Clustering Perturbation Resilient k-Median Instances

Maria Florina Balcan
School of Computer Science
Georgia Institute of Technology
Atlanta, GA 30332
ninamf@cc.gatech.edu

Yingyu Liang
School of Computer Science
Georgia Institute of Technology
Atlanta, GA 30332
yliang39@gatech.edu

Abstract

Recently, Bilu and Linial [6] formalized an implicit assumption often made when choosing a clustering objective: that the optimum clustering to the objective should be preserved under small multiplicative perturbations to distances between points. Balcan and Liang [4] generalized this to a relaxed notion where the optimal clustering after perturbation is allowed to change slightly. In this paper, we propose an efficient algorithm for k-median instances under the generalized notion, achieving theoretical guarantees that significantly improve over previous known results. Additionally, we give a sublinear-time algorithm which can return an implicit clustering from only access to a small random sample.

1 Introduction

Problems of clustering data from pairwise distance information are a classical topic in machine learning. A common approach is to optimize various objective functions such as k-median, k-means or min-sum. However, for most natural clustering objectives, finding the optimal solution is NP-hard. There has been substantial work on approximation algorithms [10, 7, 5, 8, 1] with both upper and lower bounds on the approximability of these objectives on worst case instances.

Recently, Bilu and Linial [6] suggested an exciting, alternative approach aimed at understanding the complexity of practical clustering instances. Motivated by the fact that distances are often based on a heuristic measure, they argued that interesting instances should be resilient to small perturbations in these distances. Specifically, they defined an instance to be α-perturbation resilient if perturbing pairwise distances by multiplicative factors in the range $[1, \alpha]$ does not change the optimum clustering. Balcan and Liang [4] generalized this to a weaker, relaxed, and more realistic notion of (α, ϵ)-perturbation resilience where the optimal clustering of the perturbed instance is allowed to differ from the optimal of the original in a small ϵ fraction of the points. Compared to the original perturbation resilience assumption, this is arguably a more natural though also more difficult condition to deal with.

In this paper, we propose an efficient algorithm for (α, ϵ)-perturbation resilient k-median instances, which for $\alpha > 4$ produces $(1 + O(\epsilon/\rho))$-approximation to the optimum, where ρ is the fraction of the points in the smallest cluster. This significantly improves over the bound $\alpha > 2 + \sqrt{7}$ in [4]. The algorithm is based on the key structural property that, except for ϵn bad points, most points are α times closer to their own center than to any other center. To eliminate the noise introduced by the bad points, we carefully partition the points into a list of sufficiently large blobs, each of which contains only good points from one optimal cluster. This then allows us to construct a tree on the blobs with a low-cost pruning that is a good approximation to the optimum. Additionally, the robustness to the bad points allows us to make the algorithm sublinear-time by returning an implicit clustering from only a small random sample of the input. The construction of the implicit clustering takes time poly-logarithmic in the size of the data, which makes the sublinear-time version preferable for large scale data sets.

2 Preliminaries

In a clustering instance, we are given a set S of n points in a finite metric space, and we denote $d : S \times S \to \mathbb{R}_{\geq 0}$ as the distance function. In k-median clustering, we partition S into k disjoint subsets $\mathcal{P} = \{P_1, P_2, \dots, P_k\}$ and assign a set of centers $\mathbf{p} = \{p_1, p_2, \dots, p_k\} \subseteq S$ for the subsets. The goal is to minimize the objective $\Phi(\mathcal{P}, \mathbf{p}) = \sum_{i=1}^{k} \sum_{p \in P_i} d(p, p_i)$. When the partition $\mathcal{P}$ is obtained by assigning each point to its nearest center in $\mathbf{p}$, the objective is shorten as $\Phi(\mathbf{c})$. The optimal centers are denoted as $\mathbf{c} = \{c_1, \dots, c_k\}$, the optimal clustering is denoted as $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$, and its cost is denoted as $\mathcal{OPT}$.

The core concept we study in this paper is the following (α, ϵ)-perturbation resilience notion.

Definition 1. *Let $\mathcal{C}$ be the optimal k-clustering and $\mathcal{C}'$ be another k-clustering of a set of n points. We say $\mathcal{C}'$ is ϵ-**close** to $\mathcal{C}$ if $\min_{\sigma \in \mathcal{S}_k} \sum_{i=1}^{k} |C_i \setminus C'_{\sigma(i)}| \leq \epsilon n$, where σ is a matching between indices of clusters of $\mathcal{C}'$ and those of $\mathcal{C}$.*

Definition 2. *A clustering instance (S, d) is (α, ϵ)**-perturbation resilient** to a given objective Φ if for any function $d' : S \times S \to \mathbb{R}_{\geq 0}$ such that $\forall p, q \in S, d(p,q) \leq d'(p,q) \leq \alpha d(p,q)$, the optimal clustering $\mathcal{C}'$ for Φ under d' is ϵ-close to the optimal clustering $\mathcal{C}$ for Φ under d.*

