Constructing Levels in Arrangements and Higher Order Voronoi Diagrams*

Pankaj K. Agarwal[1] Mark de Berg[2] Jiří Matoušek[3]
Otfried Schwarzkopf[2]

Abstract

We give a simple lazy randomized incremental algorithm to compute $\leq k$-levels in arrangements of x-monotone Jordan curves in the plane, and in arrangements of planes in three-dimensional space. If each pair of curves intersects in at most s points, the expected running time of the algorithm is $O(k^2\lambda_s(n/k) + \min(\lambda_s(n)\log^2 n, k^2\lambda_s(n/k)\log n))$. For the three-dimensional case the expected running time is $O(nk^2 + \min(n\log^3 n, nk^2\log n))$. The algorithm also works for computing the $\leq k$-level in a set of discs, with an expected running time of $O(nk + \min(n\log^2 n, nk\log n))$. Furthermore, we give a simple algorithm for computing the order-k Voronoi diagram of a set of n points in the plane that runs in expected time $O(k(n-k)\log n + n\log^3 n)$.

1 Introduction

Arrangements of hyperplanes have been studied for a long time in combinatorial and computational geometry and yet they have kept some of their secrets. Some of the intriguing open questions are related to the concept of *levels*. We say that a point p is at level k with respect to a set H of non-vertical hyperplanes in $\mathbb{R}^d$ if there are exactly k hyperplanes in H that lie strictly above p. The k-level of an arrangement $\mathcal{A}(H)$ of hyperplanes is the closure of all facets of $\mathcal{A}(H)$ whose interior points have level k with respect to H—see Figure 1. The $\leq k$-level of $\mathcal{A}(H)$ is

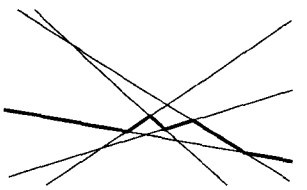

Figure 1: The 2-level in an arrangement of lines.

the union of all i-levels of $\mathcal{A}(H)$, for $0 \leq i \leq k$. The k-level and the $\leq k$-level of arrangements of monotone surfaces are defined analogously. In fact, one can give a more general definition of levels. Given a family Γ of subsets, also called *ranges*, of $\mathbb{R}^d$, define the level of a point p with respect to Γ to be the number of ranges that contain p in their interior; the k-level and $\leq k$-level in the arrangement of Γ are defined as before. For a set H of hyperplanes, if we choose ranges to be the half-spaces lying above the hyperplanes of H, then the level of p is same under the two definitions.

Although many researchers have studied combinatorial aspects of levels in arrangements, the maximum complexity of the k-level is still unknown. Even in the plane there is a large gap between the known upper and lower bound on the maximum complexity of the k-level: the best known lower bound is $\Omega(n\log(k+1))$ [Ede87], whereas the best known upper bound is $O(n\sqrt{k}/\log^*(k+1))$ [PSS92]. However, exact bounds are known for the maximum complexity of the $\leq k$-level in an arrangement of hyperplanes

*Work on this paper by P.A. has been supported by National Science Foundation Grant CCR-93-01259 and an NYI award. M.d.B. and O.S. acknowledge support by the Netherlands' Organization for Scientific Research (NWO) and partial support by ESPRIT Basic Research Action No. 7141 (project ALCOM II: *Algorithms and Complexity*). Work by J.M. was supported by Charles University Grant No. 351 and by Czech Republic Grant GAČR 201/93/2167. Part of this research was done when P.A. and O.S. visited Charles University, and when P.A. visited Utrecht University. These visits were supported by Charles University and NWO.

[1]Department of Computer Science, Box 90129, Duke University, Durham, NC 27708-0129, USA.

[2]Vakgroep Informatica, Universiteit Utrecht, Postbus 80.089, 3508 TB Utrecht, the Netherlands.

[3]Katedra aplikované matematiky, Universita Karlova, Malostranské nám. 25, 118 00 Praha 1, Czech Republic.

10th Computational Geometry 94-6/94 Stony Brook, NY, USA
© 1994 ACM 0-89791-648-4/94/0006..$3.50

in $\mathbb{R}^d$. Using a probabilistic counting method, Clarkson and Shor [CS89] proved that the maximum complexity of the $\leq k$-level is $\Theta(n^{\lfloor d/2 \rfloor} k^{\lceil d/2 \rceil})$. Following a similar technique, Sharir [Sha91] proved that the maximum complexity of the $\leq k$-level in a set of x-monotone Jordan curves, with at most s pairwise intersections, is $\Theta(k^2 \lambda_s(n/k))$. Here $\lambda_s(m)$ is the maximum length of an (m, s)-Davenport-Schinzel sequence; for any constant s, $\lambda_s(m)$ is roughly linear in m [ASS89]. For discs he proved a bound of $\Theta(nk)$ on the complexity of the $\leq k$-level.

On the algorithmic side, Mulmuley [Mul91b] gave a randomized incremental algorithm for computing the $\leq k$-level in hyperplane arrangements in any dimension. For $d \geq 4$, the expected running time of his algorithm is $O(n^{\lfloor d/2 \rfloor} k^{\lceil d/2 \rceil})$, which is optimal. For $d = 2, 3$ the expected running time of his algorithm is $O(nk^{\lceil d/2 \rceil} \log(n/k))$, which is suboptimal by a logarithmic factor. Recently, Everett et al. [ERvK93] gave an optimal $O(n \log n + nk)$ expected time randomized algorithm for computing the $\leq k$-level in an arrangement of lines in the plane. Mulmuley's algorithm can be applied to compute the $\leq k$-level of arrangements of x-monotone Jordan curves in the plane, but there does not seem to be an easy way to generalize it for computing the $\leq k$-level in more general ranges like discs. Sharir [Sha91] presented a divide-and-conquer algorithm for computing the $\leq k$-level of rather general ranges in the plane; its worst-case running time is roughly $\log^2 n$ times the maximum size of the $\leq k$-level.

