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 simple randomized incremental algorithms for computing the $\leq k$-level in an arrangement of n hyperplanes in two- and three-dimensional space. The expected running time of our algorithms is $O(nk + n\alpha(n)\log n)$ for the planar case, and $O(nk^2 + n\log^3 n)$ for the three-dimensional case. Both bounds are optimal unless k is very small. The algorithm generalizes to computing the $\leq k$-level in an arrangement of discs or x-monotone Jordan curves in the plane. Our approach can also be used to compute the k-level; this yields a randomized algorithm for computing the order-k Voronoi diagram of n points in the plane 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 $(d-1)$-cells of $\mathcal{A}(H)$ whose interior points have level k with respect to H. It is a monotone piecewise-linear surface; see Figure 1. The $\leq k$-level of $\mathcal{A}(H)$ is the complex induced by all cells of $\mathcal{A}(H)$ lying on or above the k-level. 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, which is useful in certain applications. 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; k-level and $\leq k$-level are defined as above. For a set H of hyperplanes, if we choose ranges to be the half-spaces lying below the hyperplanes of H, then the level of

*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, by Czech Republic Grant GAČR 201/93/2167 and by EC Cooperative Action IC-1000 (project ALTEC: *Algorithms for Future Technologies*). 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.

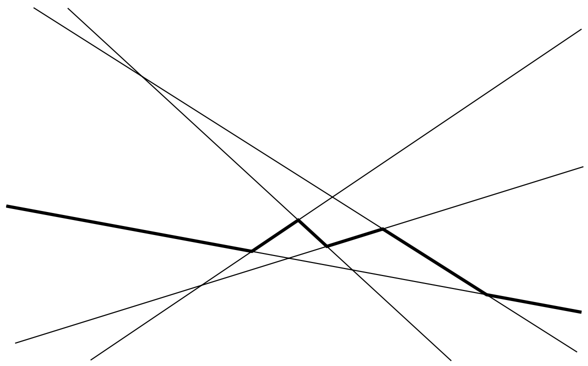

Figure 1: The 2-level in an arrangement of lines.

p is same under the two definitions. We will not distinguish between the two definitions of levels, as it will be clear from the context which of them we are referring to.

The maximum combinatorial complexity of the $\leq k$-level in an arrangement of hyperplanes in $\mathbb{R}^d$ is known precisely: using a probabilistic argument Clarkson and Shor [CS89] proved that it is $\Theta(n^{\lfloor d/2 \rfloor} k^{\lceil d/2 \rceil})$. By a similar technique, Sharir [Sha91] proved that the maximum complexity of the $\leq k$-level for a set of n discs is $\Theta(nk)$. He also proved that the complexity of the $\leq k$-level for n x-monotone Jordan curves such that each pair has at most s intersections is $O(k^2 \lambda_s(n/k))$. Here $\lambda_s(m)$ is the maximum length of an (m, s)-Davenport-Schinzel sequence; $\lambda_s(m)$ is roughly linear in m for any constant s [ASS89].

The problem of efficiently computing the $\leq k$-level has not been resolved completely. 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 exceeds the output size 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. Their algorithm does not easily extend to more general ranges.

Mulmuley's algorithm can be applied to compute the $\leq k$-level of arrangements of x-monotone Jordan curves in the plane, but it is not clear how to generalize it for computing the $\leq k$-level for 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 give a somewhat different randomized incremental algorithm for computing the $\leq k$-level whose expected running time is $O(nk + n \log n\, \alpha(n))$ in the plane and $O(nk^2 + n \log^3 n)$ in 3-space, which is worst-case optimal unless k is very small. The main difference of this algorithm compared with Mulmuley's algorithm [Mul91b, Mul93] is as follows. Mulmuley's algorithm inserts the hyperplanes one by one in a random order, and it maintains the $\leq k$-level of $\mathcal{A}(R)$, the arrangement of the already inserted hyperplanes. When adding a new hyperplane h, the algorithm first updates the arrangement locally at cells 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*. Our algorithm maintains a part of $\mathcal{A}(R)$ that is in general smaller than the $\leq k$-level. Namely, when inserting a new hyperplane, the algorithm estimates the level of each of the newly created cells of the current cell complex with respect to H, the full set of hyperplanes. Mulmuley's algorithm, in contrast, looks at the level with respect to R, the set of already inserted hyperplanes. As soon as our estimate shows that a cell lies completely outside the $\leq k$-level, it is discarded. This strategy can also be used in situations where Mulmuley's algorithm has difficulty to access the cells of the current complex that must be peeled off. This is for instance the case when one wants to compute the $\leq k$-level in an arrangement of discs.

