On Variants of k-means Clustering*

Sayan Bandyapadhya[1] and Kasturi Varadarajan[2]

1 **Department of Computer Science, University of Iowa, Iowa City, USA**
sayan-bandyapadhyay@uiowa.edu
2 **Department of Computer Science, University of Iowa, Iowa City, USA**
kasturi-varadarajan@uiowa.edu

Abstract

Clustering problems often arise in fields like data mining and machine learning. Clustering usually refers to the task of partitioning a collection of objects into groups with similar elements, with respect to a similarity (or dissimilarity) measure. Among the clustering problems, *k-means* clustering in particular has received much attention from researchers. Despite the fact that k-means is a well studied problem, its status in the plane is still open. In particular, it is unknown whether it admits a PTAS in the plane. The best known approximation bound achievable in polynomial time is $9 + \varepsilon$.

In this paper, we consider the following variant of k-means. Given a set C of points in $\mathcal{R}^d$ and a real $f > 0$, find a finite set F of points in $\mathcal{R}^d$ that minimizes the quantity $f * |F| + \sum_{p \in C} \min_{q \in F} ||p - q||^2$. For any fixed dimension d, we design a PTAS for this problem that is based on local search. We also give a "bi-criterion" local search algorithm for k-means which uses $(1 + \varepsilon)k$ centers and yields a solution whose cost is at most $(1 + \varepsilon)$ times the cost of an optimal k-means solution. The algorithm runs in polynomial time for any fixed dimension.

The contribution of this paper is two-fold. On the one hand, we are able to handle the square of distances in an elegant manner, obtaining a near-optimal approximation bound. This leads us towards a better understanding of the k-means problem. On the other hand, our analysis of local search might also be useful for other geometric problems. This is important considering that little is known about the local search method for geometric approximation.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases k-means, Facility location, Local search, Geometric approximation

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.14

1 Introduction

Clustering is the task of partitioning a set of items (or objects) into groups of similar items with respect to a similarity (or dissimilarity) measure. Due to its fundamental nature clustering has several applications in fields like data mining, machine learning, pattern recognition, and image processing [8, 11, 12, 13, 14, 19, 20, 22, 31]. Often the objects to cluster are mapped to a high-dimensional metric space and the distance between two objects represents the similarity (or dissimilarity) between the objects. Then the goal is to minimize (or maximize) a certain objective function that depends on the distances between the objects. Among the different variants of clustering, the *k-means* problem in particular has received much attention. In k-means clustering, given a set P of n points in $\mathcal{R}^d$ and an integer $k > 0$,

* This material is based upon work supported by the National Science Foundation under Grant CCF-1318996.

the goal is to find a set K of k centers in $\mathcal{R}^d$, such that the quantity

$$cost(K) = \sum_{p \in P} \min_{q \in K} ||p - q||^2$$

is minimized. k-means is known to be $\mathcal{NP}$-hard even in the plane [1, 26]. The most common heuristic for k-means is an algorithm due to Lloyd [25] which is based on an iterative refinement technique. For historical reasons this algorithm is referred to as the k-means algorithm. Though this algorithm converges to a local optimum, the quality of the solution is not guaranteed to be "good" compared to the global optimum. Thus researchers have focused on designing polynomial time approximation algorithms that give some theoretical guarantee for all inputs. In fact, there are many $(1 + \varepsilon)$-factor approximation algorithms whose time complexity depend linearly on n [16, 17, 24, 28]. Unfortunately, the time complexity of these algorithms depends exponentially on k and hence they are not suitable in practice when k is large. For arbitrary k and d, the best known approximation factor is $9 + \varepsilon$ based on a local search technique [21]. On the other hand, Makarychev *et. al* [27] have designed three "bi-criteria" approximation algorithms for k-means that use at most βk $(> k)$ centers and achieve an approximation factor $\alpha(\beta)$ $(< 9 + \varepsilon)$ that depends on β. Moreover, $\alpha(\beta)$ decreases rapidly with β (for instance $\alpha(2) < 2.59, \alpha(3) < 1.4$). These algorithms have only polynomial dependence on the dimension of input points. Recently, Awasthi *et. al* [6] have studied the inapproximabilty of k-means. They have shown that k-means is APX-hard in sufficiently high $(\Omega(\log n))$ dimensions. However, as they have pointed out in their paper, the status of this problem in constant dimensions is still not resolved. See also [5, 30] for some related work.

An insight about the difficulty of k-means can be found by comparing it to *k-median* clustering. k-median is similar to k-means except the goal is to minimize the sum of distances, instead of the sum of squares of distances. Arora *et. al* [3] presented a PTAS for k-median in the plane based on a novel technique due to Arora [2]. Kolliopoulos and Rao [23] improved the time complexity significantly to $O(\rho n \log n \log k)$, where $\rho = exp[O((1 + \log 1/\varepsilon)/\varepsilon)^{d-1}]$ and ε is the constant of the PTAS. From the results on k-median one might conclude that a reason behind the difficulty of k-means is its objective function. One reason that squares of distances are harder to handle compared to Euclidean distances is they do not satisfy the triangle inequality in general. However, the important question in this context is, "Is the objective function of k-means by itself sufficient to make this problem harder?". We are interested in addressing this question in this paper. To set up the stage we consider another famous problem, which is called the facility location problem.

Facility location is similar to the k-median problem, where we are given a set of points (clients) in $\mathcal{R}^d$. The goal is to choose another set of points (facilities) in $\mathcal{R}^d$ which "serve" the clients. Though there is no global constraint on the number of facilities, for each facility, we need to pay a fixed cost. Here the objective function to minimize is the facility costs plus the sum of the distances from each of the clients to its nearest facility. Facility location has got much attention in operations research, approximation algorithms, etc. Arora *et. al* [3] give a PTAS for facility location in the plane using the same technique that they use to solve k-median. Actually, k-median has always been considered harder compare to facility location due to the global constraint on the number of centers as mentioned in [3]. Now going back to our original question for k-means one can infer that the global constraint in k-means might also play a crucial role. Motivated by this, we define the following variant of facility location.

Sum of Squares Facility Location Problem (SOS-FL). Given a set C of points (clients) in $\mathcal{R}^d$ and a real $f > 0$, find a finite set F of points (facilities) in $\mathcal{R}^d$, such that the quantity

$$cost(F) = f * |F| + \sum_{p \in C} \min_{q \in F} ||p - q||^2$$

is minimized. Note that SOS-FL is similar to k-means except the global constraint on the number of facilities (or centers) is absent here. In this paper we study the following interesting question.

Is it possible to get a PTAS for SOS-FL in $\mathcal{R}^d$ for fixed d?

We answer this question in the affirmative. In particular for any $\varepsilon > 0$, we give a $(1+\varepsilon)$-factor approximation for SOS-FL based on a *local search method*. The significance of this result is that it addresses our earlier question regarding k-means. To be precise it lets us infer that it is the joint effect of the global constraint and the objective function that makes k-means complicated to deal with.

Local Search. Local search is a popular technique in practice, and the study of when it yields approximation guarantees has a long history in combinatorial optimization. For geometric problems, results on approximation guarantees for local search have been relatively rare until recently. Thus we know very little about local search for geometric approximation. Arya *et. al* [4] gave a $3 + \frac{2}{p}$ factor approximation for metric k-median based on a local search that swaps p facilities; a different and arguably simplified analysis was given by Gupta and Tangwongsan [15]. The $9+\varepsilon$ factor approximation for k-means [21], mentioned above, builds on the analysis by Arya *et. al* [4]. Mustafa and Ray [29] gave a local search PTAS for the discrete hitting set problem over pseudodisks and r-admissible regions in the plane. Chan and Har-Peled [9] designed a local search method for the independent set problem over fat objects, and for pseudodisks in the plane, which yields a PTAS. Recently, Cohen-Addad and Mathieu [10] showed the effectiveness of local search technique for geometric optimization by designing local search algorithms for many geometric problems including k-median and facility location. For facility location they achieved a PTAS. For k-median their approach yields a $1+\varepsilon$ factor approximation using at most $(1+\varepsilon)k$ centers. Very recently, Bhattiprolu and Har-Peled [7] designed a local search PTAS for a geometric hitting set problem over balls in $\mathcal{R}^d$. Their PTAS also works for an infinite set of balls which can be represented implicitly in a specific way mentioned in their paper. One of the bicriteria algorithms for k-means in [27], mentioned above, is based on a local search method.