In this paper we show that by modifying Mulmuley's algorithm slightly, we can obtain a simpler and faster randomized algorithm for computing the $\leq k$-level in a hyperplane arrangement whose expected running time is $O(nk^{\lceil d/2 \rceil} + n \log n \min(k^{\lceil d/2 \rceil}, \log^{\lceil d/2 \rceil} n))$, which is optimal for $k \geq \log^2 n$ if $d = 2$, for $k \geq \log^{3/2} n$ if $d = 3$, and for all values of k if $d \geq 4$. Moreover, the modified algorithm works for more general ranges like discs or pseudo-discs as well.

Let us have a brief look at Mulmuley's algorithm [Mul91b] (see also his book [Mul93]). The n hyperplanes are added one by one in random order and during this process the $\leq k$-level of the current arrangement is maintained. For all faces of the current structure a conflict list (containing the not yet inserted hyperplanes that intersect the face) is maintained. When adding a new hyperplane h, the algorithm first updates the arrangement locally at faces that are intersected by h. Then it removes cells that have "fallen off" because they are now on level $k+1$; Mulmuley calls this the peeling step.

We observe that this algorithm maintains too much information. After adding the first $n/2$ hyperplanes we expect the k-level of $\mathcal{A}(H)$ to be located near the $k/2$-level of $\mathcal{A}(R)$, where R is the set of the first $n/2$ hyperplanes; most of the $\leq k$-level of $\mathcal{A}(R)$ lying below this $k/2$-level will be peeled off in later stages. Hence, we only maintain the $\leq \kappa(r)$-level of the first r hyperplanes, where $\kappa(r)$ is approximately $k(r/n)$. However, now we have to be a bit more careful about discarding layers of cells. It turns out that the easiest way to do this is to remove the peeling step altogether, and to replace it by regular clean-up phases. This idea was inspired by the lazy randomized incremental algorithms of de Berg et al. [dBDS94]. At the same time, this makes it possible to use the algorithm in situations where it is not obvious how to access the parts of the current cell complex that must be peeled off. One such situation is for the $\leq k$-level for discs.

As stated earlier, not much is known about the complexity of the exact k-level. There is one special case where the maximum complexity of the k-level of a three-dimensional arrangement is known to be $\Theta(k(n-k))$, namely when all the hyperplanes are tangent to the unit paraboloid. This situation arises when the planes are the images of a set of points in the plane under the transformation that maps the order-k Voronoi diagram of these points to the k-level of the planes. Most known algorithms for computing the k-levels in three-dimensional space actually compute the $\leq k$-level [Mul91b, CE87, BDT93]. Since the complexity of the $\leq k$-level is $\Theta(nk^2)$ in the situation sketched above, the running time of these algorithms is $\Omega(nk^2 + n \log n)$. The randomized incremental algorithm by Aurenhammer and Schwarzkopf [AS92] maintains only the k-level, but it can be shown that any randomized incremental algorithm that maintains the k-level of the intermediate arrangements must take time $\Omega(nk^2)$ as well, since the expected number of structural changes in the k-level is $\Omega(nk^2)$ [AS92]. The only algorithm that approaches the desired $O(k(n-k))$ time bound was presented by Clarkson [Cla87]. His algorithm runs in time $O(n^{1+\varepsilon} k)$, where $\varepsilon > 0$ is an arbitrarily small constant. (The algorithm can probably be improved somewhat by using more recent results on geometric cuttings).

We present a simple randomized incremental algorithm that runs in $O(k(n-k) \log n + n \log^3 n)$ time. We obtain this result by maintaining neither the entire $\leq k$-level nor the exact k-level of the current arrangement, but some suitably chosen region that is guaranteed to contain the k-level of the full set.

2 Computing the $\leq k$-level

Let H be a set of n hyperplanes in d-dimensional space. We denote the arrangement formed by H as $\mathcal{A}(H)$ and the level of a point p with respect to H as $\ell_H(p)$. We define the level of a face of $\mathcal{A}(H)$ as the level of any of its interior points. For a subset $R \subset H$, we denote the *bottom-vertex triangulation* of the $\leq k$-level of $\mathcal{A}(R)$ by $\mathcal{L}_k(R)$. This triangulation is defined as follows. Let $\mathcal{C}$ be a cell of the $\leq k$-level and let v be the bottom vertex of $\mathcal{C}$ (that is, the lexicographically smallest vertex of the cell). If $d = 1$ then $\mathcal{C}$ is a segment and it is already a (1-dimensional) simplex. If $d > 1$, then we recursively triangulate the $(d-1)$-dimensional facets of $\mathcal{C}$ and extend each $(d-1)$-simplex into a d-simplex using vertex v. The bottom-vertex triangulation $\mathcal{L}_k(R)$ of the $\leq k$-level of $\mathcal{A}(R)$ is obtained by triangulating every cell of the $\leq k$-level in this manner. (Unbounded cells of $\mathcal{A}(R)$ require some care in this definition [Cla88].) Observe that the bottom-vertex triangulation is a cell-complex: any facet of a simplex is shared with exactly one other simplex. In this section we give an algorithm to compute $\mathcal{L}_k(H)$ for a given parameter $k < n$.

