Adaptive Computation of the Klee’s Measure in High Dimensions

Jérémy Barbay¹, Pablo Pérez-Lantero², and Javiel Rojas-Ledesma¹*

¹ Departamento de Ciencias de la Computación, Universidad de Chile, Chile. jeremy@barbay.cl, jrojas@dcc.uchile.cl.
² Escuela de Ingeniería Civil Informática, Universidad de Valparaíso, Chile. pablo.perez@uv.cl.

Abstract. The Klee’s Measure of \( n \) axis-parallel boxes in \( d \)-dimensional space is the volume of their union. It can be computed in time within \( O(n^{d/2}) \) in the worst case. We describe three techniques to boost its computation: two based on some type of “degeneracy” of the input, and one, more technical, on the inherent “easiness” of some instance.

The first technique takes advantage of instances where the intersection graph of the input has a small number \( e \) of edges and a number \( \rho \) of connected components of respective small sizes \( n_1, \ldots, n_\rho \), and yields a solution running in time within \( O(n \log^2 n + e + \sum_{i=1}^{\rho} n_i^{d/2}) \subseteq O(n^{d/2}) \).

The second one takes advantage of instances where the Maxima of the input is of small size \( h \), and yields a solution running in time within \( O(n \log^2 h + h^{d/2}) \subseteq O(n^{d/2}) \). The third and most technical technique takes advantage of instances where no \( d \)-dimensional axis-aligned hyperplane intersects more than \( k^* \) boxes in some dimension, and yields a solution running in time within \( O(n \log n + nk^{(d-2)/2}) \subseteq O(n^{d/2}) \).

1 Introduction

The Klee’s Measure of a set of \( n \) axis-parallel boxes in \( \mathbb{R}^d \) is defined as the volume of the union of the boxes in the set. In this problem, known as the Klee’s Measure problem, additionally a \( d \)-dimensional domain box \( \Gamma \) is given, making the objective to compute the Klee’s Measure within \( \Gamma \). It was first posed by Victor Klee in 1977 [15], who originally considered the problem for intervals in the real line. Bentley [5] generalized the problem to \( d \) dimensions and described an algorithm running in time within \( O(n^{d-1} \log n) \). Several years later, Overmars and Yap [17] described a solution running in time within \( O(n^{d/2} \log n) \), which remained essentially unbeaten for more than 20 years until Chan [8] presented in 2013 an algorithm running in time within \( O(n^{d/2}) \).

Special cases of this problem have been studied, such as the Hypervolume problem, where boxes are orthants of the form \( \{ (x_1, \ldots, x_d) \in \mathbb{R}^d | (x_1 \leq \alpha_1) \land \ldots \land (x_d \leq \alpha_d) \} \), being each \( \alpha_i \) a real number, which can be solved in time within \( O(n^{d/3} \text{ polylog } n) \); and CUBE-KMP [26], when the boxes are hypercubes.

* Corresponding Author
which can be solved in running time within $O(n^{(d+1)/3} \log n)$ \[8\]. Yildiz and Suri\[19\] considered $k$-GROUNDED-KMP, the case when the projection of the input boxes to the first $k$ dimensions is an instance of Hypervolume. They gave an algorithm to solve 2-GROUNDED in time within $O(n^{(d-1)/2} \log^2 n)$, for any dimension $d \geq 3$.

The best lower bound known for the computational complexity of the Klee’s Measure problem is $\Omega(n \log n)$: it is known to be tight only for dimensions $d = 1, 2$ \[11\]. Chan \[7\] conjectured that no ‘purely combinatorial’ algorithm computing the Klee’s Measure exists running in time within $O(n^{d/2-\varepsilon})$ for some $\varepsilon > 0$. He proved that if the $d$-dimensional Klee’s Measure problem can be solved in time within $T_d(n)$, then one can decide whether an arbitrary $n$-vertex graph $G = (V, E)$ contains a clique of size $d$ in time within $O(T_d(O(n^2)))$.

