High-Dimensional Axis-Aligned Bounding Box with Outliers

Ali Mostafavi* Ali Hamzeh†

Abstract

Given n points in d-dimensional space and a parameter z, we study the problem of finding the smallest axis-aligned bounding box that covers at least $n-z$ points and labels the remaining points as outliers. We consider two measures for the size of bounding box: length of the largest side and sum of lengths of the sides. We give two algorithms for the former case: one that uses at most $2z$ outliers but gives a bounding box which is at most as large as the optimal bounding box and another algorithm that uses at most z outliers but gives a box than can be twice as large as the optimal box. We also prove a matching lower bound for the approximation factor of these algorithms.

For the sum of the sides objective, we give a bi-criteria approximation algorithm that finds a box which is at most $O(\log d)$ larger than the optimal box by removing $O(z)$ points.

1 Introduction

We study the classic problem of finding the smallest axis-aligned box that contains a set of points in the setting where we are allowed to ignore z points by labelling them as outliers. The axis-aligned bounding box problem is extensively studied in literature because of its applications in pattern recognition [15], computer graphics [18, 17] and VLSI design [14]. Aggarwal *et al.* [1] gave an exact algorithm to minimize the perimeter for the case where points are in two dimensional plane which runs in $O((n-z)^2 n \log n)$ time. Eppstein and Erickson [8] showed that the time complexity can be improved to $O((n-z)^2 n)$ in the planar case and also gave an $O((n-z)n \log n + (n-z)^{d/2-1} n \log^2 (n-z))$ algorithm for the L_∞ objective (maximum side length) in higher dimensions.

Segal and Kedem [16] gave an $O(n + z(n-z)^2)$ algorithm for minimizing area and perimeter in the plane which is faster than previous ones when z is small. They also extend their algorithm to 3-dimensional space which runs in $O(n + z(n-z)^2 + (n-z)^5)$ time.

Ahn *et al.* [2] studied (p, z) *box covering* problem in the plane: find p disjoint axis-aligned rectangles that cover at least $n-z$ points minimizing the area of the largest box. They gave an $O(n+z^3)$ algorithm for the case where $p = 1$ (which is the same as our problem). They also show that this problem is NP-hard for general p. Atanassov *et al.* [3] studied many geometric problems with outliers and gave an $O(n+z^3)$ algorithm for minimum perimeter rectangle in the plane.

Kaplan *et al.* [10] developed algorithms for the minimum area and minimum perimeter rectangle with outliers which run in $O(n^{2.5} \log^2 n)$ and $O(n(n-z)^{1.5} \log (n-z) \log n)$ respectively.

de Berg *et al.* [7] considered the case where z is large ($n-z$ is small) and gave an $O(n(n-z)^2 \log n + n \log^2 n)$ time algorithm. They also studied the "dual" problem of covering the maximum number of points using a rectangle with area at most α and gave a randomized algorithm with running time $O(\frac{n}{\epsilon^4} \log^3 n \log \frac{1}{\epsilon})$ which covers at least $(1-\epsilon)\kappa^*$ points with high probability where κ^* is the maximum number of points coverable with such rectangle.

Guo and Li [9] presented an algorithm with running time $O(kz^3 + kzn + n^2 \log n)$ for the minimum area rectangle in the plane where k denotes the number of points on the first $z+1$ convex layers. This algorithm can be faster than previous ones when z and k are small.

Chan and Har-Peled [5] improved the running time for both the area and perimeter to $O(n(n-z) \log \frac{n}{n-z} \log (n-z))$ when points are in the two dimensional plane, they also gave an algorithm which finds a rectangle with area at most $1+\epsilon$ times the area of the optimal rectangle and runs in $O((1/\epsilon)^3 \log (1/\epsilon) n \log n)$.

Bae [4] studied the related *minimum width cuboidal shell* problem with outliers where cuboidal shell is defined as the area between a cube and its inward offset. They give an algorithm with running time $O(z^{2d} n)$.

In this paper, we present the first polynomial-time bi-criteria approximation algorithms for axis-aligned bounding box with outliers in high dimensional spaces. An (α, β)-approximation algorithm for axis aligned bounding box is one that achieves an objective value at most α times the optimal value by removing at most βz outliers. We consider two objectives: the maximum side length of the bounding box (L_∞) and the sum of the side lengths of the bounding box (L_1) which reduces to perimeter in 2-dimensional case.

