College of Computer & Information Science — Fall 2007
Northeastern University — 2 October 2007
CS G399: Algorithmic Power Tools I — Scribe: Jian Wen

Lecture Outline:

- The k-median problem
- Filtering algorithm for k-median
- Local search algorithm for k-median

This lecture describes the k-median problem. A filtering algorithm that yields an approximation ratio of 6 using twice the number of medians will be described. The main focus of the lecture is a 5-approximation algorithm based on local search [AGK$^+$04].

1 The k-median

The k-median problem is a variant of the uncapacitated facility location problem that we covered last week, with the difference that instead of a cost for each facility, we have a bound k on the number of facilities. Here is the definition of k-median problem:

Problem 1. Given a set of locations(clients) V, a set $\widetilde{F}$ of facilities, a cost function between two locations $C : (V \bigcup \widetilde{F}) * (V \bigcup \widetilde{F}) \to Q^+$ and an input parameter k, $0 < k \leq |\widetilde{F}|$, the k-median problem is to find a subset $F \subseteq \widetilde{F}$ and $\sigma : V \to F$ such that

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

We can write the problem down as an integer linear program, and study its fractional relaxation.

$$\begin{aligned} \min \sum_{i,j} C_{ij} x_{ij} & \\ s.t. \quad \sum_j x_{ij} \geq 1,\ \forall i & \\ x_{ij} \leq y_{ij},\ \forall i, j & \\ \sum_j y_i \leq k & \end{aligned} \tag{1}$$

Compared to the facility location, the major challenge of k-median is to satisfy the constraints. This means that we should keep a lower cost while simultaneously satisfying the condition that we can have at most k facilities selected in our solution.

2 Filtering algorithm for k-median

We can use the filtering algorithm to get an approximation solution for k-median, which is similar to the one we have used in facility location problem. We first solve the above LP. For each client i, we define a ball, whose center is the client i, with the radius

$$r_i = 2\sum_j c_{ij}x^*_{ij} \tag{2}$$

Here x^* is the optimal solution of the corresponding LP. Then we will sort all the balls in a non-increasing order of their radii. Each time for the non-overlapping ball with the maximal radius, pick a facility location in it and open it, then remove this ball and all the other balls overlapping with it. This process will be continued recursively until no all balls are processed.

Just like the analysis in the facility location problem, this algorithm will yield a solution that is 6-approximate with respect to clients' connection cost. The number of medians, however, could be as high as $2k$. This is because the only guarantee we have is that each ball has at least 1/2 a median in the LP solution. 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 the approximation ratio of 5.

The basic idea of local search is: 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. Then we will get a solution S, which is locally optimal. Our goal then is to understand how good a local optimization is.

We will define the "improvement" of a swap like this: If for each facility selected in both local optimal solution and the global optimal solution, we associate a neighborhood, defined as following:

$$N_S(s) = \{i \in V,\ \sigma(i) = s\}$$
$$N_{OPT}(o) = \{i \in V,\ \sigma^*(i) = o\}$$

Here $\sigma^*(i)$ is the optimal location of the facility serving the client i. So when we are adding $o \in OPT$ to our local solution S, the improvement means:

$$cost(S) - cost(S + \{o\}) \geq \sum_{i \in N_{OPT}(o)} [C_{i\sigma(i)} - C_{i\sigma^*(i)}] \tag{3}$$

If we consider adding each facility of OPT individually to S, we have

$$\sum_{o \in OPT} [cost(S) - cost(S + \{o\})] \geq cost(S) - cost(OPT) \tag{4}$$

Then we need to know how to bound the impact of dropping a facility of S. Toward this end, we define the notion of **capture**:

Definition 1. For a local solution S and optimal solution OPT, we say $s \in S$ **captures** $o \in OPT$ if in S at least half of o's clients are served by s, i.e.,

$$|N_S(s) \bigcap N_O(o)| > \frac{|N_O(o)|}{2}$$

We will then consider two cases. Note that a facility $o \in OPT$ is captured by at most one facility $s \in S$.

- A facility $s \in S$ captures exactly one $o \in OPT$ In this situation, we just swap this pair (s, o).
- Some of the facilities in S capture more than one facilities in OPT, and some of the choices in OPT may not have been captured by any facility in S. Here none of the facilities $s \in S$ which captures more than one optimal choices should be swapped, in such a way that no facility in S is swapped with more than 2 facilities in OPT, instead we will swap the facilities in the optimal solution with the facilities in S which capture no facilities in OPT.

We will first show that the local algorithm described above is a 3-approximation algorithm, if all facilities in OPT were of CASE 1.

The challenge in our analysis in the swapping process is that after swapping, some of the clients locations which was served by some facilities in our initial solution should be reassigned to other facilities since the old ones have been replaced by ones from the optimal solution. So here we should find a way to estimate the change of the cost in this reassignment process. So for $i \in N_S(s) - N_{OPT}(o)$ in a swapping on (s, o), we use a mapping function $\pi : V \to V$ to map $i \notin N_{OPT}(o)$ to $\pi(i) \in N_{OPT}(o)$, such that:

1. $\sigma^*(\pi(i)) = \sigma^*(i)$
2. $\sigma(\pi(i)) = \sigma(i) \qquad only\ if\ \sigma(i)\ captures\ \sigma^*(i)$

One can define such a mapping function as follows. Consider any $o \in OPT$. Let $m = |N_{OPT}(o)|$. Number the clients in $N_{OPT}(o)$ from 0 to $m - 1$ in order according to the index of the facility they are assigned to in S. Now for client $i \in N_{OPT}(o)$, set $\pi(i) = i + \lfloor m/2 \rfloor$. It is easy to verify that π satisfies the desired properties.

