

HOMEWORK - 3

PROBLEM 5.3 (Metric k -Clustering)

Input: points $X = (x_1, \dots, x_n)$, metric $distance(\cdot, \cdot)$, positive integer k

Output: subsets C_1, \dots, C_k that partition X : $\forall i, j, C_i \cap C_j = \emptyset$ and $\cup C_{i..k} = X$.

Objective: minimize the maximum cluster diameter

$$\min \{ \max_{[C_j]} \max_{[x_a, x_b \in C_j]} distance(x_a, x_b) \}$$

2-Factor Approximation

Algorithm: Pick a random node and label it as the first cluster center. Choose $k-1$ more cluster centers such that at each step pick the farthest possible vertex away from the previously chosen cluster centers. Then, create partitions based on associating node with its nearest center node. Please see the pseudo code below.

Feasibility: We partitioned vertices into k clusters.

Analysis: Let z be the furthest point from cluster centers c_1, \dots, c_k . Define Δ as the distance from z to the closest center c_i .

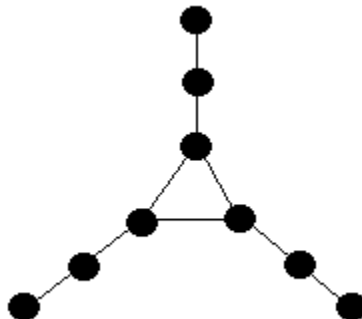
Claim1: The cost of the solution found by the Algorithm is no more than $2 \cdot \Delta$.

Proof: Since z is the furthest point from cluster centers, all other points must be within Δ of their nearest cluster center. Therefore, the radius of each cluster is at most Δ . By the triangle inequality, the diameter of each cluster cannot be more than $2 \cdot \Delta$. □

Claim2: The cost of the optimal solution is at least Δ .

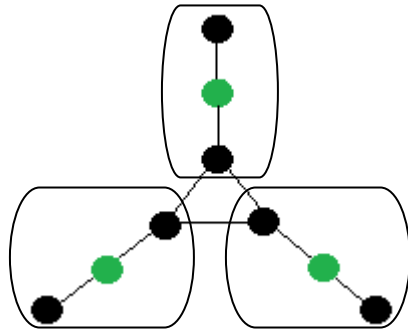
Proof: Assume we have $k+1$ points; cluster centers c_1, \dots, c_k and z all of which are at least Δ apart from each other. By the pigeonhole principle, at least two points belong to the same cluster. Therefore, the optimal clustering must be at least Δ . □

Tight Example:



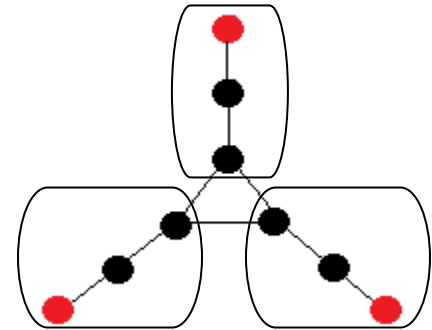
Assume all edges cost c .

Optimal solution would be;



OPT value: c

Algorithm may pick furthest points;



Approximation value: $2c$

5.3.2) Our 2-factor approximation algorithm is optimal in a very strong sense.

To show this, we can reduce from Dominating Set problem. Take an instance of DS with graph $G(V, E)$ on n nodes and a positive integer k . For the reduction, we create a weighted complete graph G' on the same vertices V where edge cost is 1 if edge is in G and edge cost is 2 if the edge is not in G .

G' satisfies the triangle inequality and this reduction satisfies the following conditions;

- If there is a DS in G of size $\leq k$, then G' has a k -cluster of cost 1 and
- If there is a DS in G of size $> k$, the optimum cost of a k -cluster in G' is 2.

In the first case, when run on G' the $(2-\epsilon, \epsilon > 0)$ -approximation algorithm must give a solution of cost 1, since we cannot use an edge of cost 2. Hence, we can distinguish between two possibilities, thus solving the DS problem.

Pseudo Code

k-cluster-2approximation (X, d, k)

1. Pick any point $c_1 \in X$ as the first cluster center.
2. **for** $j = 2$ to k
do
// the point in X , furthest from the already chosen cluster centers
3. Let $c_j = \max_{z \in X} \min_{i=1..j-1} \text{distance}(z, c_i)$
4. Associate each node with its nearest cluster center.
5. Output C_k as the set of point c_k and points associated with c_k .

Algorithm : *k-cluster-2approximation*(X, d, k)