k-Median Clustering via Metric Embedding: Towards Better Initialization with Differential Privacy

Chenglin Fan, Ping Li, Xiaoyun Li
Cognitive Computing Lab
Baidu Research
10900 NE 8th St. Bellevue, WA 98004, USA
{fanchenglin, pingli98, lixiaoyun996}@gmail.com

Abstract

[1]In clustering, the choice of initial centers is crucial for the convergence speed of the algorithms. 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. We propose a novel and efficient search algorithm which finds initial centers that can be used subsequently for the local search algorithm. The so-called HST initialization method can produce initial centers achieving lower error than those from another popular method k-median++, also with higher efficiency when k is not too small. Our HST initialization are then extended to the setting of differential privacy (DP) to generate private initial centers. We show that the error of applying DP local search followed by our private HST initialization improves prior results on the approximation error, and approaches the lower bound within a small factor. Experiments demonstrate the effectiveness of our proposed methods.

1 Introduction

Clustering is an important classic problem in unsupervised learning that has been widely studied in statistics, data mining, machine learning, network analysis, etc. (Punj and Stewart, 1983; Dhillon and Modha, 2001; Banerjee et al., 2005; Berkhin, 2006; Abbasi and Younis, 2007). The objective of clustering is to divide a set of data points into clusters, such that items within the same cluster exhibit similarities, while those in different clusters distinctly differ. 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 a cost function defined for C on dataset D. $OPT_k(D)$ is the cost of optimal clustering 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 general frameworks for clustering. 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,

[1]This work was initially submitted in 2020 and was made public in 2021.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

typically in the (continuous) 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 generally 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 (i.e., $\gamma = O(1)$) to the optimal solution (Arya et al., 2004). For general metric spaces, the state of the art approximation ratio is 2.675 for k-median Byrka et al. (2015) and 6.357 for k-means Ahmadian et al. (2017).

Initialization of cluster centers. It is well-known that the performance of clustering can be highly sensitive to initialization. If clustering starts with good initial centers with small approximation error, the algorithm may use fewer iterations to find a better solution. The k-median++ algorithm (Arthur and Vassilvitskii, 2007) iteratively selects k data points as initial centers, favoring distant points in a probabilistic way, such that the initial centers tend to be well spread over the data points (i.e., over different clusters). The produced initial centers are proved to have $O(\log k)$ multiplicative error. Follow-up works further improved its efficiency and scalability, e.g., Bahmani et al. (2012); Bachem et al. (2016); Lattanzi and Sohler (2019); Choo et al. (2020); Cohen-Addad et al. (2021); Grunau et al. (2023); Fan et al. (2023). In this work, we propose a new initialization framework, called HST initialization, which is built upon a novel search algorithm on metric embedding trees constructed from the data. Our method achieves improved approximation error compared with k-median++. Moreover, importantly, our initialization scheme can be conveniently combined with the notion of differential privacy (DP) to protect the data privacy.

Clustering with Differential Privacy. The concept of differential privacy (Dwork, 2006; McSherry and Talwar, 2007) has been popular to rigorously define and resolve the problem of keeping useful information for machine learning models, while protecting privacy for each individual. DP has been adopted to a variety of algorithms and tasks, such as regression, classification, principle component analysis, graph distance release, matrix completion, optimization, and deep learning (Chaudhuri and Monteleoni, 2008; Chaudhuri et al., 2011; Abadi et al., 2016; Ge et al., 2018; Wei et al., 2020; Dong et al., 2022; Fan and Li, 2022; Fan et al., 2022; Fang et al., 2023; Li and Li, 2023a,b). Private k-means clustering has also been widely studied, e.g., Feldman et al. (2009); Nock et al. (2016); Feldman et al. (2017), mostly in the continuous Euclidean space. 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 private Euclidean k-means error ais further improved by Stemmer and Kaplan (2018), with more advanced candidate set selection. Huang and 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); Ghazi et al. (2020) 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 the Euclidean space where n is the sample size. In discrete space, Gupta et al. (2010) proposed a private method for the classical local search heuristic, which applies to both k-medians and k-means. To cast privacy on each swapping step, the authors applied the exponential mechanism of McSherry and 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 the point set. In this work, we will show that our proposed HST initialization can improve the DP local search for k-median of Gupta et al. (2010) in terms of both approximation error and efficiency. Stemmer and Kaplan (2018); Jones et al. (2021) proposed (ϵ, δ)-differentially private solution also with constant multiplicative error but smaller additive error.

The main contributions of this work include the following:

- We introduce the Hierarchically Well-Separated Tree (HST) as an initialization tool for the k-median clustering problem. We design an efficient sampling strategy to select the initial center set from the tree, with an approximation factor $O(\log \min\{k, \triangle\})$ in the non-private setting, which is $O(\log \min\{k, d\})$ when $\log \triangle = O(\log d)$. This improves the $O(\log k)$ error of k-median++. Moreover, the complexity of our HST based method can be smaller than that of k-median++ when the number of clusters k is not too small ($k \geq \log n$), which is a common scenario in practical applications.

- We propose a differentially private version of HST initialization under the setting of Gupta et al. (2010) in discrete metric space. The so-called DP-HST algorithm finds initial centers with $O(\log n)$ multiplicative error and $O(\epsilon^{-1}\triangle k^2 \log^2 n)$ additive error. Moreover, running DP-local search starting from this initialization 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 of DP k-median (Gupta et al., 2010) within a small $O(k\log\log n)$ factor. This is the first clustering initialization method with ϵ-differential privacy guarantee and improved error rate in general metric space.
- 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 Background and Setup

The definition of differential privacy (DP) is as follows.

Definition 2.1 (Differential Privacy (DP) (Dwork, 2006))**.** *If for any two adjacent datasets D and D' with symmetric difference of size one and any $O \subset Range(\mathbb{A})$, an algorithm $\mathbb{A}$ with map f 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 (ϵ-DP).