In an adaptive analysis, the cost of an algorithm is measured as a function of, not just the input size, but of other parameters that capture the inherent simplicity or difficulty of an input instance \[1\]. An algorithm is said to be adaptive if “easy” instances are solved faster than the ‘hard’ ones. There are adaptive algorithms to solve classical problems such as Sorting \[16,3\] a permutation or a multiset, computing the Convex Hull \[14\] of a set of points in the plane and in the space, computing the Maxima of a set of $d$-dimensional vectors \[13\]. There are also adaptive algorithms for the Maximum Weight Box problem \[4\], that for any dimension $d \geq 2$, can be reduced to an instance of the Klee’s Measure problem in $2d$ dimensions.

**Hypothesis:** Even though the complexity of $O(n^{d/2})$ is the best known so far for the Klee’s Measure problem, there are many cases which can be solved in time within $O(n \lg n)$ (see Fig. \[1\] and Fig. \[2\] for examples). Some of those “easy” instances can be mere particular cases, some others can be hints of some hidden measures of difficulty of the Klee’s Measure problem. Such measures should gradually separate instances of the same size $n$ into various classes of difficulty, from easy ones solvable in time within $O(n \lg n)$ to difficult ones, which the best known algorithm solves in time within $O(n^{d/2})$.

**Results:** We describe three techniques to boost the computation of the Klee’s Measure, and analyze each in the adaptive model. For each, we identify a proper difficulty measure, which models the features which the technique is taking advantage of. The first two techniques are the simplest, taking advantage of degenerate instances, while the third one is more technical.

The first technique (described in Sect. \[2\]) is based on the intersection graph of the input set of boxes, a graph $G$ where the vertices are the boxes, and where two boxes are connected by an edge if and only if they intersect. Edelsbrunner \[10\] showed how to compute the intersection graph of a set with $n$ boxes in time within $O(n \log^{2d-3} n + e)$, where $e$ is the number of edges in the graph. This technique yields a solution computing the Klee’s Measure in time within
Fig. 1: Two ‘easy’ instances of the Klee’s Measure problem: red dashed boxes in (a) can be removed without affecting the Klee’s Measure (the shaded area) within the domain $\Gamma$, yielding a much smaller instance to solve; while the instance in (b) belongs to a family that can be solved in time within $O(n \log n)$ by a divide-and-conquer algorithm.

$O(n \log^{2d-3} n + e + \sum_{i=1}^{\rho} n_i^{d/2}) \subseteq O(n^{d/2})$, where $\rho$ denotes the number of connected components of the intersection graph, and $n_i$ denote their respective sizes for $i = [1..\rho]$.

The second technique (described in Sect. 3) is related to a classical problem in Computational Geometry: the computation of the Maxima of a set of vectors. A vector in a set $T \subset \mathbb{R}^d$ is called maximal if none of the remaining vectors in $T$ dominates it in every component. The Maxima of $T$ (denoted by $M(T)$) is the set of maximal elements in $T$. In 1985, Kirkpatrick and Seidel [13] gave an output-size sensitive algorithm for this problem, running in time within $O(n \log^{d-2} h)$, where $h$ is the size of the Maxima. In what follows, we define the Maxima of a set of boxes, and describe an algorithm computing the Klee’s Measure in time within $O(n \log^{2d-2} h + h^{d/2}) \subseteq O(n^{d/2})$, where $h$ denotes the size of the Maxima of the input set.

The third technique (described in Sect. 4) is based on the profile of the input set, which D’Amore et al. [9] defined as the minimum, over all dimensions $i$, of the maximum number of boxes intersected by a same axis aligned hyperplane orthogonal to $i$. D’Amore et al. [9] studied the separation of a set of boxes by means of a cutting axis-aligned hyperplane, and showed that, for any set of boxes with profile $k^*$, there always exists a cut with at most $\left\lfloor \frac{n+k^*}{2} \right\rfloor$ boxes intersecting each side, and that this cut can be found in time within $O(n)$. Inspired by these results, we describe an algorithm to compute the Klee’s Measure in time within $O(n \log n + nk^{*(d-2)/2})$, where $k^*$ is the profile of the input set.

