**Near-optimal Algorithms for Explainable** k**-Medians and** k**-Means**

**Konstantin Makarychev** \* 1 **Liren Shan** \* 1

**Abstract**

We consider the problem of explainable k- medians and k-means introduced by Dasgupta, Frost, Moshkovitz, and Rashtchian (ICML 2020). In this problem, our goal is to ﬁnd a *threshold de- cision tree* that partitions data into k clusters and minimizes the k-medians or k-means objective. The obtained clustering is easy to interpret be- cause every decision node of a threshold tree splits databased on a single feature into two groups. We pro~pose a new algorithm for this problem which is O(log k)~competitive with k-medians with l1 norm and O(k) competitive with k-means. This is an improvement over the previous guarantees of O(k) and O(k2 ) by Dasgupta et al (2020). We also provide a new algorithm which is O(log3/2 k) competitive fork-medians with l2 norm. Our ﬁrst algorithm is near-optimal: Dasgupta et al (2020) showed a lower bound of Ω(log k) fork-medi~ans; in this work, we prove a lower bound of Ω(k) fork-means. We also provide a lower bound of Ω(log k) fork-medians with l2 norm.

**1. Introduction**

In this paper, we investigate the problem of *explainable* k-means and k-medians clustering which was recently in- troduced by [Dasgupta, Frost, Moshkovitz, and Rashtchian](#bookmark1) [(2020)](#bookmark1). Suppose, we have a data set which we need to par- tition into k clusters. How can we do it? Of course, we could use one of many standard algorithms fork-means or k-medians clustering. However, we want to ﬁnd an *explain- able* clustering – clustering which can be easily understood by a human being. Then, k-means or k-medians clustering may not be the best options for us.

Note that though every cluster in a k-means and k-medians clustering has a simple mathematical description, this de-

\*Equal contribution 1Northwestern University, Evanston, IL, USA. Correspondence to: Liren Shan <liren- shan2023@u.northwestern.edu>.

*Proceedings of the* 38 th *International Conference on Machine Learning*, PMLR 139, 2021. Copyright 2021 by the author(s).

scriptionis not necessarily easy to interpret for a human. Every k-medians or k-means clustering is deﬁned by a set of k centers c1 , c2 , . . . , ck , where each cluster is the set of points located closer to a ﬁxed center ci than to any other center cj . That is, for points in cluster i, we must have arg minj Ⅱx - cj Ⅱ = i. Thus, in order to determine to which cluster a particular point belongs, we need to com- pute distances from point x to all centers cj . Each distance depends on all coordinates of the points. Hence, for a hu- man, it is not even easy to ﬁgure out to which cluster in k-means or k-medians clustering a particular point belongs to; let alone interpret the entire clustering.

In every day life, we are surrounded by different types of classiﬁcations. Consider the following examples from Wikipedia: *(1) Performance cars are capable of going from 0 to 60 mph in under 5 seconds*; *(2) Modern sources cur- rently deﬁne skyscrapers as being at least 100 metres or 150 metres in height; (3) Very-low-calorie diets are diets of 800 kcal or less energy intake per day, whereas low-calorie diets are between 1000-1200 kcal per day*. Note that all these deﬁnitions depend on a *single feature* which makes them easy to understand.

The above discussion leads us to the idea of [Dasgupta et al.](#bookmark1) [(2020), who proposed to use threshold (decision) trees to](#bookmark1) describe clusters (see also [Liu, Xia, and Yu(2005),](#bookmark2)[Fraiman,](#bookmark3) [Ghattas, and Svarc](#bookmark3) [(2013),](#bookmark3) [Bertsimas, Orfanoudaki, and](#bookmark4) [Wiberg](#bookmark4) [(2018), and](#bookmark4)[Saisubramanian, Galhotra, and Zilber-](#bookmark5) [stein](#bookmark5) [(2020))](#bookmark5).

A threshold tree is a binary classiﬁcation tree with k leaves. Every internal node u of the tree splits the data into two sets by comparing a single feature iu of each data point with a threshold θu. The ﬁrst set is the set of points with xiu ≤ θu ; the second set is the set of points with xiu > θu. These two sets are then recursively partitioned by the left and right children of u. Thus, each point x in the data set is eventually assigned to one of k leaves of the threshold tree T. This gives us a partitioning of the data set X into clusters P = (P1, . . . , Pk ). We note that threshold decision trees are special cases of binary space partitioning (BSP) trees and similar to k-d trees ([Bentley,1975)](#bookmark6).

[Dasgupta et al.](#bookmark1) [(2020) suggested that we measure the qual](#bookmark1)- ity of a threshold tree using the standard k-means and k- medians objectives. Speciﬁcally, the k-medians in l1 cost

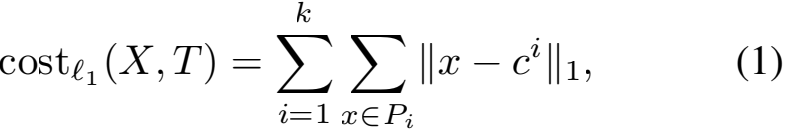
|  |  |  |  |
| --- | --- | --- | --- |
|  | k**-medians in** l1 | k**-medians in** l2 | k**-means** |

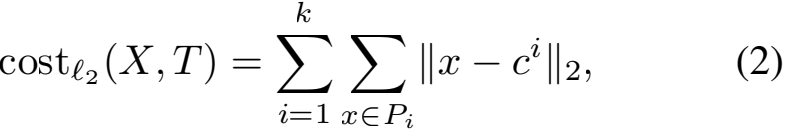
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Lower** | **Upper** | **Lower** | **Upper** | **Lower** | **Upper** |

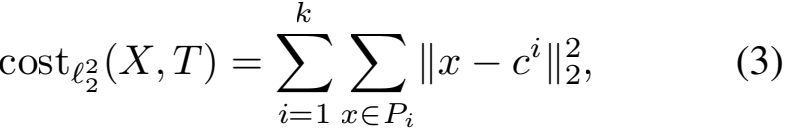
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Our results** |  | O(log k log log k) | Ω(log k) | O(log3/2 k) | Ω(k/ log k) | O(k log k log log k) |
| **Dasgupta et al. (2020)** | Ω(log k) | O(k) |  |  | Ω(log k) | O(k2 ) |

Figure 1. Summary of our results. The table shows known upper and lower bounds on the *price of explainability* fork-medians in {1 and {2 , and fork-means.

of the threshold tree T equals ([1](#bookmark8)), the k-medians in l2 cost equals ([2](#bookmark9)) and k-means cost equals ([3](#bookmark10)):







where ci is the l1 -median of cluster Pi in ([1](#bookmark8)), the l2 -median of cluster Pi in ([2](#bookmark9)), and the mean of cluster Pi in ([3)](#bookmark10).