The algorithm. To compute $\mathcal{L}_k(H)$ we first generate a random permutation $h_1, h_2, \ldots, h_n$ of H. Let $H_r := \{h_1, \ldots, h_r\}$, and let $\kappa(r)$ be an appropriate non-decreasing function with $\kappa(n) \geq k$ to be defined below. The idea of the algorithm is to maintain $\mathcal{L}_{\kappa(r)}(H_r)$ while adding the hyperplanes one by one. However, it turns out to be easier not to discard cells whose level (with respect to H_r) has grown too large after every step. Instead we get rid of these cells at regular *clean-up*-phases, namely after inserting the 2^i-th hyperplane, for $1 \leq i \leq \lfloor \log n \rfloor$, and after inserting the last hyperplane. This type of *lazy randomized incremental algorithm* was introduced recently by de Berg et al. [dBDS94]. Below we give a detailed description of the algorithm.

We maintain a collection of d-simplices that forms a d-dimensional cell complex, defined as follows. Suppose we have added the first r hyperplanes. Then the collection $\mathcal{S}_r$ forms the bottom-vertex triangulation of the cells of $\mathcal{A}(H_r)$ whose level is at most $\kappa(p)$ with respect to H_p, where p is the time when the latest clean-up was performed. Our schedule for performing clean-ups implies that p is the largest power of two that is less than r. With every simplex $\Delta \in \mathcal{S}_r$ we keep a *conflict list* $H(\Delta)$ that contains the hyperplanes in H intersecting the interior of Δ, and with each hyperplane $h \in H \setminus H_r$ we keep a list of simplices of $\mathcal{S}_r$ that h intersects. We also keep an adjacency graph $\mathcal{G}$ on the simplices. Two simplices Δ_1 and Δ_2 are connected in this graph if they share a facet. We also mark an arc of this graph if the shared facet of the two simplices connected by this arc is contained in a hyperplane. The level of points changes if and only if we pass from one simplex to an adjacent one connected by a marked arc.

Now consider the addition of hyperplane h_{r+1}. In a generic step of the algorithm (that is, when $r+1$ is not a power of 2) we must perform two tasks: update our collection of simplices (that means, re-triangulate the cells that are split by h_r) and their adjacency relations, and compute the conflict lists for the new simplices.

Let us start with the first task. Using the conflict lists we identify all the simplices in $\mathcal{S}_r$ that are intersected by h_{r+1}; those are the only ones influenced by the insertion of h_{r+1}. Because we know the adjacency relations among the simplices, we can get the intersected simplices in a number of groups, one for each cell that is intersected. Consider such a group, and let $\mathcal{C}$ be the corresponding cell. All the simplices in the group have to be deleted and replaced by a number of new simplices. Let us first take a look at the situation when $d = 2$—see Figure 2. To deal with the part of $\mathcal{C}$ that lies on the same side of h_{r+1} as the bottom vertex v of $\mathcal{C}$, we draw new diagonals from v to the two intersection points of h_{r+1} with the boundary of $\mathcal{C}$; this creates three new simplices. The part of $\mathcal{C}$ lying on the opposite side of h_{r+1} is simply triangulated from scratch. This way the bottom-vertex triangulations of the two cells that result from the splitting of $\mathcal{C}$ are constructed in time that is linear in the number of new simplices that are created. The adjacency relations among the simplices can easily be determined during the process. In dimensions greater than two things are not very different. Consider a 3-dimensional cell $\mathcal{C}$. We first re-triangulate the 2-dimensional facets that are intersected, in the way we just described. We also triangulate $\mathcal{C} \cap h_{r+1}$. Next we construct the 3-dimensional simplices. Given the triangulation of the facets of the cell, the procedure for this is analogous to the planar procedure. In general, in dimension d we first treat the parts of the $(d-1)$-facets incident to intersected simplices recursively, and then we extend the $(d-1)$-simplices that we get to d-simplices by connecting them to the correct bottom vertex.

Our second task is to compute the conflict lists of the new simplices. Let $\mathcal{C}$ be a cell that has been split by h_{r+1}, and let $\mathcal{C}'$ be one of the two new cells contained in $\mathcal{C}$. Some of the simplices of $\mathcal{C}'$ may already have existed in $\mathcal{C}$; these simplices already have the correct conflict lists. To find the conflict lists for the new simplices we collect all the hyperplanes from old

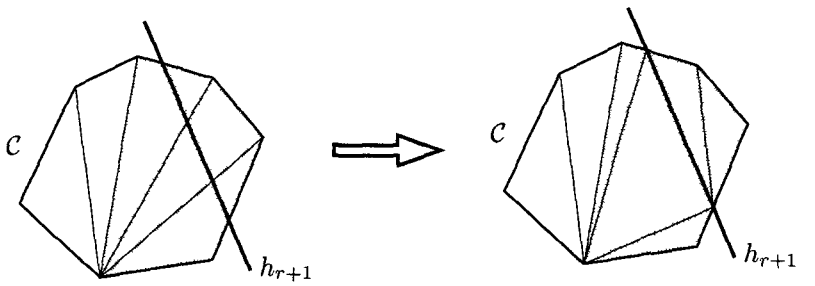

Figure 2: Re-triangulation of a cell that is split.

simplices that intersect C'. We remove the duplicates from our collection, and for each hyperplane we determine one new simplex that it intersects (its *initial simplex*). This can be done in time proportional to the total size of the conflict lists involved. Next, we perform for each hyperplane a traversal of the adjacency graph on the new simplices, starting from its initial simplex. This way we determine all the new simplices intersected by the hyperplane. After having done this for each hyperplane, we have constructed all the new conflict lists. The time we have spent is linear in the size of these lists plus the size of the old lists that disappear.