Intuitively, DP requires that after removing any data point from D (e.g., a node in a graph), the output of D' should not be too different from that of the original dataset D. The *Laplace mechanism* adds Laplace$(\eta(f)/\epsilon)$ noise to the output where $\eta(f) = \sup_{|D-D'|=1} |f(D) - f(D')|$ is the sensitivity of f, which is known to achieve ϵ-DP. The *exponential mechanism* is also a tool for many DP algorithms with discrete outputs. Let O be the output domain. 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 $P[\mathbb{A}(D) = o] \propto \exp(\frac{\epsilon q(D,o)}{2\eta(q)})$. Both mechanisms will be used in our paper.

2.1 k-Median Clustering and Local Search

In this paper, we follow the classic problem setting in the metric clustering literature, e.g. Arya et al. (2004); Gupta et al. (2010). Specifically, the definitions of metric k-median clustering problem (DP and non-DP) are stated as follow.

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

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

Let $D \subseteq U$ be a set of "demand points". The goal of DP k-median is to minimize

$$\textit{DP k-median:} \quad cost_k(F, D) = \sum_{v \in D} \min_{f \in F} \rho(v, f), \tag{2}$$

and the output F is required to be ϵ-differentially private with respect to D. We may drop "F" and use "$cost_k(U)$" or "$cost_k(D)$" if there is no risk of ambiguity.

Note that in Definition 2.2, our aim is to protect the privacy of a subset $D \subset U$. To better understand the motivation and application scenario, we provide a real-world example as below.

Example 2.3. *Consider U to be the universe of all users in a social network (e.g., Facebook, LinkedIn, etc.). Each user (account) has some public information (e.g., name, gender, interests, etc.), but also has some private personal data that can only be seen by the data server. Let D be a set of users grouped by some feature 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 differentially private algorithm to safely release the centers, while protecting the individuals in D from being identified (since the membership of D is private).*

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

The (non-private) 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 the new centers improve the cost the most, and if the improvement is more than $(1 - \alpha/k)$ for some $\alpha > 0$ [2]. We repeat the procedure until no such swap exists. Arya et al. (2004) showed that the output center set F achieves 5 approximation error to the optimal solution, i.e., $cost(F) \leq 5OPT$.

2.2 k-median++ Initialization

Although local search is able to find a solution with constant error, it takes $O(n^2)$ per iteration (Resende and Werneck, 2007) in expected $O(k \log n)$ steps (which gives total complexity $O(kn^2 \log n)$) when started from a random center set, which would be slow for large datasets. Indeed, we do not need such complicated/meticulous algorithm to reduce the cost at the beginning, i.e., when the cost is large. To accelerate the process, efficient initialization methods find a "roughly" good center set as the starting point for local search. In the paper, we compare our new initialization scheme mainly with a popular (and perhaps the most well-known) initialization method, the k-median++ (Arthur and Vassilvitskii, 2007) [3] as presented in Algorithm 2. The output centers C by k-median++ achieve $O(\log k)$ approximation error with time complexity $O(nk)$. Starting from the initialization, we only need to run $O(k \log \log k)$ steps of the computationally heavy local search to reach a constant error solution. Thus, initialization may greatly improve the clustering efficiency.

Algorithm 2: k-median++ initialization (Arthur and Vassilvitskii, 2007)

Input: Data points U, number of centers k
Randomly pick a point $c_1 \in U$ and set $F = \{c_1\}$
for $i = 2, ..., k$ **do**
  Select $c_i = u \in U$ with probability $\frac{\rho(u,F)}{\sum_{u' \in U} \rho(u',F)}$
  $F = F \cup \{c_i\}$
Output: k-median++ initial center set F

3 Initialization via Hierarchical 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 (1). 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 levels in a descending order down the tree. We use h_v to denote the level of v, and let n_i be the 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

[2] Arya et al. (2004) chooses the first swap that improves the cost by $(1 - \alpha/k)$, instead of picking the smallest cost among all such swaps as in Algorithm 1. Both methods are valid and yield the same approximation error.

[3] In the original paper of Arthur and Vassilvitskii (2007), the k-median++ algorithm was called D^1-sampling.

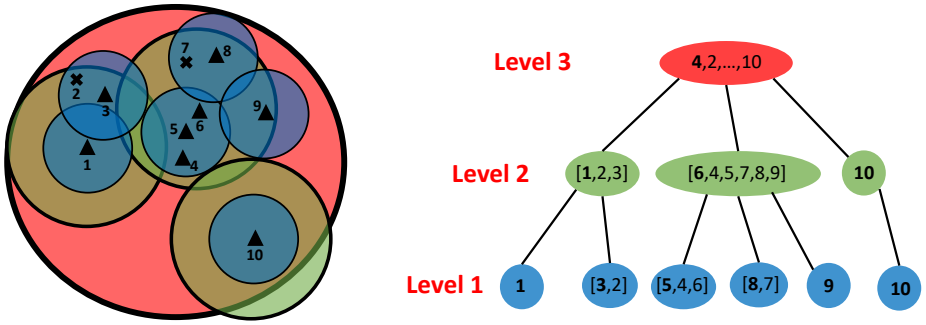

Figure 1: An example of a 3-level padded decomposition and the corresponding 2-HST. **Left:** The thickness of the ball represents the level. The colors correspond to different levels in the HST in the right panel. "△"'s are the center nodes of partitions (balls), and "×"'s are the non-center data points. **Right:** The 2-HST generated from the padded decomposition. Bold indices represent the centers.

is a probabilistic partition of U such that the diameter of each cluster $U_i \in U$ is at most β, i.e., $\rho(u, v) \leq \beta, \forall u, v \in U_i, i = 1, ..., k$. The formal definition of HST is given as below.

Definition 3.1. *Assume* $\min_{u,v \in U} \rho(u, v) = 1$ *and denote the diameter* $\triangle = \max_{u,v \in U} \rho(u, v)$. *An* α*-Hierarchically Well-Separated Tree (*α*-HST) with depth* L *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* $\triangle/\alpha^{L-i}$.