In Sect. 5 we discuss how to compare and combine these three techniques, and list some remaining open questions.
2 Intersection Graph Decomposition

We describe a technique, based on the intersection graph of the input set, to take advantage of degenerated inputs where boxes can be separated into a large number of small disjoint subsets that do not intersect.

By definition of the intersection graph (see page 2 of Sect. [1]), if two boxes belong to distinct connected components of the graph, they do not intersect. As a consequence, we can compute the **Klee’s Measure** of each connected component of the intersection graph separately, and add up those measures to obtain the **Klee’s Measure** of the complete set. This is the main idea of Algorithm [1], where we denote by SDC-KMP the algorithm described by Chan [8] to solve the **Klee’s Measure** problem.

```
Algorithm 1 intergraph_adaptive_measure

Input: A d-dimensional domain box Γ, and a set of n d-dimensional boxes B
Output: The Klee’s Measure of B within Γ
1: Compute G, the intersection graph of B
2: Let B₁,...,Bᵣ denote the sets of boxes corresponding to the ρ connected components of G, respectively
3: return ∑ᵣᵢ=1 SDC-KMP(Γ, Bᵢ)
```

We use the number of connected components and the size of each connected component of the intersection graph of the input set, as a measure of the difficulty of the input instances, and show that Algorithm [1] runs in time within the bound given in Theorem [1].

**Theorem 1.** Let $B = \{b₁,b₂,...,bₙ\}$ be a set of n boxes in $\mathbb{R}^d$, let $Γ$ be a d-dimensional box, and let $G$ be the intersection graph of $B$. The measure of the union of $B$ within $Γ$ can be computed in time within $O(n \log^{2d-3} n + e + \sum_{i=1}^{ρ} n_i^{d/2})$, where $e$ and $ρ$ are the number of edges and connected components of $G$, respectively, and $n₁,...,nₖ$ are the sizes of the $ρ$ connected components.

**Proof.** The first step of the algorithm can be done in time within $O(n \log^{2d-3} n + e)$ if we use the algorithm described by Edelsbrunner [10]. The measure of each connected component, of size $n_i$, is computed by using SDC-KMP in time within $O(n_i^{d/2})$. The time bound in the theorem follows. □

Note that in the bound from the previous theorem, the values exponentiated to $d/2$ are the $n_i$. When these values are significantly smaller than $n$, the bound from Theorem [1] is significantly better than $O(n^{d/2})$, which is the bound for the running time of SDC-KMP. One can imagine other boosting techniques based on the intersection graph: we discuss the difficulty of analyzing those in Sect. [3]. We describe in the next section another boosting technique based on a different kind of degenerated instances.
3 Maxima Filtering

We describe a technique which considers the Maxima of the input set of boxes to take advantage of instances where many boxes can be “filtered out” in small time. A box in a set $B$ is called maximal if none of the remaining boxes in $B$ completely contains it. The Maxima of $B$ (denoted by $M(B)$) is the set of maximal elements in $B$. One can observe that, by definition, elements not in the Maxima of an input set of the Klee’s Measure problem can be removed from the input set without affecting the value of the Klee’s Measure. Algorithm 2 takes advantage of this fact to compute the Klee’s Measure in time sensitive to the size of the Maxima of the input set.

```
Algorithm 2 maxima_adaptive_measure
Input: A $d$-dimensional domain box $\Gamma$, and a set of $n$ $d$-dimensional boxes $B$
Output: The Klee’s Measure of $B$ within $\Gamma$
1: Compute $M(B)$, the Maxima of $B$
2: return SDC-KMP($\Gamma, M(B)$)
```

Overmars [18] showed that if the Maxima of a set of $nd$-dimensional vectors can be computed in time within $O(T_d(n))$, then the Maxima of a set of $n$ boxes can be computed in time within $O(T_{2d}(n))$. To prove this, Overmars [18] expressed each box $b_i = [l_{i,1}, u_{i,1}] \times \ldots \times [l_{i,d}, u_{i,d}]$ as a $2d$ dimensional vector $\vec{b}_i = (-l_{i,1}, u_{i,1}, \ldots, -l_{i,1}, u_{i,d})$. Note that if $b_i, b_j$ are boxes, then $b_i$ dominates $b_j$ if and only if $\vec{b}_i$ dominates $\vec{b}_j$. We use this result in the proof of Theorem 2, where we provide a bound for the running time of Algorithm 2.