This deﬁnition raises obvious questions: Can we actually ﬁnd a good explainable clustering? Moreover, how good can it be comparing to a regular k-medians and k-means clus-

tering? Let OPTl1 (X), OPTl2 (X), and OPTl (X) be

the optimal solutions to (regular) k-medians with l1 norm, k-medians with l2 norm, and k-means, respectively. [Das-](#bookmark1) [gupta et al.](#bookmark1) [(2020) deﬁned the](#bookmark1) *price of explainability* as the ratio costl1 (X, T)/OPTl1 (X) fork-medians in l1 and

costl (X, T)/OPTl (X) for k-means. The price of ex-

plainability shows by how much the optimal unconstrained solution is better than the best explainable solution for the same data set.

In their paper, [Dasgupta et al.](#bookmark1) [(2020) gave upper and lower](#bookmark1) bounds on the price of explainability. They proved that the price of explainability is upper bounded by O(k) and O(k2 ) fork-medians in l1 and k-means, respectively. Furthermore, they designed two algorithms that given a k-medians in l1 or k-means clustering, produce an explainable clustering with cost at most O(k) and O(k2 ) times the cost of original clustering (respectively). They also provided examples for which the price of explainability of k-medians in l1 and k-means is at least Θ(log k).

**1.1. Our results**

In this work, we give almost tight bounds on the price of explainability for both k-medians in l1 and k-means. Speciﬁcally, we show how to transform any clustering to an explainable clustering with cost at most O(log k log log k)

times the original cost for the k-medians l1 objective and O(k log k log log k) for the k-means objective. Note that we get an exponential improvement over previous results for the k-medians l1 objective. Furthermore, we present an al- gorithm fork-medians in l2 with the price of explainability bounded by O(log3/2 k). We complement these results with an almost tight lower bound of Ω(k/ log k) for the k-means objective and an Ω(log k) lower bound fork-medians in l2 objective. We summarise our results in Table [1.](#bookmark7)

Below, we formally state our main results. The costs of threshold trees and clusterings are deﬁned by formulas ([1](#bookmark8)), [(2), (](#bookmark9)[3](#bookmark10)), ([4](#bookmark12)), ([5](#bookmark13)), and ([6)](#bookmark14).

**Theorem 1.1.** *There exists a polynomial-time randomized algorithm that given a data set* X *and a set of centers* C = {c1 , . . . , ck }*, ﬁnds a threshold tree* T *with expected* k*-medians in* l1 *cost at most*

E[costl1 (X, T)] ≤ O(log k log log k) · costl1 (X, C).

**Theorem 1.2.** *There exists a polynomial-time randomized algorithm that given a data set* X *and a set of centers* C = {c1 , . . . , ck }*, ﬁnds a threshold tree* T *with expected* k*-means cost at most*

E[costl (X, T)] ≤ O(k log k log log k) · costl (X, C).

We note that the algorithms by [Dasgupta et al.](#bookmark1) [(2020) also](#bookmark1) produce trees based on the given set of “reference” cen- ters c1 , . . . , ck. However, the approximation guarantees of those algorithms are O(k) and O(k2 ), respectively. Our upper bound of O(log k log log k) almost matches the lower bound of Ω(log k) given by [Dasgupta et al.](#bookmark1) [(2020)](#bookmark1). The up- per bound of O(k log k log log k) almost matches the lower bound of Ω(k/ log k) we show in Appendix D.

**Theorem 1.3.** *There exists a polynomial-time randomized algorithm that given a data set* X *and a set of centers* C = {c1 , . . . , ck }*, ﬁnds a threshold tree* T *with expected* k*-medians in* l2 *cost at most*

E[costl2 (X, T)] ≤ O(log3/2 k) · costl2 (X, C).

**1.2. Related work**

[Dasgupta et al.(2020) introduced the](#bookmark1) *explainable* k-medians and k-means clustering problems and developed Iterative

Mistake Minimization (IMM) algorithms for these problems. Later, [Frost, Moshkovitz, and Rashtchian](#bookmark15) [(2020) proposed](#bookmark15) algorithms that construct threshold trees with more thank leaves.

Decision trees have been used for interpretable classiﬁcation and clustering since 1980s. [Breiman, Friedman, Olshen, and](#bookmark16) [Stone](#bookmark16) [(1984) proposed a popular decision tree algorithm](#bookmark16) called CART for supervised classiﬁcation. For unsuper- vised clustering, threshold decision trees are used in many empirical methods based on different criteria such as infor- mation gain ([Liu et al.,2005](#bookmark2)), local 1-means cost ([Fraiman](#bookmark3) [et al.,2013), Silhouette Metric (Bertsimas et al.,](#bookmark3)[2018), and](#bookmark4) interpretability score ([Saisubramanian et al.,2020)](#bookmark5).

The k-means and k-medians clustering problems have been extensively studied in the literature. The k-means++ algo- rithm proposed by [Arthur and Vassilvitskii](#bookmark17) [(2006) is the](#bookmark17) most widely used algorithm fork-means clustering. It pro- vides an O(ln k) approximation. [Li and Svensson](#bookmark18) [(2016)](#bookmark18) provided a 1 + √3 + ε approximation for k-medians in general metric spaces, which was improved to 2.611 + ε by [Byrka, Pensyl, Rybicki, Srinivasan, and Trinh](#bookmark19) [(2014)](#bookmark19). [Ah-](#bookmark20) [madian, Norouzi-Fard, Svensson, and Ward](#bookmark20) [(2019) gave](#bookmark20) a 6.357 approximation algorithm for k-means. The k- medians and k-means problems are NP-hard ([Megiddo &](#bookmark21) [Supowit,](#bookmark21) [1984;](#bookmark21) [Dasgupta,](#bookmark22) [2008;](#bookmark22) [Aloise et al.,](#bookmark23) [2009)](#bookmark23). Re- cently, [Awasthi, Charikar, Krishnaswamy, and Sinop](#bookmark24) [(2015)](#bookmark24) showed that it is also NP-hard to approximate the k-means objective within a factor of (1 + ε) for some positive con- stant ε (see also [Lee et al.](#bookmark25) [(2017))](#bookmark25). [Bhattacharya, Goyal,](#bookmark26) [and Jaiswal](#bookmark26) [(2020) showed that the Euclidean](#bookmark26) k-medians can not be approximated within a factor of (1 + ε) for some constant ε assuming the unique games conjecture.