Our analysis will consider $\alpha = 2$-HST for conciseness, since α only affects the constants in our theoretical analysis. Figure 1 is an example 2-HST (right panel) with $L = 3$ levels, along with its underlying padded decompositions (left panel). Using Algorithm 3, a 2-HST can be built as follows: 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 (leaf), or a pre-specified tree depth is reached. It is worth mentioning that, Blelloch et al. (2017) proposed an efficient HST construction in $O(m \log n)$ time, where n and m are the number of nodes and edges in a graph, respectively. Therefore, the construction of HST can be very efficient in practice.

The first step of our method is to embed the data points into an HST (see Algorithm 4). Next, we will describe our new strategy to search for the initial centers on the tree (w.r.t. the tree metric). Before moving on, it is worth mentioning that, there are polynomial time algorithms for computing an *exact* k-median solution in the tree metric (Tamir (1996); Shah (2003)). However, the dynamic programming algorithms have high complexity (e.g., $O(kn^2)$), making them unsuitable for the purpose of fast initialization. Moreover, it is unknown how to apply them effectively to the private case. The three key merits of our new algorithm are: (1) It is more efficient than k-median++ when k is not too small, which is a very common scenario in practice; (2) It achieves $O(1)$ approximation error in the tree metric; (3) It can be easily extended to incorporating differential privacy (DP).

Algorithm 3: 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 T

3.2 HST Initialization Algorithm

Let $L = \log \Delta$ and suppose T is a level-L 2-HST in (U, ρ), where we assume L is an integer. 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 search strategy for the initial centers, NDP-HST initialization ("NDP" stands for "Non-Differentially Private"), is presented in Algorithm 4 with two phases.

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.

Algorithm 4: NDP-HST initialization

Input: U, $\triangle$, k
Initialization: $L = \log \triangle$, $C_0 = \emptyset$, $C_1 = \emptyset$
Call Algorithm 3 to build a level-L 2-HST T using 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 to C_1
  for *each* $v \in C_1$ **do**
    $C_1 = C_1 \setminus \{v\}$, if $\exists\, v' \in C_1$ such that v' is a descendant of v
$C_0 =$ FIND-LEAF(T, C_1)
Output: Initial center set $C_0 \subseteq U$

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 5 ("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).

The complexity of our search algorithm is given as follows. All proofs are placed in Appendix B.

Proposition 3.2 (Complexity)**.** *Algorithm 4 takes $O(dn \log n)$ time in the Euclidean space.*

Remark 3.3 (Comparison with k-median++)**.** *The complexity of of k-median++ is $O(dnk)$ in the Euclidean space (Arthur and Vassilvitskii, 2007). Our algorithm would be faster when $k > \log n$, which is a common scenario. Similar comparison also holds for general metrics.*

3.3 Approximation Error of HST Initialization

We provide the error analysis of our algorithm. Firstly, we show that the initial center set produced by NDP-HST 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 T, where $d_T(x, y)$ is the tree distance between nodes x and y in T. By Definition 3.1 and since $\triangle = 2^L$, in the analysis we assume

Algorithm 5: 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 $C_0 \subseteq U$

equivalently that the edge weight of the i-th level is 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 adapted to the general metrics by the following qwll-known result.

Lemma 3.4 (Bartal (1996)). *In a metric space (U, ρ) with $|U| = n$ and diameter $\triangle$, it holds that $\forall x, y \in U$, $E[\rho^T(x, y)] = O(\min\{\log n, \log \triangle\})\rho(x, y)$. In the Euclidean space $\mathbb{R}^d$, $E[\rho^T(x, y)] = O(d)\rho(x, y)$, $\forall x, y \in U$.*

Recall C_0, C_1 from Algorithm 4. We define

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

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

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

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 the 2-HST metric. The last equivalence in (5) holds because the optimal centers can always be located in k disjoint subtrees, as each leaf only contains one point. (3) is the k-median cost with 2-HST metric of the output C_0 of Algorithm 4. (4) is the optimal cost after the subtrees are chosen. That is, it represents the minimal cost to pick one center from each subtree in C_1. We first bound the error of the subtree search step and the leaf search step, respectively.

Lemma 3.5 (Subtree search). $cost_k^{T\prime}(U) \le 5OPT_k^T(U)$.

Lemma 3.6 (Leaf search). $cost_k^T(U) \le 2cost_k^{T\prime}(U)$.

Combining Lemma 3.5 and Lemma 3.6, we obtain:

Theorem 3.7 (2-HST error). *Running Algorithm 4, we have $cost_k^T(U) \le 10OPT_k^T(U)$.*

Thus, HST-initialization produces an $O(1)$ approximation to the optimal cost in the 2-HST metric. Define $cost_k(U)$ as (1) 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 based on Lemma 3.4.

Theorem 3.8. *In the general metric space, the expected k-median cost of NDP-HST (Algorithm 4) is $E[cost_k(U)] = O(\min\{\log n, \log \triangle\})OPT_k(U)$.*

Remark 3.9. *In the Euclidean space, Makarychev et al. (2019) showed that using $O(\log k)$ random projections suffices for k-median to achieve $O(1)$ error. Thus, if $\log \triangle = O(\log d)$, by Lemma 3.4, HST initialization is able to achieve $O(\log(\min\{d, k\}))$ error, which is better than $O(\log k)$ of k-median++ (Arthur and Vassilvitskii, 2007) when d is small.*

NDP-HST Local Search. We are interested in the approximation quality of standard local search (Algorithm 1), when the initial centers are produced by our NDP-HST.

Theorem 3.10. *When initialized by NDP-HST, local search achieves $O(1)$ approximation error in expected $O(k \log\log \min\{n, \triangle\})$ number of iterations for input in general metric space.*

We remark that the initial centers found by NDP-HST can be used for k-means clustering analogously. For general metrics, $E[cost_{km}(U)] = O(\min\{\log n, \log \triangle\})^2 OPT_{km}(U)$ where $cost_{km}(U)$ is the optimal k-means cost. See Appendix C for more detaileds.