In a generic step of the algorithm this is all we do. (It would be difficult to discard simplices at every step. The reason for this is the following. Since our "threshold" $\kappa(r)$ depends on r, it is not necessarily the case that all simplices lying below the new hyperplane h_{r+1} can be removed. Therefore we would have to maintain the levels of the simplices, so that we can decide in later stages whether or not to discard them. But maintaining the level of the simplices explicitly is too costly, because it involves simplices not intersected by h_{r+1}.) For $r = 2, 4, \ldots, 2^{\lfloor \log n \rfloor}, n$, however, we perform a clean-up step. We traverse the entire current cell complex using the adjacency graph $\mathcal{G}$. While visiting each simplex we compute its level and discard it if its level is larger than $\kappa(r)$, where r is the current step. There will be a slight chance that by doing so we discard a cell that later turns out to contain a part of the $\leq k$-level of $\mathcal{A}(H)$, but we will argue that the probability that this happens is small. If it happens, we will notice so in the final clean-up phase and we will restart the algorithm with a new random permutation. Since the probability of picking such an unfortunate permutation is small, this procedure changes the expected running time of the algorithm by a constant factor only.

The analysis. Consider the situation after step r of the algorithm. Let p be the largest power of two that is less than r, and define $k^* := \kappa(p)$. From now on we assume that r and, hence, p and k^* are fixed.

We use the abstract framework of de Berg et al. [dBDS94] to analyze our algorithm. Let $\mathcal{F}$ be the set of all possible simplices Δ, that is, all Δ that arise in the bottom-vertex triangulation of $\mathcal{A}(R)$ for some subset $R \subseteq H$. We denote the defining set of a simplex $\Delta \in \mathcal{F}$ by $\mathcal{D}(\Delta)$; this is the inclusion-minimal subset R for which Δ appears in the bottom-vertex triangulation of $\mathcal{A}(R)$. (In degenerate situations, different defining sets can correspond to simplices with the same set of vertices; these simplices are considered to be distinct.) We define the killing set of Δ, denoted $\mathcal{K}(\Delta)$, as the set of hyperplanes in H that intersect the interior of Δ. Furthermore, for a subset $R \subseteq H$ we define $\mathcal{T}(R)$ as the set of simplices in the bottom-vertex triangulation of the full arrangement $\mathcal{A}(R)$, and we define $\mathcal{M}(R)$ as the set of all simplices in $\mathcal{F}$ that have level at most k^* with respect to R. Notice that the simplices in $\mathcal{M}(R)$ do not have to be present in $\mathcal{T}(R)$. Finally, for $R' \subseteq R \subseteq H$ we define $\mathcal{L}(R, R') := \mathcal{T}(R) \cap \mathcal{M}(R')$. The idea is that the set $\mathcal{L}(R, R')$ corresponds to the structure maintained by the lazy algorithm: in our analysis R will be the current subset and R' will be the subset for which the latest clean-up was performed. Observe that $\mathcal{L}(R, R)$ is the same as $\mathcal{L}_{k^*}(R)$, the set of simplices of the bottom-vertex triangulation of the $\leq k^*$-level of $\mathcal{A}(R)$.

Let $b(\Delta) := |\mathcal{D}(\Delta)|$ and $\omega(\Delta) := |\mathcal{K}(\Delta)|$. It is straightforward to verify that the following three conditions hold:

(i) there is a constant $b > 0$ depending on d such that $b(\Delta) \leq b$ for all $\Delta \in \mathcal{F}$,

(ii) a region $\Delta \in \mathcal{F}$ is in $\mathcal{T}(R)$ if and only if $\mathcal{D}(\Delta) \subseteq R$ and $\mathcal{K}(\Delta) \cap R = \emptyset$,

(iii) for $R' \subseteq R \subseteq H$ we have $\mathcal{M}(R) \subseteq \mathcal{M}(R')$.

Let $h_1, \ldots, h_n$ be a random permutation of H, and let $H_r := \{h_1, \ldots, h_r\}$. Define $\mathcal{L}_r := \mathcal{L}(H_r, H_p)$. (Recall that p is the largest power of two that is less than r.) Furthermore, for $1 \leq q < r \leq n$, we define the function

$$\tau(r,q) := \max_{q<t\leq r} E\big[|\mathcal{L}(H_t, H_q)|\big]. \qquad (1)$$

De Berg et al. [dBDS94] proved that under conditions *(i)–(iii)* the following theorem holds.

Theorem 1 *Let $3 \leq r \leq n$ and let p be the largest power of two that is less than r. The expected size of $\mathcal{L}_r$ is at most $\tau(r,p)$. The expected number of simplices in $\mathcal{L}_r \setminus \mathcal{L}_{r-1}$ is bounded by $O(\frac{1}{r}\tau(r,p/2))$. The expected total conflict size of these simplices is bounded as*

$$E\big[\textstyle\sum_{\Delta\in\mathcal{L}_r\setminus\mathcal{L}_{r-1}} \omega(\Delta)\big] \leq O(\frac{n}{r^2}\tau(r,p/2)).$$

To apply this theorem we must bound the function $\tau(r,q)$ for $r/4 \leq q < r$. In other words, we must bound the expected number of simplices in the bottom-vertex triangulation $\mathcal{T}(H_t)$ that have level at most k^* with respect to H_q, for $r/4 \leq q < t \leq r$. It is sufficient to bound the expected number of vertices of $\mathcal{A}(H_t)$ at level at most k^* with respect to H_q; the number of simplices is linear in this number. Observe that H_q is a random sample of H_t. We shall reformulate our problem in an abstract setting so that we can use results on random sampling.