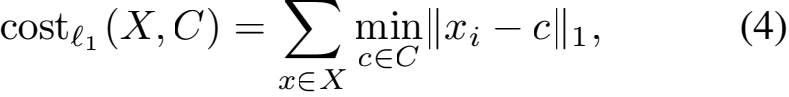
[Boutsidis et al.](#bookmark27) [(2009),](#bookmark27)[Boutsidis et al.](#bookmark28) [(2014),](#bookmark28)[Cohen et al.](#bookmark29) [(2015),](#bookmark29)[Makarychev et al.](#bookmark30) [(2019) and](#bookmark30)[Becchetti et al.](#bookmark31) [(2019)](#bookmark31) showed how to reduce the dimensionality of a data set for k-means clustering. Particularly, [Makarychev et al.](#bookmark30) [(2019)](#bookmark30) proved that we can use the Johnson–Lindenstrauss trans- form to reduce the dimensionality of k-medians in l2 and k-means to dI = O(log k). Note,however, that the Johnson– Lindenstrauss transform cannot be used for the explainable k-medians and k-means problems, because this transform does not preserve the set of features. Instead, we can use a

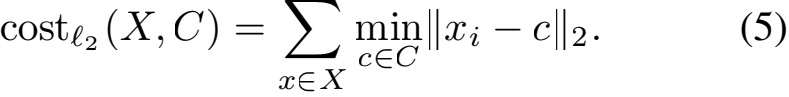
[hen et al.](#bookmark29) [(2015) to reduce the dimensionality to](#bookmark29) dI = O(k). Independently of our work, [Laber and Murtinho](#bookmark33) [(2021)](#bookmark33) proposed new algorithms for explainable k-medians with l1 and k-means objectives. Their competitive ratios are O(dlog k) and O(dk log k), respectively. Note that these competitive ratios depend on the dimension d of the space.

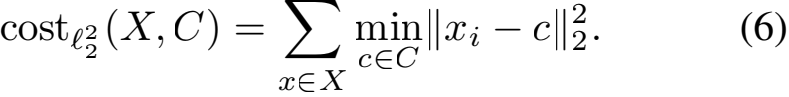
*featureselection* algorithm by [Boutsidis et al.](#bookmark28) [(2014) o](#bookmark28)r˜[Co-](#bookmark29)

**2. Preliminaries**

Given a set of points X ≤ Rd and an integer k > 1, the regular k-medians and k-means clustering problems are to ﬁnd a set C of k centers to minimize the corresponding costs: k-medians with l1 objective cost ([4](#bookmark12)), k-medians with l2 objective cost ([5](#bookmark13)), and k-means cost ([6)](#bookmark14).



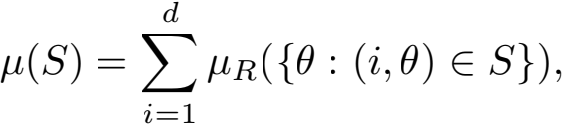




Every coordinate cut is speciﬁed by the coordinate i ∈ {1, . . . , d} and threshold θ . We denote the set of all possible cuts by Ω:

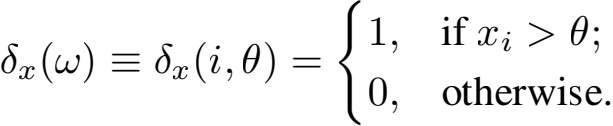
Ω = {1, · · · , d} × R.

We deﬁne the standard product measure on Ω as follows: The measure of set S C Ω equals

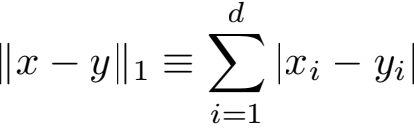


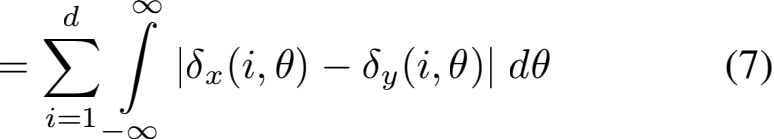
where μR is the Lebesgue measure on R.

For every cut ω = (i, θ) ∈ Ω and point x ∈ Rd , we let



In other words, δx (i, θ) is the indicator of the event {xi > θ}. Observe that x →} δx is an isometric embedding of l (d-dimensionall1 space) into L1 (Ω) (the space of integrable functions on Ω). Speciﬁcally, for x, y ∈ Rd , we have





=  |δx (ω) — δy (ω)| dμ(ω) 三 Ⅱδx — δy Ⅱ1 .

A map ϕ : Rd → Rd is coordinate cut preserving if for every coordinate cut (i, θ) ∈ Ω, there exists a coordinate cut (iI , θI ) ∈ Ω such that {x ∈ Rd : xi, ≤ θI } = {x ∈ Rd : ϕ(x)i ≤ θ} and vice versa. In the algorithm for explainable k-means, we use a cut preserving terminal embeddings of “l distance” into l1 .

**Algorithm 1** Threshold tree construction fork-medians in l1

**Input:** a data set X C Rd and set of centers C = {c1 , c2 , . . . , ck } C Rd **Output:** a threshold tree T

Set Sij = {ω ∈ Ω : δci (ω)  δcj (ω)} for all i, j ∈ {1, · · · , k}. Let t = 0.

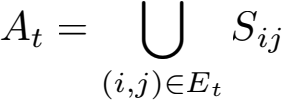
Create a tree T0 containing a root vertex r. Assign set Xr = X ∪ C to the root.

**while** Tt contains a leaf with at least two distinct centers ci and cj **do**

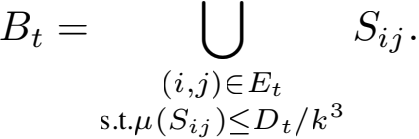
Let Et = uleaves u{(i, j) : ci , cj ∈ Xu } be the set of all not yet separated pairs of centers.

Let Dt = max(i,j)∈Et Ⅱci — cj Ⅱ1 be the maximum distance between two not separated centers.

Deﬁne two sets At , Bt C Ω as follows:



and



Le[t1 R](#bookmark35)t = At \ Bt. Pick a pair ωt = (i, θ) uniformly at random from Rt. For every leaf node u in T, split the set Xu into two sets:

*Left* = {x ∈ Xu : xi ≤ θ} and *Right* = {x ∈ Xu : xi > θ}.

If each of these sets contains at least one center from C, then create two children of u in tree T and assign sets *Left* and *Right* to the left and right child, respectively.

Denote the updated tree by Tt+1 . Update t = t + 1.

**end while**

**3. Algorithms Overview**

We now give an overview of our algorithms.

k**-medians in** l1**.** We begin with the algorithm for k- medians in l1 . We show that its competitive ratio is O(log2 k) in Section [4](#bookmark36)and then show an improved bound of O(log k log log k) in Section [5.](#bookmark37)

As the algorithm by [Dasgupta et al.](#bookmark1) [(2020), our algorithm](#bookmark1) (see Algorithm [1](#bookmark34)) builds a binary threshold tree T top-down. It starts with a tree containing only the root noder. This node is assigned the set of points Xr that contains all points in the data set X and all reference centers ci. At every round, the algorithm picks some pair ω = (i, θ) ∈ Ω (as we discuss below) and then splits data points x assigned to every *leaf* node u into two groups {x ∈ Xu : xi ≤ θ} and {x ∈ Xu : xi > θ}. Here, Xu denotes the set of points assigned to the node u. If this partition separates at least two centers ci and cj , then the algorithm attaches two children to u and assigns the ﬁrst group to the left child and the second group to the right child. The algorithm terminates when all leaves contain exactly one reference center ci. Then, we assign the points in each leaf of T to its unique reference center. Note that the unique reference center in each leaf may not be the optimal center for points contained in that

leaf. Thus, the total cost by assigning each point to the reference center in the same leaf of T is an upper bound of the cost of threshold tree T.