4 HST Initialization with Differential Privacy

In this section, we consider initialization and clustering with differential privacy (DP). Recall (2) that in this problem, 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, simply running initialization algorithms on U would

Algorithm 6: DP-HST initialization

Input: $U, D, \triangle, k, \epsilon$
Build a level-L 2-HST 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 4: find C_1; $C_0 =$ FIND-LEAF(C_1)
Output: Private initial center set $C_0 \subseteq U$

Algorithm 7: 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 6 with privacy $\epsilon/2$
Set parameter $\epsilon' = \frac{\epsilon}{4\triangle(T+1)}$
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

preserve the privacy of D. However, 1) this might be too expensive; 2) 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 proposed HST initialization can be easily combined with differential privacy and at the same time contains useful information about the demand set D, leading to improved approximation error (Theorem 4.3). 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 (DP-HST) is similar to the non-private Algorithm 4. 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 6. The DP guarantee of this initialization scheme is straightforward by the composition theory (Dwork, 2006).

Theorem 4.1. *Algorithm 6 is ϵ-differentially private.*

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

Theorem 4.2. *Algorithm 6 finds initial centers such that*

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

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 7. After DP-HST initialization, the DP local search procedure follows Gupta et al. (2010) using the exponential mechanism.

Theorem 4.3. *Algorithm 7 achieves ϵ-differential privacy. The output centers achieve $cost_k(D) \leq 6OPT_k(D) + O(\epsilon^{-1}k^2\triangle(\log\log n)\log n)$ in $O(k \log\log n)$ iterations with probability $(1 - \frac{1}{poly(n)})$.*

In prior literature, 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 $O(k \log\log n)$ (see Appendix A for an empirical justification). Notably, 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 $O(k \log\log n)$. To our knowledge, Theorem 4.3 is the first result in the literature to improve the error of DP local search in general metric space.

5 Numerical Results

5.1 Datasets and Algorithms

Discrete Euclidean space. Following previous work, 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). Our construction of graphs follows a similar approach as the synthetic *pmedinfo* graphs provided by the popular OR-Library (Beasley, 1990). The metric ρ for this experiment is the (weighted) shortest path distance. 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 with weight drawn from uniform $[0, 1]$. For every 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 Appendix A for example graphs). For this task, U has 3000 nodes, and the private set D (500 nodes) is chosen using the similar "balanced" and "imbalanced" approachs as described above. In the imbalanced case, we choose the demand set D randomly from only two clusters.

Algorithms. We compare the following clustering algorithms in both non-DP and DP setting: (1) **NDP-rand:** Local search with random initialization; (2) **NDP-kmedian++:** Local search with k-median++ initialization (Algorithm 2); (3) **NDP-HST:** Local search with NDP-HST initialization (Algorithm 4), as described in Section 3; (4) **DP-rand:** Standard DP local search algorithm (Gupta et al., 2010), which is Algorithm 7 with initial centers randomly chosen from U; (5) **DP-kmedian++:** DP local search with k-median++ initialization run on U; (6) **DP-HST:** DP local search with HST-initialization (Algorithm 7). For non-DP tasks, we set $L = 6$. For DP 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 $\epsilon = 1$ (comparisons/results with other T and ϵ are similar). We test $k \in \{2, 5, 10, 15, 20\}$. The average cost over T iterations is reported for robustness. All the results are averaged over 10 independent repetitions.

5.2 Results

The results on MNIST and graph data are given in Figure 2. Here we present the l_2-clustering on MNIST, and the simulated graph with $r = 1$ (clusters are less separable). The comparisons are similar for both l_1 metric on MNIST and $r = 100$ graph (see Figure 4 in Appendix A):

- From the left column, the initial centers found by HST has lower 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 right column, we also observe lower final cost of HST followed by local search in DP clustering. In the non-DP case, the final cost curves overlap, which means that despite HST offers better initial centers, local search can always find a good solution eventually.
- The advantage of DP-HST, in terms of both the initial and the final cost, is more significant when D is an imbalanced subset of U. As mentioned before, this is because our DP-HST initialization approach also privately incorporates the information of D.

To sum up, the proposed HST initialization scheme could perform better with various metrics and data patterns, in both non-private and private setting—in all cases, HST finds better initial centers with smaller cost than k-median++. HST considerably outperforms k-median++ in the private and imbalanced D setting, for MNIST with both l_2 and l_1 metric, and for graph with both $r = 100$ (highly separable) and $r = 1$ (less separable).

