intuition about consistent clustering, we begin with the k-CENTER objective. Given a dataset X, the goal is to identify k centers $\mathfrak{c} = \{\mathsf{c}_1, \ldots, \mathsf{c}_k\}$ that minimize: $\max_{x \in X} \min_{\mathsf{c} \in \mathfrak{c}} d(x, \mathsf{c})$. This problem is known to be NP-hard, but a simple 2-approximation algorithm exists in the batch setting (Gonzalez, 1985). In the streaming setting, when points arrive one at a time, the DOUBLING algorithm by Charikar et al. (2004) was the first algorithm discovered for this problem. The algorithm maintains an 8-approximation. Furthermore, it works in $O(\log \Delta) = O(\log n)$ phases and the total consistency cost of each phase is k; thus we get the following lemma.

Lemma 4.1. *The* DOUBLING *algorithm for the* k-CENTER *problem is* $(8, O(k \log n))$*-consistent.*

5. Main Algorithm

In this section we present our main result, an algorithm that achieves a polylogarithmic consistency factor. More precisely, we show that for every constant $p \geq 1$, it is possible to design an algorithm for the Consistent k-clustering problem under cost_p that is constant approximate, and $O(k^2 \log^4 n)$-consistent.

In the remainder of the section we first present the main ideas behind our algorithm, then prove some useful technical lemmas, and finally present the full algorithm.

5.1. Main ideas

Before delving into the details, we highlight the three main building blocks of our algorithm.

The first is the Meyerson sketch for online facility location (Meyerson, 2001). This sketch has already been used by Charikar et al. (2003) to solve the k-median problem on data streams. We show that the main ingredients of the sketch continue to work under cost_p objectives, and use it to generally reduce the number of points under considerations from n to $k \cdot \mathrm{poly} \log n$.

One caveat of this sketch is that to use it we need to have access to a good lower bound on the cost of the optimal solution at any point in time. We obtain it by running the $\Theta(p)$ approximation algorithm described by Gupta & Tangwongsan (2008) on all available points. In this way, at any point in time we have a good approximation of the optimum solution. Then we divide the progress of our algorithm into $\log n$ phases based on this lower bound and in each phase we use a different sketch.

Finally, while the Meyerson sketch maintains $O(k \log^2 n)$ possible centers, to computer the k-clustering, we have to reduce these points into exactly k final centers. We first show that this is possible and then we prove that we do not need to recluster frequently. In fact we will do it only when either a new point is added to the Meyerson sketch—$O(k \log^2 n)$ times—or when the number of points assigned to one of these elements of the Meyerson sketch doubles—$O(k \log n)$ events per sketch.

By putting all of these ingredients together, we show that the number of times we need to fully recluster is at most $O(k \log^3 n)$ per phase, or that we have $O(k^2 \log^4 n)$ cluster changes in total.

5.2. The Meyerson sketch

We present the Meyerson sketch and prove some useful properties. We assume to have access to a lower bound to the cost of the optimal solution L, such that $L^p \geq \beta \mathsf{opt}_p$, for some constant $0 \leq \beta \leq 1$. (We will remove the assumption later.) Then the algorithm works in phases, such that at any time in phase j, $L \in [2^{j-1}, 2^j)$. So in each phase j we can use the same lower bound $L_j^p = 2^{j-1}$ and have $L_j^p \geq \frac{\beta \mathsf{opt}_p}{2}$.

In each phase j we create $2 \log n$ Meyerson sketches as described in Algorithm 1. Then we combine them in a single sketch as described in Algorithm 2.