3 Clustering (α, ϵ)-Perturbation Resilient k-Median Instances

In this section we show that if the instance is (α, ϵ)-perturbation resilient, with $\alpha > 4$ and $\epsilon = O(\epsilon' \rho)$ where ρ is the fraction of the points in the smallest cluster, then we can in polynomial time output a clustering that provides a $(1 + \epsilon')$-approximation to the optimum. Formally,

Theorem 1. *If the clustering instance is (α, ϵ)-perturbation resilient for $\alpha > 4$ and $\epsilon \leq \rho/30$ where $\rho = \frac{\min_i |C_i|}{n}$, then Algorithm 1 produces a clustering which is $(1 + \frac{5\epsilon}{\rho})$-approximation to the optimal clustering with respect to the k-median objective in time $O(n^{\omega+1})$, where $O(n^{\omega})$ is the state of the art for matrix multiplication.*

This significantly improves over the bound $\alpha > 2 + \sqrt{7}$ in [4]. It also improves over the best worst-case approximation guarantees known [11] when $\epsilon' \leq \sqrt{3}$ and also beats the lower bound of $(1 + 1/e)$ on the best approximation achievable on worst case instances for the metric k-median objective [9, 10] when $\epsilon' \leq 1/e$.

In the following, we first review the structural properties utilized, and then describe our algorithm and provide a sketch of the analysis.

3.1 Structural Property

The key structural property we exploit is that, except for ϵn bad points, most points are α times closer to their own center than to any other center. Specifically, we call a point *good* if it is α times closer to its own center than to any other center in the optimal clustering; otherwise we call it *bad*. Let B_i be the set of bad points in C_i. That is, $B_i = \{p \in C_i : \exists j \neq i, \alpha d(c_i, p) > d(c_j, p)\}$. Let $G_i = C_i \setminus B_i$ be the good points in cluster C_i and let $B = \bigcup_i B_i$.

Theorem 2. *(Theorem 1 in [4]) Suppose the clustering instance is (α, ϵ)-perturbation resilient and $\min_i |C_i| > (3 + \frac{2\alpha}{\alpha - 1})\epsilon n + 9\alpha$. Then $|B| \leq \epsilon n$.*

We can see that by definition, the good points are far apart from each other. This then implies that for any good point, most of its nearest neighbors are from its own optimal cluster. Formally,

Lemma 1. *When $\alpha > 4$, for any good points $p_1, p_2 \in G_i, q \in G_j (j \neq i)$, we have $d(p_1, p_2) < d(p_1, q)$. Consequently, for any good point $p \in G_i$, all its $|G_i|$ nearest neighbors belong to $C_i \cup B$.*

3.2 Approximation Algorithm

In the following, we utilize the bound on the bad points and the property of the good points to design an efficient approximation algorithm. In order to get rid of the influence of the bad points, we

Algorithm 1 k-median, (α, ϵ) perturbation resilience

```
1: Input: Distance function $d(\cdot,\cdot)$ on $S$, the size of the smallest optimal cluster $\min_i |C_i|$, $\epsilon > 0$
2: Run Algorithm 2 to generate a list $\mathcal{L}$ of blobs.
3: Run the robust linkage procedure in [3] to get a cluster tree $T$.
4: Run dynamic programming on $T$ to get the lowest cost pruning $\tilde{\mathcal{C}}$ and its centers $\tilde{\mathbf{c}}$.
5: Output: Clustering $\tilde{\mathcal{C}}$ and its centers $\tilde{\mathbf{c}}$.
```

Algorithm 2 Generating interesting blobs

```
1: Input: Distance function $d(\cdot,\cdot)$ on $S$, $\min_i |C_i|$, $\epsilon > 0$.
2: Let $N_r(p)$ denote the $r$ nearest neighbors of $p$ in $S$.
3: Let $\mathcal{L} = \emptyset, A_S = S$. Let the initial threshold $t = \min_i |C_i|$.
4: Construct a graph $F_t$ by connecting $p, q \in A_S$ if $|N_t(p) \cap N_t(q)| > t - 2\epsilon n$.
5: Construct a graph $H_t$ by connecting points $p, q \in A_S$ that have more than $\epsilon n$ neighbors in $F_t$.
6: Add to $\mathcal{L}$ all the components $C$ of $H_t$ with $|C| \geq \frac{1}{2}\min_i |C_i|$ and remove them from $A_S$.
7: For each point $p \in A_S$, check if most of $N_t(p)$ are in $\mathcal{L}$ and if there exists $C \in \mathcal{L}$ containing a
   significant number of points in $N_t(p)$. More precisely, check if
   (1) $|N_t(p) \setminus \mathcal{L}| \leq \frac{1}{2}\min_i |C_i| + 2\epsilon n$;
   (2) $\mathcal{L}_p \neq \emptyset$ where $\mathcal{L}_p = \{C \in \mathcal{L} : |C \cap N_t(p)| \geq \frac{2}{5}|C|\}$.
   If so, assign $p$ to the blob in $\mathcal{L}_p$ of smallest median distance, and remove $p$ from $A_S$.
8: While $|A_S| > 0$, increase $t$ by 1 and go to Step 4.
9: Output: The list $\mathcal{L}$.
```

generate a list of blobs, which form a partition of the data points, and each of which contains only good points from one optimal cluster. Then we construct a tree on the list of blobs with a pruning that assigns all good points correctly. We will show that this pruning has low cost, so the lowest cost pruning of the tree is a good approximation. The details are described in Algorithm 1. We now provide a sketch of the analysis of its two key steps and the final approximation guarantee.

