CS-E400204: Approximation Algorithms **Spring (Fifth Period) 2022**

Lecture 9 (May 23): Local Search for the k-Median Problem

Lecturer: Kamyar Khodamoradi *Scribe: Zachary Friggstad*

9.1 Introduction

[1]

The k-Median problem is a variant of the k-Centre problem we have seen in Lecture 08. Here, in order to be consistent with the notion of a metric (and at the cost of being pedantic), we define the problem as follows:

- **Input**:
  - A metric (P, d), where $P = \{p_1, p_2, \ldots, p_t\}$ is the set of points and d is the distance function of the metric
  - Two disjoint sets, C, the clients set and F, the facilities set
  - A placement function $p : C \uplus F \to P$
  - Parameter k, $1 \leq k \leq |F|$
- **Output**:
  - A $S \subseteq F$, with $|S| = k$, that minimizes

$$f(S) \triangleq \sum_{j \in C} \min_{i \in S} \{d(p(i), p(j))\} = \sum_{j \in C} d(p(j(, p(S)).$$

Note that the placement function p may assign clients or facilities to the same location of the metric. Now, this is consistent with the identity property of the metric that $d(u, v) = 0 \Leftrightarrow u = v$.

We then form a complete graph $G = (C \uplus F, E)$ with the edge cost function $c : E \to \mathbb{Q}_{\geq 0}$ where $c(x, y) = d(p(x), p(y)$ for any pair $x, y \in C \uplus F$. In the rest of this lecture, we work with the cost function c for convenience, having its connection to the metric distance d in mind. The distance function c on graph G is what known as a *pseudo-metric* in the literature.

Note that rather that tying to minimize the maximum distance of a client to a facility as in k-Centre (or similarly, k-Supplier), we minimize the sum of distances between clients and their nearest facility.

A simple greedy (*local search*) algorithm for the k-Median problem is presented below. For $i \in S, i' \in F - S$ we let $S - i + i'$ denote $(S - \{i\}) \cup \{i'\}$.

We will show the above algorithm is a 5-approximation for the k-Median problem. However, it is not guaranteed to run in polynomial time. To overcome this issue, we consider a slight variant of Algorithm 1 where ϵ is a parameter we may specify.

Claim 1 *Algorithm 2 runs in polynomial time in the input size and $\frac{1}{\epsilon}$.*

[1]These are the original (lightly modified!) lecture notes of the Approximation Algorithms course taught by Zac at University of Alberta, Canada.

Algorithm 1 Local Search for k-MEDIAN

```
$S \leftarrow$ any subset of $F$ of size $k$
while $\exists\, i \in S, i' \in F - S$ s.t $f(S) > f(S - i + i')$ do
    $S \leftarrow S - i + i'$
end while
return $S$
```

Algorithm 2 Polynomial-Time Local Search for k-MEDIAN

```
If there is a solution with cost 0, then return it.
$\delta \leftarrow \epsilon/(10 \cdot k)$
$S \leftarrow$ any subset of $F$ of size $k$
while $\exists\, i \in S, i' \in F - S$ s.t $(1-\delta) \cdot f(S) > f(S - i + i')$ do
    $S \leftarrow S - i + i'$
end while
return $S$
```

Proof. First, we claim that we can easily detect if the instance has a solution with cost 0. If some client j has $c(i,j) > 0$ for all facilities $i \in F$, then there is no 0-cost solution. Otherwise, form a bipartite graph $H = (C \uplus F, E')$ where $e \in E' \Leftrightarrow c(e) = 0$. Note that due to co-location, every connected component of H is a bi-clique. Also note that in any solution, w.l.o.g, we can assume that we do not take two or more co-located facilities i, i' (the cost of any solution would not change if we remove the co-located facilities). Hence, the instance has a 0-cost solution if and only if the number of connected components of H is at most k.

Next, let

$$\Delta = \frac{n \cdot \max_{i,j}\ c(i,j)}{\min_{i,j:c(i,j)>0} c(i,j)}.$$

Note that $\log \Delta$ is polynomial in the input size (i.e. number of bits used to describe the input).