For a given swapping between (s, o), since S is locally optimal, we have

$$cost(S) - cost(S + \{o\} - \{s\}) \leq 0 \tag{5}$$

Since we need to add o while dropping s, for the clients who are served only by s but not o, the cost will increase since they should be reassigned. So

$$\begin{aligned} cost(S) - cost(S + \{o\} - \{s\}) = & \sum_{i \in N_{OPT}(o)} (C_{i\sigma(i)} - C_{i\sigma^*(i)}) \\ & + \sum_{i \in N_S(s),\ i \notin N_{OPT}(o)} (C_{i\sigma(i)} - C_{i\sigma^*(i)} - C_{\pi(i)\sigma(\pi(i))} - C_{\pi(i)\sigma^*(\pi(i))}) \end{aligned} \tag{6}$$

Then for the clients $i \in N_S(s) - N_{OPT}(o)$, after the swapping of (s, o), i should be reassigned. Here we will map i to $j = \pi(i) \in N_{OPT}(o)$. According to triangle inequality, the cost of reassigning can be estimated

$$\begin{aligned} C_{i\sigma^*(i)} + C_{j\sigma^*(j)} + C_{j\sigma(j)} - C_{i\sigma(i)} & \leq C_{i\sigma^*(i)} + C_{j\sigma^*(j)} + C_{i\sigma^*(i)} + C_{j\sigma^*(j)} + C_{i\sigma(i)} - C_{i\sigma(i)} \\ & = 2(C_{i\sigma^*(i)} + C_{j\sigma^*(j)}) \end{aligned} \tag{7}$$

From Equation 6 and Equation 7, we can get

$$\sum_{i \in N_O(o)} (C_{i\sigma(i)} - C_{i\sigma^*(i)}) \leq \sum_{i \in N_S(s),\ i \notin N_O(o)} 2(C_{i\sigma^*(i)} + C_{j\sigma^*(j)}) \tag{8}$$

For the situation that all facilities in OPT were of CASE 1, according to the swap policy mentioned above, all the facility $s \in S$ and also $o \in O$ will be swapped exactly once, since for each $o \in O$, there is exactly one $s \in S$ which captures it.

And also, since in the process of swapping for each pair (s, o) the clients $i \in N_{OPT}(o)$ will be at least half of the clients in $s \in S$, there will be at most half of the clients $i \in N_S(s) - N_{OPT}(o)$ for all the pairs swapped. After finishing all the swaps, there will totally be at most half of the clients i which should be considered separately. And also, $j = \pi(i)$ is a one-to-one mapping, so for the right hand side of Equation 8, we have

$$\sum_{i \in N_S(s),\ i \notin N_O(o)} 2(C_{i\sigma^*(i)} + C_{j\sigma^*(j)}) \leq 2 * cost(OPT) \tag{9}$$

From Equation 5, 8 and 9, we will get

$$cost(S) \leq 3 * cost(OPT)$$

For CASE 2, the policy is feasible since if the number of facilities in OPT in CASE 2 are l, there are at least $l/2$ facilities in S which will not capture any of the facilities in S, because in this case the facilities in S which capture some optimal choices will capture at least 2 facilities in OPT. We then have a set of k swaps such that each facility in OPT is swapped in once and each facility in S

is swapped out once. Now we will show that the algorithm will produce a 5-approximation solution in this case.

The policy for CASE 2 will guarantee that all the $s \in S$ will be swapped at most twice, since for the facilities in OPT at least of two of which are captured by one facility in S. So for the right hand side of Equation 8, the total number of clients $i \in N_S(s) - N_{OPT}(o)$ for all the pairs will be at most the number of all the clients. So we have

$$\sum_{i \in N_S(s),\ i \notin N_O(o)} 2(C_{i\sigma^*(i)} + C_{j\sigma^*(j)}) \leq 4 * cost(OPT) \tag{10}$$

Then we combine Equation 5 with 10, we will get

$$\begin{gathered} 4 * cost(OPT) \geq cost(S) - cost(OPT) \\ cost(S) \leq 5 * cost(OPT) \end{gathered} \tag{11}$$

The in order to keep our algorithm running into the polynomial capability, we should stop the iteration process at some point when the improvement from the swap is too trivial for the cost of running time. 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 $\frac{\varepsilon \cdot cost(S)}{k}$.

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

$$cost(S) - 5 * cost(OPT) \leq \varepsilon cost(S) \tag{12}$$

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 the iteration and S' after the iteration, we obtain:

$$cost(S') \leq \left(1 - \frac{\varepsilon}{k}\right) cost(S) \tag{13}$$

Here $\frac{\varepsilon \cdot cost(S)}{k}$ 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

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

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

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

where we use the approximation

$$\log \frac{1}{1 - \frac{\varepsilon}{k}} \approx \frac{k}{\varepsilon}.$$

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.