**Theorem 2.** Let $B = \{b_1, b_2, \ldots, b_n\}$ be a set of $n$ boxes in $\mathbb{R}^d$ and $\Gamma$ a $d$-dimensional box. The measure of the union of $B$ within $\Gamma$ can be computed in time within $O(n(\log h)^{2d-2} + h^{d/2})$, where $h$ is the size of $M(B)$.

**Proof.** As a direct consequence of the result by Overmars [18], we can use the output-size sensitive algorithm described by Kirkpatrick and Seidel [13] to compute the Maxima $M(B)$ of $B$ in time within $O(n \log^{2d-2} h)$. After that, computing the Klee’s Measure of $M(B)$, of size $h$, takes time within $O(h^{d/2})$ because of the running time of SDC-KMP. The time bound follows. □

Note again that in the bound from the previous theorem, the value exponentiated to $d/2$ is $h$, instead of $n$ as in the bound $O(n^{d/2})$ for the running time of SDC-KMP. In degenerated instances, where $h$ is significantly smaller than $n$, the bound from Theorem 2 is significantly better than $O(n^{d/2})$. One way to further improve this result would be to remove dominated elements at each recursive: we discuss the difficulties in analyzing this approach in Sect. 5. We describe in the next section another boosting technique, this one not based on degenerated instances.
4 Profile-based Partitioning

In this section, we describe a technique to compute the Klee’s Measure of a set $B$, which yields a solution sensitive to the profile of $B$. The $i$-th profile of a set of boxes $B$ (which we denote $k_i$) is defined as the maximum number of boxes that intersect a same hyperplane orthogonal to the $i$-th dimension. The profile of a set of boxes (denoted $k^*$) is defined as the minimum of $k_i$, for $i \in [1..d]$. The value of the profile $k^*$ of $B$ can be computed in time linear in the size $n$ of $B$, by a line sweeping algorithm [9].

We first describe an algorithm that, given a domain $\Gamma$, a set $B$, and an upper bound $k$ of the profile of $B$, splits $\Gamma$ into two cells intersecting a balanced number of boxes of $B$, while the boxes intersected simultaneously by the two cells are at most $k$ (Algorithm 3). We then describe an algorithm that computes the Klee’s measure (Algorithm 4) by using Algorithm 3 conveniently, and running in time sensitive to the profile of the input set. We denote by $L_i(b)$ the lowest $i$-th coordinate of a box $b$ (i.e., if $b = [l_1, u_1] \times \ldots \times [l_i, u_i] \times \ldots \times [l_d, u_d]$, then $L_i(b) = l_i$).

Algorithm 3 split-domain

Input: A domain $\Gamma$, a set of $n$ boxes $B$, and an upper bound $k$ of the profile of $B$
Output: A split of $\Gamma$ into two cells intersecting each one at most $\lceil \frac{n+k}{2} \rceil$ boxes.
1: let $i$ be a dimension where the $i$-th profile $k_i$ of $B$ satisfies $k_i \leq k$
2: $L \leftarrow \{ L_i(b) \mid b \in B \}$
3: $m \leftarrow \lceil \frac{n+k}{k} \rceil + 1$
4: let $l$ be the $m$-th smallest element of $L$
5: split $\Gamma$ into $\Gamma_L, \Gamma_R$ by the hyperplane $x_i = l$
6: return $[\Gamma_L, \Gamma_R]$

Algorithm 3 is a slightly modified version of the one described by D’Amore et al. [9]. It runs in linear time, and returns a split of $\Gamma$ into two cells intersecting each at most $\lceil \frac{n+k}{k} \rceil$ boxes:

Lemma 1. Let $B$ be a set of $n$ boxes in $\mathbb{R}^d$, $\Gamma$ a $d$-dimensional domain box, and $k$ an upper bound for the profile of $B$. The algorithm split-domain runs in time within $O(n)$, and returns a split of $\Gamma$ into two cells intersecting at most $\lceil \frac{n+k}{k} \rceil$ boxes of $B$ each.

