A Better k-means++ Algorithm via Local Search

Silvio Lattanzi [*1] **Christian Sohler** [*1]

Abstract

In this paper, we develop a new variant of k-means++ seeding that in expectation achieves a constant approximation guarantee. We obtain this result by a simple combination of k-means++ sampling with a local search strategy. We evaluate our algorithm empirically and show that it also improves the quality of the solution in practice.

1. Introduction

As a central problem in unsupervised learning clustering received a lot of attention in the past decades. The goal of clustering is to partition a given set of objects into *clusters* with the idea that objects in the same cluster should be similar to each other while objects in different clusters should be less similar. One basic formulation of clustering is the k-clustering problem with sum-of-squared-error objective function. In this problem, the objects are represented as vectors in $\mathbb{R}^d$ and we think of the squared Euclidean distance as a measure of dissimilarity. Formally, in this setting we want to find a set C of k centers that minimizes

$$\sum_{p \in P} \min_{c \in C} \|p - c\|^2.$$

The clusters are defined by assigning each point to their closest center. It is known that the optimal center of a given cluster is the average or *mean* of the cluster, which is why the problem is also often called k-means clustering.

k-means has been extensively studied in literature, and several heuristic have been proposed to solve the problem. Probably the most celebrated heuristic for k-means is the well-known Lloyd's algorithm (Lloyd, 2006). The algorithm is also often called the *k-means algorithm*. The algorithm usually performs very well in practice, but it does not provide a theoretical approximation guarantee.

[*]Equal contribution [1]Google Research, Zurich, ZH, Switzerland. Correspondence to: Silvio Lattanzi <silviol@google.com>, Christian Sohler <sohler@google.com>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

The k-means++ seeding algorithm (Arthur & Vassilvitskii, 2007) is a simple way to improve Lloyd's algorithm. The algorithm incrementally chooses a set of k centers by sampling the next center from a distribution where every point has probability proportional to its squared distance to the currently closest center. The solution computed by the seeding algorithm has expected cost of $O(\log k)$ times the cost of the optimum solution. In practice, it is then often used as a starting solution for Lloyd's algorithm (which never decreases the objective function).

k-means++ is easy to implement and is known to sometimes heavily outperform Lloyd's algorithm with random initialization in terms of the cost of the computed solution (Arthur & Vassilvitskii, 2007). However, from a theoretical point of view, the approximation guarantee of $O(\log k)$ is not fully satisfying as constant approximation algorithms exist (Ahmadian et al., 2017). Unfortunately, it is known that the analysis is tight (Arthur & Vassilvitskii, 2007) and there are also input sets such that k-means++ achieves an $o(\log k)$-approximation with probability at most $\exp(-k^{1-o(1)})$ (Brunsch & Röglin, 2011). At the same time, it is known that k-means++ provides a constant bicriteria approximation with constant probability, if one allows to sample $O(k)$ centers (Aggarwal et al., 2009). More recently, a more precise tradeoff with respect to the number of centers and the expected cost has been given by (Wei, 2016).

1.1. Our Contribution

We show that a simple combination of k-means++ seeding with $O(k \log \log k)$ rounds of local search gives a constant approximation guarantee in expectation. Our strategy is as follows. We start similarly as k-means++: We iteratively sample centers from a distribution where every point has probability proportional to its cost in the current clustering. However, after we have sampled k centers we continue for another $O(k \log \log k)$ rounds. In order to maintain a set of k centers, we exchange the newly sampled center with the old center, such that the swap improves the cost by the largest amount. If there is no improving swap, we discard the sampled center. We prove that after $O(k \log \log k)$ iterations our solution is in expectation a constant factor larger than the optimal solution.

We remark that the analysis of the local search algorithm by Kanungo et al. (Kanungo et al., 2004) already implies that after some (potentially very large) number of rounds the algorithm ends up with a constant approximation. Their algorithm improves the current solution by swapping one center with an input point. Kanungo et al. proved that this algorithm gives in polynomial time a constant approximation (Kanungo et al., 2004). Unfortunately the running time of this simple swapping strategy can be very high. The main contribution of the paper is to introduce a new improvement strategy that is provably very fast: after only $O(k \log \log k)$ rounds the resulting center has constant approximation guarantee in expectation.

We empirically evaluated our algorithm. The evaluation shows that we improve the quality of the solution significantly (8-35%) compared with the basic k-means++ seeding when both algorithms are not using Lloyd's algorithm for postprocessing. In the case that both algorithms perform 10 steps of Lloyd's algorithm after the seeding the relative improvement over k-means++ is in the range of $1 - 18\%$.

1.2. Other Related Work

Kanungo et al. analyzed and empirically evaluated their local search algorithm as well as a few other variants of k-means clustering (Kanungo et al., 2004). In their experiments the local search based algorithm was converging very slowly. They achieved the best performance using a combination of local search and Lloyd's algorithm (Lloyd, 2006). A combination of local search and Lloyd's algorithm with a coreset construction has been developed (Frahling & Sohler, 2008). In the setting of streaming algorithms, as part of the StreamKM++ algorithm a heuristic is given to quickly approximate the k-means++ seeding using coreset trees (Ackermann et al., 2012). A more recent approach to approximately sample from the k-means++ distribution uses the Metropolis-Hastings algorithm to approximately sample from the k-means++ distribution (Bachem et al., 2016). In order to make the seeding applicable in a parallel setting, the k-means|| algorithm (Bahmani et al., 2012) reduces the number of rounds by sampling centers independently with probability proportional to the squared distance. This allows to sample sets of centers in parallel.

In terms of (mostly) theoretical results, it is known that there are different constant approximation algorithms for the k-clustering problem with sum of squared errors ("k-means clustering") (Jain & Vazirani, 2001; Kanungo et al., 2004; Ahmadian et al., 2017). Currently, the best approximation ratio is 6.357 (Ahmadian et al., 2017). It is also known that it is NP-hard to obtain a solution with approximation factor less than 1.0013, i.e. there is no $(1 + \epsilon)$-approximation algorithm for arbitrary small $\epsilon > 0$, if k and d can be large (Lee et al., 2017). For constant d, one can obtain $(1 + \epsilon)$-approximation algorithms using local search (Cohen-Addad, 2018; Cohen-Addad et al., 2016; Friggstad et al., 2016) (see also (Bandyapadhyay & Varadarajan, 2016)). However, the running time of these algorithm depends doubly exponential on the dimension. If k and ϵ are considered to be constant, there are many $(1 + \epsilon)$ approximation algorithms available that are based on reducing the input size or the number of candidate solutions (see, for example, (Har-Peled & Mazumdar, 2004; Feldman et al., 2007; Chen, 2009; Kumar et al., 2010)).