Generating Blobs The first key step is to generate the list of almost "pure" blobs, which is described in Algorithm 2. Informally, the algorithm maintains a threshold t. At each threshold, for each point p that has not been added to the list, the algorithm checks its t nearest neighbors $N_t(p)$. It constructs a graph F_t by connecting any two points that have most neighbors in common. It then builds another graph H_t by connecting any two points that have sufficiently many neighbors in F_t, and adds sufficiently large components in H_t to the list. Finally, for each remaining point p, it checks if most of p's neighbors are in the list and if there are blobs containing a significant amount of p's neighbors. If so, it inserts p into such a blob with the smallest median distance. Then the threshold is increased and the above steps are repeated.

The intuition behind Algorithm 2 is as follows. The algorithm works when for any i and any good point $p \in G_i$, the $|G_i|$ nearest neighbors of p contain no good points outside C_i. To see this, assume without loss of generality that $|C_1| \leq |C_2| \leq \cdots \leq |C_k|$. When $t \leq |C_1|$, good points in different clusters do not have most neighbors in common and thus are not connected in F_t. However, they may be connected by a path of bad points. So we further build the graph H_t to disconnect such paths, which ensures that the blobs added into the list contain only good points from one optimal cluster. The final insert step (Step 7) makes sure that when $t = |C_1|$, all remaining good points in C_1 will be added to the list and will not affect the construction of blobs from other optimal clusters. We can show by induction that, at the end of the iteration $t = |C_i|$, all good points in $C_j (j \leq i)$ are added to the list. When t is large enough, any remaining bad points are inserted into the list, so the points are partitioned into a list of almost pure blobs.

Linking Blobs Another key step is to construct a tree on these blobs. Since good points are closer to good points in the same optimal cluster than to those in other clusters (Lemma 1), there exist algorithms that can build a tree with a pruning that assigns all good points correctly. In particular, we can use the robust linkage procedure in [3], which repeatedly merges the two blobs C, C' with the maximum $\mathrm{score}(C, C')$ defined as follows. For each $p \in C$, sort the other blobs in decreasing order of the median distance between p and points in the blob, and let $\mathrm{rank}(p, C')$ denote the rank of C'. Then define $\mathrm{rank}(C, C') = \mathrm{median}_{x \in C}[\mathrm{rank}(x, C')]$ and

$\mathrm{score}(C, C') = \min[\mathrm{rank}(C, C'), \mathrm{rank}(C', C)]$. Intuitively, for any blobs A, A' from the same optimal cluster and D from a different cluster, good points in A always rank A' later than D in the sorted list, so $\mathrm{rank}(A, A') > \mathrm{rank}(A, D)$. Similarly, $\mathrm{rank}(A', A) > \mathrm{rank}(A', D)$, and thus $\mathrm{score}(A', A) > \mathrm{score}(A, D)$. Then the algorithm always merges blobs from the same cluster before merging them with blobs outside, and thus there is a pruning that assigns all good points correctly.

Approximation Guarantee As described above, Algorithm 2 partitions the points into a list of blobs, each of which has size at least $\frac{1}{2}\min_i |C_i|$ and contains only good points from one optimal cluster. Let B'_i denote the bad points that are assigned to blobs containing good points in C_i. Then the robust linkage procedure on $\mathcal{L}$ guarantees that $\{G_i \cup B'_i\}_{i=1}^k$ is a pruning of the tree output (see Theorem 9 in [3]). It suffices to show that this pruning, using the optimal centers $\{c_i\}$, is a $(1 + \frac{5\epsilon}{\rho})$-approximation to $\mathcal{OPT}$. Since all good points are correctly assigned, we only need to bound the cost increased by assigning a bad point $q \in C_i$ to a blob containing good points from a different optimal cluster C_j. Intuitively, the bad point must have many nearest neighbors in that blob, then it is closer to a significant number of good points in that optimal cluster than to a significant number of good points in its own optimal cluster. Formally, we have

Lemma 2. *If a bad point $q \in B_i$ is assigned to a blob C containing good points from a different optimal clustering C_j, then there exist $m = \frac{1}{5}\min_i |C_i|$ points Z_i from C_i, and m points Z_j from C_j, such that $d(q, Z_i) \geq d(q, Z_j)$. Consequently, $d(q, c_j) - d(q, c_i) \leq \frac{\mathcal{OPT}}{m}$.*

As there are at most ϵn bad points and $m = \frac{\min_i |C_i|}{5}$, the increase of cost is at most $\frac{5\epsilon}{\rho}\mathcal{OPT}$.