1.1 Our Results and Techniques

In this work we consider both SOS-FL and k-means. The main contribution of this work is that we are able to handle the square of distances in an elegant way, which yields near optimal approximation bounds. In particular, it gives a better understanding of the classical k-means problem, whose status in the plane has remained open for a long time. We design polynomial time approximation algorithms based on the local search technique for both of these problems. For any $\varepsilon > 0$, the algorithm for SOS-FL yields a $(1+\varepsilon)$-factor approximation. For k-means, the algorithm uses at most $(1+\varepsilon)k$ centers and yields a solution whose cost is at most $(1+\varepsilon)$ times the cost of an optimal k-means solution.

The algorithm and the analysis for both of the problems are similar. However, in case of k-means there are more subtleties, which arise due to the limitation on the number of

Algorithm 1 Local Search

Require: A set of clients $C \subset \mathcal{R}^d$, an $f > 0$, and a parameter $\varepsilon > 0$.
Ensure: A set of facilities F.
1: $F \leftarrow$ the set of facilities with one facility at each client
2: **while** $\exists$ a set F_1 s.t. $cost(F_1) < (1 - \frac{1}{n})cost(F)$ and $|F_1 \setminus F| + |F \setminus F_1| \leq \frac{c}{\varepsilon^d}$ **do**
3: $\quad F \leftarrow F_1$
4: **return** F

centers. In general, both of the algorithms are based on a local search method that allows swapping in and out of constant number of facilities or centers. Like the approaches in [7, 9, 10, 29] we also use separators to prove the quality of the approximation. To be precise we use the separator from [7], which turns out to be quite suitable for the purpose of handling the square of distances. The separator is used repeatedly to partition the local and global optimal facilities simultaneously into constant sized "parts", like the analysis of the hitting set algorithm in [7]. The rest of the analysis involves assignment of clients corresponding to the local facilities of each "part" to the global facilities corresponding to that "part" only or to some "auxilliary" points. Here one should be careful that a client should not be assigned to a point "far" away from it compared to its nearest local and global facility. The choice of the separator plays a crucial role to give a bound on this cost. From a high level our approach is similar to that of the work of Cohen-Addad and Mathieu [10], which is one source of inspiration for our work. But the details of the analysis are significantly different in places. For example, they use the dissection technique from [23] as their separator and thus the assignment in their case is completely different and more complicated than ours. In this regard we would like to mention, that the dissection technique from [23] or the quadtree based approach of Arora [2] are not flexible enough to handle the square of distances. The local search algorithms for SOS-FL and k-means are described in Section 2 and Section 3, respectively.

2 PTAS for Sum of Squares Facility Location

In this section we describe a simple local search algorithm for SOS-FL. We show that the solution returned by this algorithm is within $(1 + O(\varepsilon))$-factor of the optimal solution for any $\varepsilon > 0$. Recall that in SOS-FL we are given a set C of points in $\mathcal{R}^d$ and a real $f > 0$. Let $|C| = n$. For a point p and a set R of points, let $d(p, R) = \min_{q \in R} ||p - q||$.

2.1 The Local Search Algorithm

The local search algorithm starts with the solution where one facility is placed at each client (see Algorithm 1). Note that the cost of this solution is nf. Denote by OPT the cost of any optimal solution. As $OPT \geq f$, the initial solution has cost at most $n \cdot OPT$. In each iteration the algorithm looks for local improvement. The algorithm returns the current set of facilities if there is no such improvement. Notice that in line 2, we consider swaps with at most $\frac{c}{\varepsilon^d}$ facilities, where c is a constant. Next we show that this algorithm runs in polynomial time.

We note that in each iteration of the while loop the cost of F gets dropped by a factor of at least $(1 - \frac{1}{n})$. Let the number of iterations of the while loop be t. As we start with a

solution of cost at most $n \cdot OPT$, after t iterations we have

$$cost(F) \leq \left(1 - \frac{1}{n}\right)^t \cdot n \cdot OPT.$$

As $cost(F) \geq OPT$, we have

$$OPT \leq \left(1 - \frac{1}{n}\right)^t \cdot n \cdot OPT.$$

Thus the number t of iterations of the while loop is $O(n \log n)$. Now consider a single iteration of the while loop: we need to compute if there exists an F_1 such that $cost(F_1) < (1 - \frac{1}{n}) cost(F)$ and $|F_1 \setminus F| + |F \setminus F_1| \leq \frac{c}{\varepsilon^d}$. Assuming such an F_1 exists, fix one such F_1. Let $A = F \setminus F_1$. Since $|A| \leq \frac{c}{\varepsilon^d}$, we can enumerate over the possibilities for A. Let us therefore assume we have A. We would like to find a set $A' \subset \mathcal{R}^d$ with $\lambda := |A'| \leq \frac{c}{\varepsilon^d} - |A|$ that minimizes $cost((F \setminus A) \cup A')$. Since each point in A' has d coordinates, this is an optimization problem with $\lambda \cdot d$ variables once we fix λ.

This optimization problem can be solved in polynomial time using techniques similar to the ones used by [18]. To explain this further, let the optimal A' be $\{a'_1, a'_2, \ldots, a'_\lambda\}$, let χ be the assignment that assigns each client in C to the nearest facility in $(F \setminus A) \cup A'$. For solving our optimization problem, it suffices to know χ, even though A' is unknown: we can set a'_i to be the centroid of the points $\chi^{-1}(a'_i)$. For $c \in C$, there are $\lambda + 1$ possibilities for $\chi(c)$: the nearest facility in $F \setminus A$, or one of the λ unknown facilities in A'. Naively, this suggests $(\lambda + 1)^n$ possibilites for χ. However, if we view the problem as an optimization problem in $\lambda \cdot d$ dimensions, and use standard ideas about arrangements, we can see that we only need to consider $n^{O(\lambda \cdot d)}$ possibilities for χ, and these can be enumerated in $n^{O(\lambda \cdot d)}$ time. We conclude that Algorithm 1 can be implemented in polynomial time.

▶ Remark. An alternative approach to solving the problem of finding an A' of size λ that minimizes $cost((F \setminus A) \cup A')$ is as follows. Our analysis for local search works even if we solve this problem to within an approximation of $(1 + \frac{1}{2n})$. For this purpose, we can compute a $\frac{1}{2n}$-approximate centroid set $\mathcal{D} \subset \mathcal{R}^d$ of size $n^{O(d)}$ in time $n^{O(d)}$, by adapting an algorithm of Matousek [28]. We can think of $\mathcal{D}$ as a suitable discretization of $\mathcal{R}^d$. With $\mathcal{D}$ in hand, we simply minimize $cost((F \setminus A) \cup A')$ over all subsets $A' \subset \mathcal{D}$ of size λ, in time $n^{O(\lambda \cdot d)}$.

2.2 Analysis of the Local Search Algorithm

We analyze the local search algorithm using a partitioning scheme based on the separator theorem from [7]. The idea is to partition the set of facilities in the local search solution and an optimal solution into parts of size $O(\frac{1}{\varepsilon^d})$. Now for each such small part, we assign the clients corresponding to the local facilities of that part either to the optimal facilities in that part or to the points belonging to a special set. This yields a new solution, whose symmetric difference with the local search solution contains $O(\frac{1}{\varepsilon^d})$ facilities. Thus, using the local optimality criteria, the cost of this new solution is not "small" compared to the local search solution. This gives us one inequality for each part. Then we combine the inequalities corresponding to all the parts to give a bound on the cost of the local search solution. To start with we describe the separator theorem.

2.2.1 Separator Theorem

A ball B is said to be stabbed by a point p if $p \in B$. The following theorem is due to Bhattiprolu and Har-Peled [7] which shows the existence of a "small" point set (separator), that divides a given set of points into two in a "balanced" manner.