Let S_I and S_F denote the initial and final set S used by the algorithm respectively. If t equals the number of iterations performed by the algorithm then

$$f(S_F) \le (1-\delta)^t f(S_I).$$

The claim is that $t \le \frac{10 \cdot k}{\epsilon} \ln \Delta = \delta^{-1} \cdot \ln \Delta$, which is polynomial in the input size and $\frac{1}{\epsilon}$.

Otherwise, we would have

$$(1-\delta)^t < e^{-\ln \Delta} = \frac{1}{\Delta},$$

where we have used the fact that $(1-\delta)^{1/\delta} \le \frac{1}{e}$. In other words, $\Delta < f(S_I)/f(S_F)$.

However, we know $f(S_I) \le n \cdot \max_{i,j} c(i,j)$ because each of the n clients travels distance at most the maximum distance in the metric. and we also know $f(S_F) \ge \min_{i,j:c(i,j)>0} c(i,j)$ because there is no 0-cost solution so some client has to travel distance at least the minimum nonzero distance in the metric. Thus, $f(S_I)/f(S_F) \le \Delta$ which contradicts what we just saw.

Therefore, this is a polynomial-time algorithm. ■

9.2 Analysis of the Local Search for k-Median

For a set $S \subseteq F$, we let $c(j,S) = \min_{i \in S} c(i,j)$. The *cost* of S was defined as $f(S) := \sum_{j \in C} c(j,S)$.

We first show that if the local search procedure is applied with $\delta = 0$ (Algorithm 1), then the resulting solution is a 5-approximation. However, we do not have any good bounds on the running time of this algorithm so we chose $\delta := \frac{\epsilon}{10k}$ to be a small positive number (small enough to ensure the final solution is a good approximation but large enough to guarantee polynomial running time). The analysis of this case will appear later.

Theorem 1 *Let $S \subseteq F, |S| = k$ be such that $f(S-i+i') \geq f(S)$ for any $i \in S, i' \in F-S$. Then $f(S) \leq 5 \cdot OPT$.*

The idea behind this proof is to describe a series of "test swaps". First, we give some notation and then some intuition.

Let S^* be any fixed optimum solution. For any client j, let $c^*_j := c(j, S^*)$ and $c_j := c(j, S)$ be the distance that j travels in the global and local optimum solution, respectively. That is, $\sum_{j \in C} c^*_j = f(S^*) = OPT$ and $\sum_{j \in C} c_j = f(S)$.

Let $\phi : C \to S$ map each $j \in C$ to their nearest facility in S and $\phi^* : C \to S^*$ map each $j \in C$ to their nearest facility in S^*. That is, $c(\phi(j), j) = c_j$ and $c(\phi^*(j), j) = c^*_j$. Finally, let $\sigma : S^* \to S$ map each $i \in S^*$ to its nearest facility in S, so $c(i, \sigma(i)) = c(i, S)$.

9.2.1 Intuition

Consider some client $j \in C$ and suppose $\phi^*(j) \notin S$. Consider swapping $\phi(j)$ out and $\phi^*(j)$ in. We have $c(j, S - \phi(j) + \phi(j^*)) \leq c^*_j$ because $\phi^*(j)$ is swapped in. So, the overall change in j's assignment cost when going from S to $S - \phi(j) + \phi^*(j)$ is at most $c^*_j - c_j$. We know that $0 \leq f(S - \phi(j) + \phi^*(j)) - f(S)$ because S is a locally optimum solution, so this suggests that in some way, c_j is bounded by c^*_j.

Of course, the situation is more complicated than this, but this is the basic approach: Use the optimum solution to describe some test swaps. These will then give us inequalities we can use to bound the locally optimum solution against the globally optimum solution.

9.3 A Simple Case

For simplicity, first suppose that $\sigma(i) \neq \sigma(i')$ for all $i, i' \in S^*$.