- For the L_∞ objective, we give $(1, 2)$ and $(2, 1)$-approximation algorithms. Moreover, we prove ap-

*Department of Computer Engineering, Shiraz University, a.hr.mostafavi@gmail.com

†Department of Computer Engineering, Shiraz University, ali@shirazu.ac.ir

proximating the objective function with a factor better than 2 is NP-hard if we are not allowed to use more than z outliers.

- For the L_1 objective, we give a $(O(\log d), O(1))$-approximation algorithm.

2 Definitions and Terminology

Let $P \subset \mathbb{R}^d$ be a set of points in d-dimensional euclidean space and let $n = |P|$ be the number of points. Let $l_j(P)$ denote the extent of P in the j-th dimension, that is:

$$l_j(P) = \max_{p,q \in P} p_j - q_j$$

In this paper, we explore the following extent measures of the point set:

- Maximum side length of the axis-aligned bounding box:
$$L_\infty(P) = \max_{j=1}^{d} l_j(P)$$
- The sum of side lengths of the axis-aligned bounding box:
$$L_1(P) = \sum_{j=1}^{d} l_j(P)$$

Let $f(x)$ be any function on subsets of P, the problem of minimizing f with z outliers is to find z points in P such that removal of these points minimizes f, that is:

$$Z^* = \operatorname*{arg\,min}_{Z \subset P, |Z|=z} f(P \setminus Z)$$

we denote the optimal value of this function by $\text{OPT} = f(P \setminus Z^*)$ and the optimal set of points by $P^* = P \setminus Z^*$. For convenience we denote the optimal outlier points for the L_∞ objective by $Z^*_\infty(P,z) = \arg\min_{Z \subset P, |Z|=z} L_\infty(P \setminus Z)$ and the optimal non-outlier points are $P^*_\infty(P,z) = P \setminus Z^*_\infty(P,z)$ and the optimal objective value is $L^*_\infty(P,z) = L_\infty(P^*_\infty)$. $Z^*_1(P,z)$, $P^*_1(P,z)$ and $L^*_1(P,z)$ are defined analogously. For brevity, we will omit (P,z) when their value is obvious from the context (for example instead of $L^*_\infty(P,z)$ we just write L^*_∞).

3 Approximating L_∞ in High Dimensions

In this section we give the following results for the minimum-L_∞ axis-aligned bounding box with outliers:

1. In subsection 3.1 we give an approximation algorithm that labels $2z$ points as outliers but guarantees that the L_∞ value of the remaining points is less than $L^*_\infty(P,z)$
2. In subsection 3.2 we give an approximation algorithm that labels at most z points as outliers and guarantees that the L_∞ value of the remaining points is at most $2L^*_\infty(P,z)$
3. In subsection 3.3 we prove that under some reasonable assumptions both above approximation factors are optimal.

3.1 A $(1,2)$-approximation Algorithm

We repeatedly find the dimension with maximum side length and remove two extreme points along this dimension. We claim that Algorithm 1 achieves the optimal value of L_∞ and removes at most $2z$ points.

Algorithm 1 Approximation Algorithm for L_∞ in High Dimensions

```
1: procedure $L_\infty$-APPROXIMATION1($P$)
2:     for $i \leftarrow 1...z$ do
3:         $d_i = \arg\max_{j=1}^{d} l_j(P)$
4:         $p_i^{\min}, p_i^{\max}$ = extreme points of $P$ in $d_i$
5:         $P \leftarrow P \setminus \{p_i^{\min}, p_i^{\max}\}$
6:     return $P$
```

Lemma 1 *Let $Z_i = \{p_i^{\min}, p_i^{\max}\}$ be the two points removed in the i-th iteration of the for loop in Line 2, then one of the following holds:*

- $Z_i \cap Z^*_\infty \neq \emptyset$
- $L_\infty(P) \leq L^*_\infty$

Proof. Let P^*_∞ be the optimal set of points with their z outliers removed. Suppose $Z_i \cap Z^*_\infty = \emptyset$, therefore $Z_i \subseteq P^*_\infty$ and since L_∞ is a monotone function (that is, it can never increase by deleting points), we have:

$$L_\infty(P^*_\infty) \geq L_\infty(Z_i) = L_\infty(P)$$

□

Theorem 2 *The points returned by Algorithm 1 achieve at most the optimal value of L_∞.*