Our approach can also be used to compute the k-level in an arrangement. This time a cell of the current cell complex is discarded as soon as it becomes clear that it does not intersect the k-level of $\mathcal{A}(H)$. The complexity of this algorithm depends on combinatorial bounds on the complexity of a k-level, and here the knowledge is less satisfactory than for the $\leq k$-level. In the plane, the best known lower bound is $\Omega(n \log(k+1))$ [ELSS73], whereas the best known upper bound is $O(n\sqrt{k}/\log^*(k+1))$ [PSS92]. Edelsbrunner and Welzl [EW86] gave an algorithm for computing the k-level in the plane, which was later slightly improved by Cole *et al.* [CSY87] to $O(n \log n + m \log^2 k)$ running time, where m stands for the actual complexity of the k-level. Our algorithm yields $O(n\sqrt{k \log n} + n \log^2 n)$ expected time complexity. This compares favorably to the worst case running times of previous algorithms; however, our algorithm is not output-sensitive. In principle, our algorithm for computing the k-level also works in higher dimensions, and its running time can be obtained by plugging the known bounds on the complexity of the k-level in higher dimensions [DE93, ABFK92, ZV92], into the analysis of our algorithm. We, however, do not discuss it in the paper, because our main interest lies in the special three-dimensional case discussed next.

If all the hyperplanes are tangent to the unit paraboloid, the maximum complexity of the k-level of a three-dimensional arrangement is known to be $\Theta(k(n-k))$. 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-level 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 at least $\Omega(nk^2 + n \log n)$. 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], and it 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 show that our algorithm runs in $O(k(n-k)\log n + n \log^3 n)$ expected time in this case. (We conjecture that the running time is in fact $O(k(n-k) + n \log n)$, which is asymptotically optimal, but currently we cannot prove it.)

2 Preliminaries

Arrangements. Let H be a set of n hyperplanes in d-dimensional space. We denote the arrangement of H by $\mathcal{A}(H)$. We regard it as a cell complex with cells (also called faces) of dimensions 0 to d that are relatively open. We define the *level* of a cell of $\mathcal{A}(H)$ as the level of any of its interior points.

For simplicity of exposition, we assume that the hyperplanes are in general position. This assumption can be removed by a more careful (and technically a little more complicated) treatment, or by standard perturbation arguments [Ede87].

Canonical triangulations. Let $\mathcal{C}$ be a subcomplex of $\mathcal{A}(H)$. The *canonical triangulation*[1] of $\mathcal{C}$ ([Cla88]), which we denote by $\mathcal{C}^\nabla$, is defined as follows. Let C be a j-dimensional cell of $\mathcal{C}$ (thus, a convex polytope), and let v be the bottom vertex of C, that is, the lexicographically smallest vertex of C. If $j = 1$ then C is a segment and it is already a (1-dimensional) simplex. If $j > 1$, then we recursively triangulate the $(j-1)$-dimensional faces of C and extend each $(j-1)$-simplex to a j-simplex using the vertex v. (Unbounded cells require some care in this definition [Cla88]). The canonical triangulation of a cell with m vertices has $O(m)$ simplices. The canonical triangulation $\mathcal{C}^\nabla$ of the subcomplex $\mathcal{C}$ is the simplicial complex obtained by triangulating each cell of $\mathcal{C}$ in this manner.

Let R be a subset of H. For a (relatively open) simplex $\Delta \in \mathcal{A}(R)^\nabla$, let $K(\Delta)$ denote the set of hyperplanes of H intersecting Δ, and put $w(\Delta) = |K(\Delta)| + 1$. The hyperplanes from $K(\Delta)$ are said to be *in conflict with* Δ, and $w(\Delta)$ is called the *weight* of Δ.

For each simplex Δ of the canonical triangulation there is a set $D(\Delta)$ of hyperplanes, called the *defining set* of Δ, whose size is bounded by a constant $b = b(d)$ and that defines Δ in a suitable geometric sense. For instance, for $d = 2$ the defining lines of a typical triangle Δ are the two lines whose intersection is the bottom vertex, the line containing the side opposite to the bottom vertex, and the two more lines intersecting this side at the other two vertices of Δ; so for $d = 2$ we have $b(2) = 5$. The following fact captures the property of the defining set that is often used in the analysis of randomized algorithms.

Fact 1 [CF90] *For every simplex Δ appearing in $\mathcal{A}(S)^\nabla$ for some $S \subseteq H$, the following condition holds:*

For any $R \subseteq H$, $\Delta \in \mathcal{A}(R)^\nabla$ if and only if $D(\Delta) \subseteq R$ and $R \cap K(\Delta) = \emptyset$.

When generalizing the algorithm below to computing the $\leq k$-level in more general arrangements (for example, in an arrangement of discs), we need to define an appropriate analog of the canonical triangulation having the above property.

A tool for analyzing randomized incremental algorithms. We now discuss a random sampling result that we need for the analysis of our algorithms. We do not state it in a full generality, only in the specific setting in which we need it.

Lemma 2 *For each $R \subseteq H$, let $\mathcal{C}(R)$ be a subcomplex of the arrangement $\mathcal{A}(R)$. Assume that the subcomplexes have the following "monotonicity" property:*

($*$) *Let $R' \subseteq R \subseteq H$, let C be a cell of $\mathcal{C}(R)$, and let C' be the cell of $\mathcal{A}(R')$ containing C. Then $C' \in \mathcal{C}(R')$.*

Let $f : \mathbb{N} \to \mathbb{R}^+$ be a nondecreasing function such that $f(r)$ is an upper bound for the expected total number of vertices of $\mathcal{C}(R)$, where R is a randomly chosen r-element subset of H. Then we have

[1] Sometimes the name *bottom vertex triangulation* is used in the literature.