2. Preliminaries

Let $P = \{p_1, \ldots, p_n\}$ be a point set in $\mathbb{R}^d$. For two points $p, q \in \mathbb{R}^d$ we use $\|p - q\|$ to denote their Euclidean distance. In the sum-of-squared-error problem (or k-means problem) we aim to find a set C of k centers in $\mathbb{R}^d$ that minimizes the following objective function over all such sets C:

$$\text{cost}(P, C) = \sum_{i=1}^{n} \min_{c \in C} \|p_i - c\|^2.$$

For a point set P we define its mean or center of gravity to be

$$\mu(P) = \frac{1}{|P|} \sum_{p \in P} p.$$

We note that $\mu(P)$ is an optimal solution to the 1-means problem. Finally we refer to the set of optimal centers as C^*. We use $P_1^*, \ldots, P_k^*$ to be the partitioning into clusters induced by the optimal centers and we refer to the cost of the optimal solution as Opt_k.

The following lemma is folklore:

Lemma 1. *Let $P \subseteq \mathbb{R}^d$ be a set of points and let $c \in \mathbb{R}^d$ be a center. Then we have cost$(P, \{c\}) = |P| \cdot \|c - \mu(P)\|^2 +$ cost$(P, \mu(P))$.*

We will also use the following lemma (rephrased from Corollary 21 in (Feldman et al., 2018)).

Lemma 2. *Let $\epsilon > 0$. Let $p, q \in \mathbb{R}^d$ and let $C \subseteq \mathbb{R}^d$ be a set of k centers. Then $|$cost$(\{p\}, C) -$ cost$(\{q\}, C)| \leq \epsilon \cdot$ cost$(\{p\}, C) + (1 + \frac{1}{\epsilon})\|p - q\|^2$.*

3. An improved k-means++ algorithm

We analyze the following algorithm. Starting with an empty solution C we sample a point p at random from P where the probability to sample $p \in P$ is proportional to $\text{cost}(\{p\}, C)$ (in the first step we sample uniformly at random). Then if $C < k$, we add the sampled point to C. Otherwise we check whether there exists a point $c \in C$ such that $\text{cost}(P, C \setminus \{c\} \cup \{p\}) < \text{cost}(P, C)$. If this is the case, we replace c by the point in C that reduces the cost function by

Algorithm 1 k-means++ seeding with local search

Require: P, k, Z
1: Uniformly sample $p \in P$ and set $C = \{p\}$.
2: **for** $i \leftarrow 2, 3, \ldots, k$ **do**
3: Sample $p \in P$ with probability $\frac{\text{cost}(\{p\},C)}{\sum_{q\in P}\text{cost}(\{q\},C)}$ and add it to C.
4: **end for**
5: **for** $i \leftarrow 2, 3, \ldots, Z$ **do**
6: $C = \text{LocalSearch++}(P, C)$
7: **end for**
8: **return** C

Algorithm 2 LocalSearch++

Require: P, C
1: Sample $p \in P$ with probability $\frac{\text{cost}(\{p\},C)}{\sum_{q\in P}\text{cost}(\{q\},C)}$
2: **if** $\exists q \in C$ s.t. $\text{cost}(P, C \setminus \{q\} \cup \{p\}) < \text{cost}(P, C)$ **then**
3: Let $q \in C$ be the q s.t. $\text{cost}(P, C \setminus \{q\} \cup \{p\})$ is minimized
4: $C = C \setminus \{q\} \cup \{p\}$
5: **end if**
6: **return** C

the largest amount. The pseudo-code for the algorithm is presented in Algorithm 1.

In order to implement the k-means++ sampling, we need to maintain the squared distance of every point to the current set of centers. Once we have sampled a new point we can do so by computing the distance from the new center to every input point and comparing this to the currently closest center. If the new distance is smaller, we update our data structure. This can be done in $O(dn)$ time. Using the maintained distance values we can easily implement the sampling in linear time. We first obtain rescaled weights $w_1, \ldots, w_{|P|}$ that sum up to 1. We then sample a number r uniformly at random from $[0, 1]$ and return the point with index i such that $\sum_{j<i} w_j < r \leq \sum_{j\leq i} w_j$.

In order to implement the local search, we need to compute the cost of swapping the new sample point with an old center. This requires to iterate over all clusters and for each cluster we need to compute the distance to all other centers. Thus, a local search step requires $O(dkn)$ time in the worst case, which leads to an overall running time of $O(dnkZ)$[1]. A simple heuristic to improve the running time is to remember the closest and the second closest center. This way, we can compute the change of cost more efficiently. However, maintaining the closest and second closed cluster center under insertions and deletions of cluster centers requires $O(dkn)$ time per update, since it may happen that we delete the center of a cluster that has $\Omega(n)$ points. However, if on average we delete clusters of size $O(n/k)$ we end up with a running time of $O(dnk \log\log k)$.

We are now ready to state the main result of the paper. The proof for the approximation guarantee can be found in Section 4.

Theorem 1. *Let $P \subseteq \mathbb{R}^d$ be a set of points and C be the output of Algorithm 1 with $Z \geq 100000k \log\log k$ then we have $E[cost(P, C)] \in O(cost(P, C^*))$, where C^* is the set of optimum centers. The running time of the algorithm is $O(dnk^2 \log\log k)$.*

We note that in the proofs for sake of simplicity we did not optimize the constant in our analysis so the number of iterations required by the algorithm has a large constant. In the experimental section we observe that just after few iteration the quality of the solution improves significantly.