Proof. If at any point during Algorithm 1 we have $Z_i \cap Z^*_\infty = \emptyset$, then by Lemma 1 we have already achieved the optimum value (and we will never increase this value because L_∞ is a monotone function of points). Otherwise we have $\forall i : Z_i \cap Z^*_\infty \neq \emptyset$ and $\forall i,j : Z_i \cap Z_j = \emptyset$. Therefore for each i we have removed at least one new point from Z^*_∞, so after z iterations we have removed all z points from Z^*_∞.

□

The loop at line 2 is executed z times and each execution takes $O(nd)$ time so the overall runtime of the algorithm is $O(ndz)$. This runtime can be improved when dz is $o(n)$ by exploiting the following observation:

Observation 1 *The extreme point of P^*_∞ with the lowest (highest) coordinate in j-th dimension is among the $z+1$ extreme points of P with lowest (highest) j-th coordinate.*

Proof. This follows trivially from the fact that we have at most z outliers, therefore one of the $z+1$ most extreme points must be in the optimal solution. □

Therefore, instead of considering all n points, we can consider only the points which are among the $z+1$ most extreme points in some dimension. There are at most $2d(z+1)$ such points and they can be found in $O(nd)$ time using standard selection algorithms [6]. Running the algorithm only on these points reduces the overall runtime to $O(nd + d^2z^2)$.

3.2 A $(2,1)$-approximation Algorithm

Observation 1 implies that if a point is not among the $2dz$ most extreme points, it can not be an outlier and therefore it must be included in the optimal box. Additionally, any cube with side length l can be covered with a cube with side length $2l$ centered at any point inside the cube. Therefore if all except z points can be covered with a cube of maximum side length l, then all but z points can be covered with a cube centered at any non-outlier point with side length $2l$. This suggests Algorithm 2 for the case when $n > 2dz$. We know that the point c found in line 3 can not be an outlier, and therefore, we must be able to cover $n-z$ points with a cube centered on c and length at most $2L^*_\infty$.

Algorithm 2 $(2,1)$-approximation Algorithm for L_∞ in High Dimensions

```
1: procedure $L_\infty$-APPROXIMATION2($P$)
2:     Let $P'$ be the points of $P$ with $2dz$ most extreme points in each dimension deleted
3:     Let $c$ be an arbitrary point in $P'$
4:     Find the smallest $l$ such that a cube with side length $l$ centered at $c$ can cover at least $n-z$ points in $P$
5:     Label all the points not covered in the cube found in line 4 as outliers and return the cube as the solution
```

Line 2 can be performed in $O(nd)$ time using selection algorithms. Line 4 can be performed in $O(nd \log n)$ time by performing a binary search on the distance of c to the points in P. So the overall runtime of Algorithm 2 is $O(nd \log n)$.

3.3 Hardness of Approximation

We show that unless $P = NP$, we can not approximate the value of $L^*_\infty(P,z)$ with a factor better than 2 when no approximation on z is allowed. This proves that Algorithm 2 is optimal and justifies our approximation on z in Algorithm 1.

We convert an instance of VERTEX-COVER [11] problem to an instance of minimum L_∞ axis-aligned bounding box with outliers such that the optimal value is 1 if the graph has a vertex cover of size k and is 2 otherwise (which means an approximation algorithm with a factor better than 2 can distinguish between these cases). Let (V,E,k) be an instance of VERTEX-COVER problem where V is the set of nodes and $E \subseteq V \times V$ is the set of edges. Our goal is to determine if the graph has a vertex cover of size at most k. We arbitrarily assign directions to edges of $|E|$ and convert each vertex into a point in $\mathbb{R}^{|E|}$. Note that there is a dimension corresponding to each edge in the graph, let d_e denote the dimension corresponding to edge e. Let p_v denote the point corresponding to vertex $v \in V$. We use the following rule to determine p_v:

$$p_v(d_e) = \begin{cases} -1, & \text{if } e = (v,*). \\ 1, & \text{if } e = (*,v). \\ 0, & \text{otherwise.} \end{cases}$$