4 Sublinear Time Algorithm for (α, ϵ)-Perturbation Resilient Instances

Many clustering applications have recently faced an explosion of data, and it is often expensive to run an algorithm over the entire data. Here we show that for perturbation resilient k-median instances, we can overcome this difficulty by running our algorithm on a small random sample.

More precisely, consider a clustering instance (X, d) that is (α, ϵ)-perturbation resilient to k-median. For simplicity, suppose the distances are normalized to $[0, 1]$. Let $N = |X|$ and let $\rho = \min_i |C_i|/N$ denote the fraction of the points in the smallest cluster. Let Φ_X denote the cost on X, and let $\zeta = \Phi_X(\mathbf{c})/N$ denote the average cost of the points in the optimum clustering.

Theorem 3. *Suppose (X, d) is (α, ϵ)-perturbation resilient for $\alpha > 4$, $\epsilon < \rho/100$. Then with probability $\geq 1 - \delta$, we can get an implicit clustering that is $2(1 + \frac{16\epsilon}{\rho})$-approximation in time poly$(\log \frac{N}{\delta}, k, \frac{1}{\epsilon}, \frac{1}{\zeta})$.*

The main idea is to run Algorithm 1 on a random sample S of size $n = \Theta(\frac{k}{\epsilon^2\zeta^2}\ln\frac{N}{\delta})$ to obtain the minimum cost pruning and the corresponding centers $\tilde{\mathbf{c}}$. Then the implicit clustering of the whole space X assigns each point in X to its nearest center in $\tilde{\mathbf{c}}$.

In the following, we describe the idea to show that $\tilde{\mathbf{c}}$ is a good approximation solution to the optimal centers $\mathbf{c}$ for X. First, when n is sufficiently large, with high probability, $\Phi_X(\tilde{\mathbf{c}})/N \approx \Phi_S(\tilde{\mathbf{c}})/n$ and $\Phi_X(\mathbf{c})/N \approx \Phi_S(\mathbf{c})/n$. Then it is sufficient to show $\Phi_S(\tilde{\mathbf{c}})$ is close to $\Phi_S(\mathbf{c})$. Next, we can show that Algorithm 1 builds a tree with a pruning $\mathcal{P}'$ that assigns all good points correctly. The key is to use the cost of this pruning as a bridge for comparing $\Phi_S(\tilde{\mathbf{c}})$ and $\Phi_S(\mathbf{c})$.

On one hand, $\Phi_S(\tilde{\mathbf{c}}) \leq \Phi_S(\tilde{\mathcal{C}}, \tilde{\mathbf{c}}) \leq \Phi_S(\mathcal{P}', \mathbf{c}')$. The first inequality comes from the fact that in $\Phi_S(\tilde{\mathbf{c}})$ each point is assigned to its nearest center and the second comes from the fact that $\tilde{\mathcal{C}}$ is the minimum cost pruning. On the other hand, $\Phi_S(\mathcal{P}', \mathbf{c}') \leq 2\Phi_S(\mathcal{P}', \mathbf{c}) \leq 2(1 + \frac{12\epsilon}{\rho})\Phi_S(\mathbf{c})$. The second inequality comes from an argument similar to that in Theorem 1 and the fact that $\Phi_S(\mathcal{P}', \mathbf{c})$ is different from $\Phi_S(\mathbf{c})$ only on the bad points. The first inequality comes from the triangle inequality. More precisely, for any cluster $N'_i \in \mathcal{P}'$,

$$2|N'_i| \sum_{p \in N'_i} d(p, c_i) = \sum_{p,q \in N'_i} [d(p, c_i) + d(q, c_i)] \geq \sum_{p,q \in N'_i} d(p, q) \geq \sum_{p,q \in N'_i} d(q, c'_i) = |N'_i| \sum_{q \in N'_i} d(q, c'_i).$$

Note: If we have an oracle that given a set of points C'_i finds the best center *in X* for that set, then we can save a factor of 2 in the bound.

References

[1] 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 of Computing*, 2004.

[2] P. Awasthi, A. Blum, and O. Sheffet. Center-based clustering under perturbation stability. *Information Processing Letters*, 2012.

[3] M. F. Balcan and P. Gupta. Robust hierarchical clustering. In *Proceedings of the Annual Conference on Learning Theory*, 2010.

[4] M. F. Balcan and Y. Liang. Clustering under perturbation resilience. In *Proceedings of the International Conference on Automata, Languages, and Programming*. 2012.

[5] Y. Bartal, M. Charikar, and D. Raz. Approximating min-sum k-clustering in metric spaces. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2001.

[6] Y. Bilu and N. Linial. Are stable instances easy? In *Proceedings of the Symposium on Innovations in Computer Science*, 2010.

[7] M. Charikar, S. Guha, É. Tardos, and D. B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 2002.

[8] W. F. de la Vega, M. Karpinski, C. Kenyon, and Y. Rabani. Approximation schemes for clustering problems. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2003.

[9] S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 1999.

[10] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2002.