4. Analysis

In the following we argue that the above algorithm with probability $O(1)$ reduces the cost of the current solution by a $O(1/k)$ factor in every iteration. This, in turn, implies that after $O(k \log\log k)$ iterations we have $E[\text{cost}(P, C)] \in O(\text{Opt}_k)$. Formally,

Lemma 3. *Let P be a set of points and C a set of centers with cost $cost(P, C) > 500Opt_k$. Let $C' =$ LocalSearch++(P,C) then with probability $\frac{1}{1000}$ we have $cost(P, C') \leq (1 - \frac{1}{100k})cost(P, C)$.*

We prove the lemma in Section 4.1. Now we show that the previous lemma combined with the main result in (Arthur & Vassilvitskii, 2007) gives our Theorem 1.

Proof of Theorem 1. Let $\hat{C}$ be the set of centers after the end of the first for loop in Algorithm 1 and let C be the set of centers output by Algorithm 1. By Lemma 3 we know that if before any call of LocalSearch++ the cost of the centers is bigger than 500Opt_k then with probability $\frac{1}{1000}$ we reduce the cost by a $(1 - \frac{1}{100k})$ multiplicative factor.

Now consider another random process X with initial value equal to $cost(P, \hat{C})$ and such that for $Z = 100000k \log\log k$ it reduces its value by a $\left(1 - \frac{1}{100k}\right)$ multiplicative factor with probability $1/1000$ and finally it increases its value by additive 500Opt_k. It is not hard to see that the final value of X stochastically dominates the cost of the clustering. So the final expected value of X is larger than the expected value of $cost(P, C)$ conditioned on the

[1] We note here that it is possible to remove the dependency from d using dimensionality reduction techniques (Cohen et al., 2015; Becchetti et al., 2019; Makarychev et al., 2019) and to obtain a constant approximation in $O(nk \log kZ)$ time.

initial clustering $\hat{C}$. Furthermore:

$$\begin{aligned} E[X] &= 500\text{Opt}_k + cost(P,\hat{C}) \cdot \\ &\quad \sum_{i=0}^{Z} \binom{Z}{i} \frac{1}{1000}^{i} \frac{999}{1000}^{Z-i} \left(1 - \frac{1}{100k}\right)^{i} \\ &= cost(P,\hat{C}) \left(1 - \frac{1}{100000k}\right)^{100000k \log\log k} \\ &\quad +500\text{Opt}_k \\ &\leq \frac{cost(P,\hat{C})}{\log k} + 500\text{Opt}_k \end{aligned}$$

This implies that $E[cost(P,C)|\hat{C}] \leq \frac{cost(P,\hat{C})}{\log k} + 500\text{Opt}_k$. But now:

$$\begin{aligned} E[cost(P,C)] &= \sum_{\hat{C}} E[cost(P,C)|\hat{C}]P(\hat{C}) \\ &= \sum_{\hat{C}} P(\hat{C}) \left(\frac{cost(P,\hat{C})}{\log k} + 500\text{Opt}_k\right) \\ &= \frac{E[cost(P,\hat{C})]}{\log k} + 500\text{Opt}_k \end{aligned}$$

Now the theorem follows from (Arthur & Vassilvitskii, 2007) where the authors prove that $E[cost(P,\hat{C})] \leq (8\log k + 2)\text{Opt}_k$. So $E[cost(P,C)] \leq 509\text{Opt}_k$. □

In the rest of section we show Lemma 3, the main idea behind the proof is to compare the current set of centers and the optimal set of centers and to show that if the cost of the current solution is high($> 500\text{Opt}_k$) we are likely to sample a node that will improve the clustering. Our general proof strategy is inspired to the analysis of the single swap heuristic (Kanungo et al., 2002), although we modified them to make them work in our context.

4.1. Proof of Lemma 3

We assume that the optimal solution $C^* = \{c_1^*, \ldots, c_k^*\}$ is unique (this can be enforced using proper tiebreakers) and use $P_1^*, \ldots, P_k^*$ to denote the corresponding optimal partition. We will also use $C = \{c_1, \ldots, c_k\}$ to refer to our current clustering with corresponding partition $P_1, \ldots, P_k$. When the indices are not relevant, we will drop the index and write, for example, $c \in C$.

We use some ideas from (Kanungo et al., 2002) that have been used to analyze the single swap heuristic. We say that an optimal center c^* is captured by a center $c \in C$, if c is the nearest center to c^* among all centers in C. Note that a center $c \in C$ may capture more than one optimal center and every optimal center is captured by exactly one center from C (ties are dealt with in an arbitrary way). Hence, some center in c may not capture any optimal center. Similarly to (Kanungo et al., 2002) we call these centers *lonely*. Let L be the index set of lonely centers and let H be the index set of centers capturing exactly one cluster. Wlog. we assume that for $h \in H$ we have that $c_h \in C$ captures $c_h^* \in C^*$, i.e. the indices of the clusters with a one-to-one correspondence are identical.

We will use the above definition in the following way. If a center c captures exactly one cluster of the optimal solution, we think of it as a candidate center for this cluster. In this case, if c is far away from the center of this optimal cluster, we argue that with good probability we sample a point close to the center. In order to analyze the change of cost, we will argue that we can assign all points in the cluster of c that *are not in the captured optimal cluster* to a different center without increasing their contribution by too much. This will be called the reassignment cost and is formally defined in the definition below. We will show that with good probability we sample from a cluster such that the improvement for the points in the optimal cluster is significantly bigger than the reassignment cost.

If a center is lonely, we think of it as a center that can be moved to a different cluster. Again, we will argue that with high probability we can sample points from other clusters such that the reassignment cost is much smaller than the improvement for this cluster.

Now we start to analyze the cost of reassignment of the points due to a center swap.

We would like to argue that reassigning the points currently assigned to a cluster center with index from H or L to other clusters is small. In the case of clusters with index $h \in H$, we will assign all points from P_h that are not in P_h^* to other centers. In the case of clusters with index $l \in L$ we will consider the cost of assigning all points in P_i to other clusters. We introduce the following definition to captures the cost of this reassignment.