Algorithm 1 Single Meyerson sketch

Input: A sequence of points $x_0, x_1, x_2, \ldots, x_n$. A finite p.
Output: A set S that is a constant bi-criteria approximate solution for the k-clustering problem.
$S \leftarrow \emptyset$
Let X be a set of points and let L be such that $L \geq \gamma \mathsf{opt}_p(X)$, for some constant $\gamma > 0$
for $x \in X$ **do**
  if $S == \emptyset$ **then**
    $S \leftarrow \{x\}$
  else
    Let $\delta = d(x, S)^p$
    With probability $\min\left(\frac{\delta k(1+\log n)}{L^p}, 1\right)$ add x to S
Return S

For simplicity we first analyze the property of a single Meyerson sketch. In particular we give a bound on both the

number of points selected by a single sketch, as well as the quality of the approximation. The Lemma generalizes the results in (Charikar et al., 2003; Meyerson, 2001) to all finite p and follows the general structure their proof so it is deferred to the extended version of the paper.

Lemma 5.1. *For a constant $\gamma \in (0,1)$, with probability at least $\frac{1}{2}$ the set S computed by Algorithm 1 has: (i) size at most $4k(1+\log n)\left(\frac{2^{2p+1}}{\gamma^p}+1\right)$; (ii) $\mathsf{cost}_p(S) \leq 64\mathsf{opt}_p(X)$.*

From Lemma 5.1 we know that with constant probability a single Meyerson sketch is of size $O(k \log n)$ and contains a set of points that give a good solution to our problem. Thus, if we construct $2\log n$ single Meyerson sketches in parallel, at least one of them gives a constant approximation to the optimum at every point in time with probability at least $1 - O(n^{-1})$. The observation inspired the design of Algorithm 2, whose properties are formalized next.

Lemma 5.2. *For a constant $\gamma \in (0,1)$, with probability $1 - O(n^{-1})$ the set $M = \cup_{i=1}^{2\log n} M_i$ computed by Algorithm 2 has: size at most $O(k \log^2 n)$ and $\mathsf{cost}_p(M) \leq 64\mathsf{opt}_p(X)$.*

Proof. As mentioned above, Lemma 5.1 implies that if we construct $2\log n$ single Meyerson sketches in parallel, with probability $1 - O(n^{-1})$, at least one of them gives a constant approximation to the optimum at every point in time. Furthermore in total it contains only $4k(1+\log n)\left(\frac{2^{2p+1}}{\gamma^p}+1\right)$ points.

Now in Algorithm 2 we are almost building $2\log n$ Meyerson sketches; the only difference is that we stop adding points to a single sketch when it becomes too large. This modification does not change the probability that there exist at least one single sketch that gives a constant approximation to the optimum at every point in time and has at most $4k(1+\log n)\left(\frac{2^{2p+1}}{\gamma^p}+1\right)$ points.

Thus with probability $1 - O(n^{-1})$ at least one of the sketches constructed in Lemma 5.1 gives a constant approximation to the optimum at every point in time. Merging other sketches to this sketch does not affect this property. Furthermore the number of points in each sketch is explicitly bounded by $4k(1+\log n)\left(\frac{2^{2p+1}}{\gamma^p}+1\right)$ so the total number of points in M is bounded by $8k\log n(1+\log n)\left(\frac{2^{2p+1}}{\gamma^p}+1\right)$ □

Note that in some cases we do not need to recompute all the sketches from scratch but we need only to update them, so we can define a faster update function described in Algorithm 3.

Algorithm 2 $ComputeMeyerson(X_t, \phi)$

Input: A sequence of points X_t, a lower bound to the optimum ϕ.
Output: $2\log n$ independent Meyerson sketches $M_1, \ldots, M_{2\log n}$
$L^p = \frac{\phi}{\gamma}$,
for $i \in [2\log n]$ **do**: ▷ Initialize all Meyerson sketches
  $M_i \leftarrow x_0$
for $x \in X_t$ **do**:
  for $i \in [2\log n]$ **do**: ▷ If M_i is not too large, analyze x
    if $|M_i| \leq 4k(1+\log n)\left(\frac{2^{2p+1}}{\gamma^p}+1\right)$ **then**:
      Let $\delta = d(x, M_i)^p$
      $\hat{p} = \min\left(\frac{\delta k(1+\log n)}{L^p}, 1\right)$
      Add x to M_i with probability $\hat{p}$
Return $M_1, \ldots, M_{2\log n}$