Algorithm 2 PARTITION$(\mathcal{L}, \mathcal{O}, \varepsilon)$

$\mu = \frac{\gamma}{\varepsilon^d}$, $i = 1$, $L_1 = \mathcal{L}$, $O_1 = \mathcal{O}$, $\mathcal{Z}_1 = \emptyset$
while $|L_i \cup O_i \cup \mathcal{Z}_i| > \alpha\mu$ **do**
  Let B_i, T_i be the ball and the point set computed by applying the Separator algorithm on the set $L_i \cup O_i \cup \mathcal{Z}_i$ with parameter μ
  $\mathcal{L}_i = L_i \cap B_i, \mathcal{O}_i = O_i \cap B_i$
  $L_{i+1} = L_i \setminus \mathcal{L}_i, O_{i+1} = O_i \setminus \mathcal{O}_i$
  $\mathcal{Z}_{i+1} = (\mathcal{Z}_i \setminus B_i) \cup T_i$
  $i = i + 1$
$I = i$
Let B_I be any ball that contains all the points in $L_I \cup O_I \cup \mathcal{Z}_I$
$T_I = \emptyset$

▶ **Theorem 1** ([7] Separator Theorem). *Let X be a set of points in $\mathcal{R}^d$, and $\mu > 0$ be an integer such that $|X| > \alpha\mu$, where α is a constant. There is an algorithm which can compute, in $O(|X|)$ expected time, a set $\mathcal{T}$ of $O(\mu^{1-\frac{1}{d}})$ points and a sphere $\mathcal{S}$ such that (a) the number of points of X inside $\mathcal{S}$ is $\Theta(\mu)$, and (b) any ball that intersects $\mathcal{S}$ and is stabbed by a point of X is also stabbed by a point of $\mathcal{T}$.*

The next corollary follows from Theorem 1, and will be useful for the analysis of our local search algorithm.

▶ **Corollary 2.** *Let X be a set of points in $\mathcal{R}^d$,and $\mu > 0$ be an integer such that $|X| > \alpha\mu$, where α is a constant. There is an algorithm which can compute, in $O(|X|)$ expected time, a set $\mathcal{T}$ of $O(\mu^{1-\frac{1}{d}})$ points and a ball B such that (a) $|B \cap X| = \Theta(\mu)$, and (b) for any point $p \in \mathcal{R}^d$, $d(p, \mathcal{T}) \leq \max\{d(p, X \setminus B), d(p, B \cap X)\}$.*

Proof. We use the same Algorithm in Theorem 1 to compute the sphere $\mathcal{S}$ and the set $\mathcal{T}$. Let B be the ball that has $\mathcal{S}$ as its boundary. Now consider any point $p \in \mathcal{R}^d$. Let p_1 (resp. p_2) be a point in $B \cap X$ (resp. $X \setminus B$) nearest to p. If $p \in B$, the ball B_1 centered at p and having radius $||p - p_2||$ must intersect the sphere $\mathcal{S}$, as $p_2 \notin B$. As p_2 stabs B_1, by Theorem 1 there is a point in $\mathcal{T}$ that also stabs B_1. Hence $d(p, \mathcal{T}) \leq ||p - p_2|| = d(p, X \setminus B)$. Similarly, if $p \notin B$, the ball B_2 centered at p and having radius $||p - p_1||$ intersects the sphere $\mathcal{S}$, as $p_1 \in B$. As B_2 is stabbed by p_1, by Theorem 1 there is a point in $\mathcal{T}$ that also stabs B_2. Hence $d(p, \mathcal{T}) \leq ||p - p_1|| = d(p, B \cap X)$ and the corollary follows. ◀

The algorithm in Corollary 2 will be referred to as the *Separator algorithm.*

2.2.2 The Partitioning Algorithm

For the sake of analysis fix an optimal solution $\mathcal{O}$. Let $\mathcal{L}$ be the solution computed by the local search algorithm. We design a procedure PARTITION$(\mathcal{L}, \mathcal{O}, \varepsilon)$ (see Algorithm 2) which divides the set $\mathcal{L} \cup \mathcal{O}$ into disjoint subsets of small size using the Separator algorithm. The procedure iteratively removes points from the set until the size of the set becomes less than or equal to $\alpha\mu$, where α is the constant in Corollary 2, and $\mu = \frac{\gamma}{\varepsilon^d}$ for some constant γ. This procedure is similar to the ApproximateSeparator algorithm in [7]. Next, we describe some important properties of this procedure that we will need to bound the cost of the local search solution. But before proceeding further we define some notation.

Let $T = \cup_{i=1}^{I} T_i$ be the union of the point sets computed by the Separator algorithm in PARTITION$(\mathcal{L}, \mathcal{O}, \varepsilon)$. Also let $C_l = \{p \in C | d(p, \mathcal{L}) \leq d(p, \mathcal{O})\}$ and $C_o = C \setminus C_l$. Consider

a point set $R \subset \mathcal{R}^d$. We denote the nearest neighbor voronoi diagram of R by $\mathcal{V}_R$. For $p \in R$, let $\mathcal{V}_R(p)$ be the voronoi cell of p in $\mathcal{V}_R$. Also let $C_R(p) = \mathcal{V}_R(p) \cap C$, that is $C_R(p)$ is the set of clients that are contained in the voronoi cell of p in $\mathcal{V}_R$. For $Q \subseteq R$, define $C_R(Q) = \cup_{q \in Q} C_R(q)$.

Now consider a client c. Denote its nearest neighbor in $\mathcal{O}$ (resp. $\mathcal{L}$) by $c(\mathcal{O})$ (resp. $c(\mathcal{L})$). Also let $c_O = ||c - c(\mathcal{O})||^2$ and $c_L = ||c - c(\mathcal{L})||^2$.

▶ **Definition 3.** An *assignment* is a function that maps a set of clients to the set $\mathcal{L} \cup \mathcal{O} \cup T$.

Now with all these definitions we move on towards the analysis. We begin with the following observation.

▶ **Observation 4.** Consider the procedure PARTITION$(\mathcal{L}, \mathcal{O}, \varepsilon)$. The following assertions hold.

1. $|\mathcal{L}_i \cup \mathcal{O}_i \cup T_i \cup (\mathcal{Z}_i \cap B_i)| \leq \frac{\beta}{\varepsilon^d}$ for a constant β and each $1 \leq i \leq I$
2. $I \leq \varepsilon(|\mathcal{L}| + |\mathcal{O}|)/10$
3. $|T| \leq \varepsilon(|\mathcal{L}| + |\mathcal{O}|)/10$
4. $\sum_{i=1}^{I} |T_i \cup (\mathcal{Z}_i \cap B_i)| \leq \varepsilon(|\mathcal{L}| + |\mathcal{O}|)/5$

Proof. 1. Note that $|T_i| = O(\mu^{1-\frac{1}{d}}) = O(\frac{1}{\varepsilon^{d-1}})$ and $|\mathcal{L}_i \cup \mathcal{O}_i \cup \mathcal{Z}_i| = O(\mu) = O(\frac{1}{\varepsilon^d})$ for $1 \leq i \leq I$. Thus there is a constant β such that $|\mathcal{L}_i \cup \mathcal{O}_i \cup T_i \cup (\mathcal{Z}_i \cap B_i)| \leq \frac{\beta}{\varepsilon^d}$.
2. As in each iteration we add $O(\mu^{1-\frac{1}{d}})$ points and remove $\theta(\mu)$ points, the number of iterations $I = O(\frac{|\mathcal{L}|+|\mathcal{O}|}{\mu}) = O(\varepsilon^d(\frac{|\mathcal{L}|+|\mathcal{O}|}{\gamma}))$. By choosing the constant γ sufficiently large one can ensure that $I \leq \varepsilon(|\mathcal{L}| + |\mathcal{O}|)/10$.
3. $|T| = \sum_{i=1}^{I} |T_i| = O(\frac{|\mathcal{L}|+|\mathcal{O}|}{\mu} \cdot \mu^{1-\frac{1}{d}}) = O(\varepsilon(|\mathcal{L}| + |\mathcal{O}|)/\gamma^{\frac{1}{d}}) \leq \varepsilon(|\mathcal{L}| + |\mathcal{O}|)/10$, by choosing the value of γ sufficiently large.
4. Consider any point $p \in T_i$ for some $1 \leq i \leq I$. If $p \in \mathcal{Z}_j \cap B_j$ for some $j > i$, then p is removed in iteration j and hence cannot appear in any other $\mathcal{Z}_t \cap B_t$ for $j+1 \leq t \leq I$. Thus p can appear in at most two sets of the collection $\{T_1 \cup (\mathcal{Z}_1 \cap B_1), \ldots, T_I \cup (\mathcal{Z}_I \cap B_I)\}$. It follows that $\sum_{i=1}^{I} |T_i \cup (\mathcal{Z}_i \cap B_i)| \leq 2 \sum_{i=1}^{I} |T_i| = 2|T| \leq \varepsilon(|\mathcal{L}| + |\mathcal{O}|)/5$. ◀

The next lemma states the existence of a "cheap" assignment for any client c such that its nearest neighbor $c(\mathcal{O})$ in $\mathcal{O}$ is in $\mathcal{O}_i$ and its nearest neighbor $c(\mathcal{L})$ in $\mathcal{L}$ is in $\mathcal{L}_j$ for $i < j$.