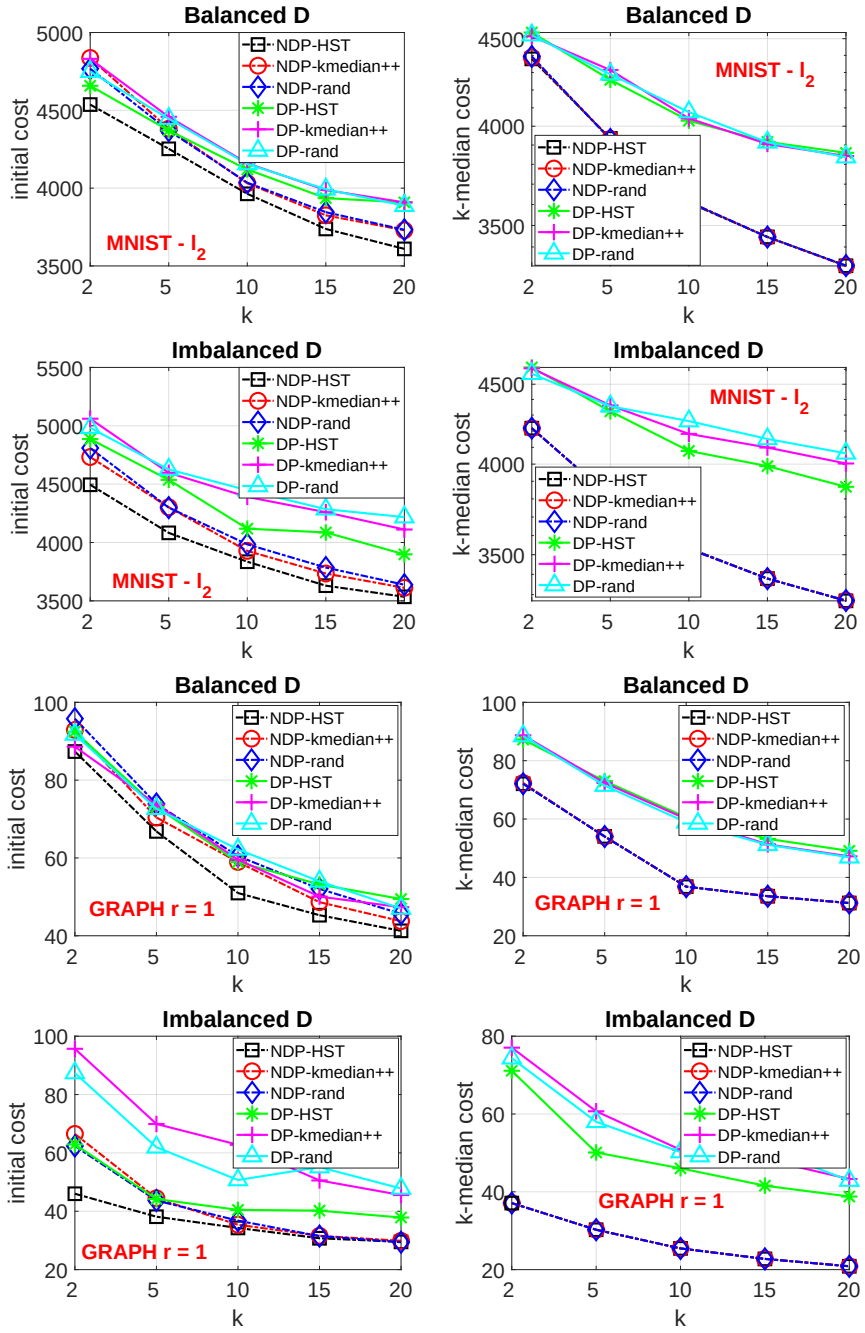

Figure 2: k-median cost on the MNIST (l_2-metric) and graph dataset ($r = 1$). **1st column:** initial cost. **2nd column:** final output cost.

6 Conclusion

We develop a new initialization framework for the k-median problem in the general metric space. Our approach, called **HST initialization**, is built upon the HST structure from metric embedding theory. We propose a novel and efficient tree search approach which provably improves the approximation error of the k-median++ method, and has lower complexity (higher efficiency) than k-median++ when k is not too small, which is a common practice. Moreover, we propose differentially private (DP) HST 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 additive error of DP local search, which is close to the theoretical lower bound within a small factor. Experiments with Euclidean metrics and graph metrics verify the effectiveness of our methods, which improve the cost of both the initial centers and the final k-median output.

References

Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 308–318, Vienna, Austria, 2016.

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

Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. In *Proceedings of the 58th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 61–72, Berkeley, CA, 2017.

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)*, pages 1027–1035, New Orleans, LA, 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 *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, pages 1459–1467, Phoenix, AZ, 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 (NIPS)*, pages 1995–2003, Lake Tahoe, NV, 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)*, pages 322–331, Sydney, Australia, 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 *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 184–193, Burlington, VT, 1996.

John E Beasley. OR-Library: distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990.

Pavel Berkhin. A survey of clustering data mining techniques. In *Grouping Multidimensional Data*, pages 25–71. Springer, 2006.

Guy E. Blelloch, Yan Gu, and Yihan Sun. Efficient construction of probabilistic tree embeddings. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 26:1–26:14, Warsaw, Poland, 2017.

Jaroslaw Byrka, Thomas W. Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median, and positive correlation in budgeted optimization. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 737–756, San Diego, CA, 2015.

Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. In *Advances in Neural Information Processing Systems (NIPS)*, pages 289–296, Vancouver, Canada, 2008.

Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. Differentially private empirical risk minimization. *J. Mach. Learn. Res.*, 12:1069–1109, 2011.

Davin Choo, Christoph Grunau, Julian Portmann, and Václav Rozhon. k-means++: few more steps yield constant approximation. In *International Conference on Machine Learning*, pages 1909–1917, 2020.

Vincent Cohen-Addad, Silvio Lattanzi, Ashkan Norouzi-Fard, Christian Sohler, and Ola Svensson. Parallel and efficient hierarchical k-median clustering. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 20333–20345, virtual, 2021.

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

Jinshuo Dong, Aaron Roth, and Weijie J Su. Gaussian differential privacy. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(1):3–37, 2022.

Cynthia Dwork. Differential privacy. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP), Part II*, pages 1–12, Venice, Italy, 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.

Chenglin Fan and Ping Li. Distances release with differential privacy in tree and grid graph. In *IEEE International Symposium on Information Theory (ISIT)*, pages 2190–2195, 2022.

Chenglin Fan, Ping Li, and Xiaoyun Li. Private graph all-pairwise-shortest-path distance release with improved error rate. In *Advances in Neural Information Processing Systems (NeurIPS)*, New Orleans, LA, 2022.

Chenglin Fan, Ping Li, and Xiaoyun Li. LSDS++ : Dual sampling for accelerated k-means++. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 9640–9649, Honolulu, HI, 2023.

Huang Fang, Xiaoyun Li, Chenglin Fan, and Ping Li. Improved convergence of differential private sgd with gradient clipping. In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR)*, Kigali, Rwanda, 2023.

