k-MEDIAN CLUSTERING VIA METRIC EMBEDDING: TOWARDS BETTER INITIALIZATION WITH PRIVACY

Anonymous authors
Paper under double-blind review

ABSTRACT

In clustering algorithms, the choice of initial centers is crucial for the quality of the learned clusters. We propose a new initialization scheme for the k-median problem in the general metric space (e.g., discrete space induced by graphs), based on the construction of metric embedding tree structure of the data. From the tree, we can extract good initial centers that can be used subsequently for the local search algorithm. Our method, named the HST initialization, can also be easily extended to the setting of differential privacy (DP) to generate private initial centers. Theoretically, the initial centers from HST initialization can achieve lower error than those from another popular initialization method, k-median++, in the non-DP setting. Moreover, with privacy constraint, we show that the error of applying DP local search followed by our private HST initialization improves previous results, and approaches the known lower bound within a small factor. Empirically, experiments are conducted to demonstrate the effectiveness of our methods.

1 INTRODUCTION

Clustering is an important problem in unsupervised learning that has been widely used in social network analysis, sensor networks, etc. (Punj & Stewart, 1983; Dhillon & Modha, 2001; Banerjee et al., 2005; Abbasi & Younis, 2007). The goal of clustering is to partition a set of data points into clusters such that items in the same cluster are expected to be similar, while items in different clusters should be different. This is concretely measured by the sum of distances (or squared distances) between each point to its nearest cluster center. One conventional notion to evaluate a clustering algorithms is: with high probability,

$$cost(C, D) \leq \gamma OPT_k(D) + \xi,$$

where C is the centers output by the algorithm and $cost(C, D)$ is some cost function defined for C on the dataset D. $OPT_k(D)$ is the cost of optimal (oracle) k-median solution on D. When everything is clear from context, we will use OPT for short. Here, γ is called *multiplicative error* and ξ is called *additive error*. Alternatively, we may also use the notion of expected cost.

Two popularly studied clustering problems are 1) the k-median problem, and 2) the k-means problem. The origin of k-median dates back to the 1970's (e.g., Kaufman et al. (1977)), where one tries to find the best location of facilities that minimizes the cost measured by the distance between clients and facilities. Formally, given a set of points D and a distance measure, the goal is to find k center points minimizing the sum of absolute distances of each sample point to its nearest center. In k-means, the objective is to minimize the sum of squared distances instead. There are two popular clustering algorithms. One heuristic is the Lloyd's algorithm (Lloyd, 1982), which is built upon an iterative distortion minimization approach. In most cases, this method can only be applied to numerical data, typically in the Euclidean space. Clustering in general metric spaces (discrete spaces) is also important and useful when dealing with, for example, the graph data, where Lloyd's method is no longer applicable. A more broadly applicable approach, the local search method (Kanungo et al., 2002; Arya et al., 2004), has also been widely studied. It iteratively finds the optimal swap between the center set and non-center data points to keep lowering the cost. Local search can achieve a constant approximation ratio ($\gamma = O(1)$) to the optimal solution for k-median (Arya et al., 2004).

Initialization of cluster centers. It is well-known that the performance of clustering can be highly sensitive to proper initialization. If clustering starts with good initial centers (i.e., with small approximation error), the algorithm may use fewer iterations to find a better solution. The k-means++

algorithm (Arthur & Vassilvitskii, 2007) iteratively selects k data points as initial centers, favoring distant points in a probabilistic way. Intuitively, the initial centers tend to be well spread over the data points (i.e., over different clusters). The produced initial center is proved to be have $O(\log k)$ multiplicative error. There are many followup works of k-means++ that improve its efficiency and scalability, e.g., Bahmani et al. (2012); Bachem et al. (2016); Lattanzi & Sohler (2019). In this work, we propose a new initialization framework based on metric embedding methods. Also, our initialization scheme can be conveniently combined with the notion of differential privacy.

Clustering with Differential Privacy. The concept of differential privacy (Dwork, 2006; McSherry & Talwar, 2007) has been popular to rigorously define and resolve the problem of keeping useful information for model learning, while protecting privacy for each individual. Private k-means problem has been widely studied, e.g., Feldman et al. (2009); Nock et al. (2016); Feldman et al. (2017), mostly in the Euclidean space. The paper (Balcan et al., 2017) considered identifying a good candidate set (in a private manner) of centers before applying private local search, which yields $O(\log^3 n)$ multiplicative error and $O((k^2 + d)\log^5 n)$ additive error. Later on, the Euclidean k-means errors are further improved to $\gamma = O(1)$ and $\xi = O(k^{1.01} \cdot d^{0.51} + k^{1.5})$ by Stemmer & Kaplan (2018), with more advanced candidate set selection. Huang & Liu (2018) gave an optimal algorithm in terms of minimizing Wasserstein distance under some data separability condition.

For private k-median clustering, Feldman et al. (2009) considered the problem in high dimensional Euclidean space. However, it is rather difficult to extend their analysis to more general metrics in discrete spaces (e.g., on graphs). The strategy of Balcan et al. (2017) to form a candidate center set could as well be adopted to k-median, which leads to $O(\log^{3/2} n)$ multiplicative error and $O((k^2+d)\log^3 n)$ additive error in high dimensional Euclidean space. Gupta et al. (2010) proposed a private method for the classical local search heuristic in discrete space, which applies to both k-medians and k-means. In order to cast privacy on each swapping step, the authors applied the exponential mechanism of McSherry & Talwar (2007). Their method produced an ϵ-differentially private solution with cost $6OPT + O(\triangle k^2 \log^2 n/\epsilon)$, where $\triangle$ is the diameter of point set.

Our contributions. We propose a new framework for center initialization of k-median in the general metric space, based on metric embedding techniques. More specifically,

- We introduce Hierarchically Well-Separated Tree (HST), a metric embedding tree structure, to the k-median clustering problem. Once the HST is constructed using the data samples, we provide a sampling strategy to select the initial center set, with an approximation factor $O(\log \min\{k, d\})$ in the non-private setting. This improves the $O(\log k)$ error of k-means/median++ in e.g., the lower dimensional Euclidean space.
- We propose an ϵ-differentially private (DP) version of our initialization method. Running DP-local search beginning from this initial center gives $O(1)$ multiplicative error and $O(\epsilon^{-1}\triangle k^2(\log\log n)\log n)$ additive error, which improves previous results towards the well-known lower bound $O(\epsilon^{-1}\triangle k \log(n/k))$ on the additive error (Gupta et al., 2010) within a small $O(k \log\log n)$ factor. To our knowledge, this is the first center initialization method with differential privacy guarantee and improved error rate.
- We conduct experiments on simulated and real-world datasets to demonstrate the effectiveness of our methods. In both non-private and private settings, our proposed HST-based approach achieves smaller cost at initialization than k-median++, which may also lead to improvements in the final clustering quality.

2 Preliminaries

Definition 2.1 (Differential Privacy (DP) (Dwork, 2006))**.** *If for any two adjacent data sets D and D' with symmetric difference one, for any $O \subset Range(\mathbb{A})$, a randomized algorithm $\mathbb{A}$ satisfies*

$$Pr[\mathbb{A}(D) \in O] \leq e^{\epsilon} Pr[\mathbb{A}(D') \in O],$$

then algorithm $\mathbb{A}$ is said to be ϵ-differentially private.

Intuitively, differential privacy requires that after removing any observation, the output of D' should not be too different from that of the original dataset D. Smaller ϵ indicates stronger privacy, which, however, usually sacrifices utility. Thus, one of the central topics in differential privacy literature is to balance the utility-privacy trade-off.

To achieve DP, it is common to introduce noise to the data or the algorithm output. The *Laplace mechanism* adds Laplace($\eta(f)/\epsilon$) noise to the output, which is known to achieve ϵ-DP. The *exponential mechanism* is also a tool for many DP algorithms. Let O be the set of feasible outputs. The utility function $q : D \times O \to \mathbb{R}$ is what we aim to maximize. The exponential mechanism outputs an element $o \in O$ with probability $P[\mathbb{A}(D) = o] \propto \exp(\frac{\epsilon q(D,o)}{2\eta(q)})$, where D is the input dataset and $\eta(f) = \sup_{|D-D'|=1} |f(D) - f(D')|$ is the sensitivity of f. Both mechanisms will be used in the design of our proposed DP approach.

k-Median Clustering. Following the framework of Arya et al. (2004); Gupta et al. (2010), the problem of k-median clustering (DP and non-DP) studied in our paper is stated as below.