(i) *Let R be a randomly chosen r-element subset of H, and let c be a constant. Then*

$$\mathbf{E}\left[\sum_{\Delta \in \mathcal{C}(R)^{\nabla}} w(\Delta)^c\right] = O\left(\left(\frac{n}{r}\right)^c f(r)\right).$$

(ii) *Consider a randomized incremental algorithm that inserts the hyperplanes of H one by one in a random order and maintains $\mathcal{C}(R)^{\nabla}$, where $R = \{h_1, h_2, \ldots, h_r\}$ is the set of already inserted hyperplanes. Let $\mathcal{N}_r$ denote the set of simplices newly created at step r, that is, $\mathcal{N}_r = \mathcal{C}(\{h_1, h_2, \ldots, h_r\})^{\nabla} \setminus \mathcal{C}(\{h_1, h_2, \ldots, h_{r-1}\})^{\nabla}$. Then for any constant $c \geq 0$ we have*

$$\mathbf{E}\left[\sum_{\Delta \in \mathcal{N}_r} w(\Delta)^c\right] = O\left(\frac{n^c}{r^{c+1}} f(r)\right).$$

Part (i) says that under the condition $(*)$, the expected average weight $w(\Delta)$ of a simplex in $\mathcal{C}(R)^{\nabla}$ is about n/r, even if we take the cth degree averages. Often the amount of work that a randomized algorithm performs at step r is directly related to the weight of the newly created simplices; part (ii) can then be used to analyze the running time.

Proof sketch. Results of this type were first obtained by Chazelle et al. [CEG+93], who essentially proved (ii) with $c = 1$ (in a more general setting) by analyzing a randomized incremental algorithm. The general case, where $c > 1$, can be obtained by an extension of their analysis, as is shown by de Berg et al. [dBDS94]. A somewhat different approach, going via the "static" part (i) and generalizing an approach of Chazelle and Friedman [CF90], is presented by Agarwal et al. [AMS94]. In order to apply the general frameworks in these papers to our situation, we need the following properties of the canonical triangulation of $\mathcal{C}(R)$:

(a) If a simplex Δ is present in $\mathcal{C}(R)^{\nabla}$, then $D(\Delta) \subseteq R$ and $K(\Delta) \cap R = \emptyset$.

(b) For any $\Delta \in \mathcal{C}(R)^{\nabla}$, we have $\Delta \in \mathcal{C}(R')^{\nabla}$ whenever $R' \subseteq R$ and $D(\Delta) \subseteq R'$.

Condition (a) is immediate from Fact 1, and condition (b) follows from Fact 1 together with property $(*)$. Under these conditions, (i) is proved by Agarwal et al. Part (ii) can now be derived from (i) by *backward analysis*: The simplices of $\mathcal{N}_r$ are those destroyed by deleting the hyperplane h_r from $R = \{h_1, h_2, \ldots, h_r\}$. Since the order of hyperplanes is random, h_r is a random hyperplane of R. Each simplex of $\mathcal{C}(R)^{\nabla}$ is only destroyed by deleting h_r if $h_r \in D(\Delta)$. Therefore, the expected sum $\sum_{\Delta} w(\Delta)^c$ over the simplices destroyed by deleting a random hyperplane of R is at most b/r times the expectation of $\sum_{\Delta \in \mathcal{C}(R)^{\nabla}} w(\Delta)^c$, and since R is a random r-element subset of H, we can use (i). □

3 Computing the $\leq k$-level and the k-level for hyperplanes

Outline of the algorithm. We first generate a random permutation $h_1, h_2, \ldots, h_n$ of H, and then insert the hyperplanes one by one in this order. Let R denote the set $\{h_1, \ldots, h_r\}$ of hyperplanes inserted in the first r steps. As the hyperplanes are inserted, the algorithm maintains the following structures:

- The canonical triangulation $\mathcal{K}_r^{\nabla}$ of a subcomplex $\mathcal{K}_r$ of $\mathcal{A}(R)$ (more precisely, we store the simplices as well as their adjacency relations).

- The conflict lists: For every simplex $\Delta \in \mathcal{K}_r^\nabla$, the list $K(\Delta)$ of hyperplanes of $H \setminus R$ intersecting its relative interior, and for every hyperplane $h \in H \setminus R$, the list of all simplices with $h \in K(\Delta)$.
- For every simplex $\Delta \in \mathcal{K}_r^\nabla$, the level ℓ_Δ of one of its (arbitrarily chosen) interior points with respect to $\mathcal{A}(H)$.

At a high level, the insertion of a new hyperplane h_r can be described as follows.

1. Find all cells of $\mathcal{K}_{r-1}$ intersected by h_r using the conflict lists. Split such cells, retriangulate them as necessary, update the conflict lists, and compute the level information for the new simplices.
2. Test each of the newly created cells whether it is *active* (the meaning of this will be described below). If a new cell is not active, its simplices and their conflict lists are discarded. The resulting complex is $\mathcal{K}_r$.

We now discuss these steps in more detail.