Proof. For the running time bound, note that Steps 3, 5 and 6 take constant time. Step 1 can be done in linear time by computing the profile in each dimension until finding the first one meting the condition. Step 2 is performed in linear time by taking the lowest $i$-th coordinate of each box, and Step 4 corresponds to the selection of the $m$-th smallest element of a list, supported in linear time [12]. The $O(n)$ bound follows.

For the second part of the lemma, observe that, since the algorithm splits the domain by the hyperplane $x_i = l$, where $l$ is the $\lceil \frac{n+k}{k} \rceil$-th smallest element
of $L$, there are at most $\lfloor \frac{n+k}{2} \rfloor$ boxes to the left of the cutting hyperplane (i.e., intersecting the interior of $\Gamma_L$). Some of them are fully contained within $\Gamma_L$, and others intersect $\Gamma_L$ and $\Gamma_R$ at the same time. However, the ones that are at the left of the cutting hyperplane and intersect $\Gamma_R$ are at most $k-1$ since they must intersect the cutting hyperplane (the -1 term comes from the fact that the hyperplane already intersects the box starting at it). Thus, there are at least $\lceil \frac{n+k}{2} \rceil - k + 1 \geq \lceil \frac{n-k}{2} \rceil$ boxes completely contained within $\Gamma_L$, and as a consequence, at most $n - \lceil \frac{n-k}{2} \rceil = \lfloor \frac{n+k}{2} \rfloor$ boxes intersecting $\Gamma_R$. \hfill \Box

Algorithm 4 computes, given a domain $\Gamma$, a set $B$, and an upper bound $k$ of the profile of $B$, the Klee’s Measure of $B$ within $\Gamma$ in time within $O \left( (n-k) \log \frac{n-k}{k} + (n-k) k^{d-2} \right) \subseteq O \left( n \log n + nk^{\frac{d}{2}} \right) \subseteq O(n^{d/2})$, as showed in Theorem 3.

**Algorithm 4 profile-adapt-kmp**

**Input:** A domain $\Gamma$, a set of boxes $B$, and an upper bound $k$ of the profile of $B$

**Output:** The Klee’s Measure of $B$ within $\Gamma$

1: if $|B| \leq 2k$ then 2: \hspace{1em} return SDC-KMP($\Gamma, M(B)$)
3: else 4: $[\Gamma_L, \Gamma_R] \leftarrow \text{split-domain}(\Gamma, B, k)$ 5: let $B_L, B_R \subseteq B$ be the subsets of boxes intersecting $\Gamma_L, \Gamma_R$ respectively 6: return profile-adapt-kmp($\Gamma_L, B_L, k$) + profile-adapt-kmp($\Gamma_R, B_R, k$)

Note that in the input of Algorithm 3 and Algorithm 4, $k$ is an upper bound of the profile of the set, and not the exact profile. This is because in Algorithm 4 before making the recursive calls in Step 6, the profiles of $B_L$ and $B_R$ are not recomputed, and they could be smaller than $k$. Recomputing these exact values do not impact negatively the asymptotic bound for the running time of Algorithm 4; it can be done in linear time, and linear time is already required by step 4. However, these exact values are not required to prove the upper bound we provide in Theorem 3 for the running time Algorithm 4 so we avoided recomputing them to keep Algorithm 4 as clean and simple as possible.

In the rest of the section we describe an expression for the running time $T(n)$ of the algorithm, and show that $T(n) \in O \left( (n-k) \log \frac{n-k}{k} + (n-k) k^{\frac{d}{2}} \right)$.