▶ **Lemma 5.** *Consider any client c, such that $c(\mathcal{O}) \in \mathcal{O}_i$, $c(\mathcal{L}) \in \mathcal{L}_j$ with $1 \leq i < j \leq I$. Also consider the sets T_j, $\mathcal{Z}_j$, and the ball B_j computed by PARTITION$(\mathcal{L}, \mathcal{O}, \varepsilon)$. There exists a point $p \in (\mathcal{Z}_j \cap B_j) \cup T_j$ such that $||c - p|| \leq \max\{||c - c(\mathcal{O})||, ||c - c(\mathcal{L})||\}$.*

Proof. To prove this lemma, we first establish the following claim.

▶ **Claim 6.** *For any $i + 1 \leq t \leq j$, there exists a point $p \in \mathcal{Z}_t$ such that $||c - p|| \leq \max\{||c - c(\mathcal{O})||, ||c - c(\mathcal{L})||\}$.*

Proof. We prove this claim using induction on the iteration number. In base case consider the iteration i. Let $X = L_i \cup O_i \cup \mathcal{Z}_i$. As $c(\mathcal{O}) \in B_i$ and $c(\mathcal{L}) \in B_j$, $c(\mathcal{O}), c(\mathcal{L}) \in X$. Now by Corollary 2, $d(c, T_i) \leq \max\{d(c, X \setminus B_i), d(c, B_i \cap X)\}$. As $c(\mathcal{O})$ is the nearest neighbor of c in $\mathcal{O}$ and $c(\mathcal{O}) \in B_i$, $d(c, B_i \cap X) \leq ||c - c(\mathcal{O})||$. Also $c(\mathcal{L})$ is the nearest neighbor of c in $\mathcal{L}$ and $c(\mathcal{L}) \notin B_i$. Thus $d(c, X \setminus B_i) \leq ||c - c(\mathcal{L})||$. Hence $d(c, T_i) \leq \max\{||c - c(\mathcal{O})||, ||c - c(\mathcal{L})||\}$. Let p be the point in T_i nearest to c. As $T_i \subseteq \mathcal{Z}_{i+1}$, $p \in \mathcal{Z}_{i+1}$ and the base case holds.

Now suppose the claim is true for any iteration $t < j - 1 \leq I - 1$. We show that the claim is also true for iteration $t + 1$. By induction, there is a point $p \in \mathcal{Z}_{t+1}$ such that $||c - p|| \leq \max\{||c - c(\mathcal{O})||, ||c - c(\mathcal{L})||\}$. Now there can be two cases: (i) $p \notin B_{t+1}$, and

(ii) $p \in B_{t+1}$. Consider the first case. In this case, by definition of $\mathcal{Z}_{t+2}$, $p \in \mathcal{Z}_{t+2}$ and the claim holds. Thus consider the second case. Let $X = L_{t+1} \cup O_{t+1} \cup \mathcal{Z}_{t+1}$. By Corollary 2, $d(c, T_{t+1}) \leq \max\{d(c, X \setminus B_{t+1}), d(c, B_{t+1} \cap X)\}$. As $p \in B_{t+1} \cap X$, $d(c, B_{t+1} \cap X) \leq ||c-p|| \leq \max\{||c - c(\mathcal{O})||, ||c - c(\mathcal{L})||\}$. Now $c(\mathcal{L}) \notin B_{t+1}$, as $t < j - 1$. Also $c(\mathcal{L}) \in L_{t+1} \subseteq X$. Thus $d(c, X \setminus B_{t+1}) \leq ||c - c(\mathcal{L})||$. It follows that $d(c, T_{t+1}) \leq \max\{||c - c(\mathcal{O})||, ||c - c(\mathcal{L})||\}$. Let q be the point in T_{t+1} nearest to c. As $T_{t+1} \subseteq \mathcal{Z}_{t+2}$, $q \in \mathcal{Z}_{t+2}$ and the claim holds also for this case. ◀

Consider the iteration j. From Claim 6 it follows that there exists a point $p \in \mathcal{Z}_j$ such that $||c - p|| \leq \max\{||c - c(\mathcal{O})||, ||c - c(\mathcal{L})||\}$. Thus if $p \in B_j$, then $p \in \mathcal{Z}_j \cap B_j$, and we are done. Note that the way B_I is chosen, $\mathcal{Z}_I \subseteq B_I$. Thus in case $j = I$, $p \in \mathcal{Z}_j \cap B_j$. Hence consider the case when $j \neq I$ and $p \notin B_j$. Let $X = L_j \cup O_j \cup \mathcal{Z}_j$. As $p \in \mathcal{Z}_j$, $p \in X$. Also by Corollary 2, $d(c, T_j) \leq \max\{d(c, X \setminus B_j), d(c, B_j \cap X)\}$. As $c(\mathcal{L}) \in B_j \cap X$, $d(c, B_j \cap X) \leq ||c - c(\mathcal{L})||$. Now $p \in X \setminus B_j$. Thus $d(c, X \setminus B_j) \leq ||c - p|| \leq \max\{||c - c(\mathcal{O})||, ||c - c(\mathcal{L})||\}$. Hence $d(c, T_j) \leq \max\{||c - c(\mathcal{O})||, ||c - c(\mathcal{L})||\}$ and the lemma follows. ◀

Now we extend Lemma 5 for any client whose nearest neighbor in $\mathcal{L}$ is in $\mathcal{L}_j$, but the nearest neighbor in $\mathcal{O}$ is not in $\mathcal{O}_j$, where $1 \leq j \leq I$.

▶ **Lemma 7.** *Consider the sets T_j, $\mathcal{Z}_j$ and the ball B_j computed by PARTITION$(\mathcal{L}, \mathcal{O}, \varepsilon)$, where $1 \leq j \leq I$. There is an assignment g of the clients in $C_{\mathcal{L}}(\mathcal{L}_j) \setminus C_{\mathcal{O}}(\mathcal{O}_j)$ to $T_j \cup (\mathcal{Z}_j \cap B_j)$ with the following properties:*

1. *for $c \in (C_{\mathcal{L}}(\mathcal{L}_j) \setminus C_{\mathcal{O}}(\mathcal{O}_j)) \cap C_l$, $||c - g(c)||^2 \leq c_O$;*
2. *for $c \in (C_{\mathcal{L}}(\mathcal{L}_j) \setminus C_{\mathcal{O}}(\mathcal{O}_j)) \cap C_o$, $||c - g(c)||^2 \leq c_L$.*

Proof. We show how to construct the assignment g for each client in $C_{\mathcal{L}}(\mathcal{L}_j) \setminus C_{\mathcal{O}}(\mathcal{O}_j)$. Consider any client $c \in C_{\mathcal{L}}(\mathcal{L}_j) \setminus C_{\mathcal{O}}(\mathcal{O}_j)$. Let $\mathcal{O}_i$ be the subset of $\mathcal{O}$ that contains $c(\mathcal{O})$. If j is equal to I, then $i < j$. Otherwise, there could be two cases: (i) $i < j$, and (ii) $i > j$. Consider the case when $i < j$ for $1 \leq j \leq I$. By Lemma 5, there is a point $p \in (\mathcal{Z}_j \cap B_j) \cup T_j$ such that $||c - p|| \leq \max\{||c - c(\mathcal{O})||, ||c - c(\mathcal{L})||\}$. Let $g(c)$ be p in this case. Now consider the case when $i > j$ such that $j < I$. Let $X = L_j \cup O_j \cup \mathcal{Z}_j$. By Corollary 2, $d(c, T_j) \leq \max\{d(c, X \setminus B_j), d(c, B_j \cap X)\}$. As $c(\mathcal{L}) \in B_j \cap X$, $d(c, B_j \cap X) \leq ||c - c(\mathcal{L})||$. Now note that $c(\mathcal{O}) \in X$. Also $c(\mathcal{O}) \notin B_j$, as $c(\mathcal{O}) \in \mathcal{O}_i$ and $i > j$. Thus $c(\mathcal{O}) \in X \setminus B_j$ and $d(c, X \setminus B_j) \leq ||c - c(\mathcal{O})||$. Hence $d(c, T_j) \leq \max\{||c - c(\mathcal{O})||, ||c - c(\mathcal{L})||\}$. Let $g(c)$ be the point in T_j nearest to c in this case.