Recall that $\mathcal{L}(H_q, H_q)$ equals $\mathcal{L}_{k^*}(H_q)$, the set of simplices of the bottom-vertex triangulation of the $\leq k^*$-level of $\mathcal{A}(H_q)$. As before, let $\mathcal{F}$ denote the set of all possible simplices, and let $\mathcal{D}(\Delta)$ denote the defining set of a simplex Δ. Again, the following condition holds:

(i') there is a constant $b > 0$ such that $b(\Delta) \leq b$ for all $\Delta \in \mathcal{F}$,

Let $\mathcal{K}(\Delta, H_t)$ be the subset of hyperplanes in H_t that intersect the interior of Δ. It is not difficult to check that the following conditions hold for every subset $R \subseteq H_t$:

(ii') if $\Delta \in \mathcal{L}(R,R)$, then $\mathcal{D}(\Delta) \subseteq R$,

(iii') if $\Delta \in \mathcal{L}(R,R)$, then $\mathcal{K}(\Delta, H_t) \cap R = \emptyset$,

(iv') if $\Delta \in \mathcal{L}(R,R)$ and R' is a subset of R with $\mathcal{D}(\Delta) \subseteq R' \subseteq R$, then $\Delta \in \mathcal{L}(R',R')$.

Define $\omega(\Delta, H_t) := |\mathcal{K}(\Delta, H_t)|$. The number of vertices we have to consider is

$$O\Big(\sum_{\Delta\in\mathcal{L}_{k^*}(H_q)} (1+\omega(\Delta,H_t))^d\Big).$$

To bound the expression above we now use the fact that H_q is a random sample of H_t of size q with $q \geq t/4$. Because conditions *(i')–(iv')* hold we can use a theorem by de Berg et al. [dBDS94] for bounding higher moments. This theorem tells us that

$$E\Big[\sum_{\Delta\in\mathcal{L}_{k^*}(H_q)} \omega(\Delta,H_t)^d\Big] = O\big((t/q)^d|\mathcal{L}_{k^*}(H_q)|\big).$$

The complexity of $\mathcal{L}_{k^*}(H_q)$, the $\leq k^*$-level of $\mathcal{A}(H_q)$, is $O(q^{\lfloor d/2\rfloor}(k^*)^{\lceil d/2\rceil})$. We can conclude that $\tau(r,q) = O(r^{\lfloor d/2\rfloor}(k^*)^{\lceil d/2\rceil})$ for $r/4 \leq q < r$. Combining this result with Theorem 1 gives us the following result. (To simplify the expression we have replaced occurrences of p and $p/2$ by r; this does not influence our bounds asymptotically because $r/2 \leq p < r$.)

Lemma 2 *The expected running time of the lazy randomized algorithm is*

$$O\big(n\sum_{r=1}^{n} r^{\lfloor d/2\rfloor-2}(\kappa(r)+1)^{\lceil d/2\rceil}\big).$$

For dimensions larger than three, we simply choose $\kappa(r) := k$. In this case, a cell that we discard cannot contain any part of the $\leq k$-level later on, so our algorithm always succeeds. We obtain an algorithm for the computation of $\leq k$-levels with expected running time $O(n^{\lfloor d/2\rfloor}k^{\lceil d/2\rceil})$. The algorithm is nearly identical to Mulmuley's algorithm and has the same optimal running time; the only difference is that we replaced his peeling steps by our clean-up steps.

For dimensions two and three, however, this choice of $\kappa(r)$ gives us an expected running time of $O(nk^{\lceil d/2\rceil}\log(n/k))$, which is suboptimal by a factor of $\log(n/k)$. We now show that a slight twist in the algorithm, namely a different choice of $\kappa(r)$, improves its expected running time for sufficiently large values of k.

In the following, the dimension d is 2 or 3. To minimize the running time we would like to choose $\kappa(r)$ as small as possible. As remarked earlier, however, there is a slight possibility that the algorithm does not give us the entire $\leq k$-level, that is, that we have discarded cells that contain a vertex of the $\leq k$-level. We must choose $\kappa(r)$ large enough so that the probability of this happening is small. To bound this probability we need the following lemma, which can be proven using tail estimates for the binomial distribution [AS93].

Lemma 3 *Let v be a vertex in an arrangement $\mathcal{A}(H)$ of n hyperplanes at level $\ell_H(v) \leq k$, and let $r < n$ and $\beta > 1$ be parameters. The probability that for*

a random sample $R \subseteq H$ of r hyperplanes the level $\ell_R(v)$ of v is more than $2\beta k\frac{r}{n}$ is bounded by

$$2[e^{\beta-1}\beta^{-\beta}]^{2k(r/n)}.$$

For $\beta \geq e^2$, this is bounded by

$$2\exp(-2\beta k(r/n)).$$

In the remainder we shall need that the probability that a vertex v at level at most k in $\mathcal{A}(H)$ has a level greater than $\kappa(r)$ in $\mathcal{A}(H_r)$ is bounded by $2/n^4$. From the lemma above it readily follows that this is true if we choose

$$\kappa(r) \;:=\; \max(2e^2k\frac{r}{n}, 4\ln n).$$

Now let us consider the probability that the algorithm fails. Our choice of $\kappa(r)$ implies that the probability that a vertex v is mistakenly discarded in any given clean-up step is at most $2/n^4$. There are $\lfloor \log n \rfloor$ clean-up phases, and there are at most n^d vertices to consider (with $d = 2$ or $d = 3$), so the probability that we ever discard a vertex unjustified is at most $2\log n/n$. In other words, we expect to succeed after we have run the algorithm a constant number of times.

