

# Clustering under Perturbation Stability in Near-Linear Time

**Pankaj K. Agarwal**

Department of Computer Science, Duke University, USA.

[pankaj@cs.duke.edu](mailto:pankaj@cs.duke.edu)

**Hsien-Chih Chang**

Department of Computer Science, Dartmouth, USA., USA.

[hsien-chih.chang@dartmouth.edu](mailto:hsien-chih.chang@dartmouth.edu)

**Kamesh Munagala**

Department of Computer Science, Duke University, USA.

[kamesh@cs.duke.edu](mailto:kamesh@cs.duke.edu)

**Erin Taylor**

Department of Computer Science, Duke University, USA.

[ect15@cs.duke.edu](mailto:ect15@cs.duke.edu)

**Emo Welzl**

Department of Computer Science, ETH Zürich, Switzerland.

[emo@inf.ethz.ch](mailto:emo@inf.ethz.ch)

---

## Abstract

We consider the problem of center-based clustering in low-dimensional Euclidean spaces under the perturbation stability assumption. An instance is  $\alpha$ -stable if the underlying optimal clustering continues to remain optimal even when all pairwise distances are arbitrarily perturbed by a factor of at most  $\alpha$ . Our main contribution is in presenting efficient exact algorithms for  $\alpha$ -stable clustering instances whose running times depend near-linearly on the size of the data set when  $\alpha \geq 2 + \sqrt{3}$ . For  $k$ -center and  $k$ -means problems, our algorithms also achieve polynomial dependence on the number of clusters,  $k$ , when  $\alpha \geq 2 + \sqrt{3} + \varepsilon$  for any constant  $\varepsilon > 0$  in any fixed dimension. For  $k$ -median, our algorithms have polynomial dependence on  $k$  for  $\alpha > 5$  in any fixed dimension; and for  $\alpha \geq 2 + \sqrt{3}$  in two dimensions. Our algorithms are simple, and only require applying techniques such as local search or dynamic programming to a suitably modified metric space, combined with careful choice of data structures.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Design and analysis of algorithms

**Keywords and phrases** clustering, stability, local search, dynamic programming, coresets, polyhedral metric, trapezoid decomposition, range query

**Digital Object Identifier** [10.4230/LIPIcs...](https://doi.org/10.4230/LIPIcs...)

**Related Version** A full version of the paper is available at <https://arxiv.org>

**Funding** Pankaj Agarwal has been partially supported by NSF grants IIS-18-14493 and CCF-20-07556. Kamesh Munagala is supported by NSF grants CCF-1637397 and IIS-1447554; ONR award N00014-19-1-2268; and DARPA award FA8650-18-C-7880.



© Pankaj K. Agarwal and Hsien-Chih Chang and Kamesh Munagala and Erin Taylor and Emo Welzl; licensed under Creative Commons License CC-BY

Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

Clustering is a fundamental problem in unsupervised learning and data summarization, with wide-ranging applications that span myriad areas. Typically, the data points are assumed to lie in a Euclidean space, and the goal in center-based clustering is to open a set of  $k$  centers to minimize the objective cost, usually a function over the distance from each data point to its closest center. The  $k$ -median objective minimizes the sum of distances; the  $k$ -means minimizes the sum of squares of distances; and the  $k$ -center minimizes the longest distance. In the worst case, all these objectives are NP-hard even in 2D [48, 50].

A substantial body of work has focused on developing polynomial-time approximation algorithms and analyzing natural heuristics for these problems. Given the sheer size of modern data sets, such as those generated in genomics or mapping applications, even a polynomial-time algorithm is too slow to be useful in practice—just computing all pairs of distances can be computationally burdensome. What we need is an algorithm whose running time is near-linear in the input size and polynomial in the number of clusters.

Because of NP-hardness results, we cannot hope to compute an optimal solution in polynomial time, but in the worst case an approximate clustering can be different from an optimal clustering. We focus on the case when the optimal clustering can be recovered under some reasonable assumptions on the input that hold in practice. Such methodology is termed “beyond worst-case analysis” and has been adopted by recent work [2, 8, 23]. In recent years, the notion of *stability* has emerged as a popular assumption under which polynomial-time optimal clustering algorithms have been developed. An instance of clustering is called *stable* if any “small perturbation” of input points does not change the optimal solution. This is natural in real datasets, where often, the optimal clustering is clearly demarcated, and the distances are obtained heuristically. Different notions of stability differ in how “small perturbation” is defined, though most of them are related. In this paper, we focus on the notions of stability introduced in Bilu and Linial [23] and Awasthi, Blum, and Sheffet [14]. A clustering instance is  $\alpha$ -perturbation resilient or  $\alpha$ -stable if the optimal clustering does not change when all distances are perturbed by a factor of at most  $\alpha$ . Similarly, a clustering instance is  $\alpha$ -center proximal if any point is at least a factor of  $\alpha$  closer to its own optimal center than any other optimal center. Awasthi, Blum, and Sheffet showed that  $\alpha$ -stability implies  $\alpha$ -center proximity [14]. This line of work designs algorithms to recover the *exact* optimal clustering—the ground truth—in polynomial time for  $\alpha$ -stable instances.

This paper also focuses on recovering the optimal clustering for stable clustering instances. But instead of focusing on polynomial-time algorithms and optimizing the value of  $\alpha$ , we ask the question: *Can algorithms be designed that compute exact solutions to stable instances of Euclidean center-based clustering that run in time near-linear in the input size?* We note that an  $(1 + \epsilon)$ -approximation solution, for an arbitrarily small constant  $\epsilon > 0$ , may differ significantly from an optimal solution (the ground truth) even for stable instances, so one cannot hope to use an approximation algorithm to recover the optimal clustering.

### 1.1 Our Results

In this paper, we make progress on the above question, and present near-linear time algorithms for finding optimal solutions of stable clustering instances with moderate values of  $\alpha$ . In particular, we show the following meta-theorem:

► **Theorem 1.** *Let  $X$  be a set of  $n$  points in  $\mathbb{R}^d$  for some constant  $d$ , let  $k \geq 1$  be an integer, and let  $\alpha \geq 2 + \sqrt{3}$  be a parameter. If the  $k$ -median,  $k$ -means, or  $k$ -center clustering instance for  $X$  is  $\alpha$ -stable, then the optimal solution can be computed in  $\tilde{O}(n \text{poly } k + f(k))$  time.*

In the above theorem, the  $\tilde{O}$  notation suppresses logarithmic terms in  $n$  and the spread of the point set. The function  $f(k)$  depends on the choice of algorithm, and we present the exact dependence below. We also omit terms depending solely on the dimension,  $d$ . Furthermore, the above theorem is robust in the sense that the algorithm is not restricted to choosing the input points as centers (*discrete setting*), and can potentially choose arbitrary points in the Euclidean plane as centers (*continuous setting*, sometimes referred to as the *Steiner point setting*)—indeed, we show that these notions are identical under a reasonable assumption on stability.

At a more fine-grained level, we present several algorithms that require mild assumptions on the stability condition. In the results below, as well as throughout the paper, we present our results both for the Euclidean plane, as well as generalizations to higher (but fixed number of) dimensions.

**Dynamic Programming.** In Section 3, we present a dynamic programming algorithm that computes the optimal clustering in  $O(nk^2 + n \text{polylog } n)$  time for  $\alpha$ -stable  $k$ -means,  $k$ -median, and  $k$ -center in any fixed dimension, provided that  $\alpha \geq 2 + \sqrt{3} + \varepsilon$  for any constant  $\varepsilon > 0$ . For  $d = 2$ , it suffices to assume that  $\alpha \geq 2 + \sqrt{3}$ .

**Local Search.** In Sections 4 and 5, we show that the standard 1-swap local-search algorithm, which iteratively swaps out a center in the current solution for a new center as long as the resulting total cost improves, computes an optimal clustering for  $\alpha$ -stable instances of  $k$ -median assuming  $\alpha > 5$ . We also show that it can be implemented in  $O(nk^2 \log^3 n \log \Delta)$  for  $d = 2$  and in  $O(nk^{2d-1} \text{polylog } n \log \Delta)$  for  $d > 2$ ;  $\Delta$  is the spread of the point set.<sup>1</sup>

**Coresets.** In the full version of the paper, we use multiplicative coresets to compute the optimal clustering for  $k$ -means,  $k$ -median and  $k$ -center in any fixed dimension, when  $\alpha \geq 2 + \sqrt{3}$ . The running time is  $O(nk^2 + f(k))$  where  $f(k)$  is an exponential function of  $k$ .

