Algorithmic Excursions: Topics in Computer Science II **Spring 2016**

Lecture 9 & 10: Local Search Algorithm for k-median Problem (contd.), k-means Problem

Lecturer: Kasturi Varadarajan *Scribe: Tanmay Inamdar*

5.1 Local Search Algorithm for k-median Problem

For convenience, for any set C, let $C - a + b = C \setminus \{a\} \cup \{b\}$. With this notation, we describe the Local Search algorithm:

Algorithm 1 Local Search (P, k)

1: $C \leftarrow$ arbitrary size-k subset of P
2: **while** $\exists \bar{c} \in C, \exists p \in P \setminus C$ s.t. $\text{Cost}(C - \bar{c} + p) < \text{Cost}(C)$ **do**
3: $\quad C \leftarrow C - \bar{c} + p$
4: **end while**

It is clear that this algorithm terminates. This is because $|C| = k$ is an invariant, and there are $\binom{n}{k}$ distinct possibilities for C. In each iteration, the $\text{Cost}(C)$ decreases, so each size-k subset of P is assigned to C at most once. This directly gives the upper bound on number of iterations to be $\binom{n}{k}$.

Note that this upper bound is exponential in k, and is equivalent to going through all size-k subsets in the worst case. Later we will discuss how to improve this.

5.1.1 Notation

Let us define some notation which will help us analyze the algorithm.
$L :=$ A solution (k-subset) returned by Local Search.
$C_{opt} :=$ An optimal solution for the k-median problem.
We will eventually show that $\text{Cost}(L) \leq 5 \cdot \text{Cost}(C_{opt})$.
For any $p \in P, C \subseteq P, \quad \text{NN}(p, C) := \bar{c} \in C$ that minimizes $d(p, \cdot)$. So $d(p, \text{NN}(p, C)) = d(p, C)$ by definition.
Also, for any $C \subseteq P, \bar{c} \in C, \quad \text{Cluster}(C, \bar{c}) := \{q \in P \mid \text{NN}(q, C) = \bar{c}\}$.
For an optimal center $\bar{o} \in C_{opt}$,

$$\text{opt}(\bar{o}) := \sum_{p \in \text{Cluster}(C_{opt}, \bar{o})} d(p, C_{opt})$$

$$\text{local}(\bar{o}) := \sum_{p \in \text{Cluster}(C_{opt}, \bar{o})} d(p, L)$$

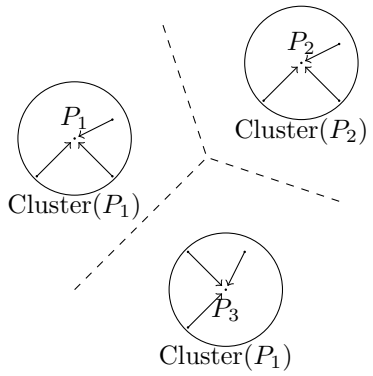


Figure 5.1: Clusters

5.1.2 Swapping Strategy

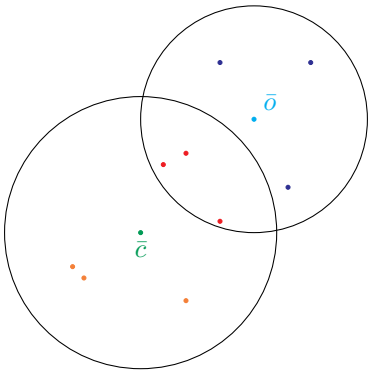


Figure 5.2: Consider the swap $\bar{c} \to \bar{o}$
Points $p \in \text{Cluster}(L, \bar{c}) \cap \text{Cluster}(C_{opt}, \bar{o})$ are assigned to closest center, i.e. $\bar{o}$.
Points $p \in \text{Cluster}(C_{opt}, \bar{o}) \backslash \text{Cluster}(L, \bar{c})$ might have a closer center in L, but they are assigned to $\bar{o}$ anyway.
Points $p \in \text{Cluster}(L, \bar{c}) \backslash \text{Cluster}(C_{opt}, \bar{o})$ will be (hopefully) assigned to some center $\alpha(q) \in L$.

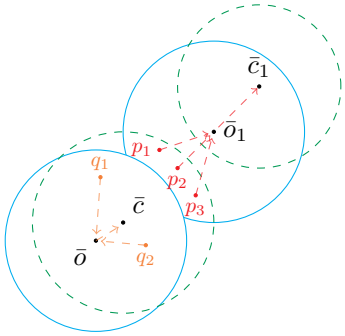


Figure 5.3: Thick circles are optimal clusters, and dashed circles are local clusters.
For p_1, p_2, p_3, $\alpha(p) = c_1 \neq \bar{c}$, and so they are "good".
For q_1, q_2, $\alpha(q) = \bar{c}$, and so they are "bad".