Consider any $i \in S^*$. We provide an upper bound on $f(S - \sigma(i) + i)$ by explicitly describing how to reassign the clients.

1. any $j \in C$ with $\phi^*(j) = i$ is assigned to i
2. any $j \in C$ with $\phi(j) = \sigma(i)$ but $\phi^*(j) \neq i$ is assigned to $\sigma(\phi^*(j))$
3. every other client is assigned to $\phi(j)$

First, we claim that this is a valid assignment (i.e. every client is assigned to a facility in $S - \sigma(i) + i$). We opened $\phi^*(j)$, so the first group of clients are properly assigned. For the second, if $\phi^*(j) \neq i$ then our assumption in this section means $\sigma(\phi^*(j)) \neq \phi(j)$. Therefore, the second group of clients are assigned to open facilities. Finally, every client in the third group has $\phi(j) \neq \sigma(i)$ and we only closed $\sigma(i)$.

Now that we know this is a valid reassignment, local optimality of S implies

$$\begin{aligned} 0 &\leq f(S-\sigma(i)+i)-f(S) \\ &\leq \sum_{j:\phi^*(j)=i}(c_j^*-c_j)+\sum_{\substack{j:\phi(j)=\sigma(i)\\ \text{and } \phi^*(j)\neq i}} c(\sigma(\phi^*(j)),j)-c_j \end{aligned} \tag{9.1}$$

The first sum corresponds to the first group of clients and the second sum to the second group of clients. The third group of clients contribute the same to both $f(S-\sigma(i)+i)$ and $f(S)$ so their contributions cancel.

We can bound the last term as follows.

Lemma 1 *For any $j \in C$, $c(\sigma(\phi^*(j)),j) \leq 2c_j^* + c_j$.*

Proof. We have

$$\begin{aligned} c(\sigma(\phi^*(j)),j) &\leq c(\phi^*(j),j)+c(\phi^*(j),\sigma(\phi^*(j))) \\ &= c_j^*+c(\phi^*(j),\sigma(\phi^*(j))) \\ &\leq c_j^*+c(\phi^*(j),\phi(j)) \\ &\leq c_j^*+c(\phi^*(j),j)+c(j,\phi(j)) \\ &= c_j^*+c_j^*+c_j. \end{aligned}$$

The first and third inequalities are by the triangle inequality. The second is because $\sigma(\phi^*(j))$ is the closest facility in S to $\phi^*(j)$, so it is no further than $\phi(j)$. ■

We can then bound expression (9.1) by

$$\begin{aligned} &\sum_{j:\phi^*(j)=i}(c_j^*-c_j)+\sum_{\substack{j:\phi(j)=\sigma(i)\\ \text{and } \phi^*(j)\neq i}} 2c_j^* \\ \leq &\sum_{j:\phi^*(j)=i}(c_j^*-c_j)+\sum_{j:\phi(j)=\sigma(i)} 2c_j^* \end{aligned}$$

Summing these over all $i \in S^*$, we see

$$\begin{aligned} 0 &\leq \sum_{i\in S^*} f(S-\sigma(i)+i)-f(S) \\ &\leq \sum_{i\in S^*}\left[\sum_{j:\phi^*(j)=i}(c_j^*-c_j)+\sum_{j:\phi(j)=\sigma(i)} 2c_j^*\right]. \end{aligned}$$

Now, for each $j \in C$ we see that $-c_j$ is counted only for $i=\phi^*(j)$. We also see that c_j^* is counted once in the first inner sum (when $i=\phi^*(j)$) and is counted once (with weight 2) in the second inner sum only when $\in S^*$ is such that $\sigma(i)=\phi_j$. Therefore, the last expression is bounded by

$$\sum_{j\in C} 3c_j^* - \sum_{j\in C} c_j = 3\cdot OPT - f(S).$$

Therefore, $f(S) \leq 3\cdot OPT$.