Algorithm 3 $UpdateMeyerson(M_1, \ldots, M_s, x_t, \phi)$

Input: A point x_t, a lower bound to the optimum ϕ and s independent Meyerson sketches $M_1, \ldots, M_s$.
Output: s independent Meyerson sketches $M_1, \ldots, M_s$
$L^p = \frac{\phi}{\gamma}$,
for $i \in [s]$ **do**: ▷ If M_i is not too large, analyze x_t
  if $|M_i| \leq 4k(1+\log n)\left(\frac{2^{2p+1}}{\gamma^p}+1\right)$ **then**:
    Let $\delta = d(x_t, M_i)^p$
    $\hat{p} = \min\left(\frac{\delta k(1+\log n)}{L^p}, 1\right)$
    Add x_t to M_i with probability $\hat{p}$
Return $M_1, \ldots, M_s$

In the rest of the paper we refer to a single Meyerson sketch as M_i and to their union as M.

5.3. From Meyerson to k clusters

Our next step is to show that in the Meyerson sketch there exists a subset of k centers that gives an approximately optimal solution. We follow the approach in Guha et al. (2000) and show that by weighing the points in the Meyerson sketch with the number of original data points assigned to them, and then running a weighted k-clustering algorithm to recluster them into k clusters, we can achieve a constant approximate solution.

Before formalizing this observation we give some additional notation. In the remainder of the section we denote the weight of a point x in the Meyerson sketch with $w(x)$, the cost of the centers used in Meyerson sketch with cost_M, and the cost of the aforementioned weighted clustering instance with cost_L. Finally we refer to the optimal set of centers for the weighted k-clustering instance as $\mathfrak{c}'$.

We begin with two technical Lemmas.

Lemma 5.3. *For any constant $p \geq 1$, $\mathsf{cost}_p(X, \mathfrak{c}') \leq 2^{p-1}\left(\mathsf{cost}_\mathsf{M} + \mathsf{cost}_\mathsf{L}\right)$*

Lemma 5.4. *For any constant $p \geq 1$, $\mathsf{cost}_\mathsf{L} \leq$*

$2^{2p-1}\left(\mathsf{cost}_{\mathsf{M}} + \mathsf{opt}_p\right)$

Note that combining Lemmas 5.3 and 5.4 the following Corollary follows.

Corollary 5.5. *For any constant $p \geq 1$, $\mathsf{cost}_p(\mathfrak{c}') \leq 2^{3p-1}\left(\mathsf{cost}_{\mathsf{M}} + \mathsf{opt}_p\right)$*

We defer the proofs of lemma 5.3 and lemma 5.4 to the extended version of the paper. Those proofs are similar in spirit to those in (Bateni et al., 2014; Guha et al., 2000), but are generalized here for all p.

Thanks to Corollary 5.5 we know that by using a Meyerson sketch, M contains a good approximation for our problem. In the next subsection we show how to use this to obtain a solution for the consistency problem.

Before doing this we define two algorithms that allow us to construct a weighted clustering instance starting from a Meyerson sketch (Algorithm 4) and to update the weights for a weighted instance (Algorithm 5).

Algorithm 4 $CreateWeightedInstance(M_1, \ldots, M_s, \phi, X_t)$

Input: A sequence of points X_t, a lower bound to the optimum ϕ and s independent Meyerson sketches $M_1, \ldots, M_s$.
Output: A weighted k-clustering instance (M, w).
Let $M = \cup_i M_i$
Assign points in X_t to the closest point in M
Let $w(y)$ to be equal to the number of points assigned to $y \in M$
Return (M, w)

Algorithm 5 $UpdateWeights(M, w, x)$

Input: A point x, the current weights w and the Meyerson sketch M.
Output: A weighted k-clustering instance (M, w).
Assign x to the closest point in M
Let m_x be the closest point to x in M
$w(m_x) = w(m_x) + 1$
Return (M, w)