Theorem 3 *Let $P = \{p_v | v \in V\}$. Then:*

$$L^*_\infty(P,k) = \begin{cases} 1, & \textit{if } (V,E) \textit{ has a vertex cover of size } k. \\ 2, & \textit{otherwise.} \end{cases}$$

Proof. Suppose (V,E) has a vertex cover $C \subset V$ where $|C| = k$. We label the points corresponding to vertices in C as outliers, let $P' = P \setminus \{p_v | v \in C\}$ be the remaining points. There are exactly two points in P with non-zero value for each edge and since we know we have removed at least one of the endpoints of each edge in P', there is at most one point in P' with non-zero value in each dimension. Therefore the points of P' can be covered with a box with maximum side length of 1.

Conversely, we can convert any solution where $L^*_\infty(P,z) = 1$ to a vertex cover by selecting the vertices corresponding to outliers in P which means that if (V,E) doesn't have a vertex cover with size k then $L^*_\infty(P,k) > 1$, but the only other possible value for $L^*_\infty(P,k)$ is 2. □

Assuming that unique games conjecture [12] is true, it is impossible to approximate VERTEX-COVER with a factor better than two [13] and we can strengthen Theorem 3 to claim that no (α,β)-approximation is possible where α and β are simultaneously less than 2. This proves that both Algorithms 1 and 2 are pareto-optimal.

4 Approximating L_1 in High Dimensions

In this section we develop bi-criteria approximation algorithms for the L_1 objective. First, we warm up by developing $(d, 2)$ and $(1, 2d)$-approximation algorithms in subsections 4.1 and 4.2 respectively. Then in subsection 4.3 we show how to combine the ideas of these algorithms to develop an algorithm that achieves a reasonable approximation factor both for the objective value and number of outliers.

4.1 A $(d, 2)$-approximation Algorithm

We claim that Algorithm 1 already gives us a $(d, 2)$ approximation factor.

Theorem 4 *Algorithm 1 is a $(d, 2)$-approximation algorithm for L_1.*

Proof. Let P^*_∞ and P^*_1 be the optimal set of points (with z outliers removed) for L_∞ and L_1 respectively and P_W be the result of applying Algorithm 1 on P. We have:

$$\begin{aligned} L_1(P_W) &= \sum_{j=1}^{d} l_j(P_W) \\ &\le d \max_{j=1}^{d} l_j(P_W) \\ &\le d \max_{j=1}^{d} l_j(P^*_\infty) \\ &\le d \max_{j=1}^{d} l_j(P^*_1) \\ &\le d \sum_{j=1}^{d} l_j(P^*_1) \\ &= dL_1(P^*_1) \end{aligned}$$

□

4.2 A $(1, 2d)$-approximation Algorithm

This algorithm is very similar to Algorithm 1, however, instead of deleting the extreme points just in the largest dimension, we delete all $2d$ extreme points in all dimensions.

Algorithm 3 Simple Approximation Algorithm for L_1 in High Dimensions

```
1: procedure $L_1$-APPROXIMATION($P$)
2:     for $i \leftarrow 1...z$ do
3:         $Z_i$ = at most $2d$ extreme points of $P$ in each dimension
4:         $P \leftarrow P \setminus Z_i$
5:     return $P$
```

We omit the full proof of correctness and approximation factor because it is essentially the same as the proof of Theorem 2. The main idea is that at each step, either we have already achieved optimality or at least one of the deleted points is in Z^*_1.

4.3 A Better Approximation Algorithm

In this section, we assume that the optimal value OPT $= L^*_1(P, z)$ is known. We will show how to remove this assumption in subsection 4.3.1. The main idea is the same as Algorithm 3, however, instead of removing the two extreme points of all dimensions, we only consider the "large" dimensions and ignore "small" dimensions by proving that their contribution to the solution can not be too large (Theorem 9). And since we know the value of OPT, we know that there can not be too many "large" dimensions in the optimal solution, therefore we know that a good portion of the deleted points are actual outliers (Theorem 8).

