Khoury College of Computer Sciences — Fall 2023
Northeastern University — 12 September 2023
CS 7870: Seminar in Theoretical Computer Science — Scribe: Rajmohan Rajaraman

Lecture Outline:

- The k-median problem
- An LP-based filtering algorithm for k-median
- A local search algorithm for k-median

This lecture considers the k-median problem. We begin with a simple algorithm based on a linear programming relaxation, which yields an approximation ratio of 6 using twice the number of medians – so technically, not a real approximation. The main focus of the lecture is a 5-approximation algorithm based on local search [AGK$^+$04]. The presentation here is partly based on [WS11, Section 9.2]

1 The k-median problem

The *k-median* problem is a widely studied problem in center-based clustering. The input to the problem is a set V of locations and distance function over pairs of locations, and the goal is to select a subset F of k locations, which we call *medians*, such that the sum, over all locations of their distance to the closest median is minimized. Here is a formal definition of k-median problem:

Problem 1. Given a set of locations(clients) V, a distance function between two locations $d : V \times V \to \mathbb{R}$ and an input parameter k, the k-median problem is to find a subset $F \subseteq V$ and $\sigma : V \to F$ such that

- $|F| = k$
- The total cost $i \in V$, $\sum_{i \in V} d(i, \sigma(i))$ is minimized.

We sometimes refer to the k centers as *facilities* and all locations as *clients*. A more general version of the problem includes a weight w_i for each client i and sets the objective as minimizing the weighted sum of the distances to the closest facility. Another generalization is where the facilities are required to be from a specified subset of V. All the results presented in this lecture extend to the general case where the clients have weights and the facilities are restricted to be from a specified subset of locations. For simplicity, we will restrict our attention to the unweighted problem with the allowance that any location can be a facility.

2 An LP-based filtering algorithm for k-median

We can write the problem down as an integer linear program, and study its fractional relaxation. For $j \in V$, let y_j be the indicator variable that we open a facility at j; that is, y_j is 1 if j is a facility and 0 otherwise.

For $i, j \in V$, let x_{ij} represent the indicator variable that client i is assigned to facility i; so, x_{ij} equals 1 if i is assigned to j and 0 otherwise. Consider the following integer linear program.

$$\begin{array}{rcl} \min & & \displaystyle\sum_{i,j \in V} d(i,j) x_{ij} \\ \displaystyle\sum_j x_{ij} & \geq & 1, \ \forall i \in V \\ x_{ij} & \leq & y_j, \ \forall i, j \in V \\ \displaystyle\sum_j y_j & \leq & k \\ x_{ij}, y_j & \in & \{0, 1\} \ \forall i, j \in V \end{array}$$

We first observe that a solution to the above integer linear program solves the k-median problem. The first inequality requires that every client is assigned to some facility. The second inequality ensures that a client to assigned to a location only if a facility is opened there. The third inequality ensures that at most k facilities are opened.

Integer linear programming is NP-complete. But if we relax the integer constraint—that is, replace $x_{ij}, y_j \in \{0, 1\}$ by $0 \leq x_{ij}, y_j \leq 1$—the resulting linear program is solvable in polynomial time. We will study linear programming in greater depth later in the course. For now, it suffices to assume that a solution to a linear program can be found in polynomial time.

We now use a simple filtering approach to get an approximation solution for k-median. We first solve the above LP to obtain a solution (x^*, y^*). For each client i, we define a ball, whose center is the client i, with the radius $r_i = 2 \sum_j d(i,j) x^*_{ij}$. Then, we sort all balls $B(i, r_i)$ in a non-decreasing order of their radii. We process the next ball in this order, pick an arbitrary location in it and open a facility there, then remove this ball and all other balls overlapping with it. This process continues until all balls are processed.

There are two things to analyze. First, what is the cost of the solution? We observe that the optimal cost equals

$$\sum_i \sum_j d(i,j) x^*_{ij} = \frac{1}{2} \sum_i r_i.$$

For any client i, if a facility is opened in $B(i, r_i)$, then the distance of i to its closest facility is at most r_i. Otherwise, the ball $B(i, r_i)$ overlapped with a ball $B(\ell, r_\ell)$ with $r_\ell \leq r_i$ such that a facility, say j, was opened within in ball $B(\ell, r_\ell)$. The distance from i to j is at most $r_i + 2r_\ell \leq 3r_i$. Therefore, the total cost of the algorithm's solution is at most

$$3 \sum_i r_i \leq 6 \text{ times optimal cost.}$$

Our second question is the following: how many facilities were opened? By an averaging argument, we know that the total sum of y_j within any ball $B(i, r_i)$ is at least $1/2$; otherwise, we get $\sum_j d(i,j) y_j \geq> r_i/2 = \sum_j d(i,j) x^*_{ij}$, a contradiction. Since the balls we select are non-overlapping, and the total sum of all y_js is at most k, we can select at most $2k$ balls, for a total of at most $2k$ facilities.