The algorithm by [Dasgupta et al.](#bookmark1) [(2020) picks splitting cuts](#bookmark1) in a greedy way. Our algorithm chooses them at random. Speciﬁcally, to pick a cut ωt ∈ Ω at round t, our algorithm ﬁnds the maximum distance Dt between two distinct centers ci , cj that belong to the same set Xu assigned to a leaf node ui.e.,

Dt = max max

Ⅱci — cj Ⅱ1 .

u is a leaf ci,cj ∈Xu

Then, we let At be the set of all ω ∈ Ω that separate at least one pair of centers; and Bt be the set of all ω ∈ Ω that separate two centers at distance at most Dt /k3 . We pick ωt uniformly at random (with respect to measure μ) from the set Rt = At \ Bt.

Every ω ∈ Rt is contained in At , which means ω separates at least one pair of centers. Thus, our algorithm terminates in at most k—1 iterations. It is easy to see that the running time of this algorithm is polynomial in the number of clusters k and dimension of the space d. In Section E, we provide a

1As we discuss in Section E, we can also let Rt = At. How- ever, this will make the analysis more involved.

variant of this algorithm with running time (kd).

k**-medians in** l2**.** Our algorithm for k-medians with l2 norm recursively partitions the data set X using the follow- ing idea. It ﬁnds the median point m of all centers in X. Then, it repeatedly makes cuts that separate centers from m. To make a cut, the algorithm chooses a random coordinate i ∈ {1; : : : ; d}, random number θ ∈ [0; R2], and random sign σ ∈ {±1}, where R is the largest distance from a cen- ter in X to the median point m. It then makes a threshold cut (i; mi + σ√θ ). After separating more than half centers from m, the algorithm recursively calls itself for each of the obtained parts. In Appendix C, we show that the *price of explainability* for this algorithm is O(log3/2 k).

k**-means.** We now move to the algorithm fork-means. This algorithm embeds the space l2 into l1 using a specially crafted *terminal embedding*' (the notion of terminal em- beddings was formally deﬁned by [Elkin et al.](#bookmark39) [(2017))](#bookmark39). The embedding satisﬁes the following property for every center c (terminal) and every point x ∈ l2 , we have

Ⅱ'(x) − '(c)Ⅱ1 ≤ Ⅱx − cⅡ ≤ 8k · Ⅱ'(x) − '(c)Ⅱ1 :

Then, the algorithm partitions the data set '(X) with centers '(c1 ); : : : ; '(ck ) using Algorithm [1.](#bookmark34) The expected cost of partitioning is at most the distortion of the embedding (8k) times the competitive guarantee (O(log k log log k)) of Algorithm [1.](#bookmark34) In Section D, we show an almost matching lower bound of Ω(k= log k) on the cost of explainability for k-means. We also remark that the terminal embedding we use in this algorithm cannot be improved. This follows from

the fact that the cost function Ⅱx − cⅡ does not satisfy the

triangle inequality; while the l1 distance Ⅱ'(x) − '(c)Ⅱ1 does.

**4. Algorithm for** k**-medians in** l1

In this section, we analyse Algorithm [1](#bookmark34) for k-medians in l1 and show that it provides an explainable clustering with cost at most O(log2 k) times the original cost. We improve this bound to O(log k log log k) in Section [5.](#bookmark37)

Recall, all centers in C are separated by the tree T returned by the algorithm, and each leaf of T contains exactly one center from C. For each point x ∈ X , we deﬁne its cost in the threshold tree T as

algl1 (x) = Ⅱx − cⅡ1 ;

where c is the center in the same leaf in T as x. Then, costl1 (X; T) ≤Σx∈X algl1 (x) (note that the original cen- ters c1 ; : : : ; ck used in the deﬁnition of algl1 (x) are not necessarily optimal for the tree T. Hence, the lefthand side is not always equal to the right hand side.). For every point x ∈ X , we also deﬁne costl1 (x; C) = minc∈C Ⅱx − cⅡ1 . Then, costl1 (X; C) =Σx∈X costl1 (x; C) (see ([4](#bookmark12))).

We prove the following theorem.

**Theorem 4.1.** *Given a set of points* X *in* Rd *and a set of centers* C = {c1 ; : : : ; ck } ⊂ Rd*, Algorithm* [*1*](#bookmark34)*ﬁnds a*

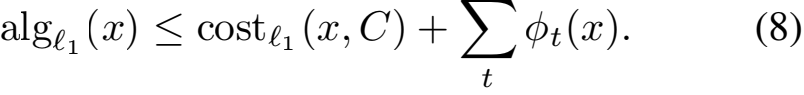
*threshold tree* T *with expected* k*-medians in* l1 *cost at most*

E[costl1 (X; T)] ≤ O(log2 k) · costl1 (X; C):

*Moreover, the same bound holds for the cost of every point* x ∈ X *i.e.,*

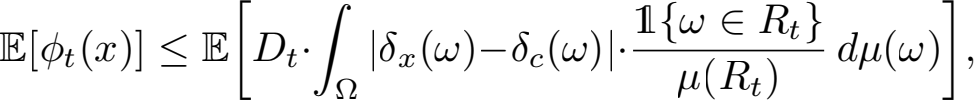
E[costl1 (x; T)] ≤ O(log2 k) · costl1 (x; C):

*Proof.* Let Tt be the threshold tree constructed by Algo- rithm [1](#bookmark34)before iteration t. Consider a point x in X. If x is separated from its original center in C by the cut generated at iteration t, thenx will be eventually assigned to some other center in the same leaf of Tt. By the triangle inequality, the new cost of x at the end of the algorithm will be at most costl1 (x; C) + Dt , where Dt is the maximum diameter of any leaf in Tt (see Algorithm [1)](#bookmark34). Deﬁne a penalty function φt (x) as follows: φt (x) = Dt if x is separated from its original center c at time t; φt (x) = 0, otherwise. Note that φt (x)  0 for at most one iteration t, and



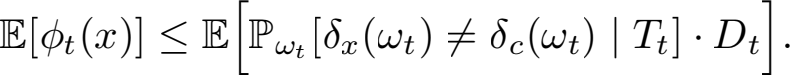
The sum in the right hand side is over all iterations of the algorithm. We bound the expected penalty φt (x) for each t.