Procedure PRUNE is called $\log_{\frac{2}{1+\epsilon}} d$ times. Each run of PRUNE takes $O(n^2 d)$ time so the overall runtime of Algorithm 4 is $O(n^2 d \log_{\frac{2}{1+\epsilon}} d)$.

Algorithm 4 Approximation Algorithm for L_1 in High Dimensions

```
1: procedure PRUNE($P$, $D$)
2:     $n \leftarrow |D|$
3:     while $|D| > \frac{(1+\epsilon)n}{2}$ do
4:         for $d$ in $D$ do
5:             if $l_d(P) > \frac{2\mathrm{OPT}}{n}$ then
6:                 Remove the two extreme points of $P$ in dimension $d$
7:             else
8:                 $D \leftarrow D \setminus \{d\}$
9:     return $P$, $D$
10: procedure $L_1$-APPROXIMATION($P$)
11:     $D = \{1, 2, ..., d\}$
12:     for $i \leftarrow 1...\lceil \log_{\frac{2}{1+\epsilon}} d \rceil$ do
13:         $P, D \leftarrow$ PRUNE($P, D$)
14:     return $P$
```

Lemma 5 *Let $D, |D| > 0$ be a set of dimensions and let D' be the set of remaining dimensions after running* PRUNE(P, D) *then $\frac{|D'|}{|D|} \le \frac{1+\epsilon}{2}$.*

Proof. Line 3 removes dimensions until this condition is satisfied (we know that it can be eventually satisfied because we can reduce the value to 0 by removing all points). □

Corollary 6 *Let D_i be D after the i-th iteration of the loop at line 12. Then $|D_i| \le (\frac{1+\epsilon}{2})^i d$.*

Proof. $|D_0|$ is d and for each i we have $|D_{i+1}| \leq \frac{1+\epsilon}{2}|D_i|$. By multiplying all these inequalities we get:

$$|D_i| \leq \left(\frac{1+\epsilon}{2}\right)^i |D_0| = \left(\frac{1+\epsilon}{2}\right)^i d$$

□

Lemma 7 *Let $P_1^* = P \setminus Z_1^*$ be the optimal set of points and $C = \{j : l_j(P_1^*) > \frac{2\text{OPT}}{n}\}$. Then $|C| \leq \frac{n}{2}$.*

Proof.

$$\begin{aligned}\text{OPT} &= \sum_{j=1}^{d} l_j(P_1^*) \geq \sum_{j\in C} l_j(P_1^*)\\ &\geq \sum_{j \in C}\frac{2\text{OPT}}{n} = |C|\frac{2\text{OPT}}{n} \implies\\ \text{OPT} &\geq |C|\frac{2\text{OPT}}{n} \implies \frac{n}{2} \geq |C|\end{aligned}$$

□

Theorem 8 *Algorithm 4 removes at most $4(\epsilon^{-1}z + 2d)$ points.*

Proof. Let Z_i be the points removed the i-th time PRUNE was called, let D_i be the input dimensions to this iteration and let α_i be the number of times while loop in Line 3 was executed. We have:

$$|Z_i| \leq 2\alpha_i|D_i| = 2(\alpha_i - 1)|D_i| + 2|D_i|$$

which means:

$$\frac{|Z_i| - 2|D_i|}{2|D_i|} \leq \alpha_i - 1$$

Now, for the first $\alpha_i - 1$ iterations, we are sure that there are at least $\frac{(1+\epsilon)|D_i|}{2}$ dimensions remaining in D which we call active dimensions (this might not be true for the last iteration because we remove points inside the loop). On the other hand, Lemma 7 tells us that there are at most $\frac{|D_i|}{2}$ dimensions in the optimal solution whose extent is more than $\frac{2\text{OPT}}{|D_i|}$. This means that at least $\frac{(1+\epsilon)|D_i|}{2} - \frac{|D_i|}{2} = \frac{\epsilon|D_i|}{2}$ of these dimensions are not optimal (the extent of points in these dimensions is still larger than the extent of optimal points in this dimension) which means at each non-final iteration we remove at least $\frac{\epsilon|D_i|}{2}$ points from Z_1^*. Thus we have:

$$|Z_i \cap Z_1^*| \geq (\alpha_i - 1)\frac{\epsilon|D_i|}{2} \geq \frac{|Z_i| - 2|D_i|}{2|D_i|}\frac{\epsilon|D_i|}{2} = \frac{\epsilon(|Z_i| - 2|D_i|)}{4}$$