5.4. The algorithm

We are now ready to formally state and prove the correctness of our main algorithm, which we present in Algorithm 6. The input of our algorithm is a sequence of points $x_0, x_1, x_2, \ldots, x_n$. Recall, that we denote the prefix up to t as X_t, and the cost of the solution using centers $\mathfrak{c}$ as $\mathsf{cost}_p(X_t, \mathfrak{c})$. Finally we assume to have access to a γ-approximation algorithm $\mathcal{A}$ for the weighted k-clustering problem for any constant p-norm (we can use for example the local search algorithm described by Gupta & Tangwongsan (2008)).

We can now state our main theorem.

Theorem 5.6. *For any constant $p \geq 1$, with probability $1 - O(n^{-1})$, Algorithm 6 returns a sequence of centers $\mathfrak{c}_0, \mathfrak{c}_1, \mathfrak{c}_2, \ldots, \mathfrak{c}_n$ such that at any point in time t $\mathsf{cost}_p(\mathfrak{c}_t, X_t) \leq \alpha^p \mathsf{opt}_p(X_t)$ for a constant α and the total inconsistency factor of the solution is $O(k^2 \log^4 n)$*

Algorithm 6 Consistent k-clustering algorithm

Input: A sequence of points $x_0, x_1, x_2, \ldots, x_n$.
Output: A sequence of centers $\mathfrak{c}_0, \mathfrak{c}_1, \mathfrak{c}_2, \ldots, \mathfrak{c}_n$
Select the first k points as centers $\mathfrak{c}_0 = \{x_0, x_1, x_2, \ldots, x_k\}$
$t \leftarrow 0$
while $\mathsf{cost}_p(\mathfrak{c}_0, X_t) = 0$ **do**:
  $\mathfrak{c}_t \leftarrow \mathfrak{c}_0$; Output $\mathfrak{c}_t$; $t \leftarrow t + 1$
$\phi \leftarrow 0$ ▷ Initialize lower bound to the optimum
$M_1, ..., M_{2\log n} \leftarrow ComputeMeyerson(X_0, \phi)$
$\mathfrak{c} \leftarrow \emptyset$; $s \leftarrow 2 \log n$
while $t \leq n$ **do**:
  Run $\mathcal{A}$ on X_t to get approximated solution $\mathfrak{c}'$
  if $\mathsf{cost}_p(X_t, \mathfrak{c}') \geq 2\phi$ **then**: ▷ New l.b. for ϕ
    $\phi \leftarrow \mathsf{cost}_p(X_t, \mathfrak{c}')$,
    $M_1, ..., M_s \leftarrow ComputeMeyerson(X_t, \phi)$
    $(M, w) \leftarrow GetWeightedProb(M_1, ..., M_s, \phi, X_t)$
    Solve (M, w) using algorithm $\mathcal{A}$
    Let $\mathfrak{c}_t$ be the set of centers computed by $\mathcal{A}$
  else: ▷ Update Meyerson and recluster if needed
    $M_1, M_2, .. \leftarrow UpdateMeyerson(M_1, .., M_s, x_t, \phi)$
    Let $M = \cup_i M_i$,
    if $x_t \in M$ **then**: ▷ x_t is in Meyerson sketch
      $(M, w) \leftarrow GetWeightedProb(M_1, ..., M_s, \phi, X_t)$
      Solve (M, w) using algorithm $\mathcal{A}$
      Let $\mathfrak{c}_t$ be the set of centers computed by $\mathcal{A}$
    else:
      $(M, w) \leftarrow UpdateWeights(M, w, x)$
      Let m_t be the closest point to x_t in M
      if $w(m_t)$ is a power of 2 **then**:
        ▷ Weight of a point "doubled"
        Solve (M, w) using algorithm $\mathcal{A}$
        Let $\mathfrak{c}_t$ be the set of computed centers
      else:
        $\mathfrak{c}_t = \mathfrak{c}_{t-1}$
  Output $\mathfrak{c}_t$; $t \leftarrow t + 1$

Proof. We start by bounding the inconsistency factor, $\sum_{i=1}^{n-1} |\mathfrak{c}_{i+1} - \mathfrak{c}_i|$.