9.3.1 The General Case

Now we consider the general case where it may be that $\sigma(i) = \sigma(i')$ for distinct $i, i' \in S^*$. The only place in the analysis for the simpler case where this could cause any problems is when describing a valid reassignment, namely we assigned some j with $\phi(j) = \sigma(i)$ to $\sigma(\phi^*(j))$. This was a valid reassignment (i.e. $\sigma(\phi^*(j))$ was still open after the swap) by the assumption that no other $i' \in S^*, i' \neq i$ has $\sigma(i') = \sigma(i)$.

However, without this assumption it might be that $\sigma(\phi^*(j)) = \sigma(i)$ even if $\phi^*(j) \neq i$, which causes a problem when we try to reassign the clients to upper bound the cost of $f(S - \sigma(i) + i)$. We now describe how to handle this below.

Partition S and S^* as follows. Let $S = O \cup M \cup N$ where $|\sigma^{-1}(i)| = 1$ for $i \in O$, $|\sigma^{-1}(i)| \geq 2$ for $i \in M$, and $|\sigma^{-1}(i)| = 0$ for $i \in N$. Also let $S^* = O^* \cup M^*$ where $\sigma(i) \in O$ for $i \in O^*$ and $\sigma(i) \in M$ for $i \in M^*$. See Figure 9.1 for an illustration.

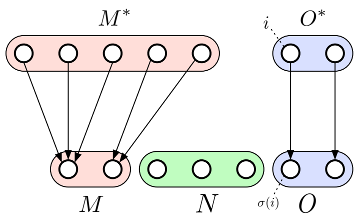

Figure 9.1: Illustration of the mapping $\sigma : S^* \to S$ and the partitioning of S and S^*.

For $i \in O^*$, we perform the swap $S - \sigma(i) + i$ as before. This generates the following bound, the proof is the same as before.

Lemma 2 *For any $i \in O^*$ we have*

$$0 \leq f(S - \sigma(i) + i) - f(S) \leq \sum_{j:\phi^*(j)=i} (c_j^* - c_j) + \sum_{j:\phi(j)=\sigma(i)} 2c_j^*.$$

For $i \in M^*$, we will choose some $i' \in N$ and perform the swap $S - i' + i$. This i' will be chosen carefully, but first let's see how to bound the cost change of this swap. It is similar to Lemma 2

Lemma 3 *For any $i \in M^*$ and any $i' \in N$ we have*

$$0 \leq f(S - i' + i) - f(S) \leq \sum_{j:\phi^*(j)=i} (c_j^* - c_j) + \sum_{j:\phi(j)=i'} 2c_j^*.$$

Proof. We begin by describing a valid assignment of clients in $S - i' + i$.

1. any $j \in C$ with $\phi^*(j) = i$ is assigned to i
2. any $j \in C$ with $\phi(j) = i'$ but $\phi^*(j) \neq i$ is assigned to $\sigma(\phi^*(j))$
3. every other client is assigned to $\phi(j)$

As before, this is a valid reassignment. The only change is the assignment for the second group of clients, but since $i' \in N$ then $\sigma(\phi^*(j)) \neq i'$ so $\sigma(\phi^*(j)) \in S - i' + i$.

The cost change is bounded as follows

$$\begin{aligned}
0 &\leq f(S - i' + i) - f(S) \\
&\leq \sum_{j:\phi^*(j)=i} (c_j^* - c_j) + \sum_{\substack{j:\phi(j)=i' \\ \text{and } \phi^*(j)\neq i}} c(\sigma(\phi^*(j)), j) - c_j \\
&\leq \sum_{j:\phi^*(j)=i} (c_j^* - c_j) + \sum_{\substack{j:\phi(j)=i' \\ \text{and } \phi^*(j)\neq i}} 2c_j^* \\
&\leq \sum_{j:\phi^*(j)=i} (c_j^* - c_j) + \sum_{j:\phi(j)=i'} 2c_j^*.
\end{aligned}$$

The second inequality follows from Lemma 1. ∎

We will use Lemma 3 in our final bound, but we have to ensure no individual c_j^* term appears too often. We need to swap in each $i \in M^*$ once to get the $-c_j$ term, but we do not want to swap out any $i' \in N$ too many times. The following ensures this is possible.