We thus obtain a $(2, 6)$-approximation solution, which means that the solution opens at most $2k$ facilities and has a 6-approximation on the clients' connection cost with respect to an optimal k-median solution. Such an algorithm is referred to a *bicriteria approximation algorithm*.

3 Local search algorithm for k-median

In this section we will introduce the local search method to solve the k-median problem with an approximation ratio of 5. The basic idea of local search is the following: we start with an arbitrary set of k facilities, which is not necessarily optimal, then we will try to improve this solution by swapping the facilities selected in our initial solution with other facilities until no improvement can be made by any swapping. In particular, we will consider one of the simplest local improvement steps: suppose our current set of facilities is S; if there exists a location $x \in S$ and a location $y \notin S$ such that the cost of $(S \setminus \{x\}) \cup \{y\}$ is less than that of S, then replace x in S by y.

We can continue this local improvement step until there is no possible improvement. Such a solution is a *local optimum*. Note that the local search algorithm completes in a finite number of steps since there are only a finite number of solutions and we are always reducing the cost of a solution in a local improvement step.

Two questions come to mind. First, how does the cost of a locally optimal solution with that of a global optimum? Can the local search algorithm be made into a polynomial time algorithm?

Theorem 1. For the metric k-median problem, any local optimum has cost at most five times that of the global optimum.

Proof. Let S be a local optimum. Thus, no swap of facilities leads to an improvement in cost. Let S^* denote a global optimum solution. We will identify k swaps among S and S^* such that the change in costs when we apply the swaps independently relate to the cost of S and S^*. Knowing that the swaps cannot decrease the cost will then yield the result that the cost of S is at most 5 times the cost of S^*.

Let σ and σ^* denote the assignment functions for S and S^*, respectively. That is, for any location i, $\sigma(i)$ (resp., $\sigma^*(i)$) is the closest facility in S (resp., S^*). We partition the facilities in S into three categories:

- Let O consist of all facilities o in S such that there is exactly one facility $o^* \in S^*$ with $\sigma(o^*) = o$. Let O^* denote the set $\{o^* \in S^* : \sigma(o^*) \in O\}$. Note that $|O^*| = |O|$.
- Let Z consist of all facilities z in S such that there is no facility $z^* \in S^*$ with $\sigma(z^*) = z$.
- Let T be $S \cup (O \cup Z)$; that is, T consists of all facilities t in S such that there are at least two facilities $t_1^*, t_2^* \in S^*$ with $\sigma(t_1^*) = \sigma(t_2^*) = t$.

We are ready to define the k swaps. The facilities in T are "close" to multiple facilities in S^*, so intuitively we should avoid swapping them out. We have the swap $(o^*, \sigma(o^*))$ for each $o^* \in O^*$. Since every facility in T has at least two facilities in $S^* - O^*$ mapped to it by σ, we have $|T| \leq |S^* - O^*|/2$. Since $|T| + |Z| = |S - O| = |S^* - O^*|$, it follows that $|Z| \geq |S^* - O^*|/2$. We include a swap (s^*, s) for each $s^* \in S^* - O^*$

by choosing an arbitrary $s \in Z$ subject to the constraint that no $s \in Z$ is in more than two swaps. Since $|Z| \geq |S^* - O^*|/2$, this can be guaranteed.

Consider any swap (s^*, s). Any client i with $\sigma^*(i) = s^*$ can be assigned to s^* yielding a change in cost of

$$d(i, \sigma^*(i)) - d(i, \sigma(i)).$$

We also need to reassign each client i such that $\sigma(i) = s$ and $\sigma^*(i) \neq s^*$; we reassign i to $\sigma(\sigma^*(i))$. For this to be feasible, we need to argue that $\sigma(\sigma^*(i)) \neq s$. Note that s is either in O or Z. If $s \in O$, and $\sigma(\sigma^*(i)) = s$, then $\sigma^*(i) = s^*$, contradicting our assumption. Otherwise, there is no $x^* \in S^*$ such that $\sigma(x^*) = s$. This ensures that the reassignment is feasible. Furthermore, the change in cost due to the reassignment of client i with $\sigma(i) = s$ and $\sigma^*(i) \neq s^*$ is at most

$$d(i, \sigma(\sigma^*(i))) - d(i, \sigma(i))$$

The total change in cost as a result of the swap (s^*, s) is at most

$$\sum_{i:\sigma^*(i)=s^*} (d(i, \sigma^*(i)) - d(i, \sigma(i))) + \sum_{i:\sigma(i)=s, \sigma^*(i)\neq s^*} d(i, \sigma(\sigma^*(i))) - d(i, \sigma(i)),$$