Definition 2.2 (k-median)**.** *Given a universe point set U and a metric $\rho : U \times U \to \mathbb{R}$, let $D \subseteq U$ be a set of demand points. The goal of DP k-median is to pick $F \subseteq U$ with $|F| = k$ to minimize*

$$\textbf{\textit{Private } k\textbf{\textit{-median:}}} \quad cost(F, D) = \sum_{v \in D} \min_{f \in F} \rho(v, f). \tag{1}$$

At the same time, the output F is required to be ϵ-differentially private to D. For standard non-private k-median, we assume $U = D$ and minimize

$$k\textbf{\textit{-median:}} \quad cost(F, U) = \sum_{v \in U} \min_{f \in F} \rho(v, f). \tag{2}$$

We may drop "F" and use "$cost(D)$" or "$cost(U)$" if there is no risk of ambiguity.

To better understand the DP clustering problem, we provide an real-world example as follows.

Example 2.1. *Consider U to be the universe of all users in a social network (e.g., Twitter). Each user (account) is public, but also has some private information that can only be seen by the data holder. Let D be users grouped by some information that might be set as private. Suppose a third party plans to collaborate with the most influential users in D for e.g., commercial purposes, thus requesting the cluster centers of D. In this case, we need a strategy to safely release the centers, while protecting the individuals in D from being identified (since the membership of D is private).*

The local search procedure for k-median proposed by Arya et al. (2004) is summarized in Algorithm 1. First we randomly pick k points in U as the initial centers. In each iteration, we search over all $x \in F$ and $y \in U$, and do the swap $F \leftarrow F - \{x\} + \{y\}$ such that $F - \{x\} + \{y\}$ improves the cost of F the most (if more than factor $(1 - \alpha/k)$ where $\alpha > 0$ is a hyper-parameter). We repeat the procedure until no such swap exists. Arya et al. (2004) showed that the output centers F achieves 5 approximation error to the optimal solution, i.e., $cost(F) \leq 5OPT$.

Algorithm 1: Local search for k-median clustering (Arya et al., 2004)

Input: Data points U, parameter k, constant α
Initialization: Randomly select k points from U as initial center set F
while $\exists\, x \in F, y \in U$ *s.t.* $cost(F - \{x\} + \{y\}) \leq (1 - \alpha/k) cost(F)$ **do**
  Select $(x, y) \in F_i \times (D \setminus F_i)$ with $\arg\min_{x,y}\{cost(F - x + y)\}$
  Swap operation: $F \leftarrow F - \{x\} + \{y\}$
Output: Center set F

3 Initialization via Hierarchically Well-Separated Tree (HST)

In this section, we propose our new initialization scheme for k-median clustering, and provide our analysis in the non-private case solving (2). The idea is based on the metric embedding theory. We will start with an introduction to the main tool used in our approach.

3.1 Hierarchically Well-Separated Tree (HST)

In this paper, for an L-level tree, we will count the levels from large to small. That is, the root is level L, and its children form level $L - 1$, and etc. We use h_v to denote the level of v, and n_i be the

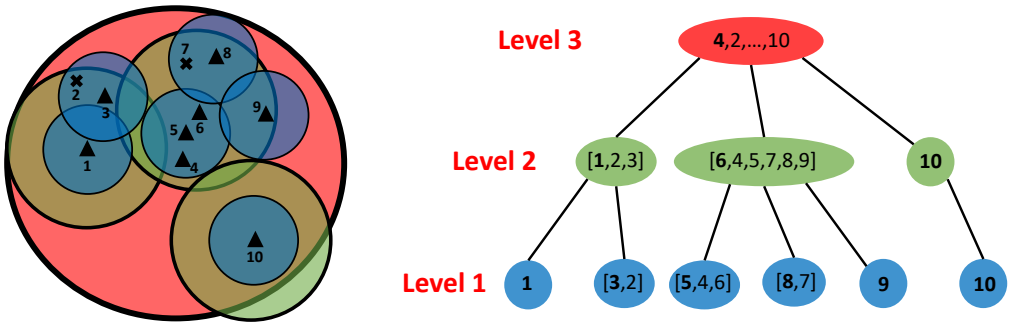

Figure 1: An illustrative example of a 3-level padded decomposition and corresponding 2-HST. **Left:** The thickness of the ball represents the level. The color corresponds to the levels in the HST in the right panel. "$\triangle$"'s are the center nodes of partitions (balls), and "$\times$"'s are non-center data points. **Right:** The resulting 2-HST generated from the padded decomposition.

number of nodes at level i. The Hierarchically Well-Separated Tree (HST) is based on the padded decompositions of a general metric space in a hierarchical manner (Fakcharoenphol et al., 2004). Let (U, ρ) be a metric space with $|U| = n$, and we will refer to this metric space without specific clarification. A β–padded decomposition of U is a probabilistic distribution of partitions of U such that the diameter of each cluster $U_i \in U$ is at most β, i.e., $\rho(u, v) \leq \beta$, $\forall u, v \in U_i$, $i = 1, ..., k$. Based on the this, we will adopt the standard 2-HST for our algorithm design.

Definition 3.1. *Assume the smallest distance between $u, v \in U$ is 1, and $\triangle = \max_{u,v \in U} \rho(u, v)$. A 2-Hierarchically Well-Separated Tree (2-HST) is an edge-weighted rooted tree T, such that an edge between any pair of two nodes of level $i - 1$ and level i has length at most $\triangle/2^{L-i}$.*

We briefly describe the construction of 2-HST, and place the detailed algorithm in Algorithm 6 in the Appendix A. We first find a padded decomposition $P_L = \{P_{L,1}, ..., P_{L,n_L}\}$ of U with parameter $\beta = \triangle/2$. The center of each partition in $P_{L,j}$ serves as a root node in level L. Then, we re-do a padded decomposition for each partition $P_{L,j}$, to find sub-partitions with diameter $\beta = \triangle/4$, and set the corresponding centers as the nodes in level $L - 1$, and so on. Each partition at level i is obtained with $\beta = \triangle/2^{L-i}$. This process proceeds until a node has a single point, or a pre-specified tree depth is reached. In Figure 1, we provide an example of $L = 3$-level 2-HST (left panel), along with its underlying padded decompositions (right panel).

3.2 HST Initialization Algorithm

Suppose T is an $L = \log \triangle$-level 2-HST in (U, ρ), where for simplicity we assume L is an integer. Using this L is mainly for the ease of presentation. We may choose other reasonable L in practice. For a node v at level i, we use $T(v)$ to denote the subtree rooted at v. Let $N_v = |T(v)|$ be the number of data points in $T(v)$. The sampling strategy for initial centers are presented in Algorithm 2.

Subtree search. The first step is to identify the subtrees that contain the k centers. To begin with, k initial centers C_1 are picked from T who have the largest $score(v) = N(v) \cdot 2^{h_v}$. This is intuitive, since to get a good clustering, we typically want the ball surrounding each center to include more data points. Next, we do a screening over C_1: if there is any ancestor-descendant pair of nodes, we remove the ancestor from C_1. If the current size of C_1 is smaller than k, we repeat the process until k centers are chosen (we do not re-select nodes in C_1 and their ancestors). This way, C_1 contains k root nodes of k disjoint subtrees. For any node w with $T(w) \cap C_1 = \emptyset$ and a node $v \in C_1$, if w is not an ancestor of v, then $score(w) \leq score(v)$.

Leaf search. After we find C_1 the set of k subtrees, the next step is to find the center in each subtree using Algorithm 3 ("FIND-LEAF"). We employ a greedy search strategy, by finding the child node with largest score level by level, until a leaf is found. This approach is intuitive since the diameter of the partition ball exponentially decays with the level. Therefore, we are in a sense focusing more and more on the region with higher density (i.e., with more data points).