During the execution of Algorithm 6 the set of centers changes if and only if one of the three following conditions is met: (i) the cost of the clustering on X_t computed by $\mathcal{A}$ increases by a factor of 2, (ii) we add a new point to a Meyerson sketch, (iii) a new point is assigned to a point of the Meyerson sketch, m_t, and the weight of m_t is a power of 2 after this addition. Note that every time we change the centers, we fully recluster, and so increase the consistency factor by k in the worst case. Therefore to prove the theorem we need to show that one of these conditions is met at most $O(k \log^4 n)$ times.

From our assumptions we know that the spread of the point set is polynomial in n, which implies the same bound on the cost of the optimum solution. Therefore, the cost of the solution computed by $\mathcal{A}$ doubles at most $O(\log n)$ times.

For the same reason we update the lower bounds, ϕ at most $O(\log n)$ times during the execution of our algorithm. This in turn implies that we rebuild the Meyerson sketches from scratch at most $O(\log n)$ times. Given that we run $O(\log n)$ Meyerson sketches in parallel, during the execution of the algorithm we use at most $O(\log^2 n)$ Meyerson sketches. Furthermore each Meyerson sketch has at most $O(k \log n)$ centers, thus in total we can add at most $O(k \log^3 n)$ points under condition (ii).

Finally note that while a Meyerson sketch is fixed, the weight of every point in the sketch can only grow. In addition, the weight is always is bounded by n, and therefore can double at most $\log n$ times per sketch point, resulting in $O(k \log^2 n)$ changes under a fixed Meyerson sketch. Therefore condition (iii) holds at most $O(k \log^4 n)$ times. So overall at least one of the conditions is satisfied at most $O(k \log^4 n)$ times, thus the algorithm is $O(k^2 \log^4 n)$-consistent.

To finish our proof we need to show that at any point in time our algorithm returns with probability $1-o(n^{-1})$ a constant approximation to the optimum. Note that by corollary 5.5 we know that for any constant $p \geq 1$ the cost of a solution computed on the Meyerson sketch can be bounded by $\mathsf{cost}_p(\mathfrak{c}') \leq 2^{3p-1}\left(\mathsf{cost}_{\mathsf{M}} + \mathsf{opt}_p\right)$. From Lemma 5.2 we know that the Meyerson sketch guarantees with probability $1 - o(n^{-1})$ that $\mathsf{cost}_{\mathsf{M}} \leq 16\mathsf{opt}_p(X)$. So we have the cost of the optimal set of centers in the Meyerson sketch at any point in time is at most $O(\alpha^p opt_p)$ w.h.p. for a constant α[4].

While we cannot compute the optimal set of centers in the Meyerson sketch, we can find an $O(p)$ approximation for every constant p by relying on the local search algorithm of Gupta & Tangwongsan (2008). Therefore, every time we recompute the centers using $\mathcal{A}$ we are sure that we obtain a constant approximation.

Finally it remains to show that when none of the three conditions are met, and we simply add a point to the solution without recomputing the centers we retain an approximately optimal solution. By Lemma 5.3 we know that for any constant $p \geq 1$, $\mathsf{cost}_p(\mathfrak{c}') \leq 2^{p-1}\left(\mathsf{cost}_{\mathsf{M}} + \mathsf{cost}_{\mathsf{L}}\right)$.

Moreover, we can always bound the cost of Meyerson sketch with $16\mathsf{opt}_p(X)$.