In both cases $||c - g(c)|| \leq \max\{||c - c(\mathcal{L})||, ||c - c(\mathcal{O})||\}$. If $c \in C_l$, $||c - g(c)||^2 \leq ||c - c(\mathcal{O})||^2 = c_O$. Otherwise, $c \in C_o$, and thus $||c - g(c)||^2 \leq ||c - c(\mathcal{L})||^2 = c_L$. Hence the lemma holds. ◀

2.2.3 Approximation Bound

The next lemma gives an upper bound on the quality of the local search solution.

▶ **Lemma 8.** $cost(\mathcal{L}) \leq (1 + O(\varepsilon))cost(\mathcal{O})$.

Proof. Fix an iteration i, where $1 \leq i \leq I$. Consider the solution $S_i = (\mathcal{L} \setminus \mathcal{L}_i) \cup \mathcal{O}_i \cup T_i \cup (\mathcal{Z}_i \cap B_i)$. By Observation 4, $|\mathcal{L}_i \cup \mathcal{O}_i \cup T_i \cup (\mathcal{Z}_i \cap B_i)| \leq \frac{\beta}{\varepsilon^d}$. Thus $|\mathcal{L} \setminus S_i| + |S_i \setminus \mathcal{L}| \leq \frac{\beta}{\varepsilon^d}$. By choosing the constant c in Algorithm 1 sufficiently large, one can ensure that $\beta \leq c$. Hence due to the local optimality condition in Algorithm 1 it follows that,

$$cost(S_i) \geq (1 - \frac{1}{n})cost(\mathcal{L})\,. \tag{1}$$

To upper bound the cost of S_i we use an assignment of the clients to the facilities in S_i. Consider a client c. There can be three cases: (i) c is nearer to a facility of $\mathcal{O}_i$ than the facilities in $(\mathcal{O} \setminus \mathcal{O}_i) \cup (\mathcal{L} \setminus \mathcal{L}_i)$, that is, $c \in C_{\mathcal{O}}(\mathcal{O}_i) \cap (C_o \cup C_{\mathcal{L}}(\mathcal{L}_i))$, (ii) c is not in $C_{\mathcal{O}}(\mathcal{O}_i)$ and c is nearer to a facility of $\mathcal{L}_i$ than the facilities in $\mathcal{L} \setminus \mathcal{L}_i$, that is, $c \in C_{\mathcal{L}}(\mathcal{L}_i) \setminus C_{\mathcal{O}}(\mathcal{O}_i)$, (iii) c does not appear in case (i) and (ii), that is, c is not in the union of $C_{\mathcal{O}}(\mathcal{O}_i) \cap (C_o \cup C_{\mathcal{L}}(\mathcal{L}_i))$ and $C_{\mathcal{L}}(\mathcal{L}_i) \setminus C_{\mathcal{O}}(\mathcal{O}_i)$. Let R_i be the set of the clients that appear in case (iii). Note that a client cannot appear in both cases (i) and (ii), as the sets $C_{\mathcal{O}}(\mathcal{O}_i) \cap (C_o \cup C_{\mathcal{L}}(\mathcal{L}_i))$ and $C_{\mathcal{L}}(\mathcal{L}_i) \setminus C_{\mathcal{O}}(\mathcal{O}_i)$ are disjoint. Also note, that if $c \in C_{\mathcal{L}}(\mathcal{L}_i)$, c must appear in case (i) or (ii). Thus if c is corresponding to case (iii), its nearest neighbor $c(\mathcal{L})$ in $\mathcal{L}$ must be in $\mathcal{L} \setminus \mathcal{L}_i$.

Now we describe the assignment. Note that we can assign the clients only to the facilities in $S_i = (\mathcal{L} \setminus \mathcal{L}_i) \cup \mathcal{O}_i \cup T_i \cup (\mathcal{Z}_i \cap B_i)$. For a client of type (i), assign it to a facility in $\mathcal{O}_i$ nearest to it. For a client of type (ii), use the assignment g in Lemma 7 to assign it to a point in $T_i \cup (\mathcal{Z}_i \cap B_i)$. For a client of type (iii), assign it to a facility in $\mathcal{L} \setminus \mathcal{L}_i$ nearest to it. Thus by Inequality 1,

$$|(\mathcal{L} \setminus \mathcal{L}_i) \cup \mathcal{O}_i \cup T_i \cup (\mathcal{Z}_i \cap B_i)|f + \sum_{c \in C_{\mathcal{O}}(\mathcal{O}_i) \cap (C_o \cup C_{\mathcal{L}}(\mathcal{L}_i))} c_O + \sum_{c \in C_{\mathcal{L}}(\mathcal{L}_i) \setminus C_{\mathcal{O}}(\mathcal{O}_i)} ||c - g(c)||^2 + \sum_{c \in R_i} c_L \geq (1 - \frac{1}{n})(|\mathcal{L}|f + \sum_{c \in C} c_L)\,. \quad (2)$$

By Lemma 7, for a client c in $(C_{\mathcal{L}}(\mathcal{L}_i) \setminus C_{\mathcal{O}}(\mathcal{O}_i)) \cap C_o$, $||c - g(c)||^2$ is at most c_L; for a client c in $(C_{\mathcal{L}}(\mathcal{L}_i) \setminus C_{\mathcal{O}}(\mathcal{O}_i)) \cap C_l$, $||c - g(c)||^2$ is at most c_O. It follows that,

$$|\mathcal{O}_i \cup T_i \cup (\mathcal{Z}_i \cap B_i)|f + \sum_{c \in C_{\mathcal{O}}(\mathcal{O}_i) \cap (C_o \cup C_{\mathcal{L}}(\mathcal{L}_i))} (c_O - c_L) + \sum_{c \in (C_{\mathcal{L}}(\mathcal{L}_i) \setminus C_{\mathcal{O}}(\mathcal{O}_i)) \cap C_o} (c_L - c_L) + \sum_{c \in (C_{\mathcal{L}}(\mathcal{L}_i) \setminus C_{\mathcal{O}}(\mathcal{O}_i)) \cap C_l} (c_O - c_L) \geq -\frac{1}{n} cost(\mathcal{L}) + |\mathcal{L}_i|f \quad (3)$$

$$\Rightarrow |\mathcal{O}_i \cup T_i \cup (\mathcal{Z}_i \cap B_i)|f + \sum_{c \in C_{\mathcal{O}}(\mathcal{O}_i) \cap (C_o \cup C_{\mathcal{L}}(\mathcal{L}_i))} (c_O - c_L) + \sum_{c \in (C_{\mathcal{L}}(\mathcal{L}_i) \setminus C_{\mathcal{O}}(\mathcal{O}_i)) \cap C_l} (c_O - c_L) \geq -\frac{1}{n} cost(\mathcal{L}) + |\mathcal{L}_i|f \quad (4)$$

$$\Rightarrow |\mathcal{O}_i \cup T_i \cup (\mathcal{Z}_i \cap B_i)|f + \sum_{c \in C_{\mathcal{O} \cup \mathcal{L}}(\mathcal{O}_i \cup \mathcal{L}_i)} (c_O - c_L) \geq -\frac{1}{n} cost(\mathcal{L}) + |\mathcal{L}_i|f\,.$$

The last inequality follows by noting that the union of $C_{\mathcal{O}}(\mathcal{O}_i) \cap (C_o \cup C_{\mathcal{L}}(\mathcal{L}_i))$ and $(C_{\mathcal{L}}(\mathcal{L}_i) \setminus C_{\mathcal{O}}(\mathcal{O}_i)) \cap C_l$ is equal to the set $C_{\mathcal{O} \cup \mathcal{L}}(\mathcal{O}_i \cup \mathcal{L}_i)$. Summing over all i we get,

$$\sum_{i=1}^{I} |\mathcal{O}_i|f + \sum_{i=1}^{I} |T_i \cup (\mathcal{Z}_i \cap B_i)|f + \sum_{i=1}^{I} \sum_{c \in C_{\mathcal{O} \cup \mathcal{L}}(\mathcal{O}_i \cup \mathcal{L}_i)} (c_O - c_L) \geq -\frac{I}{n} cost(\mathcal{L}) + \sum_{i=1}^{I} |\mathcal{L}_i|f\,. \quad (5)$$