Updating the information. In the first step, we identify all the simplices in $\mathcal{K}_r^\nabla$ intersected by h_r using the conflict lists. Since we know the adjacency relations among the simplices, we can get the intersected simplices in a number of groups, one for each cell that h_r intersects. Consider such a group, and let C be the corresponding cell. All the simplices in the group have to be deleted and replaced by a number of new simplices. We first consider the situation for $d = 2$, which is illustrated in Figure 2. To deal with the part of C that lies

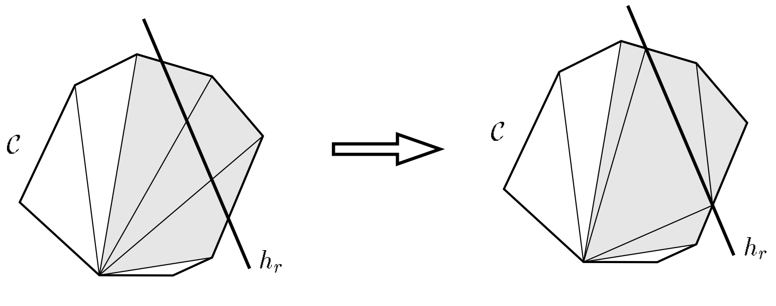

Figure 2: Re-triangulation of a cell that is split.

on the same side of h_r as the bottom vertex v of C, we draw new diagonals from v to the two intersection points of h_r with the boundary of C; this creates three new simplices. The part of C lying on the opposite side of h_r is simply triangulated from scratch. This way the canonical triangulations of the two cells that result from the splitting of 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.

Things are not very different in higher dimensions. Consider a 3-dimensional cell C. We first re-triangulate the 2-dimensional facets that are intersected, in the way we just described. We also triangulate the facet $C \cap h_r$. It remains to connect the triangles of the boundary triangulation to the correct bottom vertex (this is necessary only if they were not already connected to this vertex in the triangulation of C, of course); this is analogous to the two-dimensional case. In general, in dimension d we first treat the parts of the $(d-1)$-facets incident to intersected simplices recursively, and then we connect the $(d-1)$-simplices that still need to be extended to d-simplices to the correct bottom vertex.

Our second task is to compute the conflict lists of the new simplices. Let C be a cell that has been split by h_r into two new cells C' and C'', where C' is the cell that contains the bottom vertex of C. Let S be the set of new simplices in C' and C''. Some of the simplices of C' may already have existed in C; these simplices already have the correct conflict list and are not present in S. The union of the conflict lists for the simplices in S is the same as the union of the conflict lists of the simplices of C that were destroyed by h_r (minus h_r itself). We denote this set of hyperplanes by $K(S)$. To find the conflict lists for the simplices in S, we first determine for each hyperplane $h \in K(S)$ one simplex of S that it intersects — its *initial simplex* — as follows.

Consider a hyperplane $h \in K(S)$. If h intersects some simplex of S in an edge that is also an edge of the old cell C, then this simplex can serve as initial simplex for h. We can find it through the conflict list of a destroyed simplex of C that contained that edge. If h intersects none of the simplices of S in an edge of C, then h must separate the bottom vertex of C' from $C \cap h_r$. Hence, any simplex of C' that has a facet on $C \cap h_r$ can serve as an initial simplex for h. We conclude that we can find initial simplices for all hyperplanes in $K(S)$ in time linear in the total size of the old conflict lists. (We may find more than one initial simplex for a hyperplane, but this is no problem.)

Once we have an initial simplex for each hyperplane $h \in K(S)$, we traverse the adjacency graph of the simplices to find the other simplices in S that are intersected by h. Since the subgraph of the adjacency graph induced by these simplices is connected, the total time spent in traversals, over all hyperplanes of $K(S)$, is linear in the total size of the new conflict lists.

The last thing we have to do in step 1 is to determine for each new simplex the level of one of its interior points with respect to $\mathcal{A}(H)$. To this end, we consider a simplex of C that was destroyed by h_r. Let p be the point interior to the destroyed simplex that defined its level. Let Δ_p be the new simplex that contains p. Trivially, we now know the level of a point interior to Δ_p. To compute the level of an interior point for each of the other new simplices, we again traverse the adjacency graph. The traversal starts at Δ_p. When we step from one simplex to the next, we can update the level of an interior point in time proportional to the size of the conflict lists of the two simplices. Hence, the total time we need is linear in the total size of the conflict lists of the new simplices.

Testing active cells. Recall that ℓ_Δ denotes the level of an interior point of the simplex Δ. The level of any other point in Δ must lie in the range $I_\Delta = \ell_\Delta - w(\Delta), \ldots, \ell_\Delta + w(\Delta)$. We call Δ active if I_Δ contains a level that we wish to compute. So if we are computing the $\leq k$-level then Δ is active if $I_\Delta \cap \{0, 1, 2, \ldots, k\} \neq \emptyset$, and if we are computing the k-level then Δ is active if $k \in I_\Delta$. Since we are maintaining a subcomplex of $\mathcal{A}(R)$, we cannot discard individual simplices: a cell either has to be kept as a whole, or it has to be discarded as a whole. Of course, we should not discard cells that contain active simplices. Therefore we define a cell to be active if at least one of the simplices of its canonical triangulation is active. Note that after inserting the last hyperplane all conflict lists are empty, and so the remaining cell complex is exactly what we wish to compute. (Well, almost: when we are computing the k-level, the remaining complex consists of the cells of level k, from which the k-level can be reconstructed easily.)