According to Lemma 2 the expected running time of the algorithm with the given choice of $\kappa(r)$ will be

$$\begin{aligned} & O(n)\sum_{r=1}^{n}\frac{(\kappa(r)+1)^{\lceil d/2\rceil}}{r} \\ \leq\;& O(n)\sum_{r=1}^{n}\Big(\frac{k^{\lceil d/2\rceil}r^{\lceil d/2\rceil-1}}{n^{\lceil d/2\rceil}} + \frac{(\ln n)^{\lceil d/2\rceil}}{r}\Big) \\ =\;& O(nk^{\lceil d/2\rceil} + n(\log n)^{\lceil d/2\rceil+1}). \end{aligned}$$

For $k < 4\ln n$, this is no improvement over the trivial choice of $\kappa(r)$. Hence, our final choice for $\kappa(r)$ is

$$\kappa(r) \;:=\; \begin{cases} k, & \text{if } k < 4\ln n, \\ \max(2e^2k\frac{r}{n}, 4\ln n), & \text{otherwise.} \end{cases}$$

This gives us an expected running time of $O(nk^{\lceil d/2\rceil} + n\log n\min(k^{\lceil d/2\rceil}, \log^{\lceil d/2\rceil} n))$, which is optimal for $k \geq \log n$ in the planar case, and for $k \geq \log^{3/2} n$ in the three-dimensional case.

Before we state our theorem on computing the $\leq k$-level we discuss a generalization of the planar algorithm.

Let H be a set of n x-monotone Jordan curves in the plane with each pair of curves intersecting at most s times. To apply our algorithm to arrangements of Jordan curves we need a technique for decomposing such arrangements into constant complexity pieces, or *boxes*. We use the vertical decomposition: we extend a vertical segment upward from every intersection point of two curves until it hits another curve and extend a vertical segment downward until it hits another curve; if a vertical segment does not intersect any curve, it is extended to infinity. The same technique can be applied to the cells in the $\leq\kappa(r)$-level of the arrangement $\mathcal{A}(H_r)$. The number of resulting boxes is $O(\kappa(r)^2\lambda_s(r/\kappa(r)))$. We can now use the lazy randomized incremental algorithm described above to compute the $\leq k$-level in an arrangement of Jordan curves. We only need to change the way the decomposition is updated after an insertion, since we are now using vertical decompositions instead of bottom-vertex triangulations. We omit the details of how the vertical decomposition is updated; they are the same as in randomized incremental algorithms for computing full planar arrangements [Mul91a].

The analysis of the algorithm follows the analysis given earlier, with the maximum complexity of the $\leq k$-level in a set of hyperplanes replaced by the maximum complexity of the $\leq k$-level in a set of Jordan curves. Thus, if p is the largest power of two less than r then the expected running time is

$$O(\sum_{r=1}^{n}\frac{n}{r^2}\tau(r, p/2)),$$

where $\tau(r, p/2) = O(\kappa(r)^2\lambda_s(r/\kappa(r)))$. The same choice of $\kappa(r)$ as before gives us a running time of $O(k^2\lambda_s(n/k) + \min(\lambda_s(n)\log^2 n, k^2\lambda_s(n/k)\log n))$. This bound subsumes the case of line arrangements, since two lines intersect at most once and $\lambda_1(n) = n$.

The same algorithm also works for a set of discs. Using the fact that the complexity of the $\leq k$-level of a set of n discs is $O(nk)$, it can shown that the expected running time of the algorithm, as in the case of lines, is $O(nk + \min(n\log^2 n, nk\log n))$. We omit the easy details from here.

Theorem 4
(i) The $\leq k$-level of an arrangement of n x-monotone Jordan curves in the plane such that any pair intersects in at most s points can be computed in $O(k^2\lambda_s(n/k) + \min(\lambda_s(n)\log^2 n, k^2\lambda_s(n/k)\log n))$ time by a lazy randomized incremental algorithm.
(ii) The $\leq k$-level of an arrangement of n planes in three-dimensional space can be computed in expected time $O(nk^2 + \min(n\log^3 n, nk^2\log n))$ by a lazy randomized incremental algorithm.
(iii) The $\leq k$-level of an arrangement of n discs in the plane can be computed in expected time $O(nk + \min(n\log^2 n, nk\log n))$.

3 Computing order-k Voronoi diagrams

Let P be a set of n points in the plane. The order-k Voronoi diagram of P is the subdivision of the plane into cells such that the k closest points in P are uniquely determined inside each cell. In other words, two points p, q are in the same cell if the k points of P that are closest to p are also the k points of P that are closest to q. P can be mapped to a set $H = H(P)$ of planes in three-dimensional space as follows. Lift each point of P to the unit paraboloid $U : z = x^2 + y^2$, and take the plane that is tangent to U at that point. The order-k Voronoi diagram of P corresponds to the k-level of H—see for example Edelsbrunner's book [Ede87]. The maximum complexity of the order-k Voronoi diagram and, hence, the maximum complexity of the k-level in the situation above, is $\Theta(k(n-k))$.

We give a simple algorithm that computes the k-level for this case in $O(k(n-k)\log n + n\log^3 n)$ time. The algorithm is randomized incremental, like the one in the previous section. This time, however, we do not use the lazy paradigm: we clean up the structure after every step. Another difference with the previous algorithm is the following.

In the algorithm for computing the $\leq k$-level we observed that at time r the $\leq k$-level of the full set is expected to be contained in the $\leq kr/n$-level of the sample. Stated differently, we made an estimation of the real level of a vertex in the arrangement of sample planes based on its level in the sample. We then decided whether or not to discard the vertex based on this estimation. By making our estimations conservatively we ensured that the probability of discarding a "good" vertex was small.