therefore $4\epsilon^{-1}|Z_i \cap Z_1^*| + 2|D_i| \geq |Z_i|$. Now we can bound the number of points removed in all iterations (we use I to denote the number of iterations which is $\left\lceil \log_{\frac{2}{1+\epsilon}} d \right\rceil$):

$$\begin{aligned}\sum_{i=1}^{I}|Z_i| &\leq \sum_{i=1}^{I} 4\epsilon^{-1}|Z_i \cap Z_1^*| + 2|D_i|\\ &= \sum_{i=1}^{I} 4\epsilon^{-1}|Z_i \cap Z_1^*| + 2\sum_{i=1}^{I}|D_i|\end{aligned}$$

we have

$$\sum_{i=1}^{I}|Z_i \cap Z_1^*| \leq |Z_1^*|$$

because the Z_is are disjoint so the first sum is less than $4\epsilon^{-1}z$. For the second sum we apply Corollary 6 and sum the geometric series to get:

$$\sum_{i=1}^{I}|D_i| \leq \sum_{i=1}^{I}\left(\frac{1+\epsilon}{2}\right)^i d < \frac{2d}{1-\epsilon} < 4d$$

we assumed $\epsilon < \frac{1}{2}$ for the last inequality. Putting this all together we have:

$$\sum_{i=1}^{I}|Z_i| \leq 4(\epsilon^{-1}z + 2d)$$

□

Theorem 9 *The points returned by Algorithm 4 achieve an L_1 value of at most $\text{OPT}(2\left\lceil \log_{\frac{2}{1+\epsilon}} d \right\rceil)$.*

Proof. Each time PRUNE is called the deactivated dimensions have extent at most $\frac{2\text{OPT}}{|D_i|}$ and there are at most $|D_i|$ of them. So the total contribution of these dimensions to the objective is at most $\frac{2\text{OPT}}{|D_i|}|D_i| = 2\text{OPT}$. And PRUNE is called at most $\left\lceil \log_{\frac{2}{1+\epsilon}} d \right\rceil$ times after which there remains no active dimensions. □

4.3.1 How to Find OPT ?

In Section 4.3 we assumed we knew the value of OPT. Here we will show how to remove this assumption. Let L and U be a lower bound and an upper bound on the value of OPT. For example U can be $L_1(P)$ and L can be $\frac{1}{d}$ times the result of running Algorithm 1 on P. Let $\phi = \frac{U}{L}$. Run Algorithm 4 in parallel $\log_{1+\delta}\phi$ times each time with a guess of $L(1+\delta)^i$ for the value of OPT. For one of these guesses we have $\text{OPT} \leq L(1+\delta)^i \leq (1+\delta)\text{OPT}$ and for that guess Algorithm 4 is guaranteed to succeed with a value of at most $2(1+\delta)\left\lceil \log_{\frac{2}{1+\epsilon}} d \right\rceil \text{OPT}$.

While this method probably works well in most practical datasets, one can design adversarial datasets where the value of ϕ is unbounded. Here we will describe a method to find bounds whose ratio is guaranteed to be a polynomial. Let P_W be the result of applying Algorithm 1 on P and $W = L_\infty(P_W)$. Let $L_\infty^*(P, z)$ and

$L_1^*(P, z)$ be the optimal value of L_∞ and L_1 objectives with z outliers on point set P. We have:

$$W \leq L_\infty^*(P, z) \leq L_1^*(P, z) \tag{1}$$

On the other hand, applying Algorithm 1 removes at most $2z$ points, so $L_1(P_W)$ is an upper bound on the value of $L_1^*(P, 2z)$.

$$\begin{aligned} L_1^*(P, 2z) \leq L_1(P_W) = \sum_{j=1}^{d} l_j(P_W) \\ \leq d \max_{j=1}^{d} l_j(P_W) = dW \end{aligned} \tag{2}$$

We set $L = W, U = dW$ and we use $z' = 2z$ as the number of outliers in Algorithm 4.

$$\phi = \frac{U}{L} = \frac{dW}{W} = d \implies \log_{1+\delta} \phi = O(\delta^{-1} \log d)$$

Equation 2 ensures that our algorithm succeeds for some estimate of OPT and Equation 1 ensures that at least some of our estimates are less than OPT. So either OPT is greater than dW or there is some estimate between OPT and $(1+\delta)$OPT. Either way, our algorithm succeeds for these estimates with approximation factor given in Theorem 9. This algorithm blows up our approximation factor for the number of outliers at most by a factor of two (from $4(\epsilon^{-1}z + 2d)$ to $8(\epsilon^{-1}z + d)$).