[11] S. Li and O. Svensson. Approximating k-median via pseudo-approximation. In *Proceedings of the ACM Symposium on the Theory of Computing*, 2013.

A Proof of Theorem 1

We begin with a property of the good points proved in [2], and use it to prove the key property of good points in (α, ϵ)-perturbation resilient instances: all good points have most of their nearest neighbors in their own optimal clusters.

Lemma 3. *(Lemma 2 in [2]) For any good point points $p \in C_i$ and $q \in C_j (j \neq i)$, we have $d(p, q) > (\alpha - 1) \max\{d(p, c_i), d(q, c_j)\}$.*

Lemma 1. *When $\alpha > 4$, for any good points $p_1, p_2 \in G_i, q \in G_j (j \neq i)$, we have $d(p_1, p_2) < d(p_1, q)$. Consequently, for any good point $p \in G_i$, all its $|G_i|$ nearest neighbors belong to $C_i \cup B$.*

Proof. By Lemma 3, $d(p_1, q) > (\alpha - 1)d(p_1, c_i)$ and $d(p_2, q) > (\alpha - 1)d(p_2, c_i)$.

$$d(p_1, p_2) \quad \leq \quad d(p_1, c_i) + d(p_2, c_i) < \frac{1}{\alpha - 1}[d(p_1, q) + d(p_2, q)] \leq \frac{1}{\alpha - 1}[2d(p_1, q) + d(p_1, p_2)]$$

which leads to $d(p_1, p_2) < \frac{2}{\alpha - 2} d(p_1, q)$. When $\alpha > 4$, we have $d(p_1, p_2) < d(p_1, q)$. □

We are now ready to use the above property of the good points and the bound on the number of bad points to show that Algorithm 2 can produce a list of sufficiently large, almost pure blobs.

Lemma 4. *If $\min_i |C_i| > 30\epsilon n$, then Algorithm 2 generates a list $\mathcal{L}$ of blobs each of size at least $\frac{1}{2} \min_i |C_i|$ such that:*

- *The blobs in $\mathcal{L}$ form a partition of S.*
- *Each blob in $\mathcal{L}$ contains good points from only one optimal cluster.*

Proof. Using a careful analysis, we prove the following two claims by induction on $i \leq k$:

- For any $t \leq |G_i|$, any blob in the list $\mathcal{L}$ only contains good points from only one optimal cluster; all blobs have size at least $\frac{1}{2} \min_i |C_i|$.
- At the beginning of the iteration $t = |G_i| + 1$, any good point $p \in G_j, j \leq i$ has already been assigned to a blob in the list that contains good points only from C_j.

The first two claims imply that each blob in the list contains good points from only one optimal cluster. Moreover, at the beginning of the iteration $t = |G_k| + 1$, all good points have been assigned to one of the blobs in $\mathcal{L}$, so there are only bad points left, the number of which is smaller than $\frac{1}{2} \min_i |C_i|$. These remaining points will eventually be assigned to the blobs before $t > n$, so the blobs form a partition of S.

The claims are clearly both true initially. We show now that as long as $t \leq |G_1|$, the graphs F_t and H_t have the following properties.

- No good point p_i in cluster C_i is connected in F_t to a good point p_j in a different cluster C_j. By assumption, p_i has no neighbors outside $C_i \cup B$ and p_j has no neighbors outside $C_j \cup B$, so they share at most $\epsilon n < t - 2\epsilon n$ neighbors.
- No point q is connected in F_t to both a good point p_i in C_i and a good point p_j in a different cluster C_j. If q is connected to p_i, then $|N_t(p_i) \cap N_t(q)| > t - 2\epsilon n$. Since p_i has no neighbors outside $C_i \cup B$, $N_t(q)$ contains more than $t - 3\epsilon n \geq t/2$ points from G_i. Similarly, if q is connected to p_j, then $N_t(q)$ contains more than $t/2$ points from G_j, which is contradictory.
- All the components in H_t of size at least $\frac{1}{2} \min_i |C_i|$ will only contain good points from one optimal cluster. As there are at most ϵn bad points, any two points connected in H_t must be connected in F_t to one good point. Then by the above two properties, points on a path in H_t must be connected in F_t to good points in the same cluster, so there is no path connecting good points from different clusters.

We can use the three properties to argue the first claim: as long as $t \leq |G_1|$, each blob in $\mathcal{L}$ contains good points from at most one optimal cluster. This is true at the beginning and by the third property, for any $t \leq |G_1|$, anytime we insert a whole new blob in the list in Step 6, that blob must contain point from at most one optimal cluster. We now argue that this property is never violated as we assign points to blobs already in the list in Step 7. Suppose a good point $p \in C_i$ is inserted into $C \in \mathcal{L}$. Then $C \in \mathcal{L}_p$, which means $|N_t(p) \cap C| \geq |C|/2 > \epsilon n$. So $N_t(p) \cap C$ contains at least one good point, which must be from C_i since $N_t(p)$ contains no good points outside C_i. Then by induction C must contain only good points from C_i, and thus adding p to C does not violate the first claim.