It remains to get a bound on $\mathsf{cost}_{\mathsf{L}}$. Note that the number of points assigned to any point in M did not double since the previous reclustering. Therefore, in the weighted reclustering formulation the weight of all points increased by a factor less than 2. Therefore, $\mathsf{cost}_{\mathsf{L}}$ at this point is bounded by at most twice $\mathsf{cost}_{\mathsf{L}}$ computed when we last reclustered. Therefore, $\mathsf{cost}_p(\mathfrak{c}') \leq 2^{4p-1}\left(\mathsf{cost}_{\mathsf{M}} + \mathsf{opt}_p\right)$and the current solution remains approximately optimal. □

[4]We do not make an attempt to optimize the constant factors. As we show in the experimental section, in practice the algorithm gives a very good approximation.

6. Optimizing Consistency

How many times do we need to change the centers to obtain a good k-clustering at any point in time? In Section 5 we presented an algorithm that is $O(k^2 \log^4 n)$-consistent, while in Subsection 3.1 we showed that at least $\Omega(k \log n)$ changes are needed (assuming that Δ is polynomial in n). We give an existential result, we show that for any input sequence there exist a solution that is constant approximate and $O(k \log^2 n)$-consistent. In interest of space we deferred the proof of the lemma to the extended version of the paper.

Theorem 6.1. *For any sequence $x_0, x_1, \ldots, x_n$ there exists a sequence of solutions $\mathfrak{c}_0, \mathfrak{c}_1, \ldots, \mathfrak{c}_n$ such that $\forall i$, $\mathsf{cost}_p(X_i, \mathfrak{c}_i) \leq \alpha\mathsf{opt}_p(X_i)$ for some constant α, and the $\sum_i |\mathfrak{c}_{i+1} - \mathfrak{c}_i\| = O(k \log^2 n)$.*

7. Experiments

We demonstrate the efficacy of our algorithm by tracking both the quality of the solution and the number of reclusterings needed to maintain it on a number of diverse datasets. As we will show, the theoretical guarantees that we prove in the previous section provide a loose bound on the number of reclusterings; in practice the number of times we recompute the solution grows logarithmically with time.

Data We evaluate our algorithm on three datasets from the UCI Repository (Lichman, 2013) that vary in data size and dimensionality. (i) SKINTYPE has $245,057$ points lying in 4-dimensions. (ii) SHUTTLE has $58,000$ points in 9 dimensions. (iii) COVERTYPE has $581,012$ points in 54 dimensions. For each of the datasets we try values of k in $\{10, 50, 100\}$, and observe that the qualitative results are consistent across datasets and values of k.

Algorithm Modifications In the development of the algorithm we made a number of decisions to obtain high probability results. The key among them was to run $O(\log n)$ copies of the Meyerson sketch, since each sketch succeeds only with constant probability. We eschew this change in the implementation, and maintain just a single sketch, at the cost of incurring a worse solution quality.

Metrics and results The goal of this work is to give algorithms that maintain a good clustering, but only recluster judiciously, when necessary. To that end, we focus on two main metrics: number of reclusterings and solution quality.

Reclustering We plot the number of reclusterings as a function of time for the three different datasets in Figure 1. Note that the x-axis is on log-*scale*, and thus a straight line

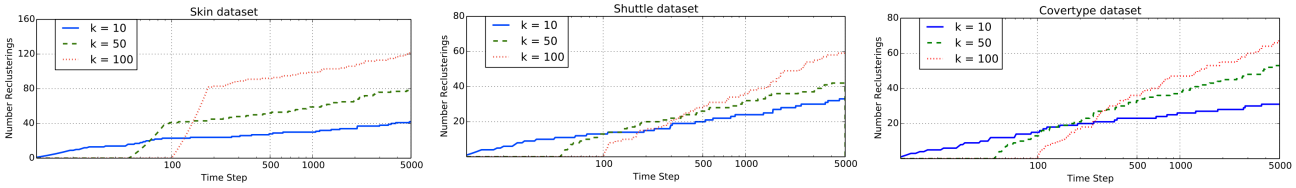

Figure 1. The number of reclusterings as a function of t for three different datasets, and various values of k, plotted on a log scale. Observe that the (i) x-axis is on Log scale, showing that after an initial warm up phase, the algorithm does a logarithmic number of reclusterings, and (ii) the rate of reclustering is slightly higher for higher values of k.