Lemma 4 *We have $|M^*| \leq 2|N|$.*

Proof. For every $i \in M$ there are at least two $i_1, i_2 \in M^*$ with $\sigma(i_1) = \sigma(i_2) = i$. Thus, $|M^*|/2 \geq |M|$.

Next, recall $|M| + |O| + |N| = k = |M^*| + |O^*|$. Clearly $|O| = |O^*|$, so $|M| + |N| = |M^*|$. Considering this and $|M^*|/2 \geq |M|$, we see $|M^*|/2 \leq |N|$. ∎

Let $\mathcal{P} \subseteq M^* \times N$ be any collection of pairs such that each $i \in M^*$ appears in exactly one pair of $\mathcal{P}$ and each $i' \in N$ appears in at most two pairs of $\mathcal{P}$. The previous lemma ensures this is possible. For example, process the $i \in M^*$ in any order. For each i, add (i, i') to $\mathcal{P}$ where i' is any facility in N does not yet appear in two pairs of $\mathcal{P}$.

Combining Lemma 2 and 3 and our careful construction of the set of test swaps $\mathcal{P}$ shows:

$$\begin{aligned} 0 &\le \sum_{i \in O^*} [f(S - \sigma(i) + i) - f(S)] + \sum_{(i,i') \in \mathcal{P}} [f(S - i' + i) - f(S)] \\ &\le \sum_{i \in O^*} \left[\sum_{j:\phi^*(j)=i} (c_j^* - c_j) + \sum_{j:\phi(j)=\sigma(i)} 2c_j^* \right] + \sum_{(i,i') \in \mathcal{P}} \left[\sum_{j:\phi^*(j)=i} (c_j^* - c_j) + \sum_{j:\phi(j)=i'} 2c_j^* \right] \qquad (9.2) \end{aligned}$$

We conclude by bounding the extent to which each c_j^* and c_j term is considered in these sums. For a client $j \in C$, we see $-c_j$ exactly when $\phi^*(j)$ is swapped in and this happens once between both outer sums: if $\phi^*(j) \in O^*$ then it is swapped in exactly once in the first outer sum and if $\phi^*(j) \in M^*$ it is swapped in exactly once in the second outer sum by how we constructed $\mathcal{P}$.

Now consider the extent to which c_j^* appears in these sums for a client j.

- If $\phi(j) \in O$: Then c_j^* appears once in the first inner sum of the first outer sum and once (with weight 2) in the second inner sum of the first outer sum. It does not appear in the second outer sum (since $\phi(j) \notin N$), so c_j^* is counted with weight at most 3 in this sum.
- $\phi(j) \in N$: Then c_j^* appears once in the first inner sum of the second outer sum and at most twice in the second inner sum of the second outer sum. This is because $\phi(j) \in N$ and at most two (i, i') pairs in $\mathcal{P}$ have $i' = \phi(j)$ by how we constructed $\mathcal{P}$. Thus, its total contribution to the bound is at most $5c_j^*$.
- $\phi(j) \in M$: Then c_j^* does not appear anywhere among these sums.

In the worst of these cases, we have that each $c_j^*, j \in C$ is considered in expression (9.2) to an extent of at most 5.

To summarize, we can bound expression (9.2) by $\sum_{j \in C} 5c_j^* - c_j = 5 \cdot OPT - f(S)$. Overall, we have shown $0 \le 5 \cdot OPT - f(S)$ which proves Theorem 1.

9.4 The Polynomial-Time Approximation

Recall that the polynomial-time local search algorithm only updates S if $f(S - i' + i) < (1 - \delta) \cdot f(S)$ for some $i' \in S, i \in F - S$ where $\delta := \frac{\epsilon}{10k}$ for some prespecified ϵ. The algorithm would run in time that is polynomial in the input size at $1/\epsilon$. We now extend the analysis above to this case.