This line of attack turns out not to work for computing the exact k-level. So we maintain a portion of the arrangement which is guaranteed to contain the k-level. In particular, we do not make an estimate of the real level of a vertex based on its level in the sample. Instead we maintain for each vertex in the current arrangement a range of values that is guaranteed to contain its real level. We only discard a vertex when k is not in this range. Thus we are sure that the algorithm produces the right answer. To bound the expected running time of our algorithm, however, we have to use a probabilistic argument.

As before, we denote by H the set of planes and by H_r the first r elements in a random permutation of H. Let Δ be a simplex in the bottom-vertex triangulation of the arrangement $\mathcal{A}(H_r)$. The *conflict list* $H(\Delta)$ of Δ is the set of planes in H that intersect the interior of Δ. Let $\ell(\Delta)$ be the level of any point in the interior of Δ with respect to $H \setminus H(\Delta)$ and let $\omega(\Delta) := |H(\Delta)|$. We call Δ *interesting* if $\ell(\Delta) \leq k \leq \ell(\Delta) + \omega(\Delta)$, and we call a cell of $\mathcal{A}(H_r)$ *active* if its bottom-vertex triangulation has at least one interesting simplex.

Our algorithm maintains the simplices of the bottom-vertex triangulation of all active cells of $\mathcal{A}(H_r)$. Since the k-level of $\mathcal{A}(H)$ can intersect only interesting simplices, we are sure that the k-level is contained in the region that we maintain. With every simplex Δ we maintain the conflict list $H(\Delta)$ and the level $\ell(\Delta)$. We also store with every vertex of each active cell its real level, that is, its level with respect to H. This will help us to compute the level $\ell(\Delta)$ when we create a new simplex Δ. Finally, we maintain the adjacency graph on the simplices; there is an arc between two simplices if they share a facet.

To add a new plane h_{r+1} we proceed as follows. First we identify all active cells intersected by h_{r+1}; using the conflict lists this can be done in time linear in the number of simplices intersected by h_{r+1}. Then we re-triangulate these cells and we compute the conflict lists for the new simplices, as described in the previous section. Every new simplex Δ contains at least one old vertex; using the level of this vertex we can compute the levels of the new vertices of Δ by examining the planes in $H(\Delta)$. Hence, we can compute the real levels of all new vertices in time that is linear in the size of the new conflict lists. Next we compute the level $\ell(\Delta)$ of each new simplex Δ; using the level of any of the vertices of Δ this can again be done by examining the planes in $H(\Delta)$. (Note that $\ell(\Delta)$ does not change for the old simplices.) Finally, we check which of the new simplices is interesting, and we discard the cells that have no interesting simplex. These two steps can be done in time linear in the total size of the conflict lists of all deleted simplices.

When the last hyperplane h_n has been added, we have $H(\Delta) = \emptyset$ for every Δ. In other words, $\omega(\Delta) = 0$ and we are left with the simplices of the bottom-vertex triangulation of $\mathcal{A}(H)$ that have level k. The facets of these simplices that bound their simplex from below and are contained in one of the planes in H form the k-level of $\mathcal{A}(H)$.

It remains to analyze the algorithm.

Lemma 5 *The expected number of simplices present at time r is $O(rk\frac{r}{n}\log r + r\log^2 r)$.*

Proof: We use an argument based on the ε-net theory. Haussler and Welzl [HW87] proved that a random sample R of r planes from a set H of n planes has the following property with probability at least

$1 - 1/n^3$: Any line segment s, which do not intersect a plane in R, intersects at most $C\frac{n}{r}\log r$ planes in H, where C is a constant.

The set H_r is a random sample of H of size r. Let us therefore assume for a moment that the above mentioned property holds for H_r, that is, every segment s that lies within a cell of $\mathcal{A}(H_r)$ intersects at most $C\frac{n}{r}\log r$ planes of H. As a consequence, for all simplices Δ present in stage r we have $\omega(\Delta) \leq 3C\frac{n}{r}\log r$. We now bound the total complexity of all active cells (still assuming that the property above holds for H_r). By definition, every active cell $\mathcal{C}$ contains at least one interesting simplex Δ. Let p be an arbitrary point inside Δ. The level of p with respect to H lies between $\ell(\Delta)$ and $\ell(\Delta)+\omega(\Delta)$. Because Δ is interesting and $\omega(\Delta) \leq 3C\frac{n}{r}\log r$ we know that the level of p lies between $k-3C\frac{n}{r}\log r$ and $k+3C\frac{n}{r}\log r$. We now observe that every vertex v of $\mathcal{C}$ can be connected to p with a segment $s = \overline{pv}$ lying inside $\mathcal{C}$. Therefore s crosses no more than $C\frac{n}{r}\log r$ planes in H, and we can conclude that the level of v lies between $k - 4C\frac{n}{r}\log r$ and $k + 4C\frac{n}{r}\log r$. The number of vertices in $\mathcal{A}(H)$ with this level is bounded by

$$\sum_{\ell=k-4C\frac{n}{r}\log r}^{k+4C\frac{n}{r}\log r} O(n\ell) = O(nk\frac{n}{r}\log r + n(\frac{n}{r}\log r)^2).$$

The probability that a given vertex of $\mathcal{A}(H)$ appears in $\mathcal{A}(H_r)$ is $\Theta((\frac{r}{n})^3)$. Hence, the total complexity of all active cells (and thus also the number of simplices present) at time r is $O(rk\frac{r}{n}\log r + r\log^2 r)$.