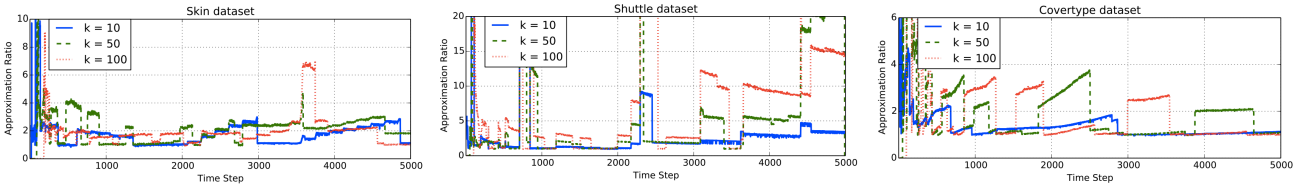

Figure 2. The approximation ratio as a function of t for three different datasets, and various values of k. Note the stair step pattern – when the algorithm choses not to recluster the approximation ratio slowly degrades. A reclustering step brings it down close to 1.

represents number of reclusterings that grows logarithmically with time. Qualitatively we make two observations, across all datasets, and values of k.

First, the rate of reclustering (defined as the fraction of time the algorithm recomputes the solution) is approximately $\log n/n$, which tends to 0 as the dataset size grows. Further, the rate is higher for higher values of k, a fact also suggested by our theoretical analysis.

Unlike the SHUTTLE and COVERTYPE datasets, the SKINTYPE dataset exhibits a change in behavior, where initially the reclustering rate is relatively high, but then it sharplly drops after about $O(2k)$ steps. This is explained by the fact that the order of the points in this data set is not random. Therefore initially the algorithm reclusters at a high rate, once all of the parts of the input space are explored, the rate of reclustering slows. When we run the algorithm on a randomly permuted instance of SKINTYPE, this phase transition in behavior disappears.

Approximation Ratio We plot the approximation ratio of the solution (as compared to the best obtained by ten runs of k-means++ (Arthur & Vassilvitskii, 2007)) in Figure 2.

For the SKIN and COVERTYPE datasets, the approximation ratio stays relatively low, largely bounded by 4, after an initial period. A more careful examination of the plots shows exactly the times when the consistent algorithm allows the solution to degrade, and when it decides to recompute the solution from scratch. The latter are indicated by sharp drops in the approximation ratio, whereas the former are the relatively flat patterns.

It is interesting to note that the additional points sometimes worsen the approximation (as indicated by the lines sloping upwards), but sometimes actually improve the approximation. This is due to the fact that decisions made by the online algorithm balance optimality at that point in time, with the potential location of points arriving in the future. The latter is most apparent in the $k = 100$ experiment of the SHUTTLE dataset.

All of the datasets sometimes exhibit large fluctuations in the approximation ratio. This is an artifact of using a single Myerson sketch, which does not capture the structure of the points with small, but constant probability.

8. Conclusions and Future Work

We introduced the notion of consistent clustering: a variant of online clustering which balances the need for maintaining an approximately optimal solution with the cost of reclustering. We proved $\Omega(k \log n)$ lower bounds, and gave algorithms for all k-clustering variants that come close to achieving this bound.

The notion of quantifying the worst case number of changes necessary to maintain a constant approximate solution in an online setting is interesting to study in contexts other than k-clustering. For example, one can consider online graph problems, such as online matching and online densest subgraph, or other types of clustering problems, such as hierarchical or correlation clustering.

References

Anagnostopoulos, Aris, Bent, Russell, Upfal, Eli, and Hentenryck, Pascal Van. A simple and deterministic competitive algorithm for online facility location. *Inf. Comput.*, 194(2):175–202, 2004.

Arthur, David and Vassilvitskii, Sergei. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pp. 1027–1035, 2007. ISBN 978-0-898716-24-5.