Step 2 of our algorithm tests whether the new cells created by the insertion of h_r are active. To this end, we store with each cell a counter indicating the number of active simplices in its canonical triangulation. Let C be a cell that was split, and let C' be one of the two new cells. To compute the counter for C' we should only spend time on its new simplices and on the simplices from C that were destroyed; we should not spend time on simplices in C' that were already present in C. But this is easy, given the counter for C and the levels of the new and old simplices. We conclude that we can test whether C' is active in time proportional to the number of new simplices in C' plus the number of destroyed simplices from C.

4 The analysis

We have seen that the total work for inserting h_r is proportional to the total size of the conflict lists of the simplices destroyed by h_r and of the newly created simplices. Since the simplices being destroyed must have been created before, the total work is proportional to $\sum_\Delta w(\Delta)$, where the summation is over all simplices created by the algorithm. Observe that it could happen that we create a new cell, spend time to triangulate it and to compute the conflict lists and the levels of all its simplices, and then immediately discard it. In other words, we may spend time on (simplices of) cells that are never active. Our analysis must take this into consideration, of course.

We are going to apply Lemma 2 to estimate the sum $\sum_\Delta w(\Delta)$ over all created simplices. The complex $\mathcal{K}_r$ maintained by the algorithm does not seem to be directly suitable for the analysis. We thus define for every $R \subseteq H$ an auxiliary complex $\bar{\mathcal{K}}(R)$, which can be used in the role of $\mathcal{C}(R)$ in Lemma 2.

For a cell C of $\mathcal{A}(R)$, its diameter $diam(C)$ is defined as the maximum number of hyperplanes of H intersecting a segment fully contained in C. We define an auxiliary set $\bar{\mathcal{K}}_0(R)$ of d-cells consisting of the cells C of $\mathcal{A}(R)$ that intersect the $\leq(k+2(d+1)\,diam(C)+d+1)$-level of $\mathcal{A}(H)$ (for the algorithm for computing the $\leq k$-level), or that intersect the region between the levels $k-2(d+1)\,diam(C)-d-1$ and $k+2(d+1)\,diam(C)+d+1$ (for the algorithm for computing the k-level). The complex $\bar{\mathcal{K}}(R)$ consists of the cells that belong to $\bar{\mathcal{K}}_0(R)$, or that share a facet with a cell of $\bar{\mathcal{K}}_0(R)$.

Lemma 3 *All simplices Δ created by the actual algorithm in the rth step are contained in $\bar{\mathcal{K}}(\{h_1,\ldots,h_r\})^\nabla \setminus \bar{\mathcal{K}}(\{h_1,\ldots,h_{r-1}\})^\nabla$; thus, a fictitious algorithm that maintains $\bar{\mathcal{K}}(R)^\nabla$ will create all simplices created by the actual algorithm.*

Proof: We prove the lemma for the algorithm that computes the k-level; the proof for the computation of the $\leq k$-level is analogous.

All cells created by the actual algorithm arise by splitting an active cell C. Let C' and C'' denote the two cells into which C is split. One of these two cells, say C', has diameter at least $(diam(C)-1)/2$. We shall prove that C' is in $\bar{\mathcal{K}}_0(\{h_1,\ldots,h_r\})$, from which the lemma follows.

Let Δ be an active simplex of C, and let p be the point in the interior of Δ defining its level ℓ_Δ. Since $w(\Delta) \leq d \cdot diam(C)$ and Δ is active, we know that

$$\ell_\Delta - d \cdot diam(C) \leq k \leq \ell_\Delta + d \cdot diam(C).$$

Let q be an arbitrary point in C'. The level of q differs from the level of p by at most $diam(C)$. Hence,

$$\text{level of } q - (d+1)\,diam(C) \le k \le \text{level of } q + (d+1)\,diam(C),$$

and since $diam(C') \ge (diam(C) - 1)/2$ this proves the lemma. $\square$

It is easy to check that the complex $\bar{\mathcal{K}}(R)$ satisfies the monotonicity condition $(*)$ in Lemma 2, so we may apply Lemma 2(ii) for $c = 1$ to $\bar{\mathcal{K}}(R)$. By Lemma 3 we can now bound the expected running time of our algorithm as follows.

Corollary 4 *Let f be a nondecreasing function such that $\mathbf{E}\left[|\bar{\mathcal{K}}(R)^{\nabla}|\right] \le f(r)$ for any $r = 1, 2, \ldots, n$, where the expectation is over a random choice of an r-element set $R \subseteq H$. Then the expected running time of the algorithm is*

$$O\left(\sum_{r=1}^{n} \frac{n}{r^2} f(r)\right).$$

$\square$

An ε-net argument. We are going to estimate the function $f(r) = \mathbf{E}\left[|\bar{\mathcal{K}}(R)^{\nabla}|\right]$. General results of Haussler and Welzl [HW87] imply that for a suitable constant $c = c(d)$ a random r-element sample $R \subseteq H$ has the following property with probability at least $1 - 1/r^d$: Any line segment s that does not intersect any hyperplane of R intersects at most $c\frac{n}{r}\log r$ hyperplanes of H. (This is usually expressed by saying that R is an ε-net with respect to segments, with $\varepsilon = c\log r/r$ [HW87].) Let ε-NET(R) be a predicate expressing this property, that is, ε-NET(R) is true if and only if R has this property. We can write, using conditional expectations