For some $\bar{c} \in L, \bar{o} \in C_{opt}$, consider the swap $\bar{c} \to \bar{o}$, where we replace L by $L - \bar{c} + \bar{o}$.

$$0 \leq \Delta(\bar{c}, \bar{o}) := \text{Cost}(L - \bar{c} + \bar{o}) - \text{Cost}(L) \tag{5.1}$$

The inequality is true because L is a local optimal solution.

For any point $p \in P$, let $\alpha(p) := \text{NN}(\text{NN}(p, C_{opt}), L)$. That is, $\alpha(p)$ is the nearest local center of the nearest optimal center of p.
If $\alpha(p) \neq \bar{c}$ for some $p \in \text{Cluster}(L, \bar{c}) \backslash \text{Cluster}(C_{opt}, \bar{o})$, then we can reassign them to $\alpha(p)$. However, it is not always the case. See Figure 5.2 for example.

Also, let $\delta_p := d(p, \alpha(p)) - d(p, L)$. Note that δ_p can be 0, but not negative.

Ignoring such a case (which will be handled separately), we will reassign points in Cluster$(L, \bar{c})$ as described in Figure 5.3. Therefore, from Equation 5.1, we get:

$$0 \leq \text{opt}(\bar{o}) - \text{local}(\bar{o}) + \sum_{q \in \text{Cluster}(L,\bar{c}) \setminus \text{Cluster}(C_{opt},\bar{o})} \delta_q \tag{5.2}$$

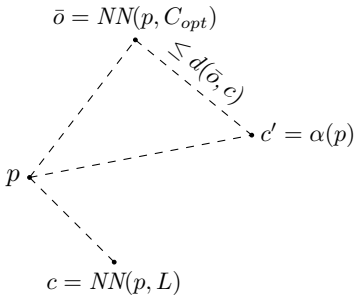
We will consider 1 swap involving each $\bar{o} \in C_{opt}$, so there will be total k swaps. Adding Equation 5.2 over all such swaps, we will get

$$\text{Cost}(L) \leq \text{Cost}(C_{opt}) + \text{“Error Term”}$$

If we can bound the “Error Term” by $c \cdot \text{Cost}(C_{opt})$, then we will get $c+1$ approximation for k-median problem. The following claim will be eventually used to bound the “Error Term”.

Claim 5.1 *For any $p \in P$, $\delta_p \leq 2 \cdot d(p, C_{opt})$*
Proof:

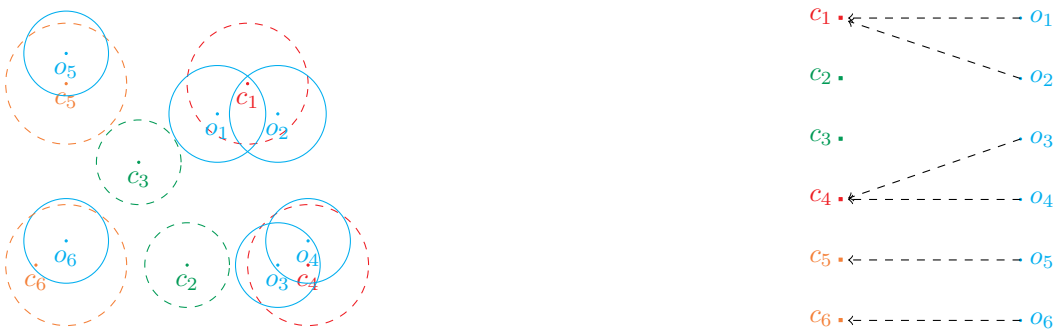
$$\begin{aligned}
d(p,\alpha(p)) &\leq d(p,\bar{o}) + d(\bar{o},\alpha(p)) && \text{(Triangle inequality)} \\
&= d(p,\bar{o}) + d(\bar{o},c') && (\alpha(p) = c') \\
&\leq d(p,\bar{o}) + d(\bar{o},c) && \\
&\quad \text{(Since } c' = \text{NN}(\bar{o}, L),\ d(\bar{o},c') \leq d(\bar{o},c)) && \\
&\leq d(p,\bar{o}) + d(\bar{o},p) + d(p,c) && \\
d(p,\alpha(p)) - d(p,c) &\leq 2d(p,\bar{o}) && \text{(Rearranging)} \\
\delta_p &\leq 2d(p,\bar{o}) && \text{(Definition of } \delta_p) \\
&= 2d(p, C_{opt}) && (\bar{o} = \text{NN}(p, C_{opt}))
\end{aligned}$$