**Lemma 4.2.** *The expected penalty* φt (x) *is upper bounded as follows:*



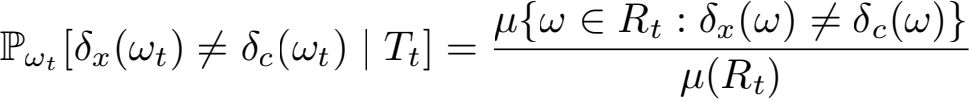
*where* c *is the closest center to the point* x *in* C*;* 1{! ∈ Rt } *is the indicator of the event* ! ∈ Rt*.*

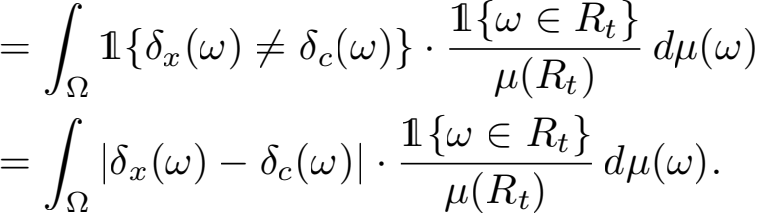
*Proof.* If x is already separated from its original center c at iteration t, then φt (x) = 0. Otherwise,x and c are separated at iteration tif for the random pair!t = (i; θ) chosen from Rt in Algorithm [1](#bookmark34), we have δx (!t )  δc (!t ). Write,



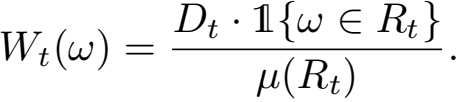
The probability that δx (!t )  δc (!t ) given Tt is bounded

as

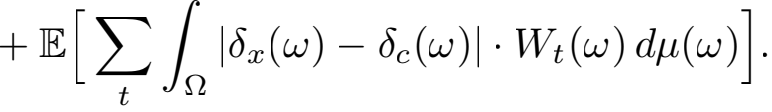




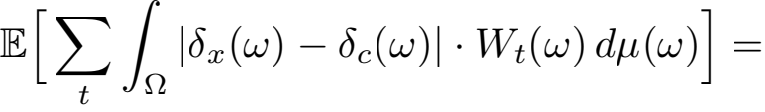
Let

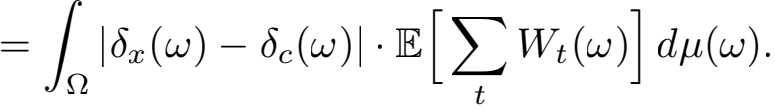


Then, by Lemma [4.2](#bookmark41)and inequality ([8](#bookmark40)), we have E[algl1 (x)] ≤ costl1 (x, C)+

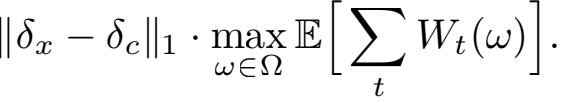


The upper bound on the expected cost of x in tree T consists of two terms: The ﬁrst term is the original cost of x. The sec- ond term is a bound on the expected penalty incurred by x. We now bound the second term as O(log2 k) · costl1 (x, C).



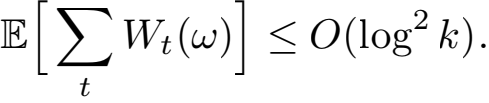


By Hlder’s inequality, the right hand side is upper bounded by the following product:



The ﬁrst multiplier in the product exactly equals Ⅱx − cⅡ1 (see Equation[7](#bookmark32)), which,in turn, equals costl1 (x, C). Hence, to ﬁnish the proof of Theorem [4.1](#bookmark38), we need to upper bound the second multiplier by O(log2 k).

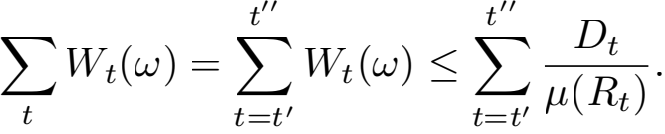
**Lemma 4.3.** *For all* ω ∈ Ω*, we have*



*Proof.* Let t/ be the ﬁrst iteration and t// be the last iteration for which Wt (ω) > 0. First, we prove that Dt,, ≥ Dt, /k3 , where Dt, and Dt,, are the maximum cluster diameters at iterations t/ and t// , respectively. Since Wt, (ω) > 0 and Wt,, (ω) > 0, we have 1{ω ∈ Rt, }  0 and 1{ω ∈ Rt,, }  0. Hence, ω ∈ Rt, and ω ∈ Rt,, . Since ω ∈ Rt,, , there exists a pair (i, j) ∈ Et,, for which ω ∈ Sij . For this pair, we have Dt,, ≥ μ(Sij ). Observe that the pair (i, j) also belongs to Et, , since Et,, ⊂ Et, . Moreover, μ(Sij ) > Dt, /k3 , because otherwise, Sij would be included in Bt, (see Algorithm [1](#bookmark34)) and, consequently, ω would not belong to Rt, = At, \ Bt, . Thus

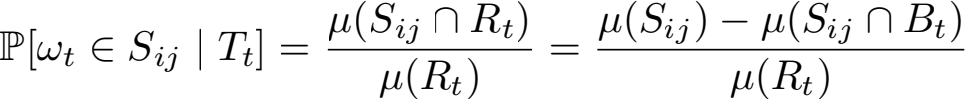
Dt,, ≥ μ(Sij ) > Dt, /k3 . (9)

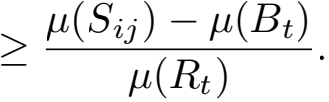
By the deﬁnition of t/ and t// , we have



Note that the largest distance Dt is a non-increasing (ran- dom) function of t. Thus, we can split the iterations of the algorithm {t/ , ..., t// } into「3 log kl phases. At phases, the maximum diameter Dt is in the range (Dt, /2s+1, Dt, /2s]. Denote the set of alliterations in phases by Phase(s).

Consider phase s. Let D = Dt, /2s. Phase s ends when all sets Sij with μ(Sij ) ≥ D/2 are removed from the set Et. Let us estimate the probability that one such set Sij is removed from Et at iteration t. Set Sij is removed from Et if the random threshold cut ωt chosen at iteration t separates centers ci and cj , or, in other words, if wt ∈ Sij . The probability of this event equals:

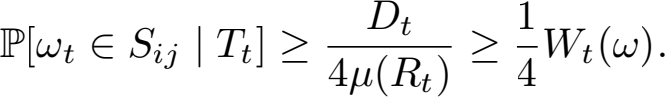




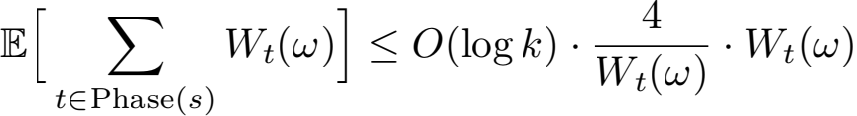
Note that μ(Sij ) > D/2 ≥ Dt /2 and μ(Bt ) < () · <

(because Bt is the union of at most () sets of measure

at most Dt /k3 each). Hence,



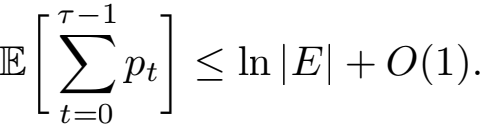
If Wt (ω) did not depend on t, then we would argue that each set Sij (with μ(Sij ) ≥ D/2) is removed from Et in at most 4/Wt (ω) iterations, in expectation, and, consequently, all sets Sij are removed in at most O(log k) · 4/Wt (ω) iterations, in expectation (note that the number of sets Sij is upper bounded by ()). Therefore,



= O(log k).