$$\begin{aligned}\mathbf{E}\left[|\bar{\mathcal{K}}(R)^{\nabla}|\right] &= \mathbf{E}\left[|\bar{\mathcal{K}}(R)^{\nabla}| \mid \varepsilon\text{-NET}(R)\right] \cdot \Pr[\varepsilon\text{-NET}(R)] \;+ \\ &\quad \mathbf{E}\left[|\bar{\mathcal{K}}(R)^{\nabla}| \mid \mathit{NOT}\ \varepsilon\text{-NET}(R)\right] \cdot \Pr[\mathit{NOT}\ \varepsilon\text{-NET}(R)]\,.\end{aligned}$$

Since $\bar{\mathcal{K}}(R)^{\nabla}$ has never more than $O(r^d)$ simplices, this is at most

$$(1 - 1/r^d) \cdot \mathbf{E}\left[|\bar{\mathcal{K}}(R)^{\nabla}| \mid \varepsilon\text{-NET}(R)\right] \;+\; O(1)\,. \tag{1}$$

Consider a cell C of $\bar{\mathcal{K}}(R)^{\nabla}$ and consider the algorithm for computing the $\le k$-level. By definition, C intersects the $\le(k + 2(d+1)\,diam(C) + d + 1)$-level of $\mathcal{A}(H)$. Hence, C is completely contained in the $\le(k + (2d+3)\,diam(C) + 2d + 1)$-level of $\mathcal{A}(H)$. Now set $\delta = (2d+3)c(n/r)\log r + 2d + 1$. With R having the ε-net property, any cell in $\mathcal{A}(R)$ has diameter at most $c(n/r)\log r$, and so all cells of $\bar{\mathcal{K}}(R)$ are contained in the $\le(k+\delta)$-level of $\mathcal{A}(H)$. Similarly, in the case of computing the k-level, all cells of $\bar{\mathcal{K}}(R)$ are contained in the region between levels $k - \delta$ and $k + \delta$. We denote the region that contains the cells of $\bar{\mathcal{K}}(R)$ in either algorithm by $T = T(r)$. Thus the quantity $|\bar{\mathcal{K}}(R)^{\nabla}|$ that we want to bound is at most proportional to the number of vertices of $\mathcal{A}(R)$ lying in the region T.

If R is a random r-element subset of H, then for any fixed vertex v of $\mathcal{A}(H)$ the probability of it being a vertex of $\mathcal{A}(R)$ is at most $(r/n)^d$. If R is conditioned to satisfy ε-NET(R), this probability can increase at most by the factor of $(1 - r^{-d})^{-1}$. Hence, the expected number of vertices of $\mathcal{A}(R)$ in T is at most $(1 - r^{-d})^{-1}(r/n)^d N_T$, where N_T stands for the number of vertices of $\mathcal{A}(H)$ in T. From (1) we get $f(r) = O(1 + N_T r^d/n^d)$. By Corollary 4, we can now bound the expected running time of the algorithm by

$$O(n) + \frac{O(1)}{n^{d-1}} \sum_{r=1}^{n} r^{d-2} N_T\,. \tag{2}$$

(This holds provided that f comes out as a nondecreasing function, which is the case in all our applications).

Specific results. We can now bound the expected running time of our algorithm in various situations by substituting appropriate bounds on N_T.

First we look at the computation of the $\leq k$-level in 3-dimensional space. Here N_T is bounded by the number of vertices of the $\leq(k+\delta)$-level, so $N_T = O(n(k+\delta)^2)$.

For the computation of the k-level, N_T is the total complexity of the levels $k-\delta$ to $k+\delta$. In the 3-dimensional case we are mainly interested in sets of planes that are the image of a set of points in the plane under the transformation that maps the order-k Voronoi diagram of the points to the k-level of the planes. In this case all planes are tangent to the unit paraboloid and $N_T = O(n(k+\delta)\delta)$. For the k-level of a set of lines in the plane, we can use an estimate due to Welzl [Wel86], saying that for any arrangement of n lines in the plane and any set $K \subseteq \{1, 2, \ldots, n\}$, the total complexity of all k-levels with $k \in K$ is $O\left(n\sqrt{\sum_{k\in K} k}\right)$. In our situation $K = \{k-\delta, \ldots, k+\delta\}$, so we get $N_T = O\left(n\sqrt{(k+\delta)\delta}\right)$.

We summarize the results of these calculations in the following theorem.