Step 2 of Algorithm 4 runs in time within $O(k^2)$ and Steps 4-5 can be performed in time within $O(n)$. If we denote by $c_1$ and $c_2$ the hidden constants in these asymptotic bounds respectively, the running time $T(n)$ of Algorithm 4 is bounded by the expression

$$T(n) \leq \begin{cases} c_1 k^2 & \text{if } n \leq 2k \\ 2T \left( \frac{n+k}{2} \right) + c_2 n & \text{if } n > 2k \end{cases} \quad (1)$$
We now prove some preliminary results to be used in the proof of a bound for $T(n)$. Let the function $h(n)$ be defined, for a known value $k$, as follows:

$$h(n) = \begin{cases} 
0 & \text{if } n \leq 2k \\
\lceil \log \frac{n-k}{k} \rceil & \text{if } n > 2k 
\end{cases}$$  

(2)

One can think of $h(n)$ as the function that describes the height of the recursion tree of Algorithm 4. Note that this height over an instance of size $n$ equals one plus the height of the recursion tree over an instance of size $(n + k)/2$. The following lemma shows that $h(n)$ behaves this way.

**Lemma 2.** For all $n > 2k$, $h(n) = h\left(\frac{n+k}{2}\right) + 1$

**Proof.** If $n > 2k$, but $\frac{n+k}{2} \leq 2k$, then $n \leq 3k$, and hence $h(n) = 1$. Since $h\left(\frac{n+k}{2}\right) = 0$, the condition holds. Otherwise $n > 2k$, $\frac{n+k}{2} > 2k$, and the following proves the lemma:

$$h(n) = \left\lceil \log \frac{n-k}{k} \right\rceil = \left\lceil \log \frac{n-k}{k} - 1 + 1 \right\rceil$$

$$= \left\lceil \log \left(\frac{n-k}{2k}\right) + 1 \right\rceil = \left\lceil \log \frac{n+k-k}{k} + 1 \right\rceil$$

$$= h\left(\frac{n+k}{2}\right) + 1 \quad \Box$$

This property of $h(n)$ can be used to prove a first upper bound for the running time $T(n)$ of Algorithm 4 which we prove by mere induction on the value of $n$ in Lemma.

**Lemma 3.** For all $n > 0$, there exists constants $c_1, c_2$ such that

$$T(n) \leq c_2 \left( (n-k)h(n) + (2^{h(n)} - 1)k \right) + 2^{h(n)}c_1k^{d/2}.$$  

**Proof.** Let $c_1$ and $c_2$ be the constants in (1). We show that the bound holds for this constants by induction on the value of $n$.

**Base case:** When $n \leq 2k$, $T(n) \leq c_1k^{d/2}$ by (1), and $h(n) = 0$ by (2). The right part of the inequality becomes

$$c_2((n-k) \times 0 + (2^0 - 1)k) + 2^0c_1k^{d/2} = c_2(0 + (1 - 1)k) + c_1k^{d/2}$$

$$= c_1k^{d/2}$$

**Inductive Hypothesis** (IH): For all $n$ such that $0 < n \leq n_0$,

$$T(n) \leq c_2 \left( (n-k)h(n) + (2^{h(n)} - 1)k \right) + 2^{h(n)}c_1k^{d/2}$$

$$+ c_1k^{d/2}.$$
Inductive Step: We now show that $T(n_0)$ also meets the bound of the lemma:

\[
T(n_0) \leq 2T\left(\frac{n_0 + k}{2}\right) + c_2n_0 \quad \text{by } [1]
\]

\[
\leq 2c_2\left(\frac{n_0 + k}{2} - k\right) + (2^{h\left(\frac{n_0 + k}{2}\right)} - 1)k
\]

\[
+ 2^{h\left(\frac{n_0 + k}{2}\right)} + 1) + c_2n_0 \quad \text{by } (IH)
\]

\[
= c_2\left(n_0 - k\right) + (2^{h\left(n_0\right)} - 1)k + 2^{h\left(n_0\right)} + 1) + c_2n_0
\]

\[
+ 2^{h\left(n_0\right)} + 1) + c_2n_0
\]

\[
= c_2\left(n_0 - k\right) + (2^{h\left(n_0\right)} - 1)k + 2^{h\left(n_0\right)} + 1) + c_2n_0
\]

\[
\leq c_2\left(n_0 - k\right) + (2^{h\left(n_0\right)} - 1)k + 2^{h\left(n_0\right)} + 1) + c_2n_0
\]

\[
\in O\left(n - k^*\right)\log\left(\frac{n - k^*}{k^*}\right) + (n - k^*)k^*\]
5 Discussion

We described here how these three techniques described above combine, and how they could be further improved.