Algorithm 2: HST initialization

Input: U, $\triangle$, k
Initialization: $L = \log \triangle$, $C_0 = \emptyset$, $C_1 = \emptyset$
Build a level-L 2-HST tree T based on input U
for *each node* v *in* T **do**
  $N_v \leftarrow |U \cap T(v)|$
  $score(v) \leftarrow N(v) \cdot 2^{h_v}$
while $|C_1| < k$ **do**
  Add top $(k - |C_1|)$ nodes with highest score in T to C_1
  for *each* $v \in C_1$ **do**
    $C_1 = C_1 \setminus \{v\}$, if $\exists\, v' \in C_1$ that v' is a descendant of v
$C_0 =$ FIND-LEAF(T, C_1)
Output: Initial center set $C_0 \subseteq U$

Algorithm 3: FIND-LEAF (T, C_1)

Input: T, C_1
Initialization: $C_0 = \emptyset$
for *each node* v *in* C_1 **do**
  while v *is not a leaf node* **do**
    $v \leftarrow \arg_w \max\{N_w, w \in ch(v)\}$, where $ch(v)$ denotes the children nodes of v
  Add v to C_0
Output: Initial center set $C_0 \subseteq U$

3.3 Approximation Error of HST Initialization

Firstly, we show that the initial center set produced by Algorithm 2 is already a good approximation to the optimal k-median solution. Let $\rho^T(x, y) = d_T(x, y)$ denote the "2-HST metric" between x and y in the 2-HST tree T, where $d_T(x, y)$ is the tree distance between nodes x and y in T. In 2-HST, by Definition 3.1 and $\triangle = 2^L$, in the analysis we assume equivalently that the edge weight of the i-th level 2^{i-1}. The crucial step of our analysis is to examine the approximation error in terms of the 2-HST metric, after which the error can be easily adapted to the general metrics by Lemma 3.1.

Lemma 3.1. *(Bartal, 1996). In a metric space (U, ρ) with $|U| = n$ and diameter $\triangle$, it holds that $E[\rho^T(x, y)] = O(\min\{\log n, \log \triangle\})\rho(x, y)$. In $\mathbb{R}^d$, $E[\rho^T(x, y)] = O(d)\rho(x, y)$.*

Recall C_0, C_1 from Algorithm 2. We define

$$cost_k^T(U) = \sum_{y \in U} \min_{x \in C_0} \rho^T(x, y), \tag{3}$$

$$cost_k^{T\prime}(U, C_1) = \min_{|F \cap T(v)| = 1, \forall v \in C_1} \sum_{y \in U} \min_{x \in F} \rho^T(x, y), \tag{4}$$

$$OPT_k^T(U) = \min_{F \subset U, |F| = k} \sum_{y \in U} \min_{x \in F} \rho^T(x, y) \equiv \min_{C_1'} \; cost_k^{T\prime}(U, C_1'). \tag{5}$$

For simplicity, we will use $cost_k^{T\prime}(U)$ to denote $cost_k^{T\prime}(U, C_1)$. Here, OPT_k^T (5) is the cost of the global optimal solution with 2-HST metric. The last equality in (5) holds because the optimal centers set can always located in k disjoint subtrees, as each leaf only contain one point. (3) is the k-median cost with 2-HST metric of the output C_0 of Algorithm 2. (4) is the oracle cost after the subtrees are chosen. That is, it represents the optimal cost to pick one center from each subtree in C_1. Firstly, we bound the approximation error of the subtree search process.

Lemma 3.2 (Subtree search)**.** $cost_k^{T\prime}(U) \leq 5OPT_k^T(U)$.

Next, we show that the greedy leaf search (Algorithm 3) only has constant extra multiplicative error.

Lemma 3.3 (Leaf search)**.** $cost_k^T(U) \leq 2cost_k^{T\prime}(U)$.

Combining Lemma 3.2 and Lemma 3.3, we have the next Theorem.

Theorem 3.4 (2-HST error)**.** *Running Algorithm 2, we have* $cost_k^T(U) \leq 10OPT_k^T(U)$.

Thus, HST-initialization produces an $O(1)$ approximation to OPT in the 2-HST metric. Define $cost_k(U)$ as (2) for our HST centers, and the optimal cost w.r.t. ρ as

$$OPT_k(U) = \min_{|F|=k} \sum_{y \in U} \min_{x \in F} \rho(x, y). \tag{6}$$

We have the following result in the general metric space based on Lemma 3.1.

Theorem 3.5. *Running Algorithm 2 gives* $E[cost_k(U)] = O(\min\{\log n, \log \triangle\})OPT_k(U)$.

Remark 3.1. *Note that Makarychev et al. (2019) proved* $O(\log k)$ *random projections suffice for* k*-median to achieve* $O(1)$ *error. Thus, if* $\triangle = O(d)$ *(e.g., bounded data), by Lemma 3.1, there exists an algorithm with HST initialization that achieves* $O(\log(\min\{d, k\}))$ *error, which is better than* $O(\log k)$ *of* k*-median++ when* d *is small.*

NDP-HST Local Search. We can apply standard local search starting from the HST initialization (instead of random) in Algorithm 1. We call it the NDP-HST method, with the following guarantee.

Theorem 3.6. *NDP-HST achieves* $O(1)$ *approximation in expected* $O(k \log \min\{\log n, \log \triangle\})$ *number of iterations for input in general metric space.*

Before ending this section, we remark that all our analysis can be easily extended to k-means clustering. Due to the space limitation, we present some related results in Appendix C.

4 HST Initialization with Differential Privacy

In this section, we consider private HST initialization method. Recall in this setting, U is the universe of data points, and $D \subset U$ is a demand set that needs to be clustered with privacy. Since U is public, running initialization algorithms on U would preserve the privacy of D. Yet, this might be too expensive, and in many cases one would probably want to incorporate some information about D in the initialization, since D could be a very imbalanced subset of U. For example, D may only contain data points from one cluster, out of tens of clusters in U. In this case, initialization on U is likely to pick initial centers in multiple clusters, which would not be helpful for clustering on D. Next, we show how our HST initialization can be easily combined with differential privacy that at the same time contains information about the demand set D, leading to improved approximation error (Theorem 4.3) and empirical clustering performance (Section 5).

Again, suppose T is an $L = \log \triangle$-level 2-HST of universe U in a general metric space. Denote $N_v = |T(v) \cap D|$ for a node point v. Our private HST initialization is similar to the non-private Algorithm 2. To gain privacy, we perturb N_v by adding i.i.d. Laplace noise:

$$\hat{N}_v = N_v + Lap(2^{(L-h_v)}/\epsilon),$$

where $Lap(2^{(L-h_v)}/\epsilon)$ is a Laplace random number with rate $2^{(L-h_v)}/\epsilon$. We will use the perturbed $\hat{N}_v$ for node sampling instead of the true value N_v, as described in Algorithm 4. The DP guarantee of this initialization scheme is straightforward by the composition of the Laplace mechanisms.

Theorem 4.1. *Algorithm 4 is* ϵ*-differentially private.*

Proof. For each level i, the subtrees $T(v, i)$ are disjoint to each other. The privacy used in i-th level is $\epsilon/2^{(L-i)}$, and the total privacy is $\sum_i \epsilon/2^{(L-i)} < \epsilon$. □

We now consider the approximation error. As the structure of the analysis is similar to the non-DP case, we present the main result here and defer the detailed proofs to Appendix B.

Theorem 4.2. *Algorithm 4 outputs an initial center set such that*

$$E[cost_k(D)] = O(\log n)(10OPT_k(D) + 10k\epsilon^{-1}\triangle \log n).$$

DP-HST Local Search. Similarly, we can use private HST initialization to improve the performance of private k-median local search, which is presented in Algorithm 5. After initialization, the DP local search procedure follows Gupta et al. (2010) using the exponential mechanism.

