Group 2: Assignment 1.1

Niv Adam, David Kaufmann, Casper Kristiansson, Nicole Wijkman

October 19, 2023

1 B-Level Problem (1.2)

Let $B = \{b_1, .., b_n\}$ and $C = \{c_1, ..., c_m\}$ be sets containing buildings and customers as defined in the task description. We define the following algorithm.

Algorithm 1: B-level problem

Function `kSuppliers`(k, B, C)**:**
  Let $b \in B$ be the building, so $\exists c \in C : d(b, c) \leq d(b', c') \forall b' \in B, c' \in C$
  Let $S = \{b\}$
  while $|S| < k$ **do**
    Let $c_i \in C$ be such that $d(c_i, S)$ is maximized
    Let $b_j \in B$ be such that $d(b_j, c_i)$ is minimized
    $S \leftarrow S \cup \{b_j\}$
  return S

1.1 Approximation Ratio

Let S_{Opt} be the optimal set of buildings. S_{Opt} partitions C into k clusters. Customer c_i belongs to cluster x if x is the point in S_{Opt} closest to c_i. Additionally, we say b_j lays within cluster x if $d(b_j, x) \leq Opt$. With this we can define the following lemmas.

Lemma 1.1. *If x, y lay within the same cluster $d(x, y) \leq 2Opt$*

Proof. See lecture slides Lecture 3 slide 45 [1]. □

Note that 1.1 is true for all points within the cluster. Independent of whether they are buildings or customers.

Lemma 1.2. *If the algorithm selects building b, a customer c exists such that $d(b, c) \leq Opt$.*

Proof. We can assume customer c is the closest to b from all customers. Then c must have been the customer furthest away from S_{Alg} before adding b. We also know that customer c belongs to some cluster x. Let us assume $d(b, c) > Opt$ and contradict it.

Case 1: $b \in S_{Opt}$. Then customer c belongs to cluster b since it is the closest building. But then the optimal solution can not be Opt since $d(b, c) > Opt$.

Case 2: $b \notin S_{Opt}$. The optimal solution can't be Opt either, since

$$\forall b_x \in B : d(b_x, c) \geq d(b, c) > Opt$$

Therefore the assumption $d(b, c) > Opt$ must be wrong. □

Note that this also is true for the first selected building, since

$$Opt = \max_{i=1..m} d(c_i, S_{Opt}) \geq \min_{\substack{i=1..m,\\ j=1..n}} d(c_i, b_j)$$

Lemma 1.3. *If the algorithm picks one building from within each cluster then $Alg \leq 2Opt$.*

Proof. See lecture slides Lecture 3 slide 47 [1]. □

Lemma 1.4. *If the algorithm selects both b_i and b_j from within the same cluster x then $Alg \leq d(b_i, b_j) + d(b_j, c)$*

Proof. Suppose b_i is picked first, then b_j. When b_j is picked, it is the building closest to the customer c that is furthest away from all previously picked buildings.

Case 1: $c \in X$ then using Lemma 1.1 $Alg \leq d(b_i, c) \leq 2Opt$

Case 2: $c \notin X$ then using Lemma 1.2 $Alg \leq d(b_i, c) \leq d(b_i, b_j) + d(b_j, c) \leq 3Opt$

So we have $Alg = \max_{k=1,\ldots,n} d(b_k, S_{Alg}) \leq d(b_i, b_j)$. □

Conclusion We are left with the following cases of how the buildings we choose with the algorithm are spread across the clusters:

Case 1: We pick two buildings b_i, b_j from within the same cluster. With Lemma 1.4 (using 1.1 and 1.2) follows that $Alg \leq 3Opt$.

Case 2: We pick a building b that is not within any cluster. Lemma 1.2 $\Rightarrow$ customer c exists with $d(c, b) \leq Opt$. Lemma 1.1 $\Rightarrow$ all customers c_y in the same cluster as c have a distance lower than $d(b, c) + d(c, c_y) \leq 3Opt$ to b. Therefore $Alg \leq 3Opt$

Case 3: We pick one building from each cluster. Lemma 1.3 $\Rightarrow Alg \leq 2Opt \leq 3Opt$

This in total gives $Alg \leq 3Opt$.

1.2 Time Analysis

1.2.1 Steps

1. Finding the shortest distance between a customer and a building requires comparing all buildings from set B with the distance of all customers from set C. This will have a time complexity of $O(|B| \cdot |C|)$.

2. The while loop will then run for $k - 1$ iterations since we have initialized S with one building. The algorithm will do the following in each iteration:

- Find the maximum distance between a customer and a set S of buildings which will take $O(k \cdot |C|)$.
- Check the closest building to it which takes $O(|B|)$

Therefore, each iteration of the while loop will take $O(k \cdot |C| + |B|)$.

1.2.2 Overall Time Complexity

The overall time complexity is the sum of the initialization step and the time spent in the while loop.

Time complexity $= O(|B| \cdot |C| + (k-1)(k \cdot |C| + |B|)) \rightarrow O(|B| \cdot |C|) + O(k^2 \cdot |C| + k \cdot |B|))$

It follows that the time complexity is in $O(|B|^2 \cdot |C|)$

References

[1] P. Austrin, "Lecture 3: Greedy approximation algorithms," September 2023.