This derivation was under the assumption that the ε-net property mentioned above holds for H_r. This is true with probability at least $1-1/n^3$. If the property is not true for H_r, we use the trivial bound of $O(r^3)$ on the number of simplices in $\mathcal{A}(H_r)$. The expected number of simplices is therefore bounded by

$$(1 - 1/n^3)\cdot O(rk\tfrac{r}{n}\log r + r\log^2 r) \;+\; (1/n^3)\cdot O(r^3)$$
$$= \; O(rk\tfrac{r}{n}\log r + r\log^2 r). \qquad \square$$

As in the previous section, the expected running time of the algorithm is bounded by the sum of the sizes of the conflict lists of all simplices ever created during the algorithm. Using Theorem 1 again (with the appropriate definitions of $\mathcal{L}$ and τ), we can prove that this quantity is bounded by

$$\sum_{r=1}^{n}\frac{n}{r^2}O(rk\frac{r}{n}\log r + r\log^2 r) = O(nk\log n + n\log^3 n).$$

For $k \leq n/2$ this is equal to $O(k(n-k)\log n + n\log^3 n)$. For $k \geq n/2$, we consider the k-level as the $(n-k)$-level in the arrangement turned upside down, leading to the following theorem.

Theorem 6 *The k-level in an arrangement of n planes in three-dimensional space that are all tangent to the unit paraboloid can be computed in expected time $O(k(n-k)\log n + n\log^3 n)$ with a randomized incremental algorithm, using $O(k(n-k)\log n + n\log^2 n)$ storage.*

Remark. We only used the fact that all planes in the set H are tangent to the unit paraboloid in the analysis of the algorithm; the algorithm itself works for arbitrary sets of planes. Hence, if the complexity of the k-level for arbitrary sets of planes is about the same as in the restricted case of planes tangent to the unit paraboloid, then the running time of our algorithm is also close to optimal in this general case.

Corollary 7 *The order-k Voronoi diagram of n points in the plane can be computed in expected time $O(k(n-k)\log n + n\log^3 n)$ with a randomized incremental algorithm that uses $O(k(n-k)\log n + n\log^2 n)$ storage.*

4 Concluding Remarks

We have presented algorithms concerning levels in arrangements. The first algorithm computes the $\leq k$-level in an arrangement of curves in the plane or planes in three-dimensional space. The second algorithm computes the exact k-level of a set of planes in three-dimensional space that are tangent to the unit paraboloid; the k-level in this situation corresponds to the order-k Voronoi diagram of a set of points in the plane. The algorithms improve and/or simplify the best known algorithms for these problems.

Both algorithms are randomized and incremental. We improve upon previous randomized incremental algorithms for these problems by being more careful in what to maintain. In particular, previous randomized incremental algorithms for these problems maintained a part of the current arrangement based on the level of that part in the current arrangement. The part that we maintain, on the other hand, is based on the level of that part in the full arrangement. This level is, of course, not known exactly during the algorithm. For the computation of the $\leq k$-level we therefore made an approximation of the level in the full arrangement based on the level in the current arrangement. For the computation of the exact k-level we maintained a range of values that is guaranteed to contain the level in the full arrangement.

Our algorithms are suboptimal when k is very small. We leave it as an open problem to develop an algorithm that is optimal for all values of k. Another open problem, even in the plane, is to compute the exact k-level in optimal time.

References

[AS92] F. Aurenhammer and O. Schwarzkopf. A simple on-line randomized incremental algorithm for computing higher order Voronoi diagrams. *Internat. J. Comput. Geom. Appl.*, 2:363–381, 1992.

[AS93] N. Alon and J. Spencer. *The Probabilistic Method.* J. Wiley & Sons, 1993.

[ASS89] P. K. Agarwal, M. Sharir, and P. Shor. Sharp upper and lower bounds on the length of general Davenport-Schinzel sequences. *J. Combin. Theory Ser. A*, 52:228–274, 1989.

[BDT93] J.-D. Boissonnat, O. Devillers, and M. Teillaud. An semidynamic construction of higher-order Voronoi diagrams and its randomized analysis. *Algorithmica*, 9:329–356, 1993.

[CE87] B. Chazelle and H. Edelsbrunner. An improved algorithm for constructing kth-order Voronoi diagrams. *IEEE Trans. Comput.*, C-36:1349–1354, 1987.

[Cla87] K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete Comput. Geom.*, 2:195–222, 1987.

[Cla88] K. L. Clarkson. A randomized algorithm for closest-point queries. *SIAM J. Comput.*, 17:830–847, 1988.

[CS89] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.

[CSY87] R. Cole, M. Sharir, and C. K. Yap. On k-hulls and related problems. *SIAM J. Comput.*, 16:61–77, 1987.

[dBDS94] M. de Berg, K. Dobrindt, and O. Schwarzkopf. On lazy randomized incremental construction. In *Proc. 25th Annu. ACM Sympos. Theory Comput.*, 1994.

[Ede87] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science.* Springer-Verlag, Heidelberg, West Germany, 1987.

[ERvK93] H. Everett, J.-M. Robert, and M. van Kreveld. An optimal algorithm for the ($\leq k$)-levels, with applications to separation and transversal problems. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 38–46, 1993.

[HW87] D. Haussler and E. Welzl. Epsilon-nets and simplex range queries. *Discrete Comput. Geom.*, 2:127–151, 1987.

[Mul91a] K. Mulmuley. A fast planar partition algorithm, II. *J. ACM*, 38:74–103, 1991.

[Mul91b] K. Mulmuley. On levels in arrangements and Voronoi diagrams. *Discrete Comput. Geom.*, 6:307–338, 1991.

[Mul93] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms.* Prentice Hall, New York, 1993.

[PSS92] J. Pach, W. Steiger, and E. Szemerédi. An upper bound on the number of planar k-sets. *Discrete Comput. Geom.*, 7:109–123, 1992.

[Sha91] M. Sharir. On k-sets in arrangements of curves and surfaces. *Discrete Comput. Geom.*, 6:593–613, 1991.