Algorithm 4: HST initialization with privacy

Input: $U, D, \triangle, k, \epsilon$
Build a level-L 2-HST tree T based on input U
for *each node v in T* **do**
  $N_v \leftarrow |D \cap T(v)|$
  $\hat{N}_v \leftarrow N_v + Lap(2^{(L-h_v)}/\epsilon)$
  $score(v) \leftarrow \hat{N}(v) \cdot 2^{h_v}$
Based on $\hat{N}_v$, apply the same strategy as Algorithm 2: find C_1; $C_0 =$ FIND-LEAF(C_1)
Output: Private initial center set $C_0 \subseteq U$

Algorithm 5: DP-HST local search

Input: U, demand points $D \subseteq U$, parameter k, ϵ, T
Initialization: F_1 the private initial centers generated by Algorithm 4 with privacy $\epsilon/2$
Set parameter $\epsilon' = \frac{\epsilon}{2\triangle(T+1)}$
for $i = 1$ *to* T **do**
  Select $(x, y) \in F_i \times (V \setminus F_i)$ with prob. proportional to $\exp(-\epsilon' \times (cost(F_i - \{x\} + \{y\}))$
  Let $F_{i+1} \leftarrow F_i - \{x\} + \{y\}$
Select j from $\{1, 2, ..., T+1\}$ with probability proportional to $\exp(-\epsilon' \times \ cost(F_j))$
Output: $F = F_j$ the private center set

Theorem 4.3. *Algorithm 5 achieves ϵ-differential privacy. The output centers admit*

$$cost_k(D) \leq 6OPT_k(D) + O(\epsilon^{-1}k^2\triangle(\log\log n)\log n)$$

with probability $(1 - 1/poly(n))$, with $T = O(k \log\log n)$ iterations.

The DP local search with random initialization (Gupta et al., 2010) has 6 multiplicative error and $O(\epsilon^{-1}\triangle k^2 \log^2 n)$ additive error. Our result improves the $\log n$ term to $\log\log n$ in the additive error. Meanwhile, the number of iterations needed is improved from $T = O(k \log n)$ to $O(k \log\log n)$. Note that, it has been shown in Gupta et al. (2010) that for k-median problem, the lower bounds on the multiplicative and additive error of any ϵ-DP algorithm are $O(1)$ and $O(\epsilon^{-1}\triangle k \log(n/k))$, respectively. Our result matches the lower bound on the multiplicative error, and the additive error is only worse than the bound by a factor of $O(k \log\log n)$ which is typically small in many cases. Thus, our private HST initialization method pushes the approximation error of private local search closer to the lower bound. To the best of our knowledge, Theorem 4.3 is the first result in literature to improve the error of DP local search in general metric space.

5 Experiments

We numerically test the proposed methods on two problems—clustering in an Euclidean space and on a graph. Our results show that the proposed HST initialization can improve the performance of using k-median++ initialization in both non-private and private clustering tasks.

5.1 Datasets and Algorithms

Discrete Euclidean space. Following Balcan et al. (2017), we test k-median clustering on the MNIST hand-written digit dataset (LeCun et al., 1998) with 10 natural clusters (digit 0 to 9). We set U as 10000 randomly chosen data points. We choose the demand set D using two strategies: 1) "balance", where we randomly choose 500 samples from U; 2) "imbalance", where D contains 500 random samples from U only from digit "0" and "8" (two clusters). We note that, the imbalanced D is a very practical setting in real-world scenarios, where data are typically not uniformly distributed. On this dataset, we test clustering with both l_1 and l_2 distance as the underlying metric.

Metric space induced by graph. Random graphs have been widely considered in testing k-median methods (Balcan et al., 2013; Todo et al., 2019). The construction of graphs follows a similar approach as the synthetic *pmedinfo* graphs provided by the popular OR-Library (Beasley, 1990).

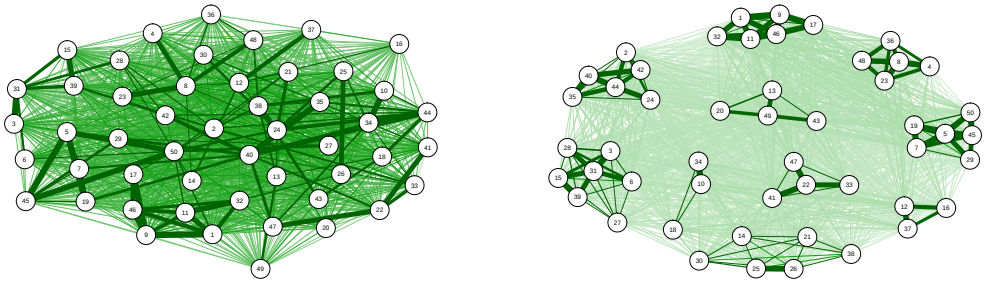

Figure 2: Example of synthetic graphs: subgraph of 50 nodes. **Left:** $r = 1$. **Right:** $r = 100$. Darker and thicker edged have smaller distance. When $r = 100$, the graph is more separable.

The metric ρ for this experiment is the shortest (weighted) path distance on graph. To generate a size n graph, we first randomly split the nodes into 10 clusters. Within each cluster, each pair of nodes is connected with probability 0.2 and weight drawn from standard uniform distribution. For each pair of clusters, we randomly connect some nodes from each cluster, with weights following uniform $[0.5, r]$. A larger r makes the graph more separable, i.e., clusters are farther from each other (see Figure 2). We present two cases: $r = 1$ and $r = 100$. For this task, U has 3000 nodes, and the private set D is chosen using similar "balanced" and "imbalanced" scheme as described above. In the imbalanced case, we choose D randomly from only two clusters.

Algorithms. We compare the following clustering algorithms in both non-DP and DP setting.

- **NDP-rand:** Local search on D with random initialization (Algorithm 1).
- **NDP-kmedian++:** Algorithm 1 with k-median++ initialization.
- **NDP-HST:** Algorithm 1 with HST initialization, as described in Section 3.
- **DP-rand:** Standard private local search algorithm (Gupta et al., 2010), which is Algorithm 5 with initial centers randomly chosen from U.
- **DP-kmedian++:** Algorithm 5 with k-median++ initialization run on U.
- **DP-HST:** Private local search with HST-initialization (Algorithm 5). For non-private tasks, we set $L = 6$. For private clustering, we use $L = 8$.

For non-DP methods, we set $\alpha = 10^{-3}$ in Algorithm 1 and the maximum number of iterations as 20. To examine the quality of initialization as well as the final centers, We report both the cost at initialization and the cost of the final output. For DP methods, we run the algorithms for $T = 20$ steps and report the results with $\epsilon = 1$. We test $k \in \{2, 5, 10, 15, 20\}$. The average cost over T iterations is reported for more robustness. All results are averaged over 10 independent repetitions.

5.2 Results

The results on MNIST dataset are given in Figure 3 for l_1 (left two columns) and l_2 (right two columns) metric. The comparisons are similar in both cases.

- We see that the initial centers generated by HST has lower k-median cost than k-median++ and random initialization, for both non-DP and DP setting, and for both balanced and imbalanced demand set D. This confirms that the proposed HST initialization is more powerful than k-median++ in finding good initial centers.
- From the final k-median cost plots, we also observe lower cost of HST approaches in DP clustering. In the non-DP case, the curves overlap, which means despite that HST offers better initial centers, local search can always find a good solution eventually.
- The advantage of HST in both initial and final cost is more significant when D is an imbalanced subset of U. As mentioned before, this is because our HST initialization approach also privately incorporates the information of D.

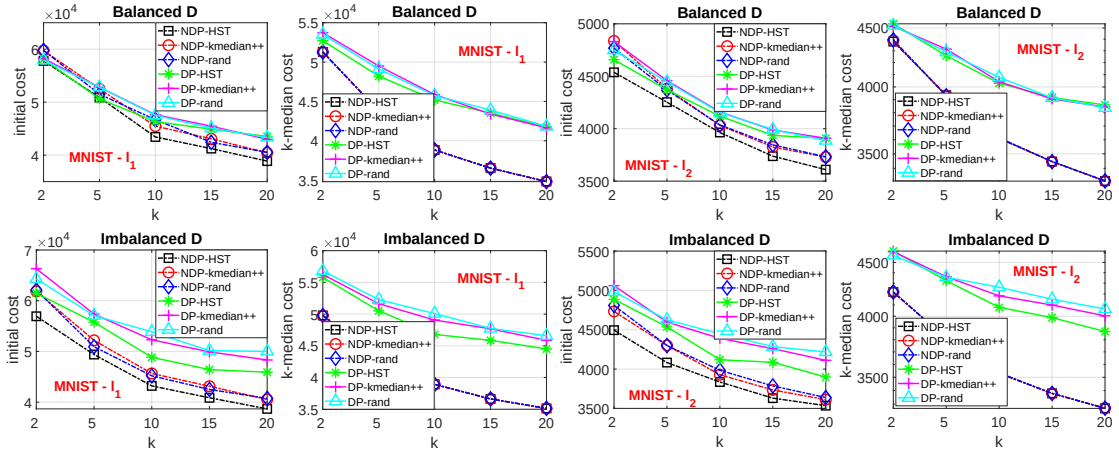

Figure 3: k-median cost on MNIST dataset. **1st row:** balanced D. **2nd row:** imbalanced D. **Column 1 & 3:** initial cost. **Column 2 & 4:** final k-median cost.