Dan Feldman, Amos Fiat, Haim Kaplan, and Kobbi Nissim. Private coresets. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 361–370, Bethesda, MD, 2009.

Dan Feldman, Chongyuan Xiang, Ruihao Zhu, and Daniela Rus. Coresets for differentially private k-means clustering and applications to privacy in mobile sensor networks. In *Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 3–15, Pittsburgh, PA, 2017.

Jason Ge, Zhaoran Wang, Mengdi Wang, and Han Liu. Minimax-optimal privacy-preserving sparse PCA in distributed systems. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1589–1598, Playa Blanca, Lanzarote, Canary Islands, Spain, 2018.

Badih Ghazi, Ravi Kumar, and Pasin Manurangsi. Differentially private clustering: Tight approximation ratios. In *Advances in Neural Information Processing Systems (NeurIPS)*, virtual, 2020.

Christoph Grunau, Ahmet Alper Özüdoğru, Václav Rozhoň, and Jakub Tětek. A nearly tight analysis of greedy k-means++. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1012–1070, Florence, Italy, 2023.

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)*, pages 1106–1125, Austin, TX, 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 (PODS)*, pages 395–408, Houston, TX, 2018.

Matthew Jones, Huy L. Nguyen, and Thy D. Nguyen. Differentially private clustering via maximum coverage. In *Proeedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*, pages 11555–11563, Virtual Event, 2021.

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 (CG)*, pages 10–18, Barcelona, Spain, 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)*, pages 3662–3671, Long Beach, CA, 2019.

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

Ping Li and Xiaoyun Li. Differential privacy with random projections and sign random projections. In *Advances in Neural Information Processing Systems (NeurIPS)*, New Orleans, LA, 2023a.

Xiaoyun Li and Ping Li. Differentially private one permutation hashing and bin-wise consistent weighted sampling. *arXiv preprint arXiv:2306.07674*, 2023b.

Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inf. Theory*, 28(2):129–136, 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)*, pages 1027–1038, Phoenix, AZ, 2019.

Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 94–103, Providence, RI, 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)*, pages 145–154, New York City, NY, 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.

Mauricio G. C. Resende and Renato Fonseca F. Werneck. A fast swap-based local search procedure for location problems. *Ann. Oper. Res.*, 150(1):205–230, 2007.

Rahul Shah. Faster algorithms for k-median problem on trees with smaller heights. *Technical Report*, 2003.

Uri Stemmer and Haim Kaplan. Differentially private k-means with constant multiplicative error. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5436–5446, Montréal, Canada, 2018.

Arie Tamir. An $o(pn^2)$ algorithm for the p-median and related problems on tree graphs. *Oper. Res. Lett.*, 19(2):59–64, 1996.

Keisuke Todo, Atsuyoshi Nakamura, and Mineichi Kudo. A fast approximate algorithm for k-median problem on a graph. In *Proceedings of the 15th International Workshop on Mining and Learning with Graphs (MLG)*, Anchorage, AK, 2019.

Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, and H. Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Trans. Inf. Forensics Secur.*, 15:3454–3469, 2020.

k-Median Clustering via Metric Embedding: Towards Better Initialization with Differential Privacy (Supplementary Material)

A More Details on Experiments

A.1 Examples of Graph Data

In Figure 3, we plot two example graphs (subgraphs of 50 nodes) with $r = 100$ and $r = 1$. When $r = 100$, the graph is highly separable (i.e., clusters are far from each other). When $r = 1$, the clusters are harder to be distinguished from each other.

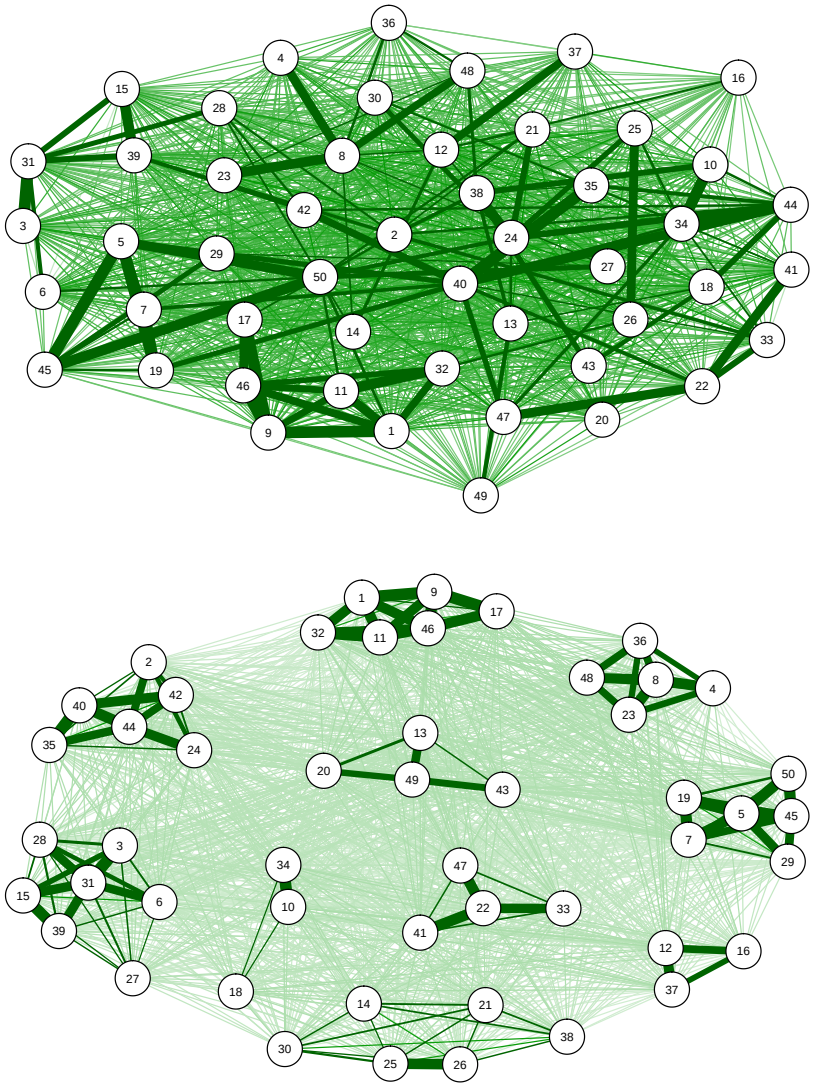

Figure 3: Example of synthetic graphs: subgraph of 50 nodes. **Upper** $r = 1$. **Bottom:** $r = 100$. Darker and thicker edged have smaller distance. When $r = 100$, the graph is more separable.