Definition 1. *Let $P \subseteq \mathbb{R}^d$ be a point set and $C \subseteq \mathbb{R}^d$ be a set of k cluster centers and let H be the subset of indices of cluster centers from $C = \{c_1, \ldots, c_k\}$ that capture exactly one cluster center of an optimal solution $C^* = \{c_1^*, \ldots, c_k^*\}$. Let P_i, P_i^*, $1 \leq i \leq k$, be the corresponding clusters. Let $h \in H$ be an index with cluster P_h and wlog. let P_h^* be the cluster in the optimal solution captured by c_h. The reassignment cost of c_h is defined as*

$$\textit{reassign}(P,C,c_h) = \textit{cost}(P \backslash P_h^*, C \backslash \{c_h\}) - \textit{cost}(P \backslash P_h^*, C)$$

For $\ell \in L$ we define the reassignment cost of c_ℓ as

$$\textit{reassign}(P,C,c_\ell) = \textit{cost}(P, C \setminus \{c_\ell\}) - \textit{cost}(P,C)$$

We will now prove the following lemma on reassignment costs.

Lemma 4. *For $r \in H \cup L$ we have*

$$\textit{reassign}(P, C, c_r) \leq \frac{21}{100}\textit{cost}(P_r, C) + 24\textit{cost}(P_r, C^*).$$

Proof. We only present the case $r \in H$. The case $r \in L$ is almost identical (in fact, simpler). We observe that $\text{reassign}(P, C, c_r) = \text{cost}(P_r \setminus P_r^*, C \setminus \{r\}) - \text{cost}(P_r, \setminus P_r^*, C)$ since vertices in clusters other than P_r will still be assigned to their current center. Our main idea is as follows. If $r \in H$, we assign every point in $P_r \cap P_i^*$, $i \neq r$, to the center that captured the center of P_i^*. While this assignment may not be optimal, its cost gives an upper bound on the cost of reassigning the points. In order to get an estimate for the cost of reassigning the points, we proceed as follows: We move every point in $P_r \cap P_i^*$, $i \neq r$, to the center of P_i^*. After this movement, the closest center of C to these points is the center that captured the center of P_i^*, which, for points not in P_r^*, cannot be r, since r is in H. We then use the fact that the squared moved distance of each point equals its contribution to the optimal solution to get an upper bound on the cost change using Lemma 2. After this, we move the points back to their original location while keeping their cluster assignments fixed. Again we can use the bound on the overall moved distance together with Lemma 2 to obtain a bound on the change of cost. Combining the two bounds we obtain an upper bound on the increase of cost that comes from reassigning the points. Details follow.

Let Q_r be the (multi)set of points obtained from $P_r \setminus P_r^*$ by moving each point in $P_i^* \cap P_r$, $i \neq r$, to c_i^*. We apply Lemma 2 with $\epsilon = 1/10$ to get an upper bound for the change of cost with respect to C that results from moving the points to Q_r. For $p \in P_r \setminus P_r^*$ let $q_p \in Q_r$ be the point of Q_r to which p has been moved. We have:

$$\begin{aligned} & |\text{cost}(\{p\}, C) - \text{cost}(\{q_p\}, C)| \\ \leq\ & \frac{1}{10}\text{cost}(\{p\}, C) + 11 \cdot \text{cost}(\{p\}, C^*). \end{aligned}$$

Summing up over all points in $P_r \setminus P_r^*$ yields

$$\begin{aligned} & |\text{cost}(P_r \setminus P_r^*, C) - \text{cost}(Q_r, C)| \\ \leq\ & \frac{1}{10}\text{cost}(P_r \setminus P_r^*, C) + 11 \cdot \text{cost}(P_r \setminus P_r^*, C^*). \end{aligned}$$

Note that after this movement all points from $P_r \setminus P_r^*$ have been assigned to centers from $C \setminus \{r\}$. Now we analyze the cost of moving the points back to their original location while maintaining this assignment. Let $Q_{r,i}$ be the points in Q_r that are nearest to center $c_i \in C$ and let $P_{r,i}$ be the set of their original locations. For $p \in P_{r,i}$ that has been moved to $q_p \in Q_{r,i}$ we have:

$$\begin{aligned} & |\text{cost}(\{q_p\}, \{c_i\}) - \text{cost}(\{p\}, \{c_i\})| \\ \leq\ & \frac{1}{10}\text{cost}(\{q_p\}, \{c_i\}) + 11 \cdot \text{cost}(\{p\}, \{q_p\}) \end{aligned}$$

Summing up over all points in P_r and the corresponding points in Q_r yields

$$\begin{aligned} & |\text{cost}(Q_r, C) - \sum_{i=1}^{k} \text{cost}(P_{r,i}, \{c_i\})| \\ \leq\ & \frac{1}{10}\text{cost}(Q_r, C) + 11 \cdot \text{cost}(P_r \setminus P_r^*, C^*) \\ \leq\ & \frac{1}{10}\left(\frac{11}{10}\text{cost}(P_r \setminus P_r^*, C) + \right. \\ & \left. 11 \cdot \text{cost}(P_r \setminus P_r^*, C^*)\right) + 11 \cdot \text{cost}(P_r \setminus P_r^*, C^*) \\ \leq\ & \frac{11}{100}\text{cost}(P_r, C) + 13 \cdot \text{cost}(P_r, C^*). \end{aligned}$$

Hence,

$$\begin{aligned} \text{reassign}(P, C, c_r) &= |\text{cost}(P_r \setminus P_r^*, C) - \sum_i \text{cost}(P_{r,i}, \{c_i\})| \\ &\leq |\text{cost}(P_r \setminus P_r^*, C) - \text{cost}(Q_r, C)| + \\ &\quad |\text{cost}(Q_r, C) - \sum_i \text{cost}(P_{r,i}, c_i)| \\ &\leq \frac{21}{100}\text{cost}(P_r, C) + 24\text{cost}(P_r, C^*). \end{aligned}$$

For the first equality note that $\sum_i \text{cost}(P_{r,i}, c_i) = \text{cost}(P_r \setminus P_r^*, C - \{r\})$. □

Now that we have a good bound on the reassignment cost we make a case distinction. Recall that we assume that for every $h \in H$ the optimal center captured by c_h is c_H^*, i.e. the indices are identical. We first consider the case that $\sum_{h \in H} \text{cost}(P_h^*, C) > \frac{1}{3}\text{cost}(P, C)$.

With the previous lemma at hand, we can focus on the centers $h \in H$ where replacing h by an arbitrary point close to the optimal cluster center of the optimal cluster captured by h improves the cost of the solution significantly. We call such clusters *good* and make this notion precise in the following definition.