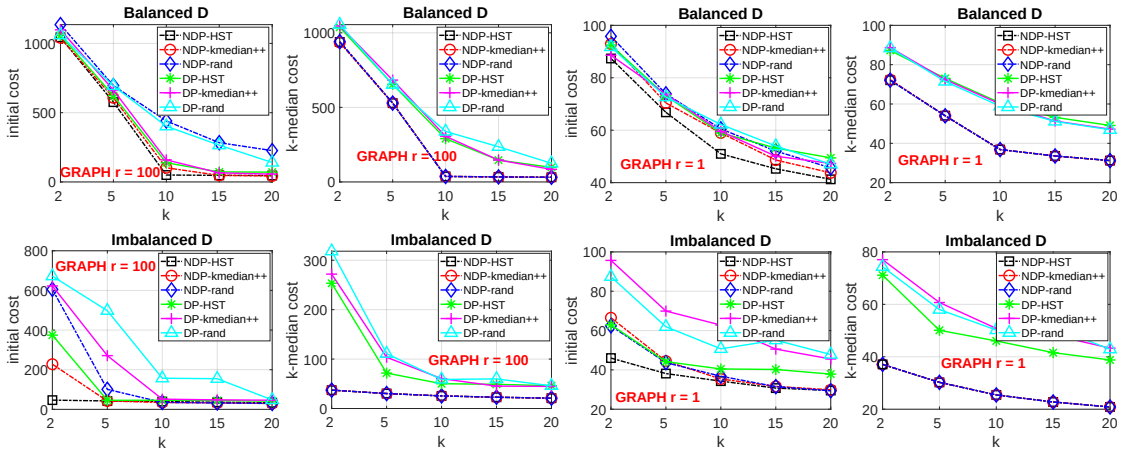

Figure 4: k-median costs on graph dataset. **1st row:** balanced D. **2nd row:** imbalanced D.

The set of results on the graph dataset is reported in Figure 4, which gives us similar conclusion. In all cases, our proposed HST method finds better initial centers with smaller cost than k-median++. In terms of the final clustering output, HST again considerably outperforms k-median++ in the private and imbalanced D setting, for both $r = 100$ (highly separable graph) and $r = 1$ (less separable graph). The improvement of HST over k-median++ is especially significant in the harder tasks when $r = 1$, i.e., the clusters are nearly mixed up.

6 Conclusion

In this paper, we propose a new initialization framework for the k-median problem in general discrete metric space. Our approach is called HST, which leverages tools from metric embedding theory. We show that in the standard k-median problem, our proposed method can achieve smaller cost than k-median++ at initialization. Moreover, we propose differentially private (DP) HST center initialization algorithm, which adapts to the private demand point set, leading to better clustering performance. When combined with subsequent DP local search heuristic, our algorithm is able to improve the addition error of prior works, which is close to the theoretical lower bound within a small factor. Experiments with Euclidean metrics and graph metrics both verify that the proposed HST initialization finds better initial centers than k-median++, and could also improve the final clustering cost in the differential privacy setting.

REFERENCES

Ameer Ahmed Abbasi and Mohamed F. Younis. A survey on clustering algorithms for wireless sensor networks. *Comput. Commun.*, 30(14-15):2826–2841, 2007.

David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pp. 1027–1035, 2007.

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.

Olivier Bachem, Mario Lucic, S. Hamed Hassani, and Andreas Krause. Approximate k-means++ in sublinear time. In Dale Schuurmans and Michael P. Wellman (eds.), *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pp. 1459–1467. AAAI Press, 2016.

Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable k-means++. *Proc. VLDB Endow.*, 5(7):622–633, 2012.

Maria-Florina Balcan, Steven Ehrlich, and Yingyu Liang. Distributed k-means and k-median clustering on general communication topologies. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pp. 1995–2003, 2013.

Maria-Florina Balcan, Travis Dick, Yingyu Liang, Wenlong Mou, and Hongyang Zhang. Differentially private clustering in high-dimensional euclidean spaces. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 322–331, 2017.

Arindam Banerjee, Srujana Merugu, Inderjit S. Dhillon, and Joydeep Ghosh. Clustering with bregman divergences. *J. Mach. Learn. Res.*, 6:1705–1749, 2005.

Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pp. 184–193. IEEE Computer Society, 1996.

John E Beasley. Or-library: distributing test problems by electronic mail. *Journal of the operational research society*, 41(11):1069–1072, 1990.

Inderjit S. Dhillon and Dharmendra S. Modha. Concept decompositions for large sparse text data using clustering. *Mach. Learn.*, 42(1/2):143–175, 2001.

Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, pp. 1–12, 2006.

Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, 2004.

D. Feldman, C. Xiang, R. Zhu, and D. Rus. Coresets for differentially private k-means clustering and applications to privacy in mobile sensor networks. In *2017 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pp. 3–16, 2017.

Dan Feldman, Amos Fiat, Haim Kaplan, and Kobbi Nissim. Private coresets. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pp. 361–370, 2009.

Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. Differentially private combinatorial optimization. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pp. 1106–1125, 2010.

Zhiyi Huang and Jinyan Liu. Optimal differentially private algorithms for k-means clustering. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*, pp. 395–408, 2018.

Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. In *Proceedings of the 18th Annual Symposium on Computational Geometry, Barcelona, Spain, June 5-7, 2002*, pp. 10–18, 2002.

Leon Kaufman, Marc Vanden Eede, and Pierre Hansen. A plant and warehouse location problem. *Journal of the Operational Research Society*, 28(3):547–554, 1977.

Silvio Lattanzi and Christian Sohler. A better k-means++ algorithm via local search. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3662–3671. PMLR, 2019.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2): 129–137, 1982.

Konstantin Makarychev, Yury Makarychev, and Ilya P. Razenshteyn. Performance of johnson-lindenstrauss transform for k-means and k-medians clustering. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pp. 1027–1038. ACM, 2019.

Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pp. 94–103, 2007.

Richard Nock, Raphaël Canyasse, Roksana Boreli, and Frank Nielsen. k-variates++: more pluses in the k-means++. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pp. 145–154, 2016.

Girish Punj and David W Stewart. Cluster analysis in marketing research: Review and suggestions for application. *Journal of marketing research*, 20(2):134–148, 1983.

Uri Stemmer and Haim Kaplan. Differentially private k-means with constant multiplicative error. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pp. 5436–5446, 2018.

Keisuke Todo, Atsuyoshi Nakamura, and Mineichi Kudo. A fast approximate algorithm for k-median problem on a graph. *15th International Workshop on Mining and Learning with Graphs*, 2019.

A CONSTRUCTING 2-HST TREE

As presented in Algorithm 6, the construction starts by applying a permutation π on U, such that in following steps the points are picked in a random sequence. We first find a padded decomposition $P_L = \{P_{L,1}, ..., P_{L,n_L}\}$ of U with parameter $\beta = \triangle/2$. The center of each partition in $P_{L,j}$ serves as a root node in level L. Then, we re-do a padded decomposition for each partition $P_{L,j}$, to find sub-partitions with diameter $\beta = \triangle/4$, and set the corresponding centers as the nodes in level $L-1$, and so on. Each partition at level i is obtained with $\beta = \triangle/2^{L-i}$. This process proceeds until a node has a single point, or a pre-specified tree depth is reached. In Figure 1, we provide an example of $L = 3$-level 2-HST (left panel), along with its underlying padded decompositions (right panel).

Algorithm 6: Build 2-HST(U, L)

Input: Data points U with diameter $\triangle$, L
Randomly pick a point in U as the root node of T
Let $r = \triangle/2$
Apply a permutation π on U `// so points will be chosen in a random sequence`
for *each* $v \in U$ **do**
  Set $C_v = [v]$
  for *each* $u \in U$ **do**
    Add $u \in U$ to C_v if $d(v, u) \leq r$ and $u \notin \bigcup_{v' \neq v} C_{v'}$
Set the non-empty clusters C_v as the children nodes of T
for *each non-empty cluster* C_v **do**
  Run 2-HST($C_v, L-1$) to extend the tree T; stop until L levels or reaching a leaf node
Output: 2-HST tree T

B PROOFS

The following composition result of differential privacy will be used in our proof.