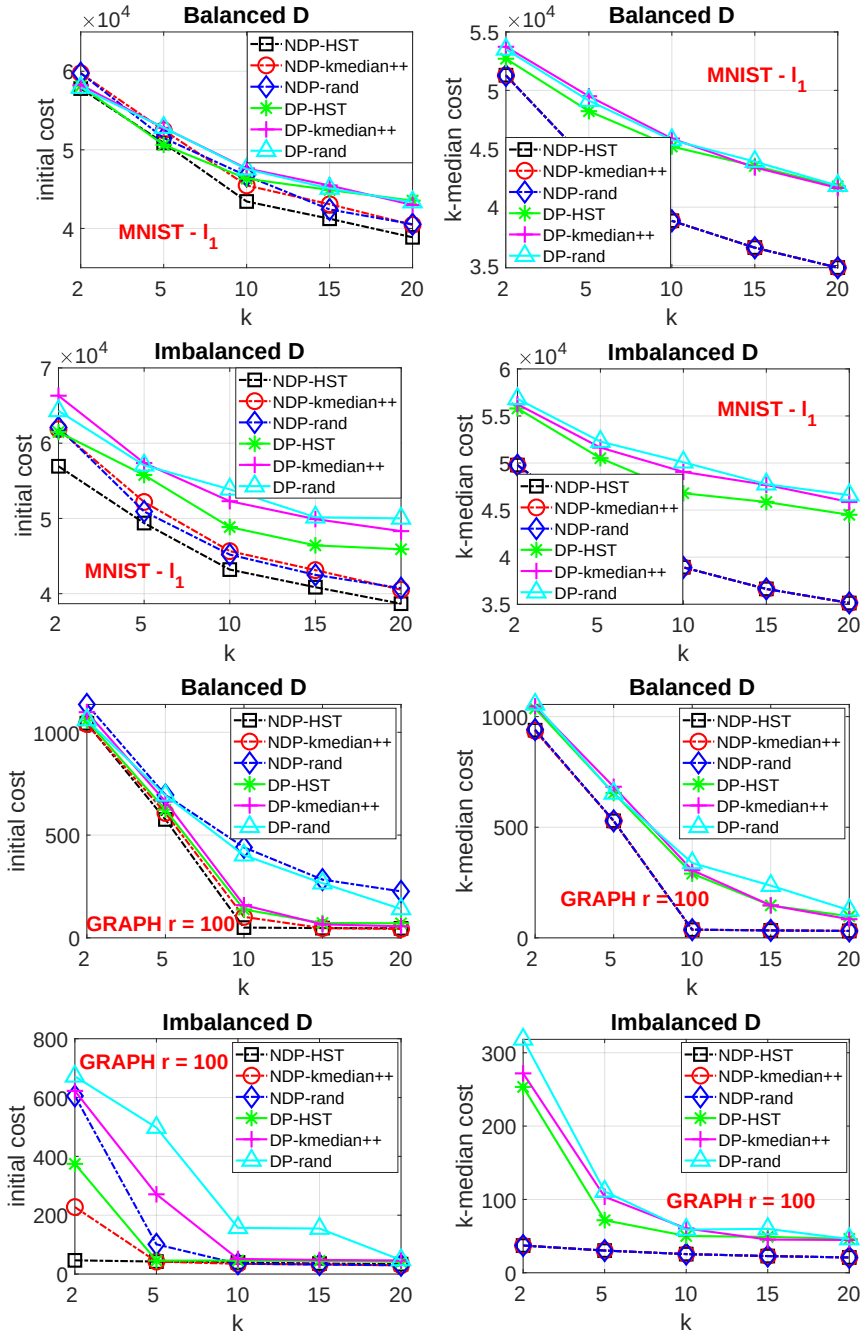

Figure 4: k-median cost on the MNIST (l_1-metric) and graph dataset ($r = 100$) **1st column:** initial cost. **2nd column:** final output cost.

A.2 More Experiments

A.3 Improved Iteration Cost of DP-HST

In Theorem 4.3, we show that under differential privacy constraints, the proposed DP-HST (Algorithm 7) improves both the approximation error and the number of iterations required to find a good solution of classical DP local search (Gupta et al., 2010). In this section, we provide some numerical results to justify the theory.

First, we need to properly measure the iteration cost of DP local search. This is because, unlike the non-private clustering, the k-median cost after each iteration in DP local search is not decreasing monotonically, due to the probabilistic exponential mechanism. To this end, for the cost sequence with length $T = 20$, we compute its moving average sequence with window size 5. Attaining the

minimal value of the moving average indicates that the algorithm has found a "local optimum", i.e., it has reached a "neighborhood" of solutions with small clustering cost. Thus, we use the number of iterations to reach such local optimum as the measure of iteration cost. The results are provided in Figure 5. We see that on all the tasks (MNIST with l_1 and l_2 distance, and graph dataset with $r = 1$ and $r = 100$), DP-HST has significantly smaller iterations cost. In Figure 6, we further report the k-median cost of the best solution in T iterations found by each DP algorithm. We see that DP-HST again provide the smallest cost. This additional set of experiments again validates the claims of Theorem 4.3, that DP-HST is able to found better initial centers in fewer iterations.