Theorem 5

(i) *The $\leq k$-level in an arrangement of n planes in $\mathbb{R}^3$ can be computed in $O(nk^2 + n\log^3 n)$ expected time.*

(ii) *The kth order Voronoi diagram for n points in the plane can be computed in $O((n-k)k\log n + n\log^3 n)$ expected time, by computing the k-level in the corresponding arrangement of n planes in $\mathbb{R}^3$.*

(iii) *The k-level in an arrangement of n lines in the plane can be computed in $O(n\sqrt{k\log n} + n\log^2 n)$ expected time.*

Our algorithm also works in other situtations, such as the computation of the $\leq k$-level of a set of lines, discs, or monotone curves in the plane. In these situations, however, there is an algorithm that obtains slightly better bounds. The details of this are described in the next section.

5 Improvements and extensions

The algorithm of the previous section is suboptimal when k is very small (polylogarithmic in n); in this case Mulmuley's algorithm is better. We can illustrate this on the algorithm for computing the $\leq k$-level in the plane, when k is a constant. The analysis of our algorithm accounts for the maintenance of the portion of $\mathcal{A}(R)$ roughly within the $\leq\delta$-level, where $\delta = O((n/r)\log r)$. According to this analysis we maintain a region of complexity $O(r\log r)$. Mulmuley's algorithm maintains the $\leq k$-level in the arrangement of the already inserted lines, which has only $O(r)$ complexity.

One cheap improvement is, of course, to run Mulmuley's algorithm in parallel with ours and look which one finishes first. This eliminates the potential advantage of our algorithm—namely that no peeling step is necessary—so it cannot be directly applied to discs, say. We indicate another route to an improvement; currently, however, it only works for the planar case (which is not so interesting for lines, since an optimal algorithm has been known there). We first explain the approach for the construction of the $\leq k$-level in an arrangement of lines in the plane. We then show that the improved algorithm also works for discs and curves.

Lazy clean-up and refined activity test. To improve the running time of the algorithm we should maintain fewer cells. In other words, the cells we maintain should be closer to the $\leq k$-level. To achieve this we will insist that each cell in the current complex $\mathcal{K}_r$ intersects the $\leq (k+2n/r)$-level (while the previous analysis used the fact that with high enough probability, each cell intersects the $\leq (k + O((n/r)\log r))$-level). There are two issues to be addressed. First, the new requirement is time-dependent: a cell that was acceptable at some step r may have to be eliminated at some later step r', though it has not been split. Second, we have to refine the test of activity for a simplex: there can be simplices intersected by many more than n/r lines at step r, so we cannot use just one interior point to estimate the level of all points in such a simplex accurately enough.

To deal with the first issue, we use the so-called *lazy* strategy [dBDS94]. That is, we do not worry about cells that are not split but should be de-activated because r has increased. Thus, when we insert a new hyperplanes we only perform an activity test for the newly created cells. Of course, cells that should be eliminated cannot be kept around for too long. We get rid of these cells at periodic clean-up steps. In particular, we do a clean-up after steps $1, 2, 4, 8$, and so on. Since we do only a logarithmic number of clean-ups, we can afford to do them in a brute-force manner: we simply traverse the entire current complex and eliminate all cells that do not contain a point of the $\leq (k + n/r)$-level. In this way, we make sure that at any step r each cell intersects the $\leq (k + 2n/r)$-level.

As for the second issue, we describe the refined activity test for a cell C at step r. Again, we test each simplex $\Delta \in C^{\nabla}$ separately. We subdivide the simplex into smaller simplices, each intersected by at most n/r lines of $H \setminus R$, and we determine the level of an interior point for each of these small simplices. Now Δ is active if and only if at least one of these points is at level less than or equal to $k + n/r$. This test is again conservative: cells that contain a part of the $\leq k$-level are always active. Subdividing Δ into smaller simplices plus determining their levels can be performed in $O(w(\Delta)^2 r/n)$ expected time, using a randomized algorithm of Chazelle and Friedman [CF90] (see also [Ma91]). At a regular insertion step, this refined activity test is performed only for the newly created cells. At the clean-up steps the test is performed for all the cells. This implies that a cell that is present at step r must have been tested (and found active) at some step r' with $r/2 \leq r' \leq r$. Hence, a cell that is present at time r (that is, after the insertion of the r-th line has been completed) contains a point of level at most $k + 2n/r$. This finishes the description of the modifications to the algorithm.

More general ranges. The modified algorithm (and also the basic algorithm) can be applied to arrangements of ranges other than halfspaces. We only need to supply a suitable notion of a canonical triangulation (a subdivision of the arrangement into constant complexity cells) and implement the steps of the algorithm suitably. As an illustration, we mention two planar cases, namely discs and x-monotone Jordan curves with a bounded number of intersections for every pair of curves.

As a canonical triangulation, we use the *vertical decomposition* of the cells: 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. We call the constant complexity cells arising in this decomposition the *trapezoids*. Updating the vertical decomposition is the same as in randomized incremental algorithms for computing full planar arrangements of curves [Mul91a]. For the refined activity test we need one more ingredient: we must be able to decompose a trapezoid into smaller trapezoids, each intersected by at most n/r curves. An obvious modification of Chazelle and Friedman algorithm [CF90] can compute such a subdivision in $O(w(\Delta)^2 r/n)$ expected time. Now we have all the tools available to implement the algorithm described above (or the basic one of the previous section).