Definition 2. *A cluster index $h \in H$ is called* good*, if*

$$\textit{cost}(P_h^*, C) - \textit{reassign}(P, C, c_h) - 9\textit{cost}(P_h^*, \{c_h^*\}) > \frac{1}{100k} \cdot \textit{cost}(P, C).$$

The above definition estimates the gain of replacing c_h by a point close to the center of P_h^* by considering a clustering

that reassigns the points in P_h that do not belong to P_h^* and assigns all points in P_h^* to the new center. Now we want to show that we have a good probability to sample a good cluster. In particular, we first argue that the sum of cost of good clusters is large.

Lemma 5. *If* $3\sum_{h\in H} cost(P_h^*, C) > cost(P, C) \geq 500 Opt_k$*, then*

$$\sum_{h\in H, h \text{ is good}} cost(P_h^*, C) \geq \frac{1}{25} cost(P, C).$$

Proof. We have $\sum_{h\in H} \text{cost}(P_h^*, C) \geq \frac{1}{3}\text{cost}(P, C)$ and by the definition of good and Lemma 4

$$\begin{aligned}\sum_{h\in H, h \text{ is not good}} \text{cost}(P_h^*, C) &\leq \sum_{h\in H} reassign(P, C, c_h) + \\ &\quad 9\text{Opt}_k + \frac{1}{100}\text{cost}(P, C) \\ &\leq \frac{22}{100}\text{cost}(P, C) + 33\text{Opt}_k.\end{aligned}$$

Using that $\text{cost}(P, C) \geq 500\text{Opt}_k$ we obtain that

$$\sum_{h\in H, h \text{ is not good}} \text{cost}(P_h^*, C) \leq \frac{143}{500} \cdot \text{cost}(P, C).$$

So $\sum_{h\in H, h \text{ is good}} \text{cost}(P_h^*, C) \geq \frac{23}{500} \cdot \text{cost}(P, C)$. The lemma follows. □

Now we show that whenever a cluster has high cost wrt. C, it suffices to consider the points close to the optimal center to get an approximation of the cost of the cluster. We will then use this fact to argue that we sample with good probability a point close to the center. In the following lemma it will be helpful to think of Q as being a (good) cluster in the optimal solution and C being the centers of the current solution.

Lemma 6. *Let $Q \subseteq \mathbb{R}^d$ be a point set and let $C \subseteq \mathbb{R}^d$ be a set of k centers and let $\alpha \geq 9$. If $cost(Q, C) \geq \alpha \cdot cost(Q, \{\mu(Q)\})$ then*

$$cost(R, C) \geq \left(\frac{\alpha - 1}{8}\right) \cdot cost(Q, \{\mu(Q)\}),$$

where $R \subseteq Q$ is the subset of Q at squared distance at most $\frac{2}{|Q|} \cdot cost(Q, \{\mu(Q)\})$ from $\mu(Q)$.

Proof. We know that the closest center in C to $\mu(Q)$ has squared distance at least $\frac{\alpha-1}{|Q|} \cdot \text{cost}(Q, \{\mu(Q)\})$ as otherwise $\text{cost}(Q, C) < \alpha \cdot \text{cost}(Q, \{\mu(Q)\})$ by Lemma 1. Hence, the squared distance of every point in R to C is at least

$$\begin{aligned}(\sqrt{\alpha - 1} - \sqrt{2})^2 \cdot \text{cost}(Q, \{\mu(Q)\})/|Q| \geq \\ (\alpha - 1)/4 \cdot \text{cost}(Q, \{\mu(Q)\})/|Q|,\end{aligned}$$

where we use that $\alpha \geq 9$ and so $\sqrt{\alpha - 1} \geq 2\sqrt{2}$. Furthermore, by averaging we get $|R| \geq |Q|/2$, which together with the inequality above implies the result. □

Now we can argue that sampling according to sum of squared distances will provide us with constant probability with a good center. Consider any index $h \in H$ with h being good. We will apply Lemma 6 with $Q = P_h^*$ and $\alpha = cost(Q, C)/cost(Q, \mu(Q))$. Note that by the definition of good, we have that $\alpha \geq 9$. Now let us define R_h^* to be the set R guaranteed by Lemma 6. We have $\text{cost}(R_h^*, C) \geq \frac{\alpha-1}{8}\text{cost}(P_h^*, \{c_h^*\}) = \frac{\alpha-1}{8\alpha}\text{cost}(P_h^*, C) \geq \frac{1}{9}\text{cost}(P_h^*, C)$ by our choice of α (observe that c_h^* equals $\mu(P_h^*)$). Since the sum of squared distances of points in good clusters is at least $1/25\text{cost}(P, C)$ by Lemma 5, we conclude that $\sum_{h\in H, h \text{ is good}} \text{cost}(R_h^*, C) \geq \frac{1}{9\cdot 25}\text{cost}(P, C)$. Thus, the probability to sample a point from $\cup_{h\in H, h \text{ is good}}\text{cost}(R_h^*, C)$ is more than $1/1000$. By the definition of good, if we sample such a point $c \in R_h^*$ we can swap it with c_h to get a new clustering of cost at most $\text{cost}(P, C\backslash\{c_h\}\cup\{c\}) \leq \text{cost}(P, C) - \text{cost}(P_h^*, C) + \text{reassign}(P, C, \{c_h\}) + \text{cost}(P_h^*, \{c\})$. By Lemma 1 we know that $\text{cost}(P_h^*, \{c\}) \leq 9\text{cost}(P_h^*, \{c_h^*\})$. Hence, with probability at least $1/1000$ the new clustering has cost at most

$$\begin{aligned}&\text{cost}(P, C) - (\text{cost}(P_h^*, C) - \text{reassign}(P, H, c_h) \\ &-9\text{cost}(P_h^*, \{c_h^*\}) \\ \leq\ &(1 - \frac{1}{100k}) \cdot \text{cost}(P, C).\end{aligned}$$

This proves our lemma in the first case.