Theorem B.1 (Composition Theorem (Dwork, 2006))**.** *If Algorithms* $\mathbb{A}_1, \mathbb{A}_2, ..., \mathbb{A}_m$ *are* $\epsilon_1, \epsilon_2, ..., \epsilon_m$ *differentially private respectively, then the union* $(\mathbb{A}_1(D), \mathbb{A}_2(D), ..., \mathbb{A}_m(D))$ *is* $\sum_{i=1}^{m} \epsilon_i$*-DP.*

B.1 PROOF OF LEMMA 3.2

Proof. Consider the intermediate output of Algorithm 2, $C_1 = \{v_1, v_2, ..., v_k\}$, which is the set of roots of the minimal subtrees each containing exactly one output center C_0. Suppose one of the optimal "root set" that minimizes (4) is $C_1^* = \{v_1', v_2', ..., v_k'\}$. If $C_1 = C_1^*$, the proof is done. Thus, we prove the case for $C_1 \neq C_1^*$. Note that $T(v), v \in C_1$ are disjoint subtrees. We have the following reasoning.

- Case 1: for some i, j', v_i is a descendant node of v_j'. Since the optimal center point f^* is a leaf node by the definition of (4), we know that there must exist one child node of v_j' that expands a subtree which contains f^*. Therefore, we can always replace v_j' by one of its child nodes. Hence, we can assume that v_i is not a descendant of v_j'.

  Note that, we have $score(v_j') \leq score(v_i)$ if $v_j' \notin C_1^* \cap C_1$. Algorithm 2 sorts all the nodes based on cost value, and it would have more priority to pick v_j' than v_i if $score(v_j') > score(v_i)$ and v_i is not a child node of v_j'.

- Case 2: for some i, j', v_j' is a descendant of v_i. In this case, optimal center point f^*, which is a leaf of $T(v_i)$, must also be a leaf node of $T(v_j')$. We can simply replace C_1 with the swap $C_1 \setminus \{v_i\} + \{v_j'\}$ which does not change $cost_k^{T'}(U)$. Hence, we can assume that v_j' is not a descendant of v_i.

- Case 3: Otherwise. By the construction of C_1, we know that $score(v_j') \leq \min\{score(v_i), i = 1, ..., k\}$ when $v_j' \in C_1^* \setminus C_1$. Consider the swap between C_1 and

C_1^*. By the definition of tree distance, we have $OPT_k^T(U) \geq \sum_{v_i \in C_1 \setminus C_1^*} N_{v_i} 2^{h_{v_i}}$, since $\{T(v_i), v_i \in C_1 \setminus C_1^*\}$ does not contain any center of the optimal solution determined by C_1^* (which is also the optimal "root set" for $OPT_k^T(U)$).

Thus, we only need to consider Case 3. Let us consider the optimal clustering with center set be $C^* = \{c_1^*, c_2^*, ..., c_k^*\}$ (each center c_j^* is a leaf of subtree whose root be c_j'), and S_j' be the leaves assigned to c_j^*. Let S_j denote the set of leaves in S_j' whose distance to c_j^* is strictly smaller than its distance to any centers in C_1. Let P_j denote the union of paths between leaves of S_j to its closest center in C_1. Let v_j'' be the nodes in P_j with highest level satisfying $T(v_j'') \cap C_1 = \emptyset$. The score of v_j'' is $2^{h_{v_j''}} N(v_j'')$. That means the swap with a center v_j' into C_1 can only reduce $4 \cdot 2^{h_{v_j''}} N(v_j'')$ to $cost_k^{T\prime}(U)$ (the tree distance between any leaf in S_j and its closest center in C_1 is at most $4 \cdot 2^{h_{v_j''}}$). We just use v_j' to represent v_j'' for later part of this proof for simplicity. By our reasoning, summing all the swaps over $C_1^* \setminus C_1$ gives

$$cost_k^{T\prime}(U) - OPT_k^T(U) \leq 4 \sum_{v_j' \in C_1^* \setminus C_1} N_{v_j'} 2^{h_{v_j'}},$$

$$OPT_k^T(U) \geq \sum_{v_i \in C_1 \setminus C_1^*} N_{v_i} 2^{h_{v_i}}.$$

Also, based on our discussion on Case 1, it holds that

$$N_{v_j'} 2^{h_{v_j'}} - N_{v_i} 2^{h_{v_i}} \leq 0.$$

Summing them together, we have $cost_k^{T\prime}(U) \leq 5OPT_k^T(U)$. □

B.2 Proof of Lemma 3.3

Proof. Since the subtrees in C_1 are disjoint, it suffices to consider one subtree with root v. With a little abuse of notation, let $cost_1^{T\prime}(v, U)$ denote the optimal k-median cost within the point set $T(v)$ with one center in 2-HST:

$$cost_1^{T\prime}(v, U) = \min_{x \in T(v)} \sum_{y \in T(v)} \rho^T(x, y), \tag{7}$$

which is the optimal cost within the subtree. Suppose v has more than one children $u, w, ...$, otherwise the optimal center is clear. Suppose the optimal solution of $cost_1^{T\prime}(v, U)$ chooses a leaf node in $T(u)$, and our HST initialization algorithm picks a leaf of $T(w)$. If $u = w$, then HST chooses the optimal one where the argument holds trivially. Thus, we consider $u \neq w$. We have the following two observations:

- Since one needs to pick a leaf of $T(u)$ to minimize $cost_1^{T\prime}(v, U)$, we have $cost_1^{T\prime}(v, U) \geq \sum_{x \in ch(v), x \neq u} N_x \cdot 2^{h_x}$ where $ch(u)$ denotes the children nodes of u.
- By our greedy strategy, $cost_1^T(v, U) \leq \sum_{x \in ch(u)} N_x \cdot 2^{h_x} \leq cost_1^{T\prime}(v, U) + N_u \cdot 2^{h_u}$.

Since $h_u = h_w$, we have

$$2^{h_u} \cdot (N_u - N_w) \leq 0,$$

since our algorithm picks subtree roots with highest scores. Then we have $cost_1^T(v, U) \leq cost_1^{T\prime}(v, U) + N_w \cdot 2^{h_w} \leq 2cost_1^{T\prime}(v, U)$. Since the subtrees in C_1 are disjoint, the union of centers for $OPT_1^T(v, U)$, $v \in C_1$ forms the optimal centers with size k. Note that, for any data point $p \in U \setminus C_1$, the tree distance $\rho^T(p, f)$ for $\forall f$ that is a leaf node of $T(v)$, $v \in C_1$ is the same. That is, the choice of leaf in $T(v)$ as the center does not affect the k-median cost under 2-HST metric. Therefore, union bound over k subtree costs completes the proof. □

B.3 PROOF OF THEOREM 4.2

Similarly, we prove the error in general metric by first analyzing the error in 2-HST metric. Then the result follows from Lemma 3.1. Let $cost_k^T(D)$, $cost_k^{T\prime}(D)$ and $OPT_k^T(D)$ be defined analogously to (3), (4) and (5), where "$y \in U$" in the summation is changed into "$y \in D$" since D is the demand set. That is,

$$cost_k^T(D) = \sum_{y \in D} \min_{x \in C_0} \rho^T(x, y), \tag{8}$$

$$cost_k^{T\prime}(D, C_1) = \min_{|F \cap T(v)|=1, \forall v \in C_1} \sum_{y \in D} \min_{x \in F} \rho^T(x, y), \tag{9}$$

$$OPT_k^T(D) = \min_{F \subset D, |F|=k} \sum_{y \in D} \min_{x \in F} \rho^T(x, y) \equiv \min_{C_1'} \ cost_k^{T\prime}(D, C_1'). \tag{10}$$

We have the following.

Lemma B.2. $cost_k^T(D) \leq 10OPT_k^T(D) + 10ck\epsilon^{-1}\triangle \log n$ *with probability* $1 - 4k/n^c$.