By Observation 4, $\sum_{i=1}^{I} |T_i \cup (\mathcal{Z}_i \cap B_i)| = O(\varepsilon(|\mathcal{L}| + |\mathcal{O}|))$. Hence we get,

$$|\mathcal{O}|f + O(\varepsilon(|\mathcal{L}| + |\mathcal{O}|))f + \sum_{c \in C} (c_O - c_L) \geq -\frac{I}{n} cost(\mathcal{L}) + |\mathcal{L}|f\,.$$

Since $I = O(\varepsilon(|\mathcal{L}| + |\mathcal{O}|)) = O(\varepsilon n)$ (Observation 4), we obtain

$$cost(\mathcal{L}) \leq (1 + O(\varepsilon))cost(\mathcal{O})\,.$$ ◀

Algorithm 3 Local Search

Require: A set of points $P \subset \mathcal{R}^d$, an integer k, a constant $\varepsilon > 0$.
Ensure: A set of centers.
1: $K \leftarrow$ the solution returned by the algorithm in [21]
2: Add arbitrary centers to K to ensure $|K| = (1 + 5\varepsilon)k$
3: **while** $\exists$ a set K_1 s.t. $|K_1| \leq (1+5\varepsilon)k$, $cost(K_1) < (1-\frac{1}{n})cost(K)$ and $|K_1 \setminus K| + |K \setminus K_1| \leq \frac{c}{\varepsilon^{2d}}$ **do**
4: $K \leftarrow K_1$
5: If needed, add arbitrary centers to K to ensure $|K| = (1 + 5\varepsilon)k$
6: **return** K

As mentioned before, the running time of Algorithm 1 is polynomial for fixed d, and hence we have established the following theorem.

▶ **Theorem 9.** *There is a local search algorithm for SOS-FL which yields a PTAS.*

3 Bi-criteria Approximation scheme for k-means

In this section we describe a local search algorithm for k-means which uses $(1 + O(\varepsilon))k$ centers and yields a solution whose cost is at most $(1 + O(\varepsilon))$ times the cost of an optimal k-means solution. The local search algorithm and its analysis are very similar to the ones for SOS-FL. Recall that in k-means we are given a set P of n points in $\mathcal{R}^d$ and an integer $k > 0$.

3.1 The Local Search Algorithm

Fix an $\varepsilon > 0$. The local search algorithm (see Algorithm 3) starts with the solution computed by the $9 + \varepsilon$ factor approximation algorithm in [21]. Upon termination, the locally optimal solution K has exactly $(1 + 5\varepsilon)k$ centers. Using an argument similar to the one in case of SOS-FL one can show that this algorithm also runs in polynomial time.

3.2 Analysis of the Local Search Algorithm

Let $\mathcal{L}$ be the solution computed by Algorithm 3. For the sake of analysis fix an optimal solution $\mathcal{O}$. We use the procedure PARTITION$(\mathcal{L}, \mathcal{O}, \varepsilon)$ to compute the sets $\mathcal{L}_i$, $\mathcal{O}_i$, $\mathcal{Z}_i \cap B_i$ and T_i for $1 \leq i \leq I$. We use the same $\mu = \frac{\gamma}{\varepsilon^d}$ in this procedure. We note, that Observation 4, Lemma 5, and Lemma 7 hold in this case also, as they directly follow from the PARTITION procedure, which works on any two input sets of points designated by $\mathcal{L}$ and $\mathcal{O}$. Let $R_i = \mathcal{L}_i \cup \mathcal{O}_i \cup T_i \cup (\mathcal{Z}_i \cap B_i)$ for $1 \leq i \leq I$. Note, by Observation 4, that $|R_i| \leq \frac{\beta}{\varepsilon^d}$. Also note that $\mathcal{Z}_i \cap B_i \subseteq T$ for each $1 \leq i \leq I$, where $T = \cup_{i=1}^{I} T_i$. Now we use the following lemma to group the balls returned by PARTITION into groups of "small" size. This lemma is similar to the Balanced Clustering Lemma in [10].

▶ **Lemma 10.** *Consider the collection $R = \{R_1, \ldots, R_I\}$ of sets with $R_j = \mathcal{L}_j \cup \mathcal{O}_j \cup T_j \cup (\mathcal{Z}_j \cap B_j)$ and $|R_j| \leq \frac{\beta}{\varepsilon^d}$ for $1 \leq j \leq I$, where β is the constant in Observation 4. There exists a collection $\mathcal{P} = \{P_1, \ldots, P_p\}$, with $P_i \subseteq R$ for $1 \leq i \leq p$, $P_i \cap P_j = \emptyset$ for any $1 \leq i < j \leq p$, and $\cup_{i=1}^{p} P_i = R$, which satisfies the following properties:*
1. $|P_i| \leq \frac{2\beta}{\varepsilon^d}$ *for* $1 \leq i \leq p$;
2. $\sum_{R_j \in P_i} |\mathcal{L} \cap R_j| \geq \sum_{R_j \in P_i} |(\mathcal{O} \cup T) \cap R_j|$ *for* $1 \leq i \leq p$.

We note that $\mathcal{L} \cap R_j = \mathcal{L}_j$ and $(\mathcal{O} \cup T) \cap R_j = \mathcal{O}_j \cup T_j \cup (\mathcal{Z}_j \cap B_j)$ for $1 \leq j \leq I$. Before proving the lemma we use it to get an approximation bound on the quality of the local search solution.

▶ **Lemma 11.** *cost*$(\mathcal{L}) \leq (1 + O(\varepsilon))$*cost*$(\mathcal{O})$.

Proof. Consider the collection $\mathcal{P}$ as mentioned in Lemma 10. Also consider any element J of $\mathcal{P}$. Let $\mathcal{L}_J = \cup_{R_i \in J}(\mathcal{L} \cap R_i)$, $\mathcal{O}_J = \cup_{R_i \in J}(\mathcal{O} \cap R_i)$, and $T_J = \cup_{R_i \in J}(T \cap R_i)$. Now consider the solution $S_J = (\mathcal{L} \setminus \mathcal{L}_J) \cup \mathcal{O}_J \cup T_J$. By Lemma 10, and using the fact that $\mathcal{L} \cap R_i$ and $\mathcal{L} \cap R_j$ are disjoint for $i \neq j$, $|\mathcal{L}_J| = \sum_{R_j \in J} |\mathcal{L} \cap R_j| \geq \sum_{R_j \in J} |(\mathcal{O} \cup T) \cap R_j| \geq |\mathcal{O}_J \cup T_J|$. Thus by definition of S_J, $|S_J| \leq |\mathcal{L}| = (1 + 5\varepsilon)k$. Now J contains at most $\frac{2\beta}{\varepsilon^d}$ sets and thus $|\mathcal{L} \setminus S_J| + |S_J \setminus \mathcal{L}| \leq |\mathcal{L}_J \cup \mathcal{O}_J \cup T_J| \leq \frac{2\beta}{\varepsilon^d}\frac{\beta}{\varepsilon^d} \leq \frac{2\beta^2}{\varepsilon^{2d}}$. By choosing the constant c in Algorithm 3 sufficiently large, one can ensure that $2\beta^2 \leq c$. Hence the local optimality condition in Algorithm 3 implies that

$$cost(S_J) \geq (1 - \frac{1}{n})cost(\mathcal{L})\,. \tag{6}$$

To upper bound the cost of S_J we use an assignment of the clients to the facilities in S_J. Consider a client c. There can be three cases: (i) c is nearer to a facility of $\mathcal{O}_J$ than the facilities in $(\mathcal{O} \setminus \mathcal{O}_J) \cup (\mathcal{L} \setminus \mathcal{L}_J)$, that is, $c \in C_1 = C_{\mathcal{O}}(\mathcal{O}_J) \cap (C_o \cup C_{\mathcal{L}}(\mathcal{L}_J))$, (ii) c is not in $C_{\mathcal{O}}(\mathcal{O}_J)$ and c is nearer to a facility of $\mathcal{L}_J$ than the facilities in $\mathcal{L} \setminus \mathcal{L}_J$, that is, $c \in C_2 = C_{\mathcal{L}}(\mathcal{L}_J) \setminus C_{\mathcal{O}}(\mathcal{O}_J)$, (iii) c does not appear in case (i) and (ii), that is, $c \in C_3 = C \setminus (C_1 \cup C_2)$. Note that if $c \in C_{\mathcal{L}}(\mathcal{L}_J)$, then $c \in C_1 \cup C_2$. Thus if $c \in C_3$, its nearest neighbor in $\mathcal{L}$ must be in $\mathcal{L} \setminus \mathcal{L}_J$. Also note that for a client $c \in C_2$, it is the case that $c(\mathcal{L}) \in \mathcal{L}_i \subseteq \mathcal{L}_J$ and $c(\mathcal{O}) \in \mathcal{O}_j \subseteq (\mathcal{O} \setminus \mathcal{O}_J)$ for some $1 \leq i \neq j \leq I$. Thus Lemma 7 is applicable to c, and $g(c) \in T_i \cup (\mathcal{Z}_i \cap B_i) \subseteq T_J$.