In the second case, we have $\sum_{h\in H} \text{cost}(P_h^*, C) < 1/3\text{cost}(P, C)$. Now let $R = \{1, \ldots, k\} \setminus H$, so we get $\sum_{r\in R} \text{cost}(P_r^*, C) \geq 2/3\text{cost}(P, C)$. Observe that R equals the index set of optimal cluster centers that were captured by centers that capture more than one optimal center. This is because every optimal center is captured by one center and R does not include H. In this case, if the index of a center of our current solution is in $R \setminus L$ we cannot easily move the cluster center without having impact on other clusters. What we do instead is to use the centers in L as candidate centers for a swap. Similar to the case above we will argue that we can swap a center from L with a point that is close to an optimal center of a cluster P_r^* for some $r \in R$.

Recall that we have already bounded the cost of reassigning a center in L so we just need to argue that the probability of sampling a good center is high enough.

In particular, we focus on the centers $r \in R$ and swap an arbitrary center $\ell \in L$ with an arbitrary point close to one of the centers in R to improve the cost of the solution. Slightly overloading notation, we call such cluster centers *good* and make this notion precise in the following definition.

Definition 3. *A cluster index $i \in \{1, \ldots, k\}$ is called* good, *if there exists a center $\ell \in L$ such that*

$$\textit{cost}(P_i^*, C) - \textit{reassign}(P, C, \ell) - 9\textit{cost}(P_i^*, \{c_i^*\}) > \frac{1}{100k} \cdot \textit{cost}(P, C).$$

The above definition estimates the cost of removing ℓ and inserting a new cluster center close to the center of P_i^* by considering a clustering that reassigns the points in P_i^* and assigns all points in P_i^* to the new center. In the following we will now argue that the sum of cost of good clusters is large, this will be useful to show that the probability of sampling such a cluster is high enough.

Lemma 7. *If $3\sum_{h \in H} \textit{cost}(P_h^*, C) \leq \textit{cost}(P, C)$ and $\textit{cost}(P, C) \geq 500\textit{Opt}_k$ we have*

$$\sum_{r \in R, r \textit{ is good}} \textit{cost}(P_r^*, C) \geq \frac{1}{20}\textit{cost}(P, C).$$

Proof. We have $\sum_{r \in R} \text{cost}(P_r^*, C) \geq 2/3\text{cost}(P, C)$. Note that $|R| \leq 2|L|$. By the definition of good and Lemma 4

$$\begin{aligned}
\sum_{r \in R, r \text{ is not good}} & \text{cost}(P_r^*, C) \\
& \leq 2|L| \min_{\ell \in L} \text{reassign}(P, C, \ell) + 9\text{Opt}_k \\
& + \frac{1}{100}\text{cost}(P, C) \\
& \leq 2\sum_{\ell \in L} \text{reassign}(P, C, \ell) + 9\text{Opt}_k \\
& + \frac{1}{100}\text{cost}(P, C) \\
& \leq \frac{43}{100}\text{cost}(P, C) + 57\text{Opt}_k.
\end{aligned}$$

Using that $\sum_{i \in \{1,\ldots,k\}} \text{cost}(P_i^*, C) \geq 500\text{Opt}_k$ we obtain that

$$\sum_{r \in R, r \text{ is not good}} \text{cost}(P_r^*, C) \leq \frac{11}{20}\text{cost}(P, C)$$

Now the bound follows by combining the previous inequality with $\sum_{r \in R} \text{cost}(P_r^*, C) \geq 2/3\text{cost}(P, C)$. □

Note that also in this case we can now argue similarly as in the other case that sampling according to sum of squared distances will provide us with constant probability with a good center using Lemma 6. In fact, since the sum of squared distances of points in good centers is at least $1/20\text{cost}(P, C)$ by Lemma 7, it follows together with Lemma 6 that we sample a point from a good cluster P_r^* that is within distance two times the average cost of the cluster with probability $\frac{1}{1000}$. By the definition of good, we know that such a point improves the cost of the current clustering by at least a factor of $(1 - \frac{1}{100k})$. Thus, Lemma 3 follows.

5. Experiments

In this experiment we study the performance of our algorithm in practice. In particular, we have two main goals:

- Study the reduction in the cost that we obtain by running our local search post-processing after k-means++
- Study the reduction in the cost that we obtain by running our local search post-processing after k-means++ and then we run the Lloyd's algorithm on the set of obtained centers

Algorithms. We compare four different algorithms:

- KM++: the classic k-means++ algorithm (Arthur & Vassilvitskii, 2007)
- LS++: our algorithm presented in Section 3
- LL-KM++: the classic k-means++ algorithm (Arthur & Vassilvitskii, 2007) followed by 10 steps of the Lloyd's algorithm (Lloyd, 2006)[2]
- LL-LS++: our algorithm presented in Section 3 followed by 10 steps of the Lloyd's algorithm (Lloyd, 2006)

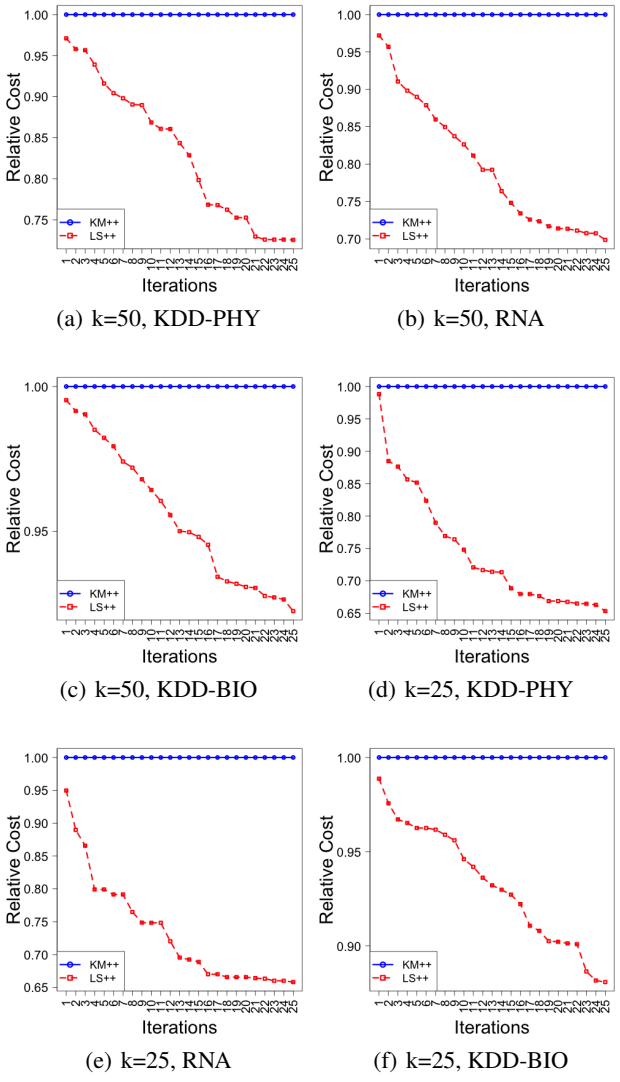
Datasets. We consider the k-means clustering for $k = 25$ or 50 on 3 different datasets:

- RNA – 8 features from 488565 RNA input sequence pairs (Uzilov et al., 2006)
- KDD-BIO – 145751 samples with 74 features measuring the match between a protein and a native sequence (KDD)
- KDD-PHY – 100000 samples with 78 features representing a quantum physic task (KDD)

We run all our experiments 3 times and we report the average of the results.

Results. In Figure 1 we compare the performance of our algorithm with the classic k-means++ for $k = 50$ and $k = 25$. We note that our algorithm improves substantially the performance of the classic k-means++ algorithm with a cost reduction that spans from the 8% to 35%. We also note that the gain that we obtain with our algorithm is stable across different datasets.

[2]We stop the Lloyd's algorithm when the incremental improvement of an iteration was small. In particular, after 10 steps of Lloyd's the improvement that we observed was less 0.4% per iteration for all considered datasets and for all different choices on the number of centers.



(a) k=50, KDD-PHY (b) k=50, RNA

(c) k=50, KDD-BIO (d) k=25, KDD-PHY

(e) k=25, RNA (f) k=25, KDD-BIO

Figure 1. Comparison between the cost of KM++ and LS++. The x axes is the number of local search iterations that we run for our algorithm. The y axes show the relative cost of the algorithm compared with the cost of KM++.

In Figure 2 we compare the performance of LL-LS++ with LL-KM++ for $k = 50$ and $k = 25$. We can observe that the reduction of the cost after Lloyd's is significantly reduced and it is between 1% and 18%. We also note that the reduction is slightly less stable, expecially for $k = 25$ and that in KDD-bio the gain is reduced substantially after running Lloyd's but the algorithm still outperform the baseline. Nevertheless we note that LL-LS++ almost always obtain better results than LL-KM++. Confirming the effectiveness of our seeding procedure.

Finally we observe that our algorithm is very efficient in practice. In fact, the time spent to run 25 iteration of our algorithm is smaller than the time spent in a *single* Lloyd's iteration.

6. Conclusions

We propose a simple variation of k-means++ algorithm based on local search and prove that the algorithm achieves a constant factor approximation. Furthermore we show experimentally the impact the effectiveness of our method.

It is an interesting open question to improve our analysis to reduce the number of local search steps needed in our analysis to obtain a constant factor approximation. In particular it would be nice to show that $O(k)$ steps are sufficient.

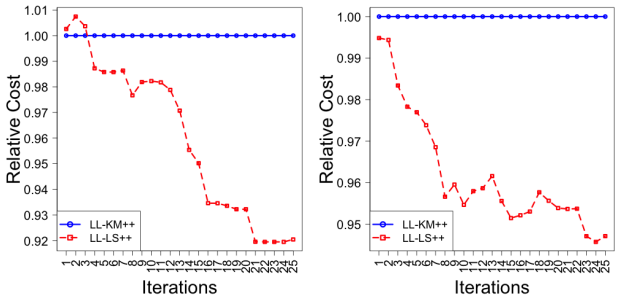
References

Kdd cup. 2004. Available at http://osmot.cs.cornell.edu/kddcup/datasets.html.

Ackermann, M. R., Märtens, M., Raupach, C., Swierkot, K., Lammersen, C., and Sohler, C. Streamkm++: A clustering algorithm for data streams. *ACM Journal of Experimental Algorithmics*, 17(1), 2012. doi: 10.1145/2133803.2184450. URL https://doi.org/10.1145/2133803.2184450.

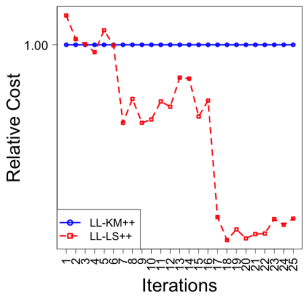
Aggarwal, A., Deshpande, A., and Kannan, R. Adaptive sampling for k-means clustering. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 12th International Workshop, APPROX 2009, and 13th International Workshop, RANDOM 2009, Berkeley, CA, USA, August 21-23, 2009. Proceedings*, pp. 15–28, 2009. doi: 10.1007/978-3-642-03685-9_2. URL https://doi.org/10.1007/978-3-642-03685-9_2.

Ahmadian, S., Norouzi-Fard, A., Svensson, O., and Ward, J. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pp. 61–

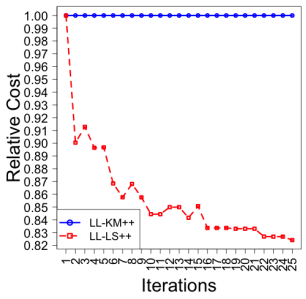


(a) k=50, KDD-PHY

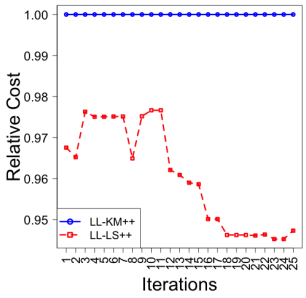
(b) k=50, RNA



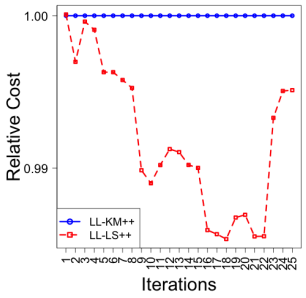
(c) k=50, KDD-BIO



(d) k=25, KDD-PHY



(e) k=25, RNA



(f) k=25, KDD-BIO

Figure 2. Comparison between the cost of LL-KM++ and LL-LS++. The x axes is the number of local search iterations that we run for our algorithm. The y axes show the relative cost of the algorithm compared with the cost of LL-KM++.