Since S is a local optimum, the change in cost owing to any swap is non-negative, so we have

$$\sum_{i:\sigma^*(i)=s^*} (d(i, \sigma^*(i)) - d(i, \sigma(i))) + \sum_{i:\sigma(i)=s, \sigma^*(i)\neq s^*} d(i, \sigma(\sigma^*(i))) - d(i, \sigma(i)) \geq 0$$

We can add this over all swaps (s^*, s) and obtain.

$$\sum_{\text{swap } (s^*,s)} \left(\sum_{i:\sigma^*(i)=s^*} (d(i, \sigma^*(i)) - d(i, \sigma(i))) + \sum_{i:\sigma(i)=s, \sigma^*(i)\neq s^*} (d(i, \sigma(\sigma^*(i))) - d(i, \sigma(i))) \right) \geq 0$$

Since each facility in S^* appears exactly once, we obtain

$$\sum_{\text{swap } (s^*,s)} \left(\sum_{i:\sigma^*(i)=s^*} (d(i, \sigma^*(i)) - d(i, \sigma(i))) \right) = C^* - C.$$

We now analyze $d(i, \sigma(\sigma^*(i))) - d(i, \sigma(i))$ whenever $\sigma(i) = s, \sigma^*(i) \neq o$. We derive

$$\begin{aligned} d(i, \sigma(\sigma^*(i))) - d(i, \sigma(i)) &\leq d(i, \sigma^*(i)) + d(\sigma^*(i), \sigma(\sigma^*(i)) - d(i, \sigma(i)) \\ &\leq d(i, \sigma^*(i)) + d(\sigma^*(i), \sigma(i)) - d(i, \sigma(i)) \\ &\leq d(i, \sigma^*(i)) + d(i, \sigma(i)) + d(i, \sigma^*(i)) - d(i, \sigma(i)) \\ &= 2d(i, \sigma^*(i)). \end{aligned}$$

Since each s appears at most twice in the swaps, each client i also appears at most twice in the following summation yielding

$$\sum_{i:\sigma(i)=s, \sigma^*(i)\neq s^*} (d(i, \sigma(\sigma^*(i))) - d(i, \sigma(i))) \leq \sum_i 4d(i, \sigma^*(i)) = 4C^*.$$

Putting all this together we obtain $5C^* - C \geq 0$ yielding the desired claim. $\square$

While the local search algorithm guarantees a 5-approximation after convergence, it is not clear how long the algorithm will take to converge. In order to attain a polynomial running time, we should stop the iteration process at some point when the improvement from any swap is small enough. For a given $\varepsilon > 0$, we will use the following stopping criterion:

- The local search will stop if no swap improves cost by more than $\frac{\varepsilon \cdot \text{cost}(S)}{k}$.

Under this criterion, the improvement of any k swaps is bounded at $\varepsilon\text{cost}(S)$. So using the same analysis as above, we obtain

$$\text{cost}(S) - 5 \cdot \text{cost}(S^*) \leq \varepsilon\text{cost}(S) \tag{1}$$

Thus, the local search algorithm with the above stopping criterion has an approximation ratio of $5+\varepsilon$, where the $\varepsilon > 0$ can be made as close to zero as needed.

We now argue that with the stopping criterion, our local search ends in polynomial time (which may depend on ε). Consider an iteration of the local search. If S is the solution before any iteration and S' after the iteration, we obtain:

$$\text{cost}(S') \leq \left(1 - \frac{\varepsilon}{k}\right) \text{cost}(S) \tag{2}$$

Here $\frac{\varepsilon \cdot \text{cost}(S)}{k}$ is the drop in cost in the iteration of local search. Thus, if our initial local search solution is S_{init}, then after t iterations of local search, our solution S will have cost at most

$$\text{cost}(S_{init}) \left(1 - \frac{\varepsilon}{k}\right)^t.$$

It is easy to obtain an $O(n)$ approximation to the k-median problem by greedily adding facilities that minimize the maximum distance of a client to its nearest facility. We can set this solution to be the initial local search solution, in which case the number of iterations before local search (with the stopping criterion) terminates is

$$t \approx \frac{k}{\varepsilon} \log\left(\frac{\text{cost}(S_{init})}{\text{cost}(OPT)}\right) \approx O\left(\frac{k}{\varepsilon} \log n\right),$$

where we use the approximation

$$\log\left(\frac{1}{1 - \frac{\varepsilon}{k}}\right) \approx \frac{k}{\varepsilon}.$$

Alternative ways of proving the polynomial time of the above local search procedure are given in [AGK$^+$04].

References

[AGK$^+$04] 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.

[WS11] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms.* Cambridge University Press, 2011.