The analysis. To analyze the modified algorithm we again define an auxiliary, possibly larger complex $\widetilde{\mathcal{K}}(R)$: a cell $C \in \mathcal{A}(R)$ belongs to $\widetilde{\mathcal{K}}(R)$ if it has a point of level $\leq (k+2n/r)$ or is adjacent to a cell with such a point, where $r = |R|$. The complex $\widetilde{\mathcal{K}}(R)$ has the monotonicity property $(*)$ needed in Lemma 2. There is also an analog of Lemma 3, namely that all cells created in the rth step of the algorithm are in $\widetilde{\mathcal{K}}(\{h_1, \ldots, h_r\}) \setminus \widetilde{\mathcal{K}}(\{h_1, \ldots, h_{r-1}\})$, and all cells present in the actually maintained collection in the rth step are in $\widetilde{\mathcal{K}}(\{h_1, \ldots, h_r\})$.

The time needed for the insertion steps (not counting the clean-up phases) is bounded by $\sum_{\Delta} w(\Delta)^2 r(\Delta)/n$, where we sum over all simplices created by the algorithm, and where $r(\Delta)$ is the moment of creation of Δ. Using Lemma 2(ii) for $c = 2$ we get the bound of

$$O\left(\sum_{r=1}^{n} \frac{n}{r^2} f(r)\right), \tag{3}$$

where $f(r)$ is a nondecreasing upper bound for $\mathbf{E}\left[|\widetilde{\mathcal{K}}(R)^{\nabla}|\right]$.

The clean-up phases can be accounted for as follows. The simplices that are discarded in a clean-up phase after inserting the rth line passed an activity test at some time $r' \geq r/2$, so the time for the activity test that discards them can be charged to that previous test. Every test gets charged at most once this way. The simplices that pass the activity test at the clean-up phase after step r belong to $\widetilde{\mathcal{K}}(R)^{\nabla}$, $R = \{h_1, \ldots, h_r\}$, and hence the time for these tests at the considered clean-up phase is at most $\sum_{\Delta \in \widetilde{\mathcal{K}}(R)^{\nabla}} (w(\Delta)^2 r/n)$. Applying Lemma 2(i) with $c = 2$ and summing up over all clean-up phases, we get a contribution of $\sum_{i=1}^{\lfloor \log_2 n \rfloor} (n/2^i) f(2^i)$, which will be of the same order as (3).

It remains to provide a good bound on the expected size of $\widetilde{\mathcal{K}}(R)^{\nabla}$. This is at most proportional to the expected number of vertices of $\widetilde{\mathcal{K}}(R)$. We could use an ε-net argument again, arguing that, with high probability, all such vertices are at level at most $k + O((n/r)\log r)$ in the arrangement of H. This, however, gives nothing better than the analysis of the basic algorithm. In the planar case, a refined argument can be given (although the resulting bound is probably still not tight). We start with the case of lines.

First, we pick a number k' in the range $[k + 2n/r, 2(k + 2n/r)]$ such that the k'-level in $\mathcal{A}(H)$ has $O(n)$ complexity (this is possible since the total complexity of all levels in this range is $O(n(k + n/r))$). Then we divide the vertices of $\widetilde{\mathcal{K}}(R)$ into two types: those within the $\leq k'$-level of H, and those outside. The expected number of vertices of the first type is $O((r/n)^2 nk')$. As for the vertices of the second type, they all belong to cells of $\mathcal{A}(R)$

intersecting L, the k'-level of $\mathcal{A}(H)$, or the adjacent cells. The collection of such cells can naturally be called the *1-zone* of L. It follows from the probabilistic argument of Clarkson and Shor [CS89] that the complexity of the 1-zone is of the same order as the complexity of the *zone* of L, that is, of the cells intersecting L, and it remains to bound the expected complexity of this zone. Let m denote the number of intersections of the lines of R with L (a contiguous segment is counted as one intersection). The number of intersections of the lines of H with L is $O(n)$ (each line meeting L does so at a vertex), and hence the expectation of m is $O(r)$. We may now use a standard trick to convert the zone of L to a single cell in an arrangement of $O(m+r)$ (possibly unbounded) segments, namely we remove from each line of R a small portion around its intersections with L. By [PSS87], the complexity of a single cell in an arrangement of n segments is $O(n\alpha(n))$, and so the expected zone complexity in our case is $O(r\alpha(r))$. We may thus conclude that $f(r) = O(r^2k/n + r\alpha(r))$, and the expected running time of the algorithm is $O(nk + n \log n\, \alpha(n))$.

The same analysis can be applied for other ranges, as long as we can bound the expected complexity of the zone of the k'-level. For the case of discs or x-monotone Jordan curves, the zone complexity can be bounded using the same argument, since one has a good bound on the complexity of a single cell in an arrangement of the corresponding (curvilinear) segments: a single cell defined by n segments of curves with at most s intersections per pair has complexity $O(\lambda_{s+2}(n))$ [GSS89]. We obtain the following result.

Theorem 6 *The $\leq k$-level in an arrangement of n discs in the plane can be computed in expected $O(nk + \lambda_4(n) \log n)$ time. The $\leq k$-level in an arrangement of n x-monotone Jordan curves with each pair having most s intersections can be computed in expected $O(k^2\lambda_s(n/k) + \lambda_{s+2}(n) \log n)$ time