72, 2017. doi: 10.1109/FOCS.2017.15. URL https://doi.org/10.1109/FOCS.2017.15.

Arthur, D. and Vassilvitskii, S. k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pp. 1027–1035, 2007. URL http://dl.acm.org/citation.cfm?id=1283383.1283494.

Bachem, O., Lucic, M., Hassani, S. H., and Krause, A. Approximate k-means++ in sublinear time. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pp. 1459–1467, 2016. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12147.

Bahmani, B., Moseley, B., Vattani, A., Kumar, R., and Vassilvitskii, S. Scalable k-means++. *PVLDB*, 5(7):622–633, 2012. doi: 10.14778/2180912.2180915. URL http://vldb.org/pvldb/vol5/p622_bahmanbahmani_vldb2012.pdf.

Bandyapadhyay, S. and Varadarajan, K. R. On variants of k-means clustering. In *32nd International Symposium on Computational Geometry, SoCG 2016, June 14-18, 2016, Boston, MA, USA*, pp. 14:1–14:15, 2016. doi: 10.4230/LIPIcs.SoCG.2016.14. URL https://doi.org/10.4230/LIPIcs.SoCG.2016.14.

Becchetti, L., Bury, M., Cohen-Addad, V., Grandoni, F., and Schwiegelshohn, C. Oblivious dimension reduction for k-means – beyond subspaces and the johnson-lindenstrauss lemma. In *STOC 2019*, 2019.

Brunsch, T. and Röglin, H. A bad instance for k-means++. In *Theory and Applications of Models of Computation - 8th Annual Conference, TAMC 2011, Tokyo, Japan, May 23-25, 2011. Proceedings*, pp. 344–352, 2011. doi: 10.1007/978-3-642-20877-5_34. URL https://doi.org/10.1007/978-3-642-20877-5_34.

Chen, K. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM J. Comput.*, 39(3):923–947, 2009. doi: 10.1137/070699007. URL https://doi.org/10.1137/070699007.

Cohen, M. B., Elder, S., Musco, C., Musco, C., and Persu, M. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pp. 163–172, 2015. doi: 10.1145/2746539.2746569. URL https://doi.org/10.1145/2746539.2746569.

Cohen-Addad, V. A fast approximation scheme for low-dimensional k-means. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pp. 430–440, 2018. doi: 10.1137/1.9781611975031.29. URL https://doi.org/10.1137/1.9781611975031.29.

Cohen-Addad, V., Klein, P. N., and Mathieu, C. Local search yields approximation schemes for k-means and k-median in euclidean and minor-free metrics. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pp. 353–364, 2016. doi: 10.1109/FOCS.2016.46. URL https://doi.org/10.1109/FOCS.2016.46.

Feldman, D., Monemizadeh, M., and Sohler, C. A PTAS for k-means clustering based on weak coresets. In *Proceedings of the 23rd ACM Symposium on Computational Geometry, Gyeongju, South Korea, June 6-8, 2007*, pp. 11–18, 2007. doi: 10.1145/1247069.1247072. URL https://doi.org/10.1145/1247069.1247072.

Feldman, D., Schmidt, M., and Sohler, C. Turning big data into tiny data: Constant-size coresets for k-means, PCA and projective clustering. *CoRR*, abs/1807.04518, 2018. URL http://arxiv.org/abs/1807.04518.

Frahling, G. and Sohler, C. A fast k-means implementation using coresets. *Int. J. Comput. Geometry Appl.*, 18(6):605–625, 2008. doi: 10.1142/S0218195908002787. URL https://doi.org/10.1142/S0218195908002787.

Friggstad, Z., Rezapour, M., and Salavatipour, M. R. Local search yields a PTAS for k-means in doubling metrics. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pp. 365–374, 2016. doi: 10.1109/FOCS.2016.47. URL https://doi.org/10.1109/FOCS.2016.47.

Har-Peled, S. and Mazumdar, S. On coresets for k-means and k-median clustering. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pp. 291–300, 2004. doi: 10.1145/1007352.1007400. URL https://doi.org/10.1145/1007352.1007400.

Jain, K. and Vazirani, V. V. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *J. ACM*, 48(2):274–296, 2001. doi: 10.1145/375827.375845. URL https://doi.org/10.1145/375827.375845.

Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, A. Y. A local search approximation algorithm for k-means clustering. In *Proceedings of the 18th Annual Symposium on Computational Geometry, Barcelona, Spain, June 5-7, 2002*, pp. 10–18, 2002. doi: 10.1145/513400.513402. URL https://doi.org/10.1145/513400.513402.

Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., and Wu, A. Y. A local search approximation algorithm for k-means clustering. *Comput. Geom.*, 28(2-3):89–112, 2004.

Kumar, A., Sabharwal, Y., and Sen, S. Linear-time approximation schemes for clustering problems in any dimensions. *J. ACM*, 57(2):5:1–5:32, 2010. doi: 10.1145/1667053.1667054. URL https://doi.org/10.1145/1667053.1667054.

Lee, E., Schmidt, M., and Wright, J. Improved and simplified inapproximability for k-means. *Inf. Process. Lett.*, 120:40–43, 2017. doi: 10.1016/j.ipl.2016.11.009. URL https://doi.org/10.1016/j.ipl.2016.11.009.

Lloyd, S. Least squares quantization in pcm. *IEEE Trans. Inf. Theor.*, 28(2):129–137, September 2006. ISSN 0018-9448. doi: 10.1109/TIT.1982.1056489. URL http://dx.doi.org/10.1109/TIT.1982.1056489.

Makarychev, K., Makarychev, Y., and Razenshteyn, I. Performance of johnson–lindenstrauss transform for k-means and k-medians clustering. In *STOC 2019*, 2019.

Uzilov, A. V., Keegan, J. M., and Mathews, D. H. Detection of non-coding rnas on the basis of predicted secondary structure formation free energy change. *BMC Bioinformatics*, 7:173, 2006. doi: 10.1186/1471-2105-7-173. URL https://doi.org/10.1186/1471-2105-7-173.

Wei, D. A constant-factor bi-criteria approximation guarantee for k-means++. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 604–612, 2016.