Proof. The result follows by combining the following Lemma B.4, Lemma B.5, and applying union bound. □

Lemma B.3. *For any node v in T, with probability* $1 - 1/n^c$, $|\hat{N}_v \cdot 2^{h_v} - N_v \cdot 2^{h_v}| \leq c\epsilon^{-1}\triangle \log n$.

Proof. Since $\hat{N}_v = N_v + Lap(2^{(L-h_v)/2}/\epsilon)$, we have

$$Pr[|\hat{N}_v - N_v| \geq x/\epsilon] = exp(-x/2^{(L-h_v)}).$$

As $L = \log \triangle$, we have

$$Pr[|\hat{N}_v - N_v| \geq x\triangle/(2^{h_v}\epsilon)] \leq exp(-x).$$

Hence, for some constant $c > 0$,

$$Pr[|\hat{N}_v \cdot 2^{h_v} - N_v \cdot 2^{h_v}| \leq c\epsilon^{-1}\triangle \log n] \geq 1 - exp(-c \log n) = 1 - 1/n^c.$$

□

Lemma B.4 (DP Subtree Search)**.** *With probability* $1 - 2k/n^c$, $cost_k^{T\prime}(D) \leq 5OPT_k^T(D) + 4ck\epsilon^{-1}\triangle \log n$.

Proof. The proof is similar to that of Lemma 3.2. Consider the intermediate output of Algorithm 2, $C_1 = \{v_1, v_2, ..., v_k\}$, which is the set of roots of the minimal disjoint subtrees each containing exactly one output center C_0. Suppose one of the optimal "root set" that minimizes (4) is $C_1^* = \{v_1', v_2', ..., v_k'\}$. Assume $C_1 \neq C_1^*$. By the same argument as the proof of Lemma 3.2, we consider for some i, j such that $v_i \neq v_j'$, where v_i is not a descendent of v_j' and v_j' is either a descendent of v_i. By the construction of C_1, we know that $score(v_j') \leq \min\{score(v_i), i = 1, ..., k\}$ when $v_j' \in C_1^* \setminus C_1$. Consider the swap between C_1 and C_1^*. By the definition of tree distance, we have $OPT_k^T(U) \geq \sum_{v_i \in C_1 \setminus C_1^*} N_{v_i} 2^{h_{v_i}}$, since $\{T(v_i), v_i \in C_1 \setminus C_1^*\}$ does not contain any center of the optimal solution determined by C_1^* (which is also the optimal "root set" for OPT_k^T). Let us consider the optimal clustering with center set be $C^* = \{c_1^*, c_2^*, ..., c_k^*\}$ (each center c_j^* is a leaf of subtree whose root be c_j'), and S_j' be the leaves assigned to c_j^*. Let S_j denote the set of leaves in S_j' whose distance to c_j^* is strictly smaller than its distance to any centers in C_1. Let P_j denote the union of paths between leaves of S_j to its closest center in C_1. Let v_j'' be the nodes in P_j with highest level satisfying $T(v_j'') \cap C_1 = \emptyset$. The score of v_j'' is $2^{h_{v_j''}} N(v_j'')$. That means the swap with a center v_j' into C_1 can only reduce $4 \cdot 2^{h_{v_j''}} N(v_j'')$ to $cost_k^{T\prime}(U)$ (the tree distance between any leaf in S_j and its closest center in C_1 is at most $4 \cdot 2^{h_{v_j''}}$). We just use v_j' to represent v_j'' for later part of this proof for simplicity. Summing all the swaps over $C_1^* \setminus C_1$, we obtain

$$cost_k^{T\prime}(U) - OPT_k^T(U) \leq 4 \sum_{v_j' \in C_1^* \setminus C_1} N_{v_j'} 2^{h_{v_j'}},$$

$$OPT_k^T(U) \geq \sum_{v_i \in C_1 \setminus C_1^*} N_{v_i} 2^{h_{v_i}}.$$

Applying union bound with Lemma B.3, with probability $1 - 2/n^c$, we have

$$N_{v'_j} 2^{h_{v'_j}} - N_{v_i} 2^{h_{v_i}} \leq 2c\epsilon^{-1} \triangle \log n.$$

Consequently, we have with probability, $1 - 2k/n^c$,

$$\begin{aligned} cost_k^{T\prime}(D) &\leq 5OPT_k^T(D) + 4c|C_1 \setminus C_1^*|\epsilon^{-1} \triangle \log n \\ &\leq 5OPT_k^T(D) + 4ck\epsilon^{-1} \triangle \log n. \end{aligned}$$

□

Lemma B.5 (DP Leaf Search)**.** *With probability $1 - 2k/n^c$, Algorithm 4 produces initial centers with $cost_k^T(D) \leq 2cost_k^{T\prime}(D) + 2ck\epsilon^{-1} \triangle \log n$.*

Proof. The proof strategy follows Lemma 3.3. We first consider one subtree with root v. Let $cost_1^{T\prime}(v, U)$ denote the optimal k-median cost within the point set $T(v)$ with one center in 2-HST:

$$cost_1^{T\prime}(v, D) = \min_{x \in T(v)} \sum_{y \in T(v) \cap D} \rho^T(x, y). \tag{11}$$

Suppose v has more than one children $u, w, ...$, and the optimal solution of $cost_1^{T\prime}(v, U)$ chooses a leaf node in $T(u)$, and our HST initialization algorithm picks a leaf of $T(w)$. If $u = w$, then HST chooses the optimal one where the argument holds trivially. Thus, we consider $u \neq w$. We have the following two observations:

- Since one needs to pick a leaf of $T(u)$ to minimize $cost_1^{T\prime}(v, U)$, we have $cost_1^{T\prime}(v, U) \geq \sum_{x \in ch(v), x \neq u} N_x \cdot 2^{h_x}$ where $ch(u)$ denotes the children nodes of u.
- By our greedy strategy, $cost_1^T(v, U) \leq \sum_{x \in ch(u)} N_x \cdot 2^{h_x} \leq cost_1^{T\prime}(v, U) + N_u \cdot 2^{h_u}$.

As $h_u = h_w$, leveraging Lemma B.3, with probability $1 - 2/n^c$,

$$\begin{aligned} 2^{h_u} \cdot (N_u - N_w) &\leq 2^{h_u}(\hat{N}_u - \hat{N}_w) + 2c\epsilon^{-1} \triangle \log n \\ &\leq 2c\epsilon^{-1} \triangle \log n. \end{aligned}$$

since our algorithm picks subtree roots with highest scores. Then we have $cost_1^T(v, D) \leq cost_k^{T\prime}(v, D) + N_w \cdot 2^{h_u} + 2c\epsilon^{-1} \triangle \log n \leq 2cost_k^{T\prime}(v, D) + 2c\epsilon^{-1} \triangle \log n$ with high probability. Lastly, applying union bound over the disjoint k subtrees gives the desired result. □

B.4 Proof of Theorem 4.3

Proof. The privacy analysis is straightforward, by using the composition theorem (Theorem B.1). Since the sensitivity of $cost(\cdot)$ is $\triangle$, in each swap iteration the privacy budget is $\epsilon/2(T+1)$. Also, we spend another $\epsilon/2(T+1)$ privacy for picking a output. Hence, the total privacy is $\epsilon/2$ for local search. Algorithm 4 takes $\epsilon/2$ DP budget for initialization, so the total privacy is ϵ.

The analysis of the approximation error follows from Gupta et al. (2010), where the initial cost is reduced by our private HST method. We need the following two lemmas.

Lemma B.6 (Gupta et al. (2010))**.** *Assume the solution to the optimal utility is unique. For any output $o \in O$ of $2\triangle\epsilon$-DP exponential mechanism on dataset D, it holds for $\forall t > 0$ that*

$$Pr[q(D, o) \leq \max_{o \in O} q(D, o) - (\ln |O| + t)/\epsilon] \leq e^{-t},$$

where $|O|$ is the size of the output set.