■

5.1.3 Drifters, Anchors, and Tyrants

Consider the bipartite graph (Figure 5.4 (b)) defined by centers in an optimal solution C_{opt}, and their nearest neighbor in L. We classify the centers in L as follows:
$c \in L$ is a Tyrant if its in-degree is strictly greater than 1.
$c \in L$ is an Anchor if its in-degree is exactly 1.
$c \in L$ is a Drifter if its in-degree is 0.
The concepts of Tyrant, Anchor, and Drifter are explained in Figure 5.4.



(a) (b)

Figure 5.4: In (a), dashed circles are clusters in L, and solid circles are clusters in C_{opt}, and (b) shows the corresponding nearest neighbor graph.
c_1, c_4 have in-degree of $2 (> 1)$, so they are Tyrants.
c_5, c_6 have in-degree of 1, so they are Anchors.
c_2, c_3 have in-degree of 0, so they are Drifters.

Let S_{opt}: optimal centers mapped to Tyrants, A_{opt}: optimal centers mapped to Anchors. By definition, there are no optimal centers mapped to Drifters.

Now, consider $\{\bar{o}_1, \bar{o}_2, \cdots\} \subseteq C_{opt}$ who have $\bar{c}$ as their nearest neighbor, i.e. $\bar{c}$ is a Tyrant. If $\bar{c}$ is swapped with any optimal center, say $\bar{o}_1$, then the points in Cluster$(\bar{o}_2, C_{opt}), \cdots$ will have no nearby center in $L - \bar{c} + \bar{o}_1$. Therefore, a Tyrant cannot be used in swapping with any optimal center.
Similar situation will arise unless an Anchor $\bar{c}'$ is swapped with any optimal center except the one which has $\bar{c}'$ as its nearest neighbor. So, an Anchor can be swapped only with its corresponding optimal center.
Drifters can be swapped with any optimal center. Therefore, if there are enough Drifters, we can use them to swap with the centers corresponding to the Tyrants. The following claim shows that enough Drifters are indeed available, if we use each Drifter in at most two swaps.

Claim 5.2 $\#Drifters \geq \frac{|S_{opt}|}{2}$.

Proof: Since $|L| = |C_{opt}| = k$, we have

$$\begin{aligned} |S_{opt}| + |A_{opt}| &= \#\text{Drifters} + \#\text{Anchors} + \#\text{Tyrants} \\ \implies |S_{opt}| &= \#\text{Drifters} + \#\text{Tyrants} && (|A_{opt}| = \#\text{Anchors}) \\ \implies |S_{opt}| &\leq \#\text{Drifters} + \frac{|S_{opt}|}{2} && (\text{Each Tyrant has in-degree} \geq 2) \end{aligned}$$

And the claim follows. ■

Now, we can finally bound the approximation guarantee of the Local Search algorithm.

Claim 5.3 $Cost(L) \leq 5 \cdot Cost(C_{opt})$.

Proof:

Recall Equation 5.2,

$$\begin{aligned} &\text{local}(\bar{o}) \leq \text{opt}(\bar{o}) + \sum_{q \in \text{Cluster}(L,\bar{c}) \setminus \text{Cluster}(C_{opt},\bar{o})} d(q, L - \bar{c} + \bar{o}) - d(q, L) \\ \implies &\text{local}(\bar{o}) \leq \text{opt}(\bar{o}) + \sum_{q \in \text{Cluster}(L,\bar{c})} \delta_q \qquad \text{(Where } \bar{c} \text{ is a Drifter or an Anchor)} \end{aligned}$$

Now, we add the above inequality over all k swaps involving all optimal centers from C_{opt}. Since Drifters are involved in at most 2 swaps, some $q \in P$ will counted in at most 2 swaps. So we upper bound the $\text{Cost}(L)$ as follows:

$$\begin{aligned} \text{Cost}(L) &\leq \text{Cost}(C_{opt}) + 2 \sum_{q \in P} \delta_q \\ &\leq \text{Cost}(C_{opt}) + 2 \sum_{q \in P} 2 \cdot d(q, C_{opt}) \qquad \text{(From } \textbf{Claim 5.1}\text{)} \\ &\leq \text{Cost}(C_{opt}) + 4 \cdot \text{Cost}(C_{opt}) \\ \text{Cost}(L) &\leq 5 \cdot \text{Cost}(C_{opt}) \end{aligned} \tag{5.3}$$

And it follows that **Algorithm 1** is a 5-approximation for the k-median problem. ■

5.1.4 Improving the Running Time, and Other Comments

Recall that the **Algorithm 1**, as described, may go through $\binom{n}{k}$ iterations in the worst case, which is exponential in k. However, we can improve the running time by incurring a small loss in the approximation guarantee.