► **Remark 2.** (1) While the current analysis of the dynamic programming based algorithm suggests that it is better than the local-search and coreset based approaches, the latter are of independent interest—our local-search analysis is considerably simpler than the previous analysis [38], and coresets have mostly been used to compute approximate, rather than exact, solutions. We also note that our analysis of the local-search algorithm is probably not tight. Furthermore, variants of all three approaches might work for smaller values of  $\alpha$ . (2) We note that the value of  $\alpha$  assumed in the above results is larger than what is known for polynomial-time algorithm (e.g.  $\alpha \geq 2$  in Angelidakis *et al.* [10]) and that in some applications the input may not satisfy our assumption, but our results are a big first step toward developing near-linear time algorithms for stable instances. We are not aware of any previous near-linear time algorithms for computing optimal clustering even for larger values of  $\alpha$ . We leave the problem of reducing the assumption on  $\alpha$  as an important open question.

**Techniques.** The key difficulty with developing fast algorithms for computing the optimal clustering is that some clusters could have a very small size compared to others. This issue persists even when the instances are stable. Imagine a scenario where there are multiple small clusters, and an algorithm must decide whether to merge these into one cluster while splitting some large cluster, or keep them intact. Now imagine this situation happening recursively, so that the algorithm has multiple choices about which clusters to recursively

<sup>1</sup>The *spread* of a point set is the ratio between the longest and shortest pairwise distances.

split. The difference in cost between these options and the size of the small clusters can be small enough that any  $(1 + \epsilon)$ -approximation can be agnostic, while an exact solution cannot. As such, work on finding exact optima use techniques such as dynamic programming [10] or local search with large number of swaps [26, 38] in order to recover small clusters. Other work makes assumptions lower-bounding the size of the optimal clusters or the spread of their centers [34].

Our main technical insight for the first two results is simple in hindsight, yet powerful: For a stable instance, if the Euclidean metric is replaced by another metric that is a good approximation, then the optimal clustering does not change under the new metric and in fact the instance remains stable albeit with a smaller stability parameter. In particular, we replace the Euclidean metric with an appropriate *polyhedral metric*—that is, a convex distance function where each unit ball is a regular polyhedron—yielding efficient procedures for the following two primitives:

- **Cost of 1-swap.** Given a candidate set of centers  $S$ , maintain a data structure that efficiently updates the total cost if center  $x \in S$  is replaced by center  $y \notin S$ .
- **Cost of 1-clustering.** Given a partition of the data points, maintain a data structure where the cost of 1-clustering (under any objectives) can be efficiently updated as partitions are merged.

We next combine the insight of changing the metrics with additional techniques. For local search, we build on the approach in [26, 31, 38] that shows local search with  $t$ -swaps for large enough constant  $t$  finds an optimal solution for stable instances in polynomial time for any fixed-dimension Euclidean space. None of the prior analysis directly extends as is to 1-swap, which is critical in achieving near-linear running time—note that even when  $t = 2$  there is a quadratic number of candidate swaps per step.

For the dynamic programming algorithm, we use the following insight: In Euclidean spaces, for  $\alpha \geq 2 + \sqrt{3}$ , the longest edge of the minimum spanning tree over the input points partitions the data set in two, such that any optimal cluster lies completely in one of the two sides of the partition. Combined with the change of metrics one can achieve near-linear running time.

Due to length constraints of the paper, the coresets result, most of algorithmic details, and many proofs can be found in the full version of the paper.

## 1.2 Related Work

All of  $k$ -median,  $k$ -means, and  $k$ -center are widely studied from the perspective of approximation algorithms and are known to be hard to approximate [36]. Indeed, for general metric spaces,  $k$ -center is hard to approximate to within a factor of  $2 - \epsilon$  [43];  $k$ -median is hard to  $(1 + 2/e)$ -approximate [44]; and  $k$ -means is hard to  $(1 + 8/e)$ -approximate in general metrics [29], and is hard to approximate within a factor of 1.0013 in the Euclidean setting [47]. Even when the metric space is Euclidean,  $k$ -means is still NP-hard when  $k = 2$  [7, 32], and there is an  $n^{\Omega(k)}$  lower bound on running time for  $k$ -median and  $k$ -means in 4-dimensional Euclidean space under the exponential-time hypothesis [27].

There is a long line of work in developing  $(1 + \epsilon)$ -approximations for these problems in Euclidean spaces. The holy grail of this work has been the development of algorithms that are near-linear time in  $n$ , and several techniques are now known to achieve this. This includes randomly shifted quad-trees [11], coresets [4, 15, 37, 40, 41], sampling [46], and local search [26, 28, 30], among others.

There are many notions of clustering stability that have been considered in literature [1, 6, 13, 17, 18, 22, 35, 45, 52]. The exact definition of stability we study here was first introduced in Awasthi *et al.* [14]; their definition in particular resembles the one of Bilu and Linial [23] for max-cut problem, which later has been adapted to other optimization problems [9, 10, 19, 49, 51]. Building on a long line of work [14, 16, 20, 21], which gradually reduced the stability parameter, Angelidakis *et al.* [10] present a dynamic programming based polynomial-time optimal algorithm for discrete 2-stable instances for all center-based objectives.

Chekuri and Gupta [25] show that a natural LP-relaxation is integral for the 2-stable  $k$ -center problem. Recent work by Cohen-Addad [31] provides a framework for analyzing local search algorithms for stable instances. This work shows that for an  $\alpha$ -stable instance with  $\alpha > 3$ , any solution is optimal if it cannot be improved by swapping  $\lceil 2/(\alpha - 3) \rceil$  centers. Focusing on Euclidean spaces of fixed dimensions, Friggstad *et al.* [38] show that a local-search algorithm with  $O(1)$ -swaps runs in polynomial time under a  $(1 + \delta)$ -stable assumption for any  $\delta > 0$ . However, none of the algorithms for stable instances of clustering so far have running time near-linear in  $n$ , even when the stability parameter  $\alpha$  is large, points lie in  $\mathbb{R}^2$ , and the underlying metric is Euclidean.

On the hardness side, solving  $(3 - \delta)$ -center proximal  $k$ -median instances in general metric spaces is NP-hard for any  $\delta > 0$  [14]. When restricted to Euclidean spaces in arbitrary dimensions, Ben-David and Reyzin [22] showed that for every  $\delta > 0$ , solving discrete  $(2 - \delta)$ -center proximal  $k$ -median instances is NP-hard. Similarly, the clustering problem for discrete  $k$ -center remains hard for  $\alpha$ -stable instances when  $\alpha < 2$ , assuming standard complexity assumption that  $\text{NP} \neq \text{RP}$  [20]. Under the same complexity assumption, discrete  $\alpha$ -stable  $k$ -means is also hard when  $\alpha < 1 + \delta_0$  for some positive constant  $\delta_0$  [38]. Deshpande *et al.* [34] showed it is NP-hard to  $(1 + \epsilon)$ -approximate  $(2 - \delta)$ -center proximal  $k$ -means instances.

## 2 Definitions and Preliminaries

**Clustering.** Let  $X = \{p_1, \dots, p_n\}$  be a set of  $n$  points in  $\mathbb{R}^d$ , and let  $\delta: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}_{\geq 0}$  be a distance function (not necessarily a metric satisfying triangle inequality). For a set  $Y \subseteq \mathbb{R}^d$ , we define  $\delta(p, Y) := \min_{y \in Y} \delta(p, y)$ . A  *$k$ -clustering* of  $X$  is a partition of  $X$  into  $k$  non-empty *clusters*  $X_1, \dots, X_k$ . We focus on center-based clusterings that are induced by a set  $S := \{c_1, \dots, c_k\}$  of  $k$  *centers*; each  $X_i$  is the subset of points of  $X$  that are closest to  $c_i$  in  $S$  under  $\delta$ , that is,  $X_i := \{p \in X \mid \delta(p, c_i) \leq \delta(p, c_j)\}$  (ties are broken arbitrarily). Assuming the nearest neighbor of each point of  $X$  in  $S$  is unique (under distance function  $\delta$ ),  $S$  defines a  $k$ -clustering of  $X$ . Sometimes it is more convenient to denote a  $k$ -clustering by its set of centers  $S$ .

The quality of a clustering  $S$  of  $X$  is defined using a cost function  $\$(X, S)$ ; cost function  $\$$  depends on the distance function  $\delta$ , so sometimes we may use the notation  $\$_\delta$  to emphasize the underlying distance function. The goal is to compute  $S^* := \arg \min_S \$(X, S)$  where the minimum is taken over all subsets  $S \subset \mathbb{R}^d$  of  $k$  points. Several different cost functions have been proposed, leading to various optimization problems. We consider the following three popular variants:

- *$k$ -median clustering*: the cost function is  $\$(X, S) = \sum_{p \in X} \delta(p, S)$ .
- *$k$ -means clustering*: the cost function is  $\$(X, S) = \sum_{p \in X} (\delta(p, S))^2$ .
- *$k$ -center clustering*: the cost function is  $\$(X, S) = \max_{p \in X} \delta(p, S)$ .

In some cases we wish  $S$  to be a subset of  $X$ , in which case we refer to the problem as the *discrete  $k$ -clustering* problem. For example, the discrete  $k$ -median problem is to compute

## XX:6 Clustering under Perturbation Stability in Near-Linear Time

217  $\arg \min_{S \subseteq X, |S|=k} \sum_{p \in X} \delta(p, S)$ . The discrete  $k$ -means and discrete  $k$ -center problems are  
 218 defined analogously.

219 Given point set  $X$ , distance function  $\delta$ , and cost function  $\$$ , we refer to  $(X, \delta, \$)$  as a  
 220 *clustering instance*. If  $\$$  is defined directly by the distance function  $\delta$ , we use  $(X, \delta)$  to denote  
 221 a clustering instance. Note that a center of a set of points may not be unique (e.g. when  $\delta$  is  
 222 defined by the  $L_1$ -metric and  $\$$  is the sum of distances) or it may not be easy to compute  
 223 (e.g. when  $\delta$  is defined by the  $L_2$ -metric and  $\$$  is the sum of distances).

224 **Stability.** Let  $X$  be a point set in Euclidean space  $\mathbb{R}^d$ . For  $\alpha \geq 1$ , a clustering instance  
 225  $(X, \delta, \$)$  is  *$\alpha$ -stable* if for any *perturbed distance function*  $\tilde{\delta}$  (not necessary a metric) satisfying  
 226  $\delta(p, q) \leq \tilde{\delta}(p, q) \leq \alpha \cdot \delta(p, q)$  for all  $p, q \in \mathbb{R}^d$ , any optimal clustering of  $(X, \delta, \$)$  is also an  
 227 optimal clustering of  $(X, \tilde{\delta}, \$)$ . Note that the cluster centers as well as the cost of optimal  
 228 clustering may be different for the two instances. We exploit the following property of  
 229 stability, which follows directly from its definition.

230 ► **Lemma 3.** *Let  $(X, \delta)$  be an  $\alpha$ -stable clustering instance with  $\alpha > 1$ . Then the optimal*  
 231 *clustering  $O$  of  $(X, \delta)$  is unique.*

232 **Metric approximations.** The next lemma, which we rely on heavily throughout the paper,  
 233 is the observation that a change of metric preserves the optimal clustering as long as the  
 234 new metric is a  $\beta$ -approximation of the original metric satisfying  $\beta < \alpha$ .

235 ► **Lemma 4.** *Given point set  $X$ , let  $\delta$  and  $\delta'$  be two metrics satisfying  $\delta(p, q) \leq \delta'(p, q) \leq$   
 236  $\beta \cdot \delta(p, q)$  for all  $p$  and  $q$  in  $X$  for some  $\beta$ . Let  $(X, \delta)$  be an  $\alpha$ -stable clustering instance with  
 237  $\alpha > \beta$ . Then the optimal clustering of  $(X, \delta)$  is also the optimal clustering of  $(X, \delta')$ , and  
 238 vice versa. Furthermore,  $(X, \delta')$  is an  $(\alpha/\beta)$ -stable clustering instance.*

239 **Polyhedral metric.** In light of the metric approximation lemma, we would like to approxi-  
 240 mate the Euclidean metric without losing too much stability, using a collection of convex  
 241 distance functions generalizing the  $L_\infty$ -metric in Euclidean space. Let  $N \subseteq S^{d-1}$  be a  
 242 centrally-symmetric set of  $\gamma$  unit vectors (that is, if  $u \in N$  then  $-u \in N$ ) such that for any  
 243 unit vector  $v \in S^{d-1}$ , there is a vector  $u \in N$  within angle  $\arccos(1 - \epsilon) = O(\sqrt{\epsilon})$ . The  
 244 number of vectors needed in  $N$  is known to be  $O(\epsilon^{-(d-1)/2})$ . We define the *polyhedral metric*  
 245  $\delta_N: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}_{\geq 0}$  to be  $\delta_N(p, q) := \max_{u \in N} \langle p - q, u \rangle$ .

246 Since  $N$  is centrally symmetric,  $\delta_N$  is symmetric and thus a metric. The unit ball under  
 247  $\delta_N$  is a convex polyhedron, thus the name polyhedral metric. By construction, an easy  
 248 calculation shows that for any  $p$  and  $q$  in  $\mathbb{R}^d$ ,  $\|p - q\| \geq \delta_N(p, q) \geq (1 - \epsilon) \cdot \|p - q\|$ . By scaling  
 249 each vector in  $N$  by a  $1 + \epsilon$  factor, we can ensure that  $(1 + \epsilon) \cdot \|p - q\| \geq \delta_N(p, q) \geq \|p - q\|$ .  
 250 By taking  $\epsilon$  to be small enough, the optimal clustering for  $\alpha$ -stable clustering instance  
 251  $(X, \|\cdot\|, \$)$  is the same as that for  $(X, \delta_N, \$)$  by Lemma 4, and the new instance  $(X, \delta_N, \$)$  is  
 252  $(1 - \epsilon)\alpha$ -stable if the original instance  $(X, \|\cdot\|, \$)$  is  $\alpha$ -stable.

253 **Center proximity.** A clustering instance  $(X, \delta)$  satisfies  *$\alpha$ -center proximity property* [14] if  
 254 for any distinct optimal clusters  $X_i$  and  $X_j$  with centers  $c_i$  and  $c_j$  and any point  $p \in X_i$ ,  
 255 one has  $\alpha \cdot \delta(p, c_i) < \delta(p, c_j)$ . Awasthi, Blum, and Sheffet showed that any  $\alpha$ -stable instance  
 256 satisfies  $\alpha$ -center proximity [14, Fact 2.2] (also [10, Theorem 3.1] under metric perturbation).  
 257 Optimal solutions of  $\alpha$ -stable instances satisfy the following separation properties.<sup>2</sup>

<sup>2</sup>We give an additional list of known separation properties in the full version of the paper.



- 258 ■  $\alpha$ -center proximity implies that  $(\alpha - 1) \cdot \delta(p, c_i) < \delta(p, q)$  for any  $p \in X_i$  and any  $q \notin X_i$ .  
 259 For  $\alpha \geq 2$ , a point is closer to its own center than to any point of another cluster.<sup>3</sup>
- 260 ■ For  $\alpha \geq 2 + \sqrt{3}$ ,  $\alpha$ -center proximity implies that  $\delta(p, p') < \delta(p, q)$  for any  $p, p' \in$   
 261  $X_i$  and any  $q \notin X_i$ . In other words, from any point  $p$  in  $X$ , any *intra*-cluster distance to  
 262 a point  $p'$  is shorter than any *inter*-cluster distance to a point  $q$ .<sup>3</sup>

263 We make use of the following stronger intra-inter distance property on  $\alpha$ -stable instances,  
 264 which allows us to compare *any* intra-distance between two points in  $X_i$  and *any* inter-distance  
 265 between a point in  $X_i$  and a point in  $X_j$ .

266 ► **Lemma 5.** *Let  $(X, \delta)$  be an  $\alpha$ -stable instance,  $\alpha > 1$ , and let  $X_1$  be a cluster in an optimal*  
 267 *clustering with  $q \in X \setminus X_1$  and  $p, p', p'' \in X_1$ . If  $\delta$  is a metric, then  $\delta(p, p') \leq \delta(p'', q)$  for*  
 268  *$\alpha \geq 2 + \sqrt{5}$ . If  $\delta$  is the Euclidean metric in  $\mathbb{R}^d$ , then  $\delta(p, p') \leq \delta(p'', q)$  for  $\alpha \geq 2 + \sqrt{3}$ .*

269 Finally, we note that it is enough to consider the discrete version of the clustering problem  
 270 for stable instances.

271 ► **Lemma 6.** *For any  $\alpha$ -stable instance  $(X, \delta, \$)$  with  $\alpha \geq 2 + \sqrt{3}$ , any continuous optimal*  
 272  *$k$ -clustering is a discrete optimal  $k$ -clustering and vice versa.*

### 273 3 Efficient Dynamic Programming

274 We now describe a simple, efficient algorithm for computing the optimal clustering for  
 275 the  $k$ -means,  $k$ -center, and  $k$ -median problem assuming the given instance is  $\alpha$ -stable for  
 276  $\alpha \geq 2 + \sqrt{3}$ . Roughly speaking, we make the following observation: if there are at least two  
 277 clusters, then the two endpoints of the longest edge of the minimum spanning tree of  $X$  belong  
 278 to different clusters, and no cluster has points in both subtrees of the minimum spanning  
 279 tree delimited by the longest edge. We describe the dynamic programming algorithm in  
 280 Section 3.1 and then describe the procedure for computing cluster costs in Section 3.2. We  
 281 summarize the results in this section by the following theorem.

282 ► **Theorem 7.** *Let  $X$  be a set with  $n$  points lying in  $\mathbb{R}^d$  and  $k \geq 1$  an integer. If the  $k$ -means,*  
 283  *$k$ -median, or  $k$ -center instance for  $X$  under the Euclidean metric is  $\alpha$ -stable for  $\alpha \geq 2 + \sqrt{3} + \varepsilon$*   
 284 *for any constant  $\varepsilon > 0$ , then the optimal clustering can be computed in  $O(nk^2 + n \text{ polylog } n)$*   
 285 *time. For  $d = 2$  the assumption can be relaxed to  $\alpha \geq 2 + \sqrt{3}$ .*

#### 286 3.1 Fast Dynamic Programming

287 The following lemma is the key observation for our algorithm.

288 ► **Lemma 8.** *Let  $(X, \delta, \$)$  be an  $\alpha$ -stable  $k$ -clustering instance with  $\alpha \geq 2 + \sqrt{3}$  and  $k \geq 2$ ,*  
 289 *and let  $T$  be the minimum spanning tree of  $X$  under metric  $\delta$ . Then (1) The two endpoints*  
 290  *$u$  and  $v$  of the longest edge  $e$  in  $T$  do not belong to the same cluster; (2) each cluster lies in*  
 291 *the same connected component of  $T \setminus \{e\}$ .*

292 **Algorithm.** We fix the metric  $\delta$  and the cost function  $\$$ . For a subset  $Y \subseteq X$  and for an  
 293 integer  $j$  between 1 and  $k - 1$ , let  $\mu(Y; j)$  denote the optimal cost of an  $j$ -clustering on  
 294  $Y$  (under  $\delta$  and  $\$$ ). Recall that our definition of  $j$ -clustering required all clusters to be

---

<sup>3</sup>They are known as *weak center proximity* [20] and *strict separation property* [18, 22] respectively.

non-empty, so it is not defined for  $|Y| < j$ . For simplicity, we assume that  $\mu(Y; j) = \infty$  for  $|Y| < j$ . Let  $T$  be the minimum spanning tree on  $X$  under  $\delta$ , let  $uv$  be the longest edge in  $T$ ; let  $X_u$  and  $X_v$  be the set of vertices of the two components of  $T \setminus \{uv\}$ . Then  $\mu(X; k)$  satisfies the following recurrence relation:

$$\mu(X; k) = \begin{cases} \mu(X; 1) & \text{if } k = 1, \\ \infty & \text{if } k > |X|, \\ \min_{1 \leq i < k} \{\mu(X_u; i) + \mu(X_v; k - i)\} & \text{if } |X| > 1 \text{ and } k > 1. \end{cases} \quad (1)$$

Using recurrence (1), we compute  $\mu(X; k)$  as follows. Let  $R$  be a *recursion tree*, a binary tree where each node  $v$  in  $R$  is associated with a subtree  $T_v$  of  $T$ . If  $v$  is the root of  $R$ , then  $T_v = T$ . Recursion tree  $R$  is defined recursively as follows. Let  $X_v \subseteq X$  be the set of vertices of  $T$  in  $T_v$ . If  $|X_v| = 1$ , then  $v$  is a leaf. Each interior node  $v$  of  $T$  is also associated with the longest edge  $e_v$  of  $T_v$ . Removal of  $e_v$  decomposes  $T_v$  into two connected components, each of which is associated with one of the children of  $v$ . After having computed  $T$ ,  $R$  can be computed in  $O(n \log n)$  time by sorting the edges in decreasing order of their costs.<sup>4</sup>

For each node  $v \in R$  and for every  $i$  between 1 and  $k - 1$ , we compute  $\mu(X_v; i)$  as follows. If  $v$  is a leaf, we set  $\mu(X_v; 1) = 0$  and  $\mu(X_v; i) = \infty$  otherwise. For all interior nodes  $v$ , we compute  $\mu(X_v; 1)$  using the algorithms described in Section 3.2. Finally, if  $v$  is an interior node and  $i > 1$ , we compute  $\mu(X_v; i)$  using the recurrence relation (1). Recall that if  $w$  and  $z$  are the children of  $v$ , then  $\mu(X_w; \ell)$  and  $\mu(X_z; r)$  for all  $\ell$  and  $r$  have been computed before we compute  $\mu(X_v; i)$ .

Let  $\tau(n)$  be the time spent in computing  $T$  plus the total time spent in computing  $\mu(X_v, 1)$  for all nodes  $v \in R$ . Then the overall time taken by the algorithm is  $O(nk^2 + \tau(n))$ . What is left is to compute the minimum spanning tree  $T$  and all  $\mu(X_v, 1)$  efficiently.

### 3.2 Efficient Implementation

In this section, we show how to obtain the minimum spanning tree and compute  $\mu(X_v; 1)$  efficiently for 1-mean, 1-center, and 1-median when  $X \subseteq \mathbb{R}^d$ . We can compute the Euclidean minimum spanning tree  $T$  in  $O(n \log n)$  time in  $\mathbb{R}^2$  [54]. We can then compute  $\mu(X_v; 1)$  efficiently either under Euclidean metric (for 1-mean), or switch to the  $L_1$ -metric and compute  $\mu(X_v; 1)$  efficiently using Lemma 4 (for 1-center and 1-median).

There are two difficulties in extending the 2D data structures to higher dimensions. No near-linear time algorithm is known for computing the Euclidean minimum spanning tree for  $d \geq 3$ , and we can work with the  $L_1$ -metric only if  $\alpha \geq \sqrt{d}$  (Lemma 4). We address both of these difficulties by working with a polyhedral metric  $\delta_N$ . Let  $\alpha \geq 2 + \sqrt{3} + \Omega(1)$  be the stability parameter. By taking the number of vectors in  $N$  (defined by the polyhedral metric) to be large enough, we can ensure that  $(1 - \epsilon)\|p - q\| \leq \delta_N(p, q) \leq \|p - q\|$  for all  $p, q \in \mathbb{R}^d$ . By Lemma 4,  $X$  is an  $\alpha$ -stable instance under  $\delta_N$  for  $\alpha \geq 2 + \sqrt{3}$ . We first compute the minimum spanning tree of  $X$  in  $O(n \text{ polylog } n)$  time under  $\delta_N$  using the result of Callahan and Kosaraju [24], and then compute  $\mu(X_v, 1)$ .

**Data structure.** We compute  $\mu(X_v; 1)$  in a bottom-up manner. When processing a node  $v$  of  $R$ , we maintain a dynamic data structure  $\Psi_v$  on  $X_v$  from which  $\mu(X_v; 1)$  can be computed

<sup>4</sup>Tree  $R$  is nothing but the minimum spanning tree constructed by Kruskal's algorithm.



quickly. The exact form of  $\Psi_v$  depends on the cost function to be described below. Before that, we analyze the running time  $\tau(n)$  spent on computing every  $\mu(X_v; 1)$ . Let  $w$  and  $z$  be the two children of  $v$ . Suppose we have  $\Psi_w$  and  $\Psi_z$  at our disposal and suppose  $|X_w| \leq |X_z|$ . We insert the points of  $X_w$  into  $\Psi_z$  one by one and obtain  $\Psi_v$  from which we compute  $\mu(X_v; 1)$ . Suppose  $Q(n)$  is the update time of  $\Psi_v$  as well as the time taken to compute  $\mu(X_v; 1)$  from  $\Psi_v$ . The total number of insert operations performed over all nodes of  $R$  is  $O(n \log n)$  because we insert the points of the smaller set into the larger set at each node of  $R$  [42, 53]. Hence  $\tau(n) = O(Q(n) \cdot n \log n)$ . We now describe the data structure for each specific clustering problem.

**1-mean.** We work with the  $L_2$ -metric. Here the center of a single cluster consisting of  $X_v$  is the centroid  $\sigma_v := (\sum_{p \in X_v} p) / |X_v|$ , and  $\mu(X_v; 1) = \sum_{p \in X_v} \|p\|^2 - |X_v| \cdot \|\sigma_v\|^2$ . At each node  $v$ , we maintain  $\sum_{p \in X_v} p$  and  $\sum_{p \in X_v} \|p\|^2$ . Point insertion takes  $O(1)$  time so  $Q(n) = 1$ .

**1-center.** As mentioned in the beginning of the section, we can work with the  $L_1$ -metric for  $d = 2$ . We wish to find the smallest  $L_1$ -disc (a diamond) that contains  $X_v$ . Let  $e^+ = (1, 1)$  and  $e^- = (-1, 1)$ . Then the radius  $\rho_v$  of the smaller  $L_1$ -disc containing  $X_v$  is

$$\rho_v = \frac{1}{2} \max \left\{ \max_{p \in X_v} \langle p, e^+ \rangle - \min_{p \in X_v} \langle p, e^+ \rangle, \max_{p \in X_v} \langle p, e^- \rangle - \min_{p \in X_v} \langle p, e^- \rangle \right\}. \quad (2)$$

We maintain the four terms  $\max_{p \in X_v} \langle p, e^+ \rangle$ ,  $\min_{p \in X_v} \langle p, e^+ \rangle$ ,  $\max_{p \in X_v} \langle p, e^- \rangle$ , and  $\min_{p \in X_v} \langle p, e^- \rangle$  at  $v$ . A point can be inserted in  $O(1)$  time and  $\rho_v$  can be computed from these four terms in  $O(1)$  time. Therefore,  $Q(n) = O(1)$ . For  $d > 2$ , we work with a polyhedral metric and compute the smallest ball  $B(X_v)$  that contains  $X_v$ . For full details, see the full version of the paper.

**1-median.** Similar to 1-center, we work with the polyhedral metric. Fix a node  $v$  of  $T$ . For a point  $x \in \mathbb{R}^d$ , let  $F_v(x) = \sum_{p \in X_v} \delta_N(x, p)$  which is a piecewise-linear function. Our goal is to compute  $\xi_v^* = \arg \min_{x \in \mathbb{R}^d} F_v(x)$ . Our data structure is a dynamic range-tree [3] used for orthogonal range searching that can insert a point in  $O(\log n)$  time. Using multi-dimensional parametric search [5],  $\xi_v^*$  can be computed in  $O(\text{poly log } n)$  time after each update; see the full version of the paper for details.

## 4 $k$ -Median: Single-Swap Local Search

We customize the standard local-search framework for the  $k$ -clustering problem [30, 31, 39]. In order to recover the optimal solution, we must define near-optimality more carefully. Let  $(X, \delta)$  be an instance of  $\alpha$ -stable  $k$ -median in  $\mathbb{R}^2$  for  $\alpha > 5$ . By Lemma 6, it suffices to consider the *discrete*  $k$ -median problem. In Section 4, we describe a simple local-search algorithm for finding the optimal clustering of  $(X, \delta)$ . In Section 4 we show that the algorithm terminates within  $O(k \log(n\Delta))$  iterations. We obtain the following.

► **Theorem 9.** *Let  $(X, \delta)$  be an  $\alpha$ -stable instance of the  $k$ -median problem for some  $\alpha > 5$  where  $X$  is a set of  $n$  points in  $\mathbb{R}^2$  equipped with  $L_p$ -metric  $\delta$ . The 1-swap local search algorithm terminates with the optimal clustering in  $O(k \log(n\Delta))$  iterations.*

## XX:10 Clustering under Perturbation Stability in Near-Linear Time

**Local-search algorithm.** The local-search algorithm maintains a  $k$ -clustering induced by a set  $S$  of  $k$  cluster centers. At each step, it finds a pair of points  $x \in X$  and  $y \in S$  such that  $\$(X, S + x - y)$  is minimized. If  $\$(X, S + x - y) \geq \$(X, S)$ , it stops and returns the  $k$ -clustering induced by  $S$ . Otherwise it replaces  $S$  with  $S + x - y$  and repeats the above step. The pair  $(x, y)$  will be referred to as a **1-swap**.

**Local-search analysis.** The high-level structure of our analysis follows Friggstad *et al.* [39], however new ideas are needed for 1-swap. In this subsection, we denote a  $k$ -clustering by the set of its cluster centers. Let  $S$  be a fixed  $k$ -clustering, and let  $O$  be the optimal clustering. For a subset  $Y \subseteq X$ , we use  $\$(Y)$  and  $\$(Y)$  to denote  $\$(Y, S)$  and  $\$(Y, O)$ , respectively. Similarly, for a point  $p \in X$ , we use  $\text{nn}(p)$  and  $\text{nn}^*(p)$  to denote the nearest neighbor of  $p$  in  $S$  and in  $O$ , respectively; define  $\delta(p)$  to be  $\delta(p, S)$  and  $\delta^*(p)$  to be  $\delta(p, O)$ . We partition  $X$  into four subsets as follows:

$$\begin{aligned} X_{00} &:= \{p \in X \mid \text{nn}(p) \in S \setminus O, \text{nn}^*(p) \in O \setminus S\}; \\ X_{01} &:= \{p \in X \mid \text{nn}(p) \in S \setminus O, \text{nn}^*(p) \in S \cap O\}; \\ X_{10} &:= \{p \in X \mid \text{nn}(p) \in S \cap O, \text{nn}^*(p) \in O \setminus S\}; \\ X_{11} &:= \{p \in X \mid \text{nn}(p) \in S \cap O, \text{nn}^*(p) \in S \cap O\}. \end{aligned}$$

Observe that for any point  $p$  in  $X_{11}$ ,  $\text{nn}(p) = \text{nn}^*(p)$  and  $\$(p) = \$(p)$ ; for any point  $p$  in  $X_{01}$ , one has  $\$(p) \leq \$(p)$ ; and for any point  $p$  in  $X_{10}$ , one has  $\$(p) \geq \$(p)$ . Costs  $\delta(p)$  and  $\delta^*(p)$  are not directly comparable for point  $p$  in  $X_{00}$ . A  $k$ -clustering  $S$  is  **$C$ -good** for some parameter  $C \geq 0$  if  $\$(X) \leq \$(X) + C \cdot \$(X_{00})$ .

► **Lemma 10.** Any  $C$ -good clustering  $S$  for an  $\alpha$ -stable clustering instance  $(X, \delta, \$)$  must be optimal for  $\alpha \geq C + 1$ .

**Proof.** Define a perturbed distance function  $\tilde{\delta}: X \times X \rightarrow \mathbb{R}_{\geq 0}$  with respect to the given clustering  $S$  as follows:

$$\tilde{\delta}(p', p) := \begin{cases} \alpha \cdot \delta(p', p) & \text{if } p \neq \text{nn}(p'), \\ \delta(p', p) & \text{otherwise.} \end{cases}$$

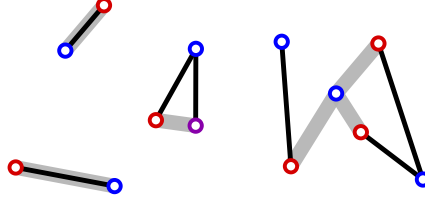
Note that  $\tilde{\delta}$  is not symmetric. Let  $\tilde{\$}(\cdot, \cdot)$  denote the cost function under the perturbed distance function  $\tilde{\delta}$ . The optimal clustering under perturbed cost function is the same as the original optimal clustering  $O$  by the stability assumption. Since  $\text{nn}(p) = \text{nn}^*(p)$  if and only if  $p \in X_{11}$ , the cost of  $O$  under the perturbed cost can be written as:

$$\tilde{\$}(X, O) = \alpha \cdot \$(X_{00}, O) + \alpha \cdot \$(X_{01}, O) + \alpha \cdot \$(X_{10}, O) + \$(X_{11}, O).$$

By definition of perturbed distance  $\tilde{\delta}$ ,  $\tilde{\$}(X, S) = \$(X, S)$ . Now, by the assumption that clustering  $S$  is  $C$ -good,

$$\begin{aligned} \tilde{\$}(X, S) &= \$(X, S) \leq \$(X, O) + C \cdot \$(X_{00}, O) \\ &\leq (C + 1) \cdot \$(X_{00}, O) + \$(X_{01}, O) + \$(X_{10}, O) + \$(X_{11}, O) \\ &\leq \tilde{\$}(X, O); \end{aligned}$$

the last inequality follows by taking  $\alpha \geq C + 1$ . This implies that  $S$  is an optimal clustering for  $(X, \tilde{\delta})$ , and thus is equal to  $O$ . ◀



■ **Figure 1.** Illustration of candidate swaps  $\mathcal{S}$  in  $\mathbb{R}^2$ . The blue dots belong to set  $S$ , the red dots belong to set  $O$ ; the only purple dot is in  $S \cap O$ . The thick gray segments indicate pairs inside the stars; each star has exact one blue dot as its center. The black pairs are the candidate swaps. Notice that the partitions of  $S$  and  $O$  form connected components.

408 Next, we prove a lower bound on the improvement in the cost of a clustering that is not  
 409  $C$ -good after performing a 1-swap. Following Arya *et al.* [12], define the set of *candidate*  
 410 *swaps*  $\mathcal{S}$  as follows: For each center  $i$  in  $S$ , consider the *star*  $\Sigma_i$  centered at  $i$  defined as the  
 411 collection of pairs  $\Sigma_i := \{(i, j) \in S \times O \mid \text{nn}(j) = i\}$ . Denote  $\text{center}(j)$  to be the center of the  
 412 star where  $j$  belongs; in other words,  $\text{center}(j) = i$  if  $j$  belongs to  $\Sigma_i$ .

413 For  $i \in S$ , let  $O_i := \{j \in O \mid \text{center}(j) = i\}$  be the set of centers of  $O$  in star  $\Sigma_i$ . If  $|O_i| = 1$ ,  
 414 then we add the only pair  $(i, j) \in \Sigma_i$  to the candidate set  $\mathcal{S}$ . Let  $S_\emptyset := \{i \in S \mid O_i = \emptyset\}$ .  
 415 Let  $O_{>1}$  contain centers in  $O$  that belong to a star of size greater than 1. We pick  $|O_{>1}|$   
 416 pairs from  $S_\emptyset \times O_{>1}$  such that each point of  $O_{>1}$  is matched only once and each point of  $S_\emptyset$   
 417 is matched at most twice and add them to  $\mathcal{S}$ ; this is feasible because  $|S_\emptyset| \geq |O_{>1}|/2$ . Since  
 418 each center in  $O$  belongs to exactly one pair of  $\mathcal{S}$ ,  $|\mathcal{S}| = k$ . By construction, if  $|\Sigma_i| \geq 2$ , then  
 419  $i$  does not belong to any candidate swap. See Figure 1.

420 ► **Lemma 11.** For each point  $p$  in  $X_{01}$ ,  $X_{10}$ , or  $X_{11}$ , the set of candidate swaps  $\mathcal{S}$  satisfies

$$421 \quad \sum_{(i,j) \in \mathcal{S}} (\delta(p) - \delta'(p)) \geq \delta(p) - \delta^*(p); \quad (3)$$

423 and for each point  $p$  in  $X_{00}$ , the set of candidate swaps  $\mathcal{S}$  satisfies

$$424 \quad \sum_{(i,j) \in \mathcal{S}} (\delta(p) - \delta'(p)) \geq (\delta(p) - \delta^*(p)) - 4\delta^*(p), \quad (4)$$

426 where  $\delta'$  is the cost function on  $X$  defined with respect to  $S' := S - i + j$ , and  $\delta'(p)$  is the  
 427 distance between  $p$  and its nearest neighbor in  $S'$ .

428 **Proof.** For point  $p$  in  $X_{11}$ , both  $\text{nn}(p)$  and  $\text{nn}^*(p)$  are in  $S'$ , so  $\delta'(p) = \delta(p) = \delta^*(p)$ . For  
 429 point  $p$  in  $X_{01}$ ,  $\delta(p) \leq \delta^*(p)$ ; when  $\text{nn}(p)$  is being swapped out by some in 1-swap  $S'$ ,  $\text{nn}^*(p)$   
 430 must be in  $S'$ . For point  $p$  in  $X_{10}$ ,  $\delta(p) \geq \delta^*(p)$ ; center  $\text{nn}(p)$  will never be swapped out by  
 431 any 1-swap in  $\mathcal{S}$ , so  $\delta'(p) \leq \delta(p)$ . By construction of  $\mathcal{S}$ , there is exactly one choice of  $S'$  that  
 432 swaps  $\text{nn}^*(p)$  in; for that particular swap we have  $\delta'(p) = \delta^*(p)$ . In all three cases one has  
 433 inequality (3). Our final goal is to prove inequality (4). Consider a swap  $(i, j)$  in  $\mathcal{S}$ . There  
 434 are three cases to consider:

- 435 ■  $j = \text{nn}^*(p)$ . There is exactly one swap for which  $j = \text{nn}^*(p)$ . In this case  $\delta(p) \leq \delta^*(p)$ ,  
 436 therefore  $\delta(p) - \delta'(p) \geq \delta(p) - \delta^*(p)$ .
- 437 ■  $j \neq \text{nn}^*(p)$  and  $i \neq \text{nn}(p)$ . Since  $\text{nn}(p) \in S'$ ,  $\delta'(p) \leq \delta(p)$ . Therefore  $\delta(p) - \delta'(p) \geq 0$ .
- 438 ■  $j \neq \text{nn}^*(p)$  and  $i = \text{nn}(p)$ . By construction, there are most two swaps in  $\mathcal{S}$  that may  
 439 swap out  $\text{nn}(p)$ . We claim that  $i \neq \text{center}(\text{nn}^*(p))$ . Indeed, if  $i = \text{center}(\text{nn}^*(p))$ , then  
 440 by construction,  $\Sigma_i = \{(i, \text{nn}^*(p))\}$  because the center of star of size greater than one is

## XX:12 Clustering under Perturbation Stability in Near-Linear Time

never added to a candidate swap. But this contradicts the assumption that  $j \neq \text{nn}^*(p)$ . The claim implies that  $\text{center}(\text{nn}^*(p)) \in S'$  and thus  $\delta'(p) \leq \delta(p, \text{center}(\text{nn}^*(p)))$ . We obtain a bound on  $\delta(p, \text{center}(\text{nn}^*(p)))$  as follows:

$$\begin{aligned} \delta(p, \text{center}(\text{nn}^*(p))) &\leq \delta(p, \text{nn}^*(p)) + \delta(\text{nn}^*(p), \text{center}(\text{nn}^*(p))) \\ &\leq \delta^*(p) + \delta(\text{nn}^*(p), \text{nn}(p)) \leq \delta^*(p) + (\delta^*(p) + \delta(p)) = \delta(p) + 2\delta^*(p). \end{aligned}$$

Therefore,  $\delta(p) - \delta'(p) \geq \delta(p) - \delta(p, \text{center}(\text{nn}^*(p)))$ . Putting everything together, we obtain:

$$\sum_{S' \in \mathcal{S}} (\delta(p) - \delta'(p)) \geq (\delta(p) - \delta^*(p)) + 0 + 2(\delta(p) - \delta(p) - 2\delta^*(p)) = \delta(p) - 5\delta^*(p).$$

450

Using Lemma 11, we can prove the following.

► **Lemma 12.** *Let  $S$  be a  $k$ -clustering of  $(X, \delta)$  that is not  $C$ -good for some fixed constant  $C > 4 + \epsilon$  with arbitrarily small  $\epsilon > 0$ . There is always a 1-swap  $S'$  such that  $\$(X) - \$^*(X) \leq (1 - \epsilon/(1 + \epsilon)k) \cdot (\$(X) - \$^*(X))$ , where  $\$$  is the cost function defined with respect to  $S'$ .*

**Proof.** By Lemma 11 one has  $\$(X) - \$'(X) \geq (\$(X) - \$^*(X) - \Psi(X_{00}))/k$  for some 1-swap  $S'$  and its corresponding cost function  $\$'(\cdot)$ . Since  $S$  is not  $C$ -good,  $\$(X) - \$^*(X) > C \cdot \$^*(X_{00})$ . Rearranging and plugging the definition of  $\Psi(\cdot)$ , we have

$$\begin{aligned} \$'(X) - \$^*(X) &\leq \$(X) - \$^*(X) - (\$(X) - \$^*(X) - \Psi(X_{00}))/k \\ &\leq \$(X) - \$^*(X) - (\$(X) - \$^*(X) - 4 \cdot \$^*(X_{00}))/k \\ &\leq \$(X) - \$^*(X) - (\$(X) - \$^*(X) + (M - 1) \cdot (\$(X) - \$^*(X)) - 4M \cdot \$^*(X_{00}))/Mk \\ &\leq \left(1 - \frac{\epsilon}{(1 + \epsilon)k}\right) \cdot (\$(X) - \$^*(X)), \end{aligned}$$

where the last inequality holds by taking  $M$  to be arbitrarily large (say  $M > 1 + 1/\epsilon$ ). ◀

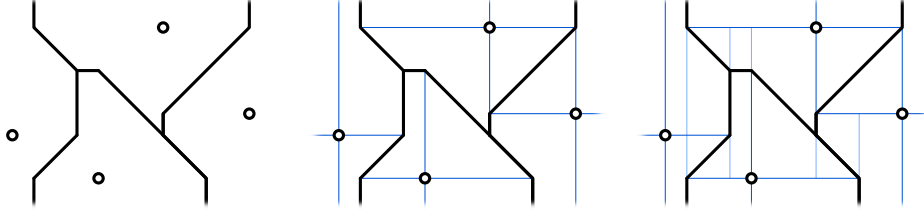
## 5 Efficient Implementation of Local Search

We describe an efficient implementation of each step of the local-search algorithm in this section. By Lemma 4, it suffices to implement the algorithm using a polyhedral metric  $\delta_N$ . We show that each step of 1-swap can be implemented in  $O(nk^{2d-1} \text{polylog } n)$  time under the assumption that  $\alpha > 5$ . We obtain the following:

► **Theorem 13.** *Let  $(X, \delta)$  be an  $\alpha$ -stable instance of the  $k$ -median problem where  $X \subset \mathbb{R}^d$  and  $\delta$  is the Euclidean metric. For  $\alpha > 5$ , the 1-swap local search algorithm computes the optimal  $k$ -clustering of  $(X, \delta)$  in  $O(nk^{2d-1} \text{polylog } n)$  time.*

For simplicity, we present a slightly weaker result for  $d = 2$  using the  $L_1$ -metric, as it is straightforward to implement and more intuitive. Using the  $L_1$ -metric requires  $\alpha > 5\sqrt{2}$ . The extension to higher dimensional Euclidean space using the polyhedral metric is described in the full version of the paper, which works for  $\alpha > 5$ .

**Voronoi diagram under  $L_1$  norm.** First, we fix a point  $x \in X \setminus S$  to insert and a center  $y \in S$  to drop. Define  $S' := S + x - y$ . We build the  $L_1$  Voronoi diagram  $V$  of  $S'$ . The cells of  $V$  may not be convex, but they are *star-shaped*: for any  $c \in S'$  and for any point



■ **Figure 2.**  $L_1$  Voronoi diagram  $V$ , quadrant decomposition  $\tilde{V}$ , and trapezoid decomposition  $V^{\parallel}$ .

479  $x \in \text{Vor}(c)$ , the segment  $cx$  lies completely in  $\text{Vor}(c)$ . Furthermore, all line segments on the  
 480 cell boundaries of  $V$  must have slopes belonging to one of the four possible values: vertical,  
 481 horizontal, diagonal, or antidiagonal.

482 Next, decompose each Voronoi cell  $\text{Vor}(c)$  into four *quadrants* centered at  $c$ . Denote the  
 483 resulting subdivision of  $V$  as  $\tilde{V}$ . We compute a *trapezoidal decomposition*  $V^{\parallel}$  of the diagram  
 484  $\tilde{V}$  by drawing a vertical segment from each vertex of  $\tilde{V}$  in both directions until it meets an  
 485 edge of  $V$ ;  $V^{\parallel}$  has  $O(k)$  trapezoids, see Figure 2. For each trapezoid  $\tau \in V^{\parallel}$ , let  $X_{\tau} := X \cap \tau$ .  
 486 The cost of the new clustering  $S'$  can be computed as  $\$(X, S') = \sum_{\tau \in V^{\parallel}} \$(X_{\tau}, S')$ .

487 **Range-sum queries.** Now we discuss how to compute  $\$(X_{\tau}, S')$ . Each trapezoid  $\tau$  in cells  
 488  $\text{Vor}(c)$  is associated with a vector  $u(\tau) \in \{\pm 1\}^2$ , depending on which of the four quadrants  
 489  $\tau$  belongs to with respect to the axis-parallel segments drawn passing through the center  $c$   
 490 of the cell. If  $\tau$  lies in the top-right quadrant then  $u(\tau) = (1, 1)$ . Similarly if  $\tau$  lies in the  
 491 top-left (resp. bottom-left, bottom-right) then  $u(\tau) = (-1, 1)$  (resp.  $(-1, -1)$ ,  $(1, -1)$ ).

$$492 \quad \$(X_{\tau}, S') = \sum_{x \in X_{\tau}} \|x - c\|_1 = \sum_{x \in X_{\tau}} \langle x - c, u(\tau) \rangle = \sum_{x \in X_{\tau}} \langle x, u(\tau) \rangle - |X_{\tau}| \cdot \langle c, u(\tau) \rangle. \quad (5)$$

493 We preprocess  $X$  into a data structure that answers the following query:

494 ■ **TRAPEZOIDSUM**( $\tau, u$ ): Given a trapezoid  $\tau$  and a vector  $u \in \{\pm 1\}^2$ , return  $|X \cap \tau|$  as  
 495 well as  $\sum_{x \in X \cap \tau} \langle x, u \rangle$ .

496 The above query can be viewed as a 3-oriented polygonal range query [33]. We construct  
 497 a 3-level range tree  $\Psi$  on  $X$ . Omitting the details (which can be found in [33]),  $\Psi$  can be  
 498 constructed in  $O(n \log^2 n)$  time and uses  $O(n \log^2 n)$  space. Each node  $\xi$  at the third level of  
 499  $\Psi$  is associated with a subset  $X_{\xi} \subseteq X$ . We store  $w(\xi, u) := \sum_{x \in X_{\xi}} \langle x, u \rangle$  for each  $u \in \{\pm 1\}^2$   
 500 and  $|X_{\xi}|$  at  $\xi$ . For a trapezoid  $\tau$ , the query procedure identifies in  $O(\log^3 n)$  time a set  $\Xi_{\tau}$  of  
 501  $O(\log^3 n)$  third-level nodes such that  $X \cap \tau = \cup_{\xi \in \Xi_{\tau}} X_{\xi}$  and each point of  $X \cap \tau$  appears as  
 502 exactly one node of  $\Xi_{\tau}$ . Then  $\sum_{x \in X_{\tau}} \langle x, u \rangle = \sum_{\xi \in \Xi_{\tau}} w(\xi, u)$  and  $|X_{\tau}| = \sum_{\xi \in \Xi_{\tau}} |X_{\xi}|$ .

503 With the information stored at the nodes in  $\Xi_{\tau}$ , **TRAPEZOIDSUM**( $\tau, u$ ) query can be  
 504 answered in  $O(\log^3 n)$  time. By performing **TRAPEZOIDSUM**( $\tau, u(\tau)$ ) query for all  $\tau \in V^{\parallel}$ ,  
 505  $\$(X_{\tau}, S')$  can be computed in  $O(k \log^3 n)$  time since  $V^{\parallel}$  has a total of  $O(k)$  trapezoids.

506 We summarize the implementation of 1-swap algorithm in Figure 3. The 1-swap procedure  
 507 considers at most  $nk$  different  $k$ -clusterings. Therefore we obtain the following.

508 ► **Lemma 14.** Let  $(X, \delta, \$)$  be a given clustering instance where  $\delta$  is the  $L_1$  metric, and  
 509 let  $S$  be a given  $k$ -clustering. After  $O(n \log n)$  time preprocessing, we find a  $k$ -clustering  
 510  $S' := S + x - y$  minimizing  $\$(X, S')$  among all choices of  $(x, y)$  in  $O(nk^2 \log^3 n)$  time.

**1-SWAP( $X, S$ ):**  
*input:* Point set  $X$  and centers  $S$   
 for each point  $x \in X \setminus S$  and center  $y \in S$ :  
 $S' \leftarrow S + x - y$   
 $V \leftarrow L_1$  Voronoi diagram of  $S'$   
 $\tilde{V} \leftarrow$  decompose each cell  $\text{Vor}(c)$  into four quadrants centered at  $c$   
 $V^\parallel \leftarrow$  trapezoidal decomposition of  $\tilde{V}$   
 for each trapezoid  $\tau \in V^\parallel$ :  
 $\$(X_\tau, S') \leftarrow \text{TRAPEZOIDSUM}(\tau, u(\tau))$   
 $\$(X, S') \leftarrow \sum_{\tau \in V^\parallel} \$(X_\tau, S')$   
 return  $(x, y)$  with the lowest  $\$(X, S + x - y)$

■ **Figure 3.** Efficient implementation of 1-swap under 1-norm.

## References

- 1 M. Ackerman and S. Ben-David. Clusterability: A theoretical study. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, volume 5 of *JMLR Proceedings*, pages 1–8, 2009.
- 2 P. Afshani, J. Barbay, and T. M. Chan. Instance-optimal geometric algorithms. *Journal of the ACM (JACM)*, 64(1):3, 2017.
- 3 P. K. Agarwal, J. Erickson, et al. Geometric range searching and its relatives. *Contemporary Mathematics*, 223:1–56, 1999.
- 4 P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Geometric approximation via coresets. *Combinatorial and computational geometry*, 52:1–30, 2005.
- 5 P. K. Agarwal and J. Matoušek. Ray shooting and parametric search. *SIAM Journal on Computing*, 22(4):794–806, 1993.
- 6 N. Ailon, A. Bhattacharya, R. Jaiswal, and A. Kumar. Approximate clustering with same-cluster queries. In A. R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, volume 94 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 40:1–40:21, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- 7 D. Aloise, A. Deshpande, P. Hansen, and P. Popat. NP-hardness of Euclidean sum-of-squares clustering. *Machine learning*, 75(2):245–248, 2009.
- 8 O. Angel, S. Bubeck, Y. Peres, and F. Wei. Local max-cut in smoothed polynomial time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 429–437. ACM, 2017.
- 9 H. Angelidakis, P. Awasthi, A. Blum, V. Chatziafratis, and C. Dan. Bilu-Linial stability, certified algorithms and the independent set problem. Preprint, October 2018.
- 10 H. Angelidakis, K. Makarychev, and Y. Makarychev. Algorithms for stable and perturbation-resilient problems. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 438–451. ACM, 2017.
- 11 S. Arora, P. Raghavan, and S. Rao. Approximation schemes for Euclidean  $k$ -medians and related problems. In *STOC*, volume 98, pages 106–113, 1998.
- 12 V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for  $k$ -median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, Jan. 2004.
- 13 H. Ashtiani, S. Kushagra, and S. Ben-David. Clustering with same-cluster queries. In *Advances in neural information processing systems*, pages 3216–3224, 2016.
- 14 P. Awasthi, A. Blum, and O. Sheffet. Center-based clustering under perturbation stability. *Information Processing Letters*, 112(1–2):49–54, 2012.



- 547 15 M. Bădoiu, S. Har-Peled, and P. Indyk. Approximate clustering via core-sets. In *Proceedings*  
548 *of the thirty-fourth annual ACM symposium on Theory of computing*, pages 250–257. ACM,  
549 2002.
- 550 16 A. Bakshi and N. Chepurko. Polynomial time algorithm for 2-stable clustering instances.  
551 Preprint, July 2016.
- 552 17 M.-F. Balcan, A. Blum, and A. Gupta. Approximate clustering without the approximation.  
553 In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages  
554 1068–1077. Society for Industrial and Applied Mathematics, 2009.
- 555 18 M.-F. Balcan, A. Blum, and S. Vempala. A discriminative framework for clustering via  
556 similarity functions. In *Proceedings of the fortieth annual ACM symposium on Theory of*  
557 *computing*, pages 671–680. ACM, 2008.
- 558 19 M.-F. Balcan and M. Braverman. Nash equilibria in perturbation-stable games. *Theory of*  
559 *Computing*, 13(1):1–31, 2017.
- 560 20 M.-F. Balcan, N. Haghtalab, and C. White.  $k$ -center clustering under perturbation resilience. In  
561 I. Chatzigiannakis, M. Mitzenmacher, Y. Rabani, and D. Sangiorgi, editors, *43rd International*  
562 *Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *Leibniz*  
563 *International Proceedings in Informatics (LIPIcs)*, pages 68:1–68:14, Dagstuhl, Germany, 2016.  
564 Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- 565 21 M. F. Balcan and Y. Liang. Clustering under perturbation resilience. *SIAM Journal on*  
566 *Computing*, 45(1):102–155, 2016.
- 567 22 S. Ben-David and L. Reyzin. Data stability in clustering: A closer look. *Theoretical Computer*  
568 *Science*, 558(1):51–61, 2014.
- 569 23 Y. Bilu and N. Linial. Are stable instances easy? *Combinatorics, Probability and Computing*,  
570 21(5):643–660, 2012.
- 571 24 P. B. Callahan and S. R. Kosaraju. Faster algorithms for some geometric graph problems in  
572 higher dimensions. In *Proceedings of the fourth annual ACM-SIAM symposium on Discrete*  
573 *algorithms*, pages 291–300. Society for Industrial and Applied Mathematics, 1993.
- 574 25 C. Chekuri and S. Gupta. Perturbation resilient clustering for  $k$ -center and related prob-  
575 lems via LP relaxations. In E. Blais, K. Jansen, J. D. P. Rolim, and D. Steurer, editors,  
576 *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*  
577 *(APPROX/RANDOM 2018)*, volume 116 of *Leibniz International Proceedings in Informatics*  
578 *(LIPIcs)*, pages 9:1–9:16, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer  
579 Informatik.
- 580 26 V. Cohen-Addad. A fast approximation scheme for low-dimensional  $k$ -means. In *Proceedings*  
581 *of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages  
582 430–440, Philadelphia, PA, USA, 2018. Society for Industrial and Applied Mathematics.
- 583 27 V. Cohen-Addad, A. de Mesmay, E. Rotenberg, and A. Roytman. The bane of low-  
584 dimensionality clustering. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium*  
585 *on Discrete Algorithms*, SODA '18, pages 441–456, Philadelphia, PA, USA, 2018. Society for  
586 Industrial and Applied Mathematics.
- 587 28 V. Cohen-Addad, A. E. Feldmann, and D. Saulpic. Near-linear time approximation schemes  
588 for clustering in doubling metrics. Preprint, June 2019.
- 589 29 V. Cohen-Addad, A. Gupta, A. Kumar, E. Lee, and J. Li. Tight FPT Approximations for  
590  $k$ -Median and  $k$ -Means. In C. Baier, I. Chatzigiannakis, P. Flocchini, and S. Leonardi, editors,  
591 *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*,  
592 volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 42:1–42:14,  
593 Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- 594 30 V. Cohen-Addad, P. N. Klein, and C. Mathieu. Local search yields approximation schemes for  
595  $k$ -means and  $k$ -median in Euclidean and minor-free metrics. *SIAM Journal on Computing*,  
596 48(2):644–667, 2019.

- 597 31 V. Cohen-Addad and C. Schwiegelshohn. On the local structure of stable clustering instances.  
598 In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages  
599 49–60. IEEE, 2017.
- 600 32 S. Dasgupta. The hardness of  $k$ -means clustering. Technical report, Department of Computer  
601 Science and Engineering, University of California, 09 2008.
- 602 33 M. De Berg, M. Van Kreveld, M. Overmars, and O. Schwarzkopf. Computational geometry.  
603 In *Computational geometry*, pages 1–17. Springer, 1997.
- 604 34 A. Deshpande, A. Louis, and A. Vikram Singh. On Euclidean  $k$ -means clustering with  $\alpha$ -center  
605 proximity. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages  
606 2087–2095, 2019.
- 607 35 A. Dutta, A. Vijayaraghavan, and A. Wang. Clustering stable instances of Euclidean  $k$ -means.  
608 In *Advances in Neural Information Processing Systems*, pages 6500–6509, 2017.
- 609 36 T. Feder and D. Greene. Optimal algorithms for approximate clustering. In *Proceedings of the*  
610 *twentieth annual ACM symposium on Theory of computing*, pages 434–444. ACM, 1988.
- 611 37 D. Feldman, M. Monemizadeh, and C. Sohler. A ptas for  $k$ -means clustering based on weak  
612 coresets. In *Proceedings of the twenty-third annual symposium on Computational geometry*,  
613 pages 11–18. ACM, 2007.
- 614 38 Z. Friggstad, K. Khodamoradi, and M. R. Salavatipour. Exact algorithms and lower bounds  
615 for stable instances of Euclidean  $k$ -means. In *Proceedings of the Thirtieth Annual ACM-SIAM*  
616 *Symposium on Discrete Algorithms, SODA '19*, pages 2958–2972, Philadelphia, PA, USA, 2019.  
617 Society for Industrial and Applied Mathematics.
- 618 39 Z. Friggstad, M. Rezapour, and M. R. Salavatipour. Local search yields a PTAS for  $k$ -means  
619 in doubling metrics. *SIAM Journal on Computing*, 48(2):452–480, 2019.
- 620 40 S. Har-Peled. No, coreset, no cry. In *International Conference on Foundations of Software*  
621 *Technology and Theoretical Computer Science*, pages 324–335. Springer, 2004.
- 622 41 S. Har-Peled and S. Mazumdar. On coresets for  $k$ -means and  $k$ -median clustering. In  
623 *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 291–300.  
624 ACM, 2004.
- 625 42 D. Harel and R. E. Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM J.*  
626 *Comput.*, 13(2):338–355, 1984.
- 627 43 D. S. Hochbaum and D. B. Shmoys. A best possible heuristic for the  $k$ -center problem. *Math.*  
628 *Oper. Res.*, 10(2):180–184, May 1985.
- 629 44 K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems.  
630 In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, pages  
631 731–740. ACM, 2002.
- 632 45 A. Kumar and R. Kannan. Clustering with spectral norm and the  $k$ -means algorithm. In  
633 *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*  
634 *(FOCS)*, pages 299–308. IEEE, 2010.
- 635 46 A. Kumar, Y. Sabharwal, and S. Sen. A simple linear time  $(1 + \epsilon)$ -approximation  
636 algorithm for  $k$ -means clustering in any dimensions. In *Annual Symposium on Foundations of*  
637 *Computer Science*, volume 45, pages 454–462. IEEE COMPUTER SOCIETY PRESS, 2004.
- 638 47 E. Lee, M. Schmidt, and J. Wright. Improved and simplified inapproximability for  $k$ -means.  
639 *Information Processing Letters*, 120:40–43, 2017.
- 640 48 M. Mahajan, P. Nimbhorkar, and K. Varadarajan. The planar  $k$ -means problem is NP-hard.  
641 *Theoretical Computer Science*, 442:13–21, 2012.
- 642 49 K. Makarychev, Y. Makarychev, and A. Vijayaraghavan. Bilu-Linial stable instances of  
643 max cut and minimum multiway cut. In *Proceedings of the twenty-fifth annual ACM-SIAM*  
644 *symposium on Discrete algorithms*, pages 890–906. SIAM, 2014.
- 645 50 N. Megiddo and K. J. Supowit. On the complexity of some common geometric location  
646 problems. *SIAM journal on computing*, 13(1):182–196, 1984.

- 647 51 M. Mihalák, M. Schöngens, R. Šrámek, and P. Widmayer. On the complexity of the metric  
648 TSP under stability considerations. In *International Conference on Current Trends in Theory  
649 and Practice of Computer Science*, pages 382–393. Springer, 2011.
- 650 52 R. Ostrovsky, Y. Rabani, L. J. Schulman, and C. Swamy. The effectiveness of Lloyd-type  
651 methods for the  $k$ -means problem. *Journal of the ACM (JACM)*, 59(6):28, 2012.
- 652 53 D. D. Sleator and R. E. Tarjan. A data structure for dynamic trees. *J. Comput. Syst. Sci.*,  
653 26(3):362–391, 1983.
- 654 54 C. D. Toth, J. O’Rourke, and J. E. Goodman, editors. *Handbook of discrete and computational  
655 geometry*. Chapman and Hall/CRC, 2017.