Now we describe the assignment. For a client in C_1, assign it to a facility in $\mathcal{O}_J$ nearest to it. For a client $c \in C_2$, use the assignment g in Lemma 7. For a client in C_3, assign it to a facility in $\mathcal{L} \setminus \mathcal{L}_J$ nearest to it. Thus by Inequality 6,

$$\sum_{c \in C_1} c_O + \sum_{c \in C_2} ||c - g(c)||^2 + \sum_{c \in C_3} c_L \geq (1 - \frac{1}{n}) \sum_{c \in C} c_L$$

$$\Rightarrow \sum_{c \in C_1} (c_O - c_L) + \sum_{c \in (C_2 \cap C_o)} (c_L - c_L) + \sum_{c \in (C_2 \cap C_l)} (c_O - c_L) \geq -\frac{1}{n} cost(\mathcal{L})$$

$$\Rightarrow \sum_{c \in C_1} (c_O - c_L) + \sum_{c \in (C_2 \cap C_l)} (c_O - c_L) \geq -\frac{1}{n} cost(\mathcal{L})$$

$$\Rightarrow \sum_{c \in C_{\mathcal{O} \cup \mathcal{L}}(\mathcal{O}_J \cup \mathcal{L}_J)} (c_O - c_L) \geq -\frac{1}{n} cost(\mathcal{L})\,.$$

The last inequality follows by noting that $C_1 \cup (C_2 \cap C_l) = C_{\mathcal{O} \cup \mathcal{L}}(\mathcal{O}_J \cup \mathcal{L}_J)$. Adding over all $J \in \mathcal{P}$ we get,

$$\sum_{J \in \mathcal{P}} \sum_{c \in C_{\mathcal{O} \cup \mathcal{L}}(\mathcal{O}_J \cup \mathcal{L}_J)} (c_O - c_L) \geq -\frac{|\mathcal{P}|}{n} cost(\mathcal{L})$$

$$\Rightarrow \sum_{c \in C}(c_O - c_L) \geq -\frac{|\mathcal{P}|}{n} cost(\mathcal{L}) .$$

From Observation 4, it follows that $|\mathcal{P}| \leq I = O(\varepsilon(|\mathcal{L}| + |\mathcal{O}|)) = O(\varepsilon k)$. Thus,

$$cost(\mathcal{L}) \leq (1 + O(\varepsilon)) cost(\mathcal{O}) .$$ ◀

As mentioned before, the running time of Algorithm 1 is polynomial for fixed d, and hence we have established the following theorem.

▶ **Theorem 12.** *There is a polynomial time local search algorithm for k-means that uses $(1 + O(\varepsilon))k$ facilities and returns a solution with cost at most $(1 + O(\varepsilon))$ times the cost of the optimal k-means solution.*

Next we present the proof of Lemma 10.

3.2.1 Proof of Lemma 10

For purposes of exposition we introduce some more notation. For a set $r \subseteq \mathcal{L} \cup \mathcal{O} \cup T$, let $u(r) = |\mathcal{L} \cap r| - |(\mathcal{O} \cup T) \cap r|$. For a collection Ψ of sets, let $u(\Psi) = \sum_{r \in \Psi} |\mathcal{L} \cap r| - |(\mathcal{O} \cup T) \cap r|$. Using this notation we rewrite the statement of Lemma 10 as follows.

▶ **Lemma 13.** *Consider the collection $R = \{R_1, \ldots, R_I\}$ of sets with $R_j = \mathcal{L}_j \cup \mathcal{O}_j \cup T_j \cup (\mathcal{Z}_j \cap B_j)$ and $|R_j| \leq \frac{\beta}{\varepsilon^d}$ for $1 \leq j \leq I$, where β is the constant in Observation 4. There exists a collection $\mathcal{P} = \{P_1, \ldots, P_p\}$, with $P_i \subseteq R$ for $1 \leq i \leq p$, $P_i \cap P_j = \emptyset$ for any $1 \leq i < j \leq p$, and $\cup_{i=1}^{p} P_i = R$, which satisfies the following properties:*
1. *$|P_i| \leq \frac{2\beta}{\varepsilon^d}$ for $1 \leq i \leq p$, where β is the constant in Observation 4;*
2. *$u(P_i) \geq 0$ for $1 \leq i \leq p$.*

Proof. Note that for each j, $u(R_j) \in [-\frac{\beta}{\varepsilon^d}, \frac{\beta}{\varepsilon^d}]$, as $|R_j| \leq \frac{\beta}{\varepsilon^d}$. Now by Observation 4, $\sum_{j=1}^{I} |T_j \cup (\mathcal{Z}_j \cap B_j)| \leq \varepsilon(|\mathcal{L}| + |\mathcal{O}|)/5 \leq \varepsilon((1 + 5\varepsilon)k + k)/5 \leq 2\varepsilon k$. Thus,

$$u(R) \geq |\mathcal{L}| - |\mathcal{O}| - \sum_{j=1}^{I} |T_j \cup (\mathcal{Z}_j \cap B_j)| \geq (1 + 5\varepsilon)k - k - 2\varepsilon k \geq 3\varepsilon k.$$

Now we describe the construction of the collection $\mathcal{P}$. For any j, if $u(R_j)$ equals 0, we add $\{R_j\}$ to $\mathcal{P}$ as an element. Note that such an element satisfies the desired properties. Now consider all the sets $R_j \in R$ such that $|u(R_j)| \geq 1$. Denote by R' the collection of such sets. Note that $u(R') = u(R)$. We process R' using the construction in Algorithm 4.

In each iteration of the outer while loop in line 2, we remove at most $l = \frac{2\beta}{\varepsilon^d}$ sets from R' and add the collection Ψ of these sets to $\mathcal{P}$ as an element. These l sets are chosen carefully so that $u(\Psi)$ is non-negative. To ensure this, at first $l/2$ sets are chosen to get the collection Ψ' such that $-l/2 \leq u(\Psi') \leq l/2$. Then we add at most $l/2$ more sets $\{r\}$ with $u(r) > 0$ (while loop in lines 17-19) to obtain a collection Ψ with $u(\Psi) \geq 0$. Assuming the loop invariant $u(R') \geq 0$ at the beginning of each iteration of the outer while loop, such r must exist. Later we will argue that this loop invariant holds. This ensures that the algorithm exits the while loop in lines 17-19 with a Ψ such that $u(\Psi) \geq 0$. Also we stop the addition of sets as soon as $u(\Psi)$ becomes non-negative. This ensures that $u(\Psi) \leq \frac{\beta}{\varepsilon^d}$. Note that $|\Psi| \leq l = \frac{2\beta}{\varepsilon^d}$. Thus Ψ satisfies all the desired properties.