Lemma B.7 (Arya et al. (2004))**.** *For any set $F \subseteq D$ with $|F| = k$, there exists some swap (x, y) such that the local search method admits*

$$cost_k(F, D) - cost_k(F - \{x\} + \{y\}, D) \geq \frac{cost_k(F, D) - 5OPT(D)}{k}.$$

From Lemma B.7, we know that when $cost_k(F_i, D) > 6OPT(D)$, there exists a swap (x, y) s.t.

$$cost_k(F_i - \{x\} + \{y\}, D) \leq (1 - \frac{1}{6k})cost_k(F_i, D).$$

At each iteration, there are at most n^2 possible outputs (i.e., possible swaps), i.e., $|O| = n^2$. Using Lemma B.6 with $t = 2\log n$, for $\forall i$,

$$Pr[cost_k(F_{i+1}, D) \geq cost_k(F^*_{i+1}, D) + 4\frac{\log n}{\epsilon'}] \geq 1 - 1/n^2,$$

where $cost_k(F^*_{i+1}, D)$ is the minimum cost among iteration $1, 2, ..., t+1$. Hence, we have that as long as $cost(F_i, D) > 6OPT(D) + \frac{24k\log n}{\epsilon'}$, the improvement in cost is at least by a factor of $(1 - \frac{1}{6k})$. By Theorem 4.2, we have $cost_k(F_1, D) \leq C(\log n)(6OPT(D) + 6k\triangle\log n/\epsilon)$ for some constant $C > 0$. Let $T = 6Ck\log\log n$. We have that

$$\begin{aligned} E[cost(F_i, D)] &\leq (6OPT(D) + 6k\epsilon^{-1}\triangle\log n)C(\log n)(1 - 1/6k)^{6Ck\log\log n} \\ &\leq 6OPT(D) + 6k\epsilon^{-1}\triangle\log n \leq 6OPT(D) + \frac{24k\log n}{\epsilon'}. \end{aligned}$$

Therefore, with probability at least $(1 - T/n^2)$, there exists an $i <= T$ s.t. $cost(F_i, D) \leq 6OPT(D) + \frac{24k\log n}{\epsilon'}$. Then by using the Lemma B.7, one will pick an F_j with additional additive error $4\ln n/\epsilon'$ to the $\min\{cost(F_j, D), j = 1, 2, ..., T\}$ with probability $1 - 1/n^2$. Consequently, we know that the expected additive error is

$$24k\triangle\log n/\epsilon' + 4\log n/\epsilon' = O(\epsilon^{-1}k^2\triangle(\log\log n)\log n),$$

with probability $1 - 1/poly(n)$.

□

C Extend HST Initialization to k-Means

Naturally, our HST method can also be applied to k-means clustering problem. In this section, we extend the HST to k-means and provide some brief analysis similar to k-median. We present the analysis in the non-private case, which can then be easily adapted to the private case. Define the following costs for k-means.

$$cost^T_{km}(U) = \sum_{y\in U}\min_{x\in C_0}\rho^T(x, y)^2, \tag{12}$$

$$cost^T_{km}{}'(U, C_1) = \min_{|F\cap T(v)|=1, \forall v\in C_1}\sum_{y\in U}\min_{x\in F}\rho^T(x, y)^2, \tag{13}$$

$$OPT^T_{km}(U) = \min_{F\subset U, |F|=k}\sum_{y\in U}\min_{x\in F}\rho^T(x, y)^2 \equiv \min_{C'_1} cost^T_{km}{}'(U, C'_1). \tag{14}$$

For simplicity, we will use $cost^T_{km}{}'(U)$ to denote $cost^T_{km}{}'(U, C_1)$ if everything is clear from context. Here, OPT^T_{km} (14) is the cost of the global optimal solution with 2-HST metric.

Lemma C.1 (Subtree search)**.** $cost^T_{km}{}'(U) \leq 17OPT^T_{km}(U)$.

Proof. The analysis is similar with the proof of Lemma 3.2. Thus, we only need stated the difference. Let us just use some notations the same as in Lemma 3.2 here. Let us consider the clustering with center set be $C^* = \{c^*_1, c^*_2, ..., c^*_k\}$ (each center c^*_j is a leaf of subtree whose root be c'_j), and S'_j be the leaves assigned to c^*_j in optimal k-means clustering in tree metric. Let S_j denote the set of leaves in S'_j whose distance to c^*_j is strictly smaller than its distance to any centers in C_1. Let P_j denote the union of paths between leaves of S_j to its closest center in C_1. Let v''_j be the nodes in P_j with highest level satisfying $T(v''_j) \cap C_1 = \emptyset$. The score of v''_j is $2^{h_{v''_j}}N(v''_j)$. That means the swap with a center v'_j into C_1 can only reduce $(4\cdot 2^{h_{v''_j}})^2N(v''_j)$ to $cost^T_{km}{}'(U)$. We just use v'_j to

represent v_j'' for later part of this proof for simplicity. By our reasoning, summing all the swaps over $C_1^* \setminus C_1$ gives

$$cost_{km}^T{}'(U) - OPT_{km}^T(U) \leq \sum_{v_j' \in C_1^* \setminus C_1} N_{v_j'} \cdot (4 \cdot 2^{h_{v_j'}})^2,$$

$$OPT_{km}^T(U) \geq \sum_{v_i \in C_1 \setminus C_1^*} N_{v_i} (2^{h_{v_i}})^2.$$

Also, based on our discussion on Case 1, it holds that

$$N_{v_j'} 2^{h_{v_j'}} - N_{v_i} 2^{h_{v_i}} \leq 0.$$

Summing them together, we have $cost_{km}^T{}'(U) \leq 17 OPT_{km}^T(U)$. □

Next, we show that the greedy leaf search strategy (Algorithm 3) only leads to an extra multiplicative error of 2.

Lemma C.2 (Leaf search)**.** $cost_{km}^T(U) \leq 2 cost_{km}^T{}'(U)$.

Proof. Since the subtrees in C_1 are disjoint, it suffices to consider one subtree with root v. With a little abuse of notation, let $cost_1^{T\prime}(v, U)$ denote the optimal k-means cost within the point set $T(v)$ with one center in 2-HST:

$$cost_1^{T\prime}(v, U) = \min_{x \in T(v)} \sum_{y \in T(v)} \rho^T(x, y)^2, \tag{15}$$

which is the optimal cost within the subtree. Suppose v has more than one children $u, w, ...$, otherwise the optimal center is clear. Suppose the optimal solution of $cost_1^{T\prime}(v, U)$ chooses a leaf node in $T(u)$, and our HST initialization algorithm picks a leaf of $T(w)$. If $u = w$, then HST chooses the optimal one where the argument holds trivially. Thus, we consider $u \neq w$. We have the following two observations:

- Since one needs to pick a leaf of $T(u)$ to minimize $cost_1^{T\prime}(v, U)$, we have $cost_1^{T\prime}(v, U) \geq \sum_{x \in ch(v), x \neq u} N_x \cdot (2^{h_x})^2$ where $ch(u)$ denotes the children nodes of u.
- By our greedy strategy, $cost_1^T(v, U) \leq \sum_{x \in ch(u)} N_x \cdot (2^{h_x})^2 \leq cost_1^{T\prime}(v, U) + N_u \cdot (2^{h_u})^2$.

Since $h_u = h_w$, we have

$$2^{h_u} \cdot (N_u - N_w) \leq 0,$$

since our algorithm picks subtree roots with highest scores. Then we have $cost_1^T(v, U) \leq cost_1^{T\prime}(v, U) + N_w \cdot (2^{h_w})^2 \leq 2 cost_1^{T\prime}(v, U)$. Since the subtrees in C_1 are disjoint, the union of centers for $OPT_1^T(v, U)$, $v \in C_1$ forms the optimal centers with size k. Note that, for any data point $p \in U \setminus C_1$, the tree distance $\rho^T(p, f)$ for $\forall f$ that is a leaf node of $T(v)$, $v \in C_1$ is the same. That is, the choice of leaf in $T(v)$ as the center does not affect the k-median cost under 2-HST metric. Therefore, union bound over k subtree costs completes the proof. □

We are ready to state the error bound for our proposed HST initialization (Algorithm 2), which is a natural combination of Lemma C.1 and Lemma C.2.

Theorem C.3 (HST initialization)**.** $cost_{km}^T(U) \leq 34 OPT_{km}^T(U)$.

We have the following result based on Lemma 3.1.

Theorem C.4. *In a general metric space,*

$$E[cost_{km}(U)] = O(\min\{\log n, \log \triangle\})^2 OPT_{km}(U).$$