Bansal, Nikhil, Buchbinder, Niv, Madry, Aleksander, and Naor, Joseph (Seffi). A polylogarithmic-competitive algorithm for the k-server problem. *J. ACM*, 62(5):40:1–40:49, November 2015. ISSN 0004-5411.

Bateni, MohammadHossein, Bhaskara, Aditya, Lattanzi, Silvio, and Mirrokni, Vahab S. Distributed balanced clustering via mapping coresets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 2591–2599, 2014.

Charikar, Moses, O'Callaghan, Liadan, and Panigrahy, Rina. Better streaming algorithms for clustering problems. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pp. 30–39, 2003.

Charikar, Moses, Chekuri, Chandra, Feder, Tomás, and Motwani, Rajeev. Incremental clustering and dynamic information retrieval. *SIAM J. Comput.*, 33(6):1417–1440, 2004.

Czumaj, Artur, Lammersen, Christiane, Monemizadeh, Morteza, and Sohler, Christian. $(1+\epsilon)$-approximation for facility location in data streams. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pp. 1710–1728, 2013.

Feldman, Jon, Mehta, Aranyak, Mirrokni, Vahab S., and Muthukrishnan, S. Online stochastic matching: Beating 1-1/e. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pp. 117–126, 2009.

Ferguson, Thomas S. Who solved the secretary problem? *Statist. Sci.*, 4(3):282–289, 1989.

Fiat, Amos, Karp, Richard, Luby, Micheal, McGeoch, Lyle, Sleator, Daniel, and Young, Neal E. Competitive paging algorithms. *Journal of Algorithms*, 12(4):685–699, 1991. doi: 10.1016/0196-6774(91)90041-V.

Fotakis, Dimitris. On the competitive ratio for online facility location. *Algorithmica*, 50(1):1–57, 2008.

Gonzalez, Teofilo F. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.

Guha, Sudipto, Mishra, Nina, Motwani, Rajeev, and O'Callaghan, Liadan. Clustering data streams. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pp. 359–366, 2000.

Gupta, Anupam and Tangwongsan, Kanat. Simpler analyses of local search algorithms for facility location. *CoRR*, abs/0809.2554, 2008.

Kanungo, Tapas, Mount, David M., Netanyahu, Nathan S., Piatko, Christine D., Silverman, Ruth, and Wu, Angela Y. A local search approximation algorithm for k-means clustering. *Comput. Geom.*, 28(2-3):89–112, 2004.

Karp, Richard M., Vazirani, Umesh V., and Vazirani, Vijay V. An optimal algorithm for on-line bipartite matching. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pp. 352–358, 1990.

Kesselheim, Thomas, Kleinberg, Robert D., and Niazadeh, Rad. Secretary problems with non-uniform arrival order. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pp. 879–888, 2015.

Kleinberg, Robert D. A multiple-choice secretary algorithm with applications to online auctions. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, pp. 630–631, 2005.

Li, Shi and Svensson, Ola. Approximating k-median via pseudo-approximation. *SIAM J. Comput.*, 45(2):530–547, 2016.

Liang, Percy and Klein, Dan. Online em for unsupervised models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pp. 611–619, 2009. ISBN 978-1-932432-41-1.

Liberty, Edo, Sriharsha, Ram, and Sviridenko, Maxim. An algorithm for online k-means clustering. In *Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments, ALENEX 2016, Arlington, Virginia, USA, January 10, 2016*, pp. 81–89, 2016.

Lichman, M. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Mahdian, Mohammad and Yan, Qiqi. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pp. 597–606, 2011.

Manasse, Mark S., McGeoch, Lyle A., and Sleator, Daniel Dominic. Competitive algorithms for server problems. *J. Algorithms*, 11(2):208–230, 1990.

Meyerson, Adam. Online facility location. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pp. 426–431, 2001.

Sleator, Daniel Dominic and Tarjan, Robert Endre. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28(2):202–208, 1985.