Comparing and Combining the Techniques: None of the three measures of difficulty that we considered is better than the other ones on every instance. For example, an instance of $n$ boxes in the class of Fig. 2a has a MAXIMA of size $n$, and its profile is within $\Omega(n)$. With respect to these measures this is a ‘hard’ instance: Algorithm 2 and Algorithm 4 run in time within $O(n d/2)$. However, since none of the boxes in the instance intersect, its intersection graph has no edges, $n$ connected components, and the size of each connected component is one. With respect to this parameters this is an “easy” instance: Algorithm 1 runs in time within $O(n \log 2^{d/2} - 3 n)$. Similar examples exists for the MAXIMA-size and profile based measures.

Since this measures are independent, we can easily combine them to obtain an algorithm sensitive to the three measures, at the same time, as follows: (i.) Compute the intersection graph of the input set; (ii.) Compute the MAXIMA of each connected component; (iii.) Find the KLEE’S MEASURE of the MAXIMA of each connected component using Algorithm 4 and return the summation of these measures. This way of proceeding yields an algorithm with running time within $O(n \log 2^{d/2} - 3 n + e + \sum_{i=1}^{\rho} n_i \log 2^{d/2} h_i + h_i k_i d/2)$, where $e$ and $\rho$ are the number of edges and connected components of the intersection graph, $n_i$ is the size of the $i$-th connected components, and $h_i, k_i$ are the size and profile of its maxima, respectively, for $i = [1..\rho]$.

Further improvements: Each of the three boosting techniques that we analyzed can be improved, even though it is not yet clear how to analyze those improvements.

The first technique (described in Sect. 2) yields an algorithm running in time within $O(n \log h)^{2^{d/2} + h d/2}$, where $h$ is the size of the MAXIMA of the input set. This bound can probably be improved. For example, if instead of filtering items not in the MAXIMA only once, we do it as part of the simplification step of the algorithm SDC-KMP described by Chan [8], the expression for the running time of this algorithm becomes $T(n) = 2 T(\frac{h}{n^{d/2}}) + O(n \log 2^{d/2} h)$. This running time is still within $O(n^{d/2})$ in the worst case, and also within $O(n \log 2^{d/2} h + h d/2)$. It is never worse (asymptotically) and it is be better for some cases, but how to analyze formally this improvement is left as an open question.

In the second technique (described in Sect. 3) we considered the number $e$ of edges and $\rho$ of connected components of the intersection graph of the input, and the sizes $n_1, \ldots, n_\rho$ of the $\rho$ connected components. We described an algorithm running in time $O(n \log 2^{d/2} - 3 n + e + \sum_{i=1}^{\rho} n_i d/2)$. There are, possibly, other measures of difficulty based on the intersection graph of the input set. For instance, when a graph is a forest, it can be split in two disjoint subgraphs, with at most $n/2$ vertices, by removing at most 2 vertices. This idea can be used to compute
the Klee’s Measure of a set, which intersection graph is a forest (like the one illustrated in Fig. 2b), in time within $O(n \log n)$. The same time bound can be obtained if in the intersection graph all cycles are disjoint, or if there is a small number of non-disjoint cycles. Whether this behavior can be generalized into a boosting technique for the Klee’s Measure problem, and formally analyzed via an adaptive difficulty measure is left open.

The third technique (described in Sect. 4) is based on the profile of the set, defined as the maximum number $k^*$ (minimized over all dimensions) of boxes in the input set that intersect any $d$-dimensional axis-aligned hyperplane. The technique yields a solution running in time within $O \left( \left( n - k^* \right) \log \frac{n - k^*}{k^*} + nk^* \frac{d^2}{2} \right)$.

The algorithm SDC-KMP described by Chan [8] seems to be already adaptive to the profile of the input set. It has a limitation though: it necessarily cycles over the dimensions in order to ensure running in time within $O(n^{d/2})$. If there are few dimensions where the profile of the set is small, Algorithm 4 will perform considerably better than SDC-KMP. Algorithm 4 could be further improved if, instead of considering an upper bound for the profile in each subproblem, we use the exact value of the profile. However, it is not yet clear how to analyze this improvements.