It can be shown that the 2-approximation algorithm for k-center problem, discussed in the previous lecture, is a $2n$-approximation for k-median problem. So in the 1st line of **Algorithm 1**, we initialize C to be the solution returned by that algorithm, as the initial guess.

Later, in the condition of while loop (Line 2), we impose a stricter condition on the new C – namely that the decrease in the cost of the algorithm is at least $\delta\lambda^*$, where λ^* is some estimate on $\text{Cost}(C_{opt})$, say $\frac{\text{Cost(Initial C)}}{2n}$.

This guarantees that the number of iterations is polynomial, but the Equation 5.2 no longer holds true. Specifically, it now becomes:

$$\text{local}(\bar{o}) - \text{opt}(\bar{o}) \leq \left(\sum_{q \in \text{Cluster}(L,\bar{c})} \delta_q \right) + \delta\lambda^* \tag{5.4}$$

Which means that Equation 5.3 will now become:

$$\text{Cost}(L) \leq 5 \cdot \text{Cost}(C_{opt}) + \delta k \text{Cost}(C_{opt}) \tag{5.5}$$

Now, we can choose δ-small enough, so that the **Algorithm 1** is a $(5+\epsilon)$ approximation.

Theorem 5.4 **Algorithm 1** *with the specified modifications is a* $(5+\epsilon)$*-approximation for* k*-median problem, and it has the running time of* $O\left(n^2 k^3 \frac{\log n}{\epsilon}\right)$. ■

Algorithm 1 is a 1-swap heuristic, that is, the set C can change by 1 center in each iteration. One can have p-swap heuristics, where C can change by p centers in each iteration. Such an algorithm will have $\left(3 + \frac{2}{p} + \epsilon\right)$-approximation guarantee, with the running time being exponentially dependent on p.

5.2 k-means Problem in $\mathbb{R}^d$

5.2.1 Description of the Problem

Given $P \subseteq \mathbb{R}^d$ and a positive integer k, find a set C of k centers that minimizes

$$\text{Cost}(C) := \sum_{q \in P} \min_{c \in C} ||q - c||^2$$

Note that the objective function is similar to that of k-median problem, except that the distances are now squared. However, squares of Euclidean distances do not form a metric. Specifically, they do not satisfy the triangle inequality. Therefore, same algorithm cannot be directly used.

5.2.2 k-means Heuristic or Lloyd's Method

The following is a well-known heuristic for k-means problem, and it is widely used in practice due to its simplicity. Specifically, Step 4, i.e. calculation of the centroid for a given set of points is a trivial calculation.

Algorithm 2 k-means Heuristic ($P \subseteq \mathbb{R}^d, k$)

1: $C \leftarrow$ arbitrary size-k subset of P.
2: **repeat** until the value of objective function stops decreasing
3: Assign each point $p \in P$ to the nearest center in C. *//Fix the centers, optimize the clusters*
4: For each cluster, let its centroid be its new center. *//Fix the clusters, optimize the centers*

Homework Problem: Describe an example where the k-means Heuristic fails to find:
1) Optimum solution, 2) The "right clustering".

When points are in $\mathbb{R}^d$, and where k is fixed (constant), there are $(1+\epsilon)$-approximation algorithms for k-median and k-means problem, which have running time of $O\left(nd \exp\left(k/\epsilon\right)\right)$. They are of 2 types:
1. "Guess centers" *(Possible term paper topic)*
2. "Coresets" (Possibly covered in the course later)

For k-median problem, if points are in $\mathbb{R}^2$ (or $\mathbb{R}^d$ for fixed d), there is a $(1+\epsilon)$-approximation algorithm that is polynomial in both n and k. However, achieving a similar result for k-means is an open problem.

Possible Term Paper Topics

Following topics w.r.t. k-means heuristic:

- How quickly does the algorithm terminate? (There is an example that takes exponential time)
- How can it be sped up in big data scenarios? (Where n and d may be very large)
- What type of approximation guarantees can one give?
- $(1+\epsilon)$-approximation using "Guess centers" approach.