Theorem 2 *Let $\delta = \frac{\epsilon}{10k}$ for some $0 \le \epsilon \le 1$. For any $S \subseteq F, |S| = k$ such that $f(S - i' + i) \ge (1 - \delta) \cdot f(S)$ for any $i' \in S, i \in F - S$ we have $f(S) \le (5 + \epsilon) \cdot OPT$.*

Proof. Consider any test swap $S \to S - i' + i$. We then have

$$0 \le f(S - i' + i) - (1 - \delta) \cdot f(S) = f(S - i' + i) - f(S) + \delta \cdot f(S).$$

Let $\mathcal{P}'$ be the pairs of facilities swapped in the analysis in Section 9.3.1 (i.e. the $(i, \sigma(i))$ swaps for $i \in O^*$ and all swaps in $\mathcal{P}$). Note that $|\mathcal{P}'| = k$.

The previous analysis shows

$$0 \leq \sum_{(i,i') \in \mathcal{P}'} [f(S - i' + i) - f(S) + \delta \cdot f(S)] \leq 5 \cdot OPT - f(S) + k \cdot \delta \cdot f(S). \tag{9.3}$$

Rearranging and recalling $\delta = \frac{\epsilon}{10k}$, we see $(1 - \epsilon/10) f(S) \leq 5 \cdot OPT$. Because $\epsilon \leq 1$, we have

$$f(S) \leq \frac{5}{1 - \epsilon/10} \cdot OPT \leq (5 + \epsilon) \cdot OPT.$$

■

Notice the role of the number of swaps in the approximation ratio. In the bound in (9.3), the number of times the extra $\delta \cdot f(S)$ term was added is equal to the number of test swaps, namely k. Thus, δ was chosen so that (# of test swaps) $\cdot\, \delta = k \cdot \delta$ is appropriately small.

This is the general idea behind many local search algorithms: analyze the case where the swaps are made if *any* improvement can be made. As long as the number of test swaps in the analysis is polynomially small, this leads to a polynomial-time algorithm by choosing δ to be an appropriately small value that is roughly ϵ divided by the number of test swaps.

9.5 Discussion

The analysis of the local search algorithms for k-MEDIAN was initially performed by Arya et al. [AGK+04]. They showed a more general local-search algorithm that yields a $(3+\varepsilon)$-approximation ($3+2/p$ for swap size p). This was the best approximation for k-median for roughly a decade until Li and Svensson obtained an LP-based approximation (with an additional trick) and obtained a $1 + \sqrt{3} + \epsilon \approx 2.732$ approximation [LS13]. Currently, the best approximation is by Byrka et al [BPR+14] who obtain roughly a 2.675-approximation[2]. Also, very recently, Cohen-Addad et al. showed a local search with respect to a potential function that can achieve a factor of $2.836 + \varepsilon$. On the negative side, it is hard to approximation k-median within a factor better than $1 + 2/e \approx 1.736$ [JMS02].

The analysis in these lecture notes is due to Gupta and Tangwonsan [GT08].

References

[AGK+04] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit, Local search heuristics for k-median and facility location problems, SIAM Journal on Computing, 33(3):544–562, 2004.

[BPR+14] J. Byrka, T. Pensyl, B. Rybicki, A. Srinivasan, and K. Trinh, An improved approximation for k-median, and positive correlation in budgeted optimization, To appear in Proceedings of ACM-SIAM Symposium on Discrete Algorithms, 2015.

[GT08] A. Gupta and K. Tangwonsan, Simpler analysis of local search algorithms for facility location, CoRR:abs/0809.2554, `http://arxiv.org/abs/0809.2554`, 2008.

[JMS02] K. Jain, M. Mahdian, and A. Saberi, A new greedy approach for facility location problems, In Proceedings of ACM Symposium on Theory of Computing, 2002.

[LS13] S. Li and O. Svensson, Approximating k-median by pseudo-approximation, In Proceedings of ACM Symposium on Theory of Computing, 2013.

[2]For a while, it was believed the factor for this algorithm was 2.611, but then a tiny bug in the proof was discovered, raising the factor slightly to 2.675.