We now show the second claim: after the iteration $t = |G_1|$, all the good points in C_1 have already been assigned to a blob in the list that only contains good points from C_1. There are two cases. First, if at the beginning of the iteration $t = |G_1|$, there are still at least $\frac{1}{2}\min_i |C_i|$ points from the good point set G_1 that do not belong to blobs in the list. Any such good point has all $|G_1|$ neighbors in $C_1 \cup B$. Then any two such good points share at least $2|G_1| - |C_1 \cup B| \geq |G_1| - |B| \geq t - 2\epsilon n$ neighbors. So they will connect to each other in F_t and then in H_t, and thus we will add one blob to $\mathcal{L}$ containing all these points. Second, it could be that at the beginning of the iteration $t = |G_1|$, all but less than $\frac{1}{2}\min_i |C_i|$ good points in G_1 have been assigned to a blob in the list. Denote these points as E. Any point $p \in E$ has no neighbors outside $C_1 \cup B$. Then $|N_t(p) \setminus \mathcal{L}| \leq |E| + |B| \leq \frac{1}{2}\min_i |C_i| + 2\epsilon n$. Also, there exists a blob C containing good points from C_1 such that $C \in \mathcal{L}_p$. Otherwise, $N_t(p)$ contains at most $\frac{2}{5}(|C_1 \cup B|) < |C_1| - \frac{1}{2}|C_1| - 2\epsilon n$ points in $C_1 \cap \mathcal{L}$, while it contains at most $|E|$ good points in $C_1 \setminus \mathcal{L}$ and contains no points outside $C_1 \cup B$. In total, $N_t(p)$ has less than t points, which is contradictory. So $\mathcal{L}_p \neq \emptyset$ and p will be added to the list in Step 7.

We then iterate the argument on the remaining set A_S. The key point is that for $t \geq |G_i|, i > 1$, we have that all the good points in $C_1, C_2, \ldots, C_i$ have already been assigned to blobs in $\mathcal{L}$. □

Lemma 1 and 4 show that Algorithm 2 produces a list of sufficiently large, almost pure blobs. Then the robust linkage procedure in [3] can build a tree on these blobs with a pruning that assigns all good points correctly. Now it suffices to show that this pruning is a good approximation, for which we need to bound the cost increased by the bad points assigned incorrectly. The following property of these bad points turns out to be useful. Intuitively, Algorithm 2 is designed such that whenever a bad point is added to a blob containing good points from a different cluster, it must be closer to a significantly amount of points in that cluster than to a significantly amount of points in its own cluster. Then the cost increased by incorrectly assigning each such bad point is small, resulting in a good approximation.

Lemma 5. *Suppose for any good point $p \in G_i$, all its $|G_i|$ nearest neighbors in S are from $C_i \cup B$, and $\min_i |C_i| > 30\epsilon n$. When running Algorithm 2 with $\gamma = 1$, if a bad point $q \in B_i$ is assigned to a blob C containing good points from a different optimal clustering C_j, then there exist $m = \frac{1}{5}\min_i |C_i|$ points Z_i from C_i, and m points Z_j from C_j, such that $d(q, Z_i) \geq d(q, Z_j)$.*

Proof. There are two cases: q is added into C in (1) Step 6 or (2) Step 7.

Case 1 There must be a path in H_t connecting q to a good point in C_j at threshold t. For any edge (x, y) in H_t, since x, y share at least ϵn neighbors in F_t and there are at most ϵn bad points, they share at least one good point as neighbor in F_t. As shown in the proof of Lemma 4, no point can connect to good points from different clusters, so in F_t all points on the path must connect to good points in C_j. In particular, q is connected in F_t to a good point $p \in G_j$. Then $|N_t(p) \cap N_t(q)| > t - 2\epsilon n$. Since p is still in A_S, $t \leq |G_j|$, and thus $N_t(p)$ contains no points outside $C_j \cup B$. This means that at least $t - 3\epsilon n \geq m$ points in $N_t(q)$ are good points in C_j, then we can select m points Z_j from $N_t(q) \cap G_j$. We also have that at most $2\epsilon n$ points in $N_t(q)$ are points in C_i, so we can select m points Z_i from $C_i \setminus N_t(q)$.

Case 2 There are three subcases when q is inserted into C at threshold t.
(1) There is no good points from C_i in the list. Since $|N_t(q) \setminus \mathcal{L}| \leq \frac{1}{2}\min_i |C_i| + 2\epsilon n$, $N_t(q)$ contains at most this number of good points in C_i. This means at least $\frac{1}{2}\min_i |C_i| - 2\epsilon n > m$ good points in C_i are outside $N_t(q)$, from which we can select Z_i. On the other hand, we can select Z_j as follows. When inserting q into C, we have $|N_t(q) \cap C| \geq \frac{2}{5}|C| \geq m + \epsilon n$. Since C contains only good points from C_j and some bad points, $N_t(q) \cap C$ contains at least m good points in C_j, from which