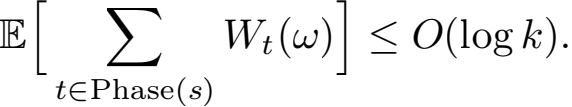
However, we cannot assume that Wt (ω) is a constant. In- stead, we use the following claim with E = {0, . . . , k − 1} × {0, . . . , k − 1}, E = {(i, j) ∈ Et : μ(Sij ) ≥ D/2}, and pt = Wt (ω)/4.

**Claim 4.4.** *Consider two stochastic processes* Et *and* pt *adapted to ﬁltration* Ft*. The values of* Et *are subsets of some ﬁnite non-empty set* E*. The values of* pt *are numbers in* [0, 1]*. Suppose that for every step* t*,* Et+1 ⊂ Et *and for every* e ∈ Et*,* Pr[e  Et+1 | Ft] ≥ pt*. Let* τ *be the (stopping) time* t *when* Et = ?*. Then,*



We prove this claim in Appendix A.

By Claim [4.4,](#bookmark43)



The expected sum of Wt over all phases is upper bounded by O(log2 k), since the number of phases is upper bounded by O(log k). We note that if the number of phases is upper bounded by L, then the expected sum of Wt over all phases is upper bounded by O(Llog k). This concludes the proofs of Lemma [4.3](#bookmark42)and Theorem [4.1.](#bookmark38)  

**5. Improved Analysis for** k**-medians in** l1

In this section, we provide an improved analysis of our algorithm fork-medians in l1 .

**Theorem 5.1.** *Given a set of points* X *in* Rd *and set of cen- ters* C = {c1 , . . . , ck } C Rd*, Algorithm* [*1*](#bookmark34)*ﬁnds a threshold*

*tree* T *with expected* k*-medians* l1 *cost at most*

E[costl1 (X, T)] ≤ O(log k log log k) · costl1 (X, C).

*Proof.* In the proof of Theorem [4.1](#bookmark38), we used a pessimistic estimate on the penalty a point x ∈ X incurs when it is sep- arated from its original center c. Speciﬁcally, we bounded the penalty by the maximum diameter of any leaf in the tree Tt. In the current proof, we will use an additional bound: The distance from x to the closest center after separation. Suppose, that x is separated from its original center c. Let cI be the closest center to x after we make cut ωt at step t. That is, cI is the closest center to xin the same leaf of the threshold tree Tt+1 . Note that after we make additional cuts, x may be separated from its new center cI as well, and the cost of x may increase. However, as we already know, the expected cost of x may increase in at most O(log2 k) times in expectation (by Theorem [4.1)](#bookmark38). Here, we formally apply Theorem [4.1](#bookmark38) to the leaf where x is located and treat cI as the original center of x. Therefore, if x is separated from c by a cut ωt at step t, then the expected cost of x in the end of the algorithm is upper bounded by

E[algl1 (x) j Tt , ωt] ≤ O(log2 k) · ⅡcI — xⅡ1 (10) 

In the formula above, we used the following deﬁnition:

D(x, ω) is the distance from x to the closest center cI in

the same leaf of Tt as x which is not separated from x by the cut ω i.e., δx (ω) = δc, (ω). If there are no such centers cI (i.e., cut ω separates x from all centers), then we let D(x, ω) = 0. Note that in this case, our algorithm will never make cut ω, since it always makes sure that the both parts of the cut contain at least one center from C. Similarly

to D(x, ω), we deﬁne D(x, ω): D(x, ω) is the

distance from x to the farthest center cII in the same leaf

of Tt as x which is not separated from x by the cut ω .

We also let D(x, ω) = 0 if there is no such cII . Note

that D(x, ω) is an upper bound on the cost of x in the

eventual threshold tree T if cut ω separated x from c at step t.

We now have three bounds on the expected cost of x in the ﬁnal tree T given that the algorithm separates x from its original center c at step t with cut ω . The ﬁrst bound is

D(x, ω); the second bound is O(log2 k) · D(x, ω),

and the third bound is Ⅱx— cⅡ1 +Dt. We use the ﬁrst bound

if D(x, ω) ≤ 2Ⅱx — cⅡ1 . We call such cuts ω light cuts.

We use the second bound if D(x, ω) > 2Ⅱx — cⅡ1 but

D(x, ω) ≤ Dt / log4 k. We call such cuts ω medium

cuts. We use the third bound if D(x, ω) > 2Ⅱx — cⅡ1

and D(x, ω) > Dt / log4 k. We call such cuts ω heavy

cuts.

Note that in the threshold tree returned by the algorithm, one and only one of the following may occur: (1) xis separated from the original center c by a light, medium, or heavy cut; (2) x is not separated from c. We now estimate expected penalties due to light, medium, or heavy cuts.

If the algorithm makes a light cut, then the maximum cost of point x in T is at most 2Ⅱx — cⅡ1 = 2costl1 (x, C). So we should not worry about such cuts. If the algorithm makes a medium cut, then the expected additional penalty for x is upper bounded by

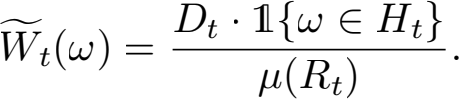
D(x, ωt ) · O(log2 k) ≤ O(φt (x)/ log2 k),

where φt (x) is the function from the proof of Theorem [4.1.](#bookmark38) Thus, the total expected penalty due to a medium cut (added up over all steps of the algorithm) is Ω(log2 k) times smaller than the penalty we computed in the proof of Theorem [4.1.](#bookmark38) Therefore, the expected penalty due to a medium cut is at most O(Ⅱx — cⅡ1 ).

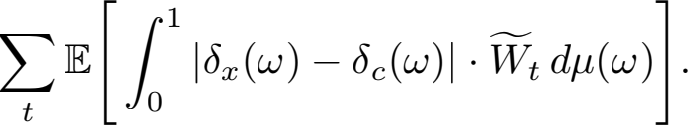
We now move to heavy cuts. Denote the set of possible heavy cuts for xin Rt by Ht. That is, if x is not separated from its original center c by step t, then

Ht = {ω ∈ Rt : D(x, ω) > Dt / log4 k and D(x, ω) > 2Ⅱx — cⅡ1 }.