There are instances that are “easy” to other measures we did not considered, and ‘hard’ to the ones we did. See for example the class of instances illustrated in Fig. 2b any instance in the class has an intersection graph with only one connected component, the size of its maxima is $n$, and its profile is within $\Omega(n)$. Instances in this class however can be solved in time within $O(n \log n)$ since their intersection graph is a tree.
References

1. Afshani, P., Barbay, J., Chan, T.M.: Instance-optimal geometric algorithms. In: Proceedings of the Annual IEEE Symposium on Foundations of Computer Science (FOCS). pp. 129–138 (2009)
2. Agarwal, P.K.: An improved algorithm for computing the volume of the union of cubes. In: Proceedings of the Annual Symposium on Computational Geometry (SoCG). pp. 230–239. ACM, New York, NY, USA (2010)
3. Barbay, J.: From time to space: Fast algorithms that yield small and fast data structures. In: Space-Efficient Data Structures, Streams, and Algorithms (IanFest). pp. 97–111 (2013)
4. Barbay, J., Chan, T.M., Navarro, G., Pérez-Lantero, P.: Maximum-weight planar boxes in $O(n^2)$ time (and better). Information Processing Letters (IPL) 114(8), 437–445 (2014)
5. Bentley, J.L.: Algorithms for Klee’s rectangle problems. Unpublished notes (1977)
6. Bringmann, K.: An improved algorithm for Klee’s measure problem on fat boxes. Computational Geometry: Theory and Applications (CGTA) 45(5-6), 225–233 (2012)
7. Chan, T.M.: A (slightly) faster algorithm for klee’s measure problem. In: Proceedings of the Annual Symposium on Computational Geometry (SoCG). pp. 94–100 (2008)
8. Chan, T.M.: Klee’s measure problem made easy. In: Proceedings of the Annual IEEE Symposium on Foundations of Computer Science (FOCS). pp. 410–419 (2013)
9. d’Amore, F., Nguyen, V.H., Roos, T., Widmayer, P.: On optimal cuts of hyperrectangles. Computing 55(3), 191–206 (1995)
10. Edelsbrunner, H.: A new approach to rectangle intersections part i. Journal of Computer Mathematics (JCM) 13(3-4), 209–219 (1983)
11. Fredman, M.L., Weide, B.: On the complexity of computing the measure of $\cup_i [a_i, b_i]$. Communications of the ACM (CACM) 21(7), 540–544 (Jul 1978)
12. Hoare, C.A.R.: Algorithm 65: Find. Communications of the ACM (CACM) 4(7), 321–322 (1961)
13. Kirkpatrick, D.G., Seidel, R.: Output-size sensitive algorithms for finding maximal vectors. In: Proceedings of the Annual Symposium on Computational Geometry (SoCG). pp. 89–96 (1985)
14. Kirkpatrick, D.G., Seidel, R.: The ultimate planar convex hull algorithm. SIAM Journal on Computing (JC) 15(1), 287–299 (Feb 1986)
15. Klee, V.: Can the measure of $\cup_i [a_i, b_i]$ be computed in less than $o(n \log n)$ steps? The American Mathematical Monthly (AMM) 84(4), 284–285 (1977)
16. Moffat, A., Petersson, O.: An overview of adaptive sorting. Australian Computer Journal (AJC) 24(2), 70–77 (1992)
17. Overmars, M.H., Yap, C.: New upper bounds in Klee’s measure problem. SIAM Journal on Computing (JC) 20(6), 1034–1045 (1991)
18. Overmars, M.H.: On the equivalence of rectangle containment, rectangle enclosure and ECDF-searching. Tech. rep., University of Utrecht, Department of Computer Science (01 1981)
19. Yildiz, H., Suri, S.: On Klee’s measure problem for grounded boxes. In: Proceedings of the Annual Symposium on Computational Geometry (SoCG). pp. 111–120. ACM, New York, NY, USA (2012)