Now consider the selection of the $l/2$ sets of Ψ'. We select these sets sequentially, one in each iteration of the for loop in lines 4-15. Consider a particular iteration of this for loop. There can be two cases: (i) $u(\Psi') \geq 0$, and (ii) $u(\Psi') < 0$. In case (i) if there is a set r with

Algorithm 4

1: $l = \frac{2\beta}{\varepsilon^d}$
2: **while** $|R'| > l$ **do**
3: $\Psi' \leftarrow \{r\}$, where r is any element in R' with $u(r) > 0$; $R' \leftarrow R' \setminus \{r\}$
4: **for** $i = 1$ to $l/2$ **do**
5: **if** $u(\Psi') \geq 0$ **then**
6: **if** $u(r) > 0$ for each $r \in R'$ **then**
7: Add Ψ' to $\mathcal{P}$
8: **for each** $r \in R'$ **do**
9: Add $\{r\}$ to $\mathcal{P}$; $R' \leftarrow R' \setminus \{r\}$
10: **return**
11: $r \leftarrow$ any element in R' with $u(r) < 0$
12: $\Psi' \leftarrow \Psi' \cup \{r\}$; $R' \leftarrow R' \setminus \{r\}$
13: **else if** $u(\Psi') < 0$ **then**
14: $r \leftarrow$ any element in R' with $u(r) > 0$
15: $\Psi' \leftarrow \Psi' \cup \{r\}$; $R' \leftarrow R' \setminus \{r\}$
16: $\Psi \leftarrow \Psi'$
17: **while** $u(\Psi) < 0$ **do**
18: $r \leftarrow$ any element in R' with $u(r) > 0$
19: $\Psi \leftarrow \Psi \cup \{r\}$; $R' \leftarrow R' \setminus \{r\}$
20: Add Ψ to $\mathcal{P}$
21: Add R' to $\mathcal{P}$

$u(r) < 0$, we choose it. If there is no such set r, we add Ψ' to $\mathcal{P}$ and for any set $r \in R'$, $\{r\}$ is added to $\mathcal{P}$ as an element. The algorithm terminates. In case (ii) we choose a set r with $u(r) > 0$. Assuming the loop invariant $u(R') \geq 0$ at the beginning of each iteration of the outer while loop, such an r must exist. Hence in both cases we can ensure that at the end of each iteration of the for loop in lines 4-15 $u(\Psi') \in [-\frac{\beta}{\varepsilon^d}, \frac{\beta}{\varepsilon^d}]$.

Let M denote the number of iterations of the outer while loop. Then in each step except the last, we remove at least $l/2$ sets from R'. Since R' has at most I sets initially, $(M-1)\frac{l}{2} \leq I \Rightarrow M \leq \frac{2I}{l} + 1$.

Now we argue that after iteration $0 \leq j \leq M$,

$$u(R') \geq (\frac{2I}{l} + 1)\frac{l}{2} - j\frac{l}{2}. \tag{7}$$

Since $(\frac{2I}{l} + 1)\frac{l}{2} - j\frac{l}{2} \geq (\frac{2I}{l} + 1)\frac{l}{2} - M\frac{l}{2} \geq 0$, this would imply $u(R') \geq 0$ after iteration $j \leq M$. This establishes the loop invariant and also shows that the set R' added to $\mathcal{P}$ in line 21 has $u(R') \geq 0$, completing the proof of the lemma.

We now show (7) by induction. The inequality is true for $j = 0$, since before iteration 1,

$$u(R') = u(R) \geq 3\varepsilon k \geq I + \frac{\beta}{\varepsilon^d} = (\frac{2I}{l} + 1)\frac{l}{2}.$$

Consider a j such that $1 \leq j \leq M$ and suppose (7) is true after iteration $j-1$. Then at the beginning of iteration j, we have $u(R') \geq (\frac{2I}{l} + 1)\frac{l}{2} - (j-1)\frac{l}{2}$. If the condition in line 6 is true in iteration j, then the algorithm terminates. Since R' becomes empty after this iteration, (7) trivially holds. If in iteration j we add Ψ to $\mathcal{P}$ in line 20, then $u(\Psi) \leq \frac{\beta}{\varepsilon^d} = \frac{l}{2}$. Thus, after iteration j,

$$u(R') \geq (\frac{2I}{l} + 1)\frac{l}{2} - (j-1)\frac{l}{2} - \frac{l}{2} = (\frac{2I}{l} + 1)\frac{l}{2} - j\frac{l}{2}.$$ ◀

References

1. Daniel Aloise, Amit Deshpande, Pierre Hansen, and Preyas Popat. Np-hardness of euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, 2009. doi:10.1007/s10994-009-5103-0.
2. Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, 1998. doi:10.1145/290179.290180.
3. Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for euclidean k-medians and related problems. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC'98, pages 106–113, New York, NY, USA, 1998. ACM. doi:10.1145/276698.276718.
4. Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM J. Comput.*, 33(3):544–562, 2004. doi:10.1137/S0097539702416402.
5. Pranjal Awasthi, Avrim Blum, and Or Sheffet. Stability yields a PTAS for k-median and k-means clustering. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 309–318, 2010. doi:10.1109/FOCS.2010.36.
6. Pranjal Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali Kemal Sinop. The hardness of approximation of euclidean k-means. In *31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands*, pages 754–767, 2015. doi:10.4230/LIPIcs.SOCG.2015.754.
7. Vijay V. S. P. Bhattiprolu and Sariel Har-Peled. Separating a voronoi diagram via local search. *CoRR*, abs/1401.0174, 2014. URL: http://arxiv.org/abs/1401.0174.
8. Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. Syntactic clustering of the web. *Computer Networks*, 29(8-13):1157–1166, 1997. doi:10.1016/S0169-7552(97)00031-7.
9. Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48(2):373–392, 2012. doi:10.1007/s00454-012-9417-5.
10. Vincent Cohen-Addad and Claire Mathieu. Effectiveness of local search for geometric optimization. In *31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands*, pages 329–343, 2015. doi:10.4230/LIPIcs.SOCG.2015.329.
11. Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990. doi:10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9.
12. Richard O Duda, Peter E Hart, et al. Pattern classification and scene analysis. *J. Wiley and Sons*, 1973.
13. Christos Faloutsos, Ron Barber, Myron Flickner, Jim Hafner, Wayne Niblack, Dragutin Petkovic, and William Equitz. Efficient and effective querying by image content. *J. Intell. Inf. Syst.*, 3(3/4):231–262, 1994. doi:10.1007/BF00962238.
14. Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996.
15. Anupam Gupta and Kanat Tangwongsan. Simpler analyses of local search algorithms for facility location. *CoRR*, abs/0809.2554, 2008. URL: http://arxiv.org/abs/0809.2554.
16. Sariel Har-Peled and Akash Kushal. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry*, 37(1):3–19, 2007. doi:10.1007/s00454-006-1271-x.

17 Sariel Har-Peled and Soham Mazumdar. 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*, pages 291–300, 2004. doi:10.1145/1007352.1007400.

18 Mary Inaba, Naoki Katoh, and Hiroshi Imai. Applications of weighted voronoi diagrams and randomization to variance-based *k*-clustering (extended abstract). In *Proceedings of the Tenth Annual Symposium on Computational Geometry, Stony Brook, New York, USA, June 6-8, 1994*, pages 332–339, 1994. doi:10.1145/177424.178042.

19 Anil K. Jain, Robert P. W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(1):4–37, 2000. doi:10.1109/34.824819.

20 Anil K. Jain, M. Narasimha Murty, and Patrick J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, 1999. doi:10.1145/331499.331504.

21 Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. *Comput. Geom.*, 28(2-3):89–112, 2004. doi:10.1016/j.comgeo.2004.03.003.

22 L. Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley, 1990.

23 Stavros G. Kolliopoulos and Satish Rao. A nearly linear-time approximation scheme for the euclidean k-median problem. *SIAM J. Comput.*, 37(3):757–782, 2007. doi:10.1137/S0097539702404055.

24 Amit Kumar, Yogish Sabharwal, and Sandeep Sen. Linear time algorithms for clustering problems in any dimensions. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, pages 1374–1385, 2005. doi:10.1007/11523468_111.

25 S. Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, Mar 1982. doi:10.1109/TIT.1982.1056489.

26 Meena Mahajan, Prajakta Nimbhorkar, and Kasturi R. Varadarajan. The planar k-means problem is np-hard. *Theor. Comput. Sci.*, 442:13–21, 2012. doi:10.1016/j.tcs.2010.05.034.

27 Konstantin Makarychev, Yury Makarychev, Maxim Sviridenko, and Justin Ward. A bi-criteria approximation algorithm for k means. *CoRR*, abs/1507.04227, 2015. URL: http://arxiv.org/abs/1507.04227.

28 J. Matoušek. On approximate geometric k-clustering. *Discrete & Computational Geometry*, 24(1):61–84. doi:10.1007/s004540010019.

29 Nabil H. Mustafa and Saurabh Ray. Improved results on geometric hitting set problems. *Discrete & Computational Geometry*, 44(4):883–895, 2010. doi:10.1007/s00454-010-9285-9.

30 Rafail Ostrovsky, Yuval Rabani, Leonard J. Schulman, and Chaitanya Swamy. The effectiveness of lloyd-type methods for the k-means problem. *J. ACM*, 59(6):28, 2012. doi:10.1145/2395116.2395117.

31 Michael J. Swain and Dana H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991. doi:10.1007/BF00130487.