Otherwise, let Ht = ?. Deﬁne a density function t (ω) similarly to Wt (ω):

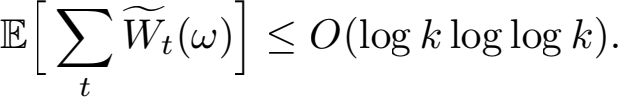


Then, the expected penalty due to a heavy cut is bounded, similarly to Lemma [4.2](#bookmark41), by



Therefore, to ﬁnish the proof of Theorem [1.1](#bookmark11), we need to prove the following analog of Lemma [4.3.](#bookmark42)

**Lemma 5.2.** *For all* ω ∈ Ω*, we have*



*Proof.* As in the proof of Lemma [4.3](#bookmark42), consider the ﬁrst and last steps when Wt (ω) > 0. Denote these steps by t\* and t\*\* , respectively. In the proof of Lemma [4.3](#bookmark42), we had a bound Dt// ≥ Dt/ /k3 (see inequality ([9](#bookmark44))). We now show a stronger bound on t\* and t\*\* .

**Claim 5.3.** *We have* Dt\*\* ≥ Dt\*/2 log4 k*.*

This claim implies that the number of phases deﬁned in Lemma [4.3](#bookmark42)is bounded by O(log log k), which immediately implies Lemma [5.2.](#bookmark47) So, to complete the proof, it remains to show Claim [5.3.](#bookmark49)

*Proof of Claim* [*5.3*](#bookmark49)First, note that 1{ω ∈ Ht\*\* } > 0 and,

consequently, cut ω is heavy at step t\*\* . Thus, Dn (x, ω)

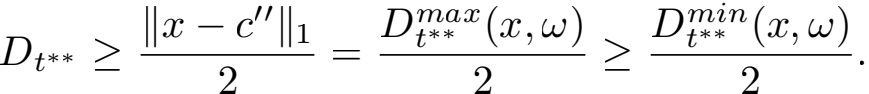
is positive. Hence, this cut separates c from at least one other center c/ in the same leaf of the current threshold tree Tt\*\* . Let c// be the farthest such center from point x. Then,

Ⅱc// − xⅡ1 = Dx (x, ω). Since centers candc// are not

separated prior to step t\*\* , we have

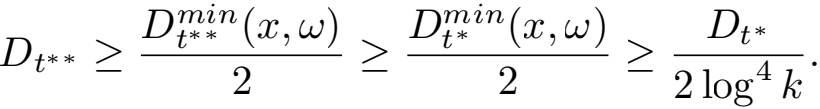
Dt\*\* ≥ Ⅱc − c// Ⅱ1 ≥ Ⅱx − c// Ⅱ1 − Ⅱx − cⅡ1 .

Since ω is a heavy cut and not a light cut, Ⅱx − c// Ⅱ1 > 2Ⅱx − cⅡ1 . Thus,



Now, observe that the random process Dn (x, ω) is non-

decreasing (for ﬁxedx and ω) since the distance from x to the closest center c/ cannot decrease over time. Therefore,



In the last inequality, we used that ω is a heavy cut at time t\* . This ﬁnishesthe proof of Claim [5.3.](#bookmark49)

**6. Terminal Embedding of** l **into** l1

In this section, we show how to construct a coordinate cut preserving terminal embedding of l (squared Euclidean distances) into l1 with distortion O(k) for every set of ter- minals K ⊂ Rd of size k.

Let K be a ﬁnite subset of points in Rd. We say that ϕ : x →} ϕ(x) is a terminal embedding of l into l1 with a set of terminals K and distortion α if for every terminal y in K and every point x in Rd , we have

Ⅱϕ(x) − ϕ(y)Ⅱ1 ≤ Ⅱx − yⅡ ≤ α · Ⅱϕ(x) − ϕ(y)Ⅱ1 .

**Lemma 6.1.** *For every ﬁnite set of terminals* K *in* Rd*, there exists acoordinate cut preserving terminal embedding of* l *into* l1 *with distortion* 8|K|*.*

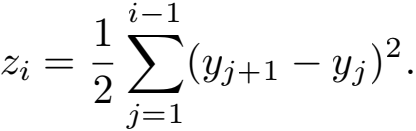
*Proof.* We ﬁrst prove a one dimensional analog of this the- orem (which corresponds to the case when all points and centers are in one dimensional space).

**Lemma 6.2.** *For every ﬁnite set of real numbers* K*, there exists a cut preserving embedding* ψK : R → R *such that for every* x ∈ R *and*y ∈ K*, we have*

|ψK (x) − ψK (y)| ≤ |x − y|2 (11)

≤ 8|K| · |ψK (x) − ψK (y)| .

*Proof.* Let kbe the size of K andy1 , . . . , yk be the elements of K sorted in increasing order. We ﬁrst deﬁne ψK on pointsin K and then extend this map to the entire real line R. We map each yi to zi deﬁned as follows: z1 = 0 and for i = 2, . . . , k,



Now consider an arbitrary number x in R. Let yi be the closest point to x in K. Let εx = sign(x − yi ). Then, x = yi + εx |x − yi |. Note that εx = 1 if x is on the right to

yi , and εx = −1, otherwise. Let the function ψK be ψK (x) = zi + εx (x − yi )2 .

For x = (yi + yi+1)/2, both yi and yi+1 are the closest points to xin K. In this case, we have

zi + εx (x − yi )2 = zi+1 + εx (x − yi+1)2 ,

which means ψK (x) is well-deﬁned. An example of the ter- minal embedding function ψK (x) is shown in Figure [2.](#bookmark50) We show that this function ψK is a cut preserving embedding satisfying inequality ([11](#bookmark48)) in Lemma B.1.

Using the above lemma, we can construct a terminal em-

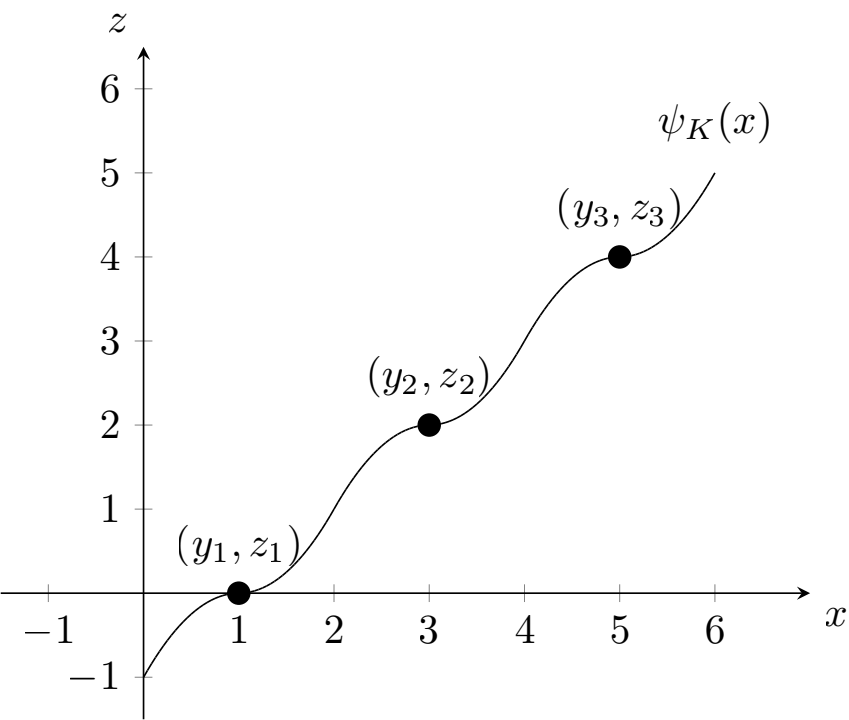
bedding ψ from d-dimensional l into d-dimensional l1

as follows. For each coordinate i ∈ {1, 2, · · · , d}, let Ki be the set of the i-th coordinates for all terminals in K. Deﬁne one dimensional terminal embeddings ψi for

all coordinates i. Then, ψ maps every point x ∈ l to

ψ(x) = (ψ1 (x), · · · , ψd (x)). We show that this terminal embedding ψ is coordinate cut preserving in Lemma B.2.