5 Conclusion

We presented simple bi-criteria approximation algorithms for minimum axis-aligned bounding box in high-dimensional euclidean spaces for the maximum side length and sum of side lengths objectives. While this problem has been previously studied in low-dimensional settings, as far as we are aware this is the first work that studies this problem in high-dimensional euclidean spaces. We proved that our algorithms for the maximum side length objective are pareto-optimal under some reasonable assumptions.

We plan on (i) improving the approximation factors for the sum of side length algorithm, (ii) prove matching lower bounds for sum of side length objective, and (iii) studying other high-dimensional problems in the presence of outliers.

References

[1] Alok Aggarwal, Hiroshi Imai, Naoki Katoh, and Subhash Suri. Finding k points with minimum diameter and related problems. *Journal of Algorithms*, 12(1):38–56, 1991.

[2] Hee-Kap Ahn, Sang Won Bae, Erik D Demaine, Martin L Demaine, Sang-Sub Kim, Matias Korman, Iris Reinbacher, and Wanbin Son. Covering points by disjoint boxes with outliers. *Computational Geometry*, 44(3):178–190, 2011.

[3] Rossen Atanassov, Prosenjit Bose, Mathieu Couture, Anil Maheshwari, Pat Morin, Michel Paquette, Michiel Smid, and Stefanie Wuhrer. Algorithms for optimal outlier removal. *Journal of Discrete Algorithms*, 7(2):239–248, 2009.

[4] Sang Won Bae. Minimum-width cuboidal shells with outliers. *Journal of Computing Science and Engineering*, 14(1):1–8, 2020.

[5] Timothy M. Chan and Sariel Har-Peled. Smallest k-enclosing rectangle revisited. *Discrete and Computational Geometry*, 66(2):769–791, 2021.

[6] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms.* MIT press, 2009.

[7] Mark De Berg, Sergio Cabello, Otfried Cheong, David Eppstein, and Christian Knauer. Covering many points with a small-area box. *Journal of Computational Geometry*, 10(1):207–222, 2019.

[8] David Eppstein and Jeff Erickson. Iterated nearest neighbors and finding minimal polytopes. *Discrete & Computational Geometry*, 11(3):321–350, 1994.

[9] Zhengyang Guo and Yi Li. Minimum enclosing parallelogram with outliers. *arXiv preprint arXiv:2003.01900*, 2020.

[10] Haim Kaplan, Sasanka Roy, and Micha Sharir. Finding axis-parallel rectangles of fixed perimeter or area containing the largest number of points. *Computational Geometry: Theory and Applications*, 81(52):1–11, 2019.

[11] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[12] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 767–775, 2002.

[13] Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2- ε. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

[14] Maharaj Mukherjee and Kanad Chakraborty. A polynomial-time optimization algorithm for a rectilinear partitioning problem with applications in VLSI design automation. *Information Processing Letters*, 83(1):41–48, 2002.

[15] Sung Hee Park and Jae-young Kim. Unsupervised Clustering with Axis-Aligned Rectangular Regions. `http://cs229.stanford.edu/proj2009/ParkKim.pdf`.

[16] Michael Segal and Klara Kedem. Enclosing k points in the smallest axis parallel rectangle. *Information Processing Letters*, 65(2):95–99, 1998.

[17] Daiki Takeshita. AABB pruning: Pruning of neighborhood search for uniform grid using axis-aligned bounding box. *The Journal of the Society for Art and Science*, 19(1):1–8, 2020.

[18] Bruce Walter, Kavita Bala, Milind Kulkarni, and Keshav Pingali. Fast agglomerative clustering for rendering. *RT'08 - IEEE/EG Symposium on Interactive Ray Tracing 2008, Proceedings*, pages 81–86, 2008.