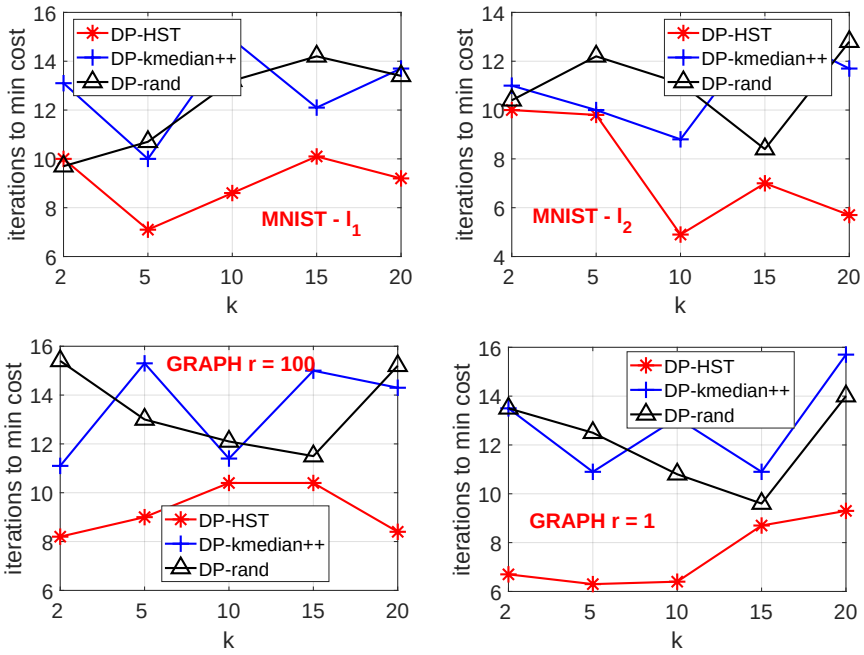

Figure 5: Iteration cost to reach a locally optimal solution, on MNIST and graph datasets with different k. The demand set is an imbalanced subset of the universe.

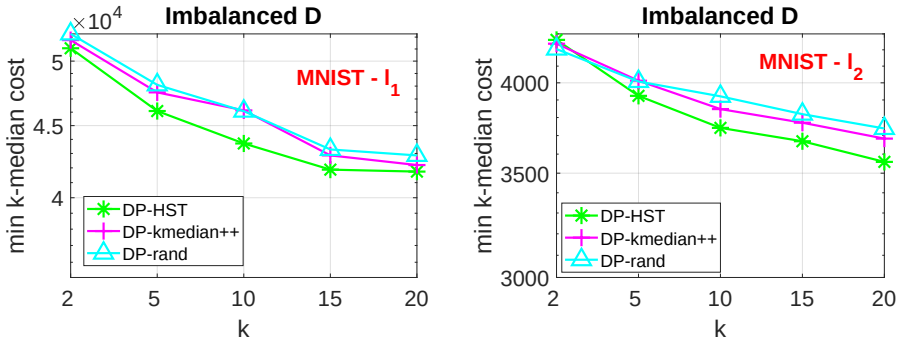

Figure 6: The k-median cost of the best solution found by each differentially private algorithm. The demand set is an imbalanced subset of the universe. Same comparison holds on graph data.

B Technical 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 3.5

Proof. Consider the intermediate output of Algorithm 4, $C_1 = \{v_1, v_2, ..., v_k\}$, 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 4 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 $cost_k^{T\prime}(U)$. 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 $cost_k^{T\prime}(U) \leq 5 OPT_k^T(U)$. □

B.2 Proof of Lemma 3.6

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 Proposition 3.2

Proof. It is known that the 2-HST can be constructed in $O(dn \log n)$ (Bartal, 1996). The subtree search in Algorithm 4 involves at most sorting all the nodes in the HST based on the score, which takes $O(nlogn)$. We use a priority queue to store the nodes in C_1. When we insert a new node v into queue, its parent node (if existing in the queue) would be removed from the queue. The number of nodes is $O(n)$ and each operation (insertion, deletion) in a priority queue based on score has $O(\log n)$ complexity. Lastly, the total time to obtain C_0 is $O(n)$, as the FIND-LEAF only requires a top down scan in k disjoint subtrees of T. Summing parts together proves the claim. □

B.4 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 3.4. 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 3.5. Consider the intermediate output of Algorithm 4, $C_1 = \{v_1, v_2, ..., v_k\}$, 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 3.5, 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, $1 - 2k/n^c$,

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

which proves the claim. □

Lemma B.5 (DP Leaf Search)**.** *With probability* $1 - 2k/n^c$*, Algorithm 6 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 3.6. 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.5 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 6 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 $i \leq T$ 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.

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

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

$$OPT_{km}^{T}(U) = \min_{F \subset U, |F|=k} \sum_{y \in U} \min_{x \in F} \rho^T(x,y)^2 \equiv \min_{C_1'} \ cost_{km}^{T}{}'(U, C_1'). \tag{14}$$

For simplicity, we will use $cost_{km}^{T}{}'(U)$ to denote $cost_{km}^{T}{}'(U, C_1)$ if everything is clear from context. Here, OPT_{km}^{T} (14) is the cost of the global optimal solution with 2-HST metric.

Lemma C.1 (Subtree search)**.** $cost_{km}^{T}{}'(U) \leq 17 OPT_{km}^{T}(U)$.

Proof. The analysis is similar with the proof of Lemma 3.5. Thus, we mainly highlight the difference. Let us just use some notations the same as in Lemma 3.5 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 $(4 \cdot 2^{h_{v_j''}})^2 N(v_j'')$ to $cost_{km}^{T}{}'(U)$. 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 5) 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 $cost_1^T(v, U) \leq cost_1^{T\prime}(v, U) + N_w \cdot (2^{h_w})^2 \leq 2cost_1^{T\prime}(v, U)$. 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. □

We are ready to state the error bound for our proposed HST initialization (Algorithm 4), 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 3.4.

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

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