For explainable k-means clustering, we ﬁrst use the terminal embedding of l into l1 . Then, we apply Algorithm [1](#bookmark34) to the instance after the embedding. By using this terminal embedding, we can get the following result.

Arthur, D. and Vassilvitskii, S. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

Awasthi, P., Charikar, M., Krishnaswamy, R., and Sinop, A. K. The hardness of approximation of euclidean k- means. *arXiv preprint arXiv:1502.03316*, 2015.

Becchetti, L., Bury, M., Cohen-Addad, V., Grandoni, F., and Schwiegelshohn, C. Oblivious dimension reduction fork- means: beyond subspaces and the johnson-lindenstrauss lemma. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1039–1050, 2019.

Bentley, J. L. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18 (9):509–517, 1975.

Figure 2. Terminal embedding function ψK (x) for K = {1, 3, 5}.

**Theorem 6.3.** *Given a set of points* X *in* Rd *and a set of centers* C *in* Rd*, Algorithm* [*1*](#bookmark34)*with terminal embedding ﬁnds*

*a threshold tree* T *with expected* k*-means cost at most*

E[costl (X, T)] ≤ O(k log k log log k) · costl (X, C).

*Proof.* Let ϕ be the terminal embedding of l into l1 with terminals C. Let T/ be the threshold tree returned by our al- gorithm on the instance after embedding. Since the terminal embedding ϕ is coordinate cut preserving, the threshold tree T/ also provides a threshold tree T on the original k-means instance. Let ϕ(C) be the set of centers after embedding. For any point x ∈ X , the expected cost of x is at most

E[costl (x, T)] ≤ 8k · E[costl1 (ϕ(x), T/ )]

Bertsimas, D., Orfanoudaki, A., and Wiberg, H. Inter- pretable clustering via optimal trees. *arXiv preprint arXiv:1812.00539*, 2018.

Bhattacharya, A., Goyal, D., and Jaiswal, R. Hardness of approximation of euclidean k-median. *arXiv preprint arXiv:2011.04221*, 2020.

Boutsidis, C., Mahoney, M. W., and Drineas, P. An im- proved approximation algorithm for the column subset selection problem. In *Proceedings of the twentieth an- nualACM-SIAM symposium on Discrete algorithms*, pp. 968–977. SIAM, 2009.

Boutsidis, C., Zouzias, A., Mahoney, M. W., and Drineas, P. Randomized dimensionality reduction for k-means clustering. *IEEE Transactions on Information Theory*, 61 (2):1045–1062, 2014.

≤ O(k log k log log k) · costl1 (ϕ(x), ϕ(C)) ≤ O(k log k log log k) · costl (x, C),

where the ﬁrst and third inequality is from the terminal embedding in Lemma [6.1](#bookmark46)and the second inequality is due to Theorem [5.1.](#bookmark45)

**Acknowledgements**

Konstantin Makarychev and Liren Shan were supported by NSF Awards CCF-1955351 and CCF-1934931.

**References**

Ahmadian, S., Norouzi-Fard, A., Svensson, O., and Ward, J. Better guarantees fork-means and euclidean k-median by primal-dual algorithms. *SIAM Journal on Computing*, 49(4):FOCS17–97, 2019.

Aloise, D., Deshpande, A., Hansen, P., and Popat, P. Np- hardness of euclidean sum-of-squares clustering. *Ma- chine learning*, 75(2):245–248, 2009.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. Classi- ﬁcation and regression trees, 1984.

Byrka, J., Pensyl, T., Rybicki, B., Srinivasan, A., and Trinh, K. An improved approximation fork-median, and posi- tive correlation in budgeted optimization. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Dis- crete algorithms*, pp. 737–756. SIAM, 2014.

Cohen, M. B., Elder, S., Musco, C., Musco, C., and Persu, M. Dimensionality reduction fork-means clustering and low rank approximation. In *Proceedings of the forty- seventh annual ACM symposium on Theory of computing*, pp. 163–172, 2015.

Dasgupta, S. *The hardness of* k*-means clustering*. Depart- ment of Computer Science and Engineering, University of California, San Diego, 2008.

Dasgupta, S., Frost, N., Moshkovitz, M., and Rashtchian,

C. Explainable k-means and k-medians clustering. In

*International Conference on Machine Learning*, pp. 7055–

7065. PMLR, 2020.

Elkin, M., Filtser, A., and Neiman, O. Terminal embeddings. *Theoretical Computer Science*, 697:1–36, 2017.

Fraiman, R., Ghattas, B., and Svarc, M. Interpretable clus- tering using unsupervised binary trees. *Advances in Data Analysis and Classiﬁcation*, 7(2):125–145, 2013.

Frost, N., Moshkovitz, M., and Rashtchian, C. Exkmc: Ex- panding explainable k-means clustering. *arXiv preprint arXiv:2006.02399*, 2020.

Laber, E. and Murtinho, L. On the price of explain- ability for some clustering problems. *arXiv preprint arXiv:2101.01576*, 2021.

Lee, E., Schmidt, M., and Wright, J. Improved and simpli- ﬁed inapproximability fork-means. *Information Process- ing Letters*, 120:40–43, 2017.

Li, S. and Svensson, O. Approximating k-median via pseudo-approximation. *SIAM Journal on Computing*, 45(2):530–547, 2016.

Liu, B., Xia, Y., and Yu, P. S. Clustering via decision tree construction. In *Foundations and advances in data* *mining*, pp. 97–124. Springer, 2005.

Makarychev, K., Makarychev, Y., and Razenshteyn, I. Per- formance of johnson-lindenstrauss transform fork-means and k-medians clustering. In *Proceedings of the 51stAn- nualACM SIGACT Symposium on Theory of Computing*, pp. 1027–1038, 2019.

Megiddo, N. and Supowit, K. J. On the complexity of some common geometric location problems. *SIAM journal on* *computing*, 13(1):182–196, 1984.

Saisubramanian, S., Galhotra, S., and Zilberstein, S. Bal- ancing the tradeoff between clustering value and inter- pretability. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 351–357, 2020.