we can select Z_j. Since Z_j are from $N_t(q)$ and Z_i are outside $N_t(q)$, we have $d(q, Z_i) \geq d(q, Z_j)$.
(2) There exists $C' \in \mathcal{L}$ containing good points from C_i, but $C' \notin \mathcal{L}_p$. This means $|B(q,t) \cap C'| \leq \frac{2}{5}|C'|$, so there are at least $\frac{3}{5}|C'| \geq m + \epsilon n$ points in C' are outside $N_t(q)$. At least m of these points are good points from C_i, since C' contains only good points from C_i and at most ϵn bad points. On the other hand, we can select Z_j as in the first subcase.
(3) There exists $C' \in \mathcal{L}_p$ containing good points from C_i. Since q is assigned to C rather than C' according to median distances, we know that at least half of the points Z'_j from C are closer to q than at least half of the points Z'_i from C'. Since there are at most ϵn bad points, we can select m good points Z_j from Z'_j and select m good points Z_i from Z'_i. Note that Z_j are all from G_j and Z_i are all from G_i, so $d(q, Z_i) \geq d(q, Z_j)$. □

We are now ready to prove the approximation guarantee of our algorithm.

Theorem 1. *If the clustering instance is (α, ϵ)-perturbation resilient for $\alpha > 4$ and $\epsilon \leq \rho/30$ where $\rho = \frac{\min_i |C_i|}{n}$, then Algorithm 1 produces a clustering which is $(1 + \frac{5\epsilon}{\rho})$-approximation to the optimal clustering with respect to the k-median objective in time $O(n^{\omega+1})$, where $O(n^{\omega})$ is the state of the art for matrix multiplication.*

Proof. By Lemma 4, Algorithm 2 partition the points into a list of blobs, each of which has size at least $\frac{1}{2}\min_i |C_i|$ and contains only good points from one optimal cluster. Let B'_i denote the bad points that are assigned to blobs containing good points in C_i. By Lemma 1, Theorem 9 in [3] can be applied to $\mathcal{L}$, by which we know that $\{(C_i \cap G) \cup B'_i\}$ is a pruning of the tree. Suppose the cost of the optimum is $\mathcal{OPT}$. We now show that this pruning, using the original centers $\{c_i\}$, is a $(1 + \frac{5\epsilon}{\rho})$-approximation to $\mathcal{OPT}$.

Suppose a bad point $q \in C_i$ is assigned to a blob C containing good points from a different optimal cluster C_j. By Lemma 5, there exist $m = \frac{1}{5}\min_i |C_i|$ points Z_i from C_i, and m points Z_j from C_j, such that $d(q, Z_i) \geq d(q, Z_j)$. Then the increase in cost due to q is bounded as follows:

$$d(q, c_j) - d(q, c_i) \leq \frac{d(q, Z_j) + d(c_j, Z_j)}{m} - \frac{d(q, Z_i) - d(c_i, Z_i)}{m} \leq \frac{1}{m}[d(c_j, Z_j) + d(c_i, Z_i)] \leq \frac{\mathcal{OPT}}{m}.$$

As there are at most ϵn bad points and $m = \frac{\min_i |C_i|}{5}$, the cost increased is at most $\frac{5\epsilon}{\rho}\mathcal{OPT}$.

Running Time In Algorithm 2, for each $p \in S$, we first sort all the other points in ascending order of distances in time $O(n^2 \log n)$. At each threshold t, think of a directed t-regular graph E_t, where, for each point q in the t nearest neighbors of a point p, there is a directed edge from p to q in E_t. Let A_E denote the adjacent matrix for E_t, and let $N = A_E A_E^T$. Then N_{pq} is the number of common neighbors between p and q, which can be used in constructing F_t. Computing N takes time $O(n^{\omega})$, the state of the art for matrix multiplication. The same method can be used to compute the number of common neighbors in F_t and construct H_t. Since there are $O(n)$ thresholds, the total time for constructing F_t and H_t is $O(n^{\omega+1})$. For the other steps, adding a blob takes time $O(n^2)$ and inserting a point takes time $O(n^2)$. These steps can be performed at most $O(n)$ times, so they take $O(n^3)$ time. In total, Algorithm 2 takes time $O(n^{\omega+1})$. Since the robust linkage algorithm takes time at most $O(n^{\omega+1})$ ([3]), and the dynamic programming takes time $O(n^3)$, the running time of Algorithm 1 is $O(n^{\omega+1})$. □

B Proof of Theorem 3

Theorem 3. *Suppose (X, d) is (α, ϵ)-perturbation resilient for $\alpha > 4$, $\epsilon < \rho/100$. Then with probability $\geq 1 - \delta$, we can get an implicit clustering that is $2(1 + \frac{16\epsilon}{\rho})$-approximation in time poly$(\log \frac{N}{\delta}, k, \frac{1}{\epsilon}, \frac{1}{\zeta})$.*

Proof. We sample a set S of size $n = \Theta(\frac{k}{\epsilon^2\zeta^2} \ln \frac{N}{\delta})$ and run Algorithm 1 on S to obtain the minimum cost pruning $\tilde{\mathcal{C}}$ and its centers $\tilde{\mathbf{c}}$. The implicit clustering of the whole space X then assigns each point in X to its nearest neighbor in $\tilde{\mathbf{c}}$. Recall that if we partition A into $\mathcal{P}$, the cost using centers $\mathbf{p}$ is denoted as $\Phi_A(\mathcal{P}, \mathbf{p})$. If we partition A by assigning points to nearest centers in $\mathbf{p}$, the cost is denoted as $\Phi_A(\mathbf{p})$. We will show that the cost of implicit clustering $\Phi_X(\tilde{\mathbf{c}})$ approximates the optimum $\Phi_X(\mathbf{c})$.

First, we will prove that when n is sufficiently large, with high probability, $\Phi_X(\tilde{\mathbf{c}})/N \approx \Phi_S(\tilde{\mathbf{c}})/n$ and $\Phi_X(\mathbf{c})/N \approx \Phi_S(\mathbf{c})/n$. Formally, for every set of centers $\mathbf{p}$, if $n = \Theta(\frac{k}{\upsilon^2\zeta^2}\log\frac{N}{\delta})$ where $0 < \upsilon < 1$, then

$$\Pr\left[\left|\frac{\Phi_S(\mathbf{p})}{n} - \frac{\Phi_X(\mathbf{p})}{N}\right| > \upsilon\frac{\Phi_X(\mathbf{p})}{N}\right] \leq 2\exp\{-2\upsilon^2\zeta^2 n\} \leq \frac{\delta}{4N^k}.$$

By the union bound, we have with probability at least $1-\delta/4$, $(1-\upsilon)\Phi_X(\tilde{\mathbf{c}})/N \leq \Phi_S(\tilde{\mathbf{c}})/n$ and $\Phi_S(\mathbf{c})/n \leq (1+\upsilon)\Phi_X(\mathbf{c})/N$. We can choose $\upsilon = \epsilon/20$, then it is sufficient to show $\Phi_S(\tilde{\mathbf{c}}) \leq 2(1+\frac{12\epsilon}{\rho})\Phi_S(\mathbf{c})$.

Next, since $\tilde{\mathcal{C}}$ may be different from $\mathcal{C}\cap S$, we need to find a bridge for comparing $\Phi_S(\tilde{\mathbf{c}})$ and $\Phi_S(\mathbf{c})$. Now, we turn to analyze Algorithm 1 on S to find such a bridge. First, we know that X has at most ϵN bad points. Since n is sufficiently large, with probability at least $1-\delta/4$, S has at most $2\epsilon n$ bad points. Similarly, with probability at least $1-\delta/4$, for any $i, |C_i \cap S| > 60\epsilon n$. These ensure that Algorithm 1 can successfully produce a tree with a pruning $\mathcal{P}'$ that assigns all good points in S correctly, as shown in Theorem 1. Suppose in S, $\mathbf{c}'$ are the optimal centers for $\mathcal{P}'$. Then we can use $\Phi_S(\mathcal{P}', \mathbf{c}')$ as a bridge for comparing $\Phi_S(\tilde{\mathbf{c}})$ and $\Phi_S(\mathbf{c})$.

On one hand, $\Phi_S(\tilde{\mathbf{c}}) \leq \Phi_S(\tilde{\mathcal{C}}, \tilde{\mathbf{c}}) \leq \Phi_S(\mathcal{P}', \mathbf{c}')$. The first inequality comes from the fact that in $\Phi_S(\tilde{\mathbf{c}})$ each point is assigned to its nearest center and the second comes from that $\tilde{\mathcal{C}}$ is the minimum cost pruning.

On the other hand, $\Phi_S(\mathcal{P}', \mathbf{c}') \leq 2\Phi_S(\mathcal{P}', \mathbf{c}) \leq 2(1+\frac{12\epsilon}{\rho})\Phi_S(\mathbf{c})$. The first inequality comes from the triangle inequality. More precisely, for any $N_i' \in \mathcal{P}'$,

$$\begin{aligned}
2|N_i'|\sum_{p\in N_i'} d(p,c_i) &= \sum_{q\in N_i'}\sum_{p\in N_i'}[d(p,c_i)+d(q,c_i)] \geq \sum_{p\in N_i'}\sum_{q\in N_i'} d(p,q) \\
&\geq \sum_{p\in N_i'}\sum_{q\in N_i'} d(q,c_i') = |N_i'|\sum_{q\in N_i'} d(q,c_i')
\end{aligned}$$

which then leads to $2\sum_{p\in N_i'} d(p,c_i) \geq \sum_{q\in N_i'} d(q,c_i')$, and thus $\Phi_S(\mathcal{P}', \mathbf{c}') \leq 2\Phi_S(\mathcal{P}', \mathbf{c})$. The second inequality comes from an argument similar to that in Theorem 1 and the fact that $\Phi_S(\mathcal{P}', \mathbf{c})$ is different from $\Phi_S(\mathbf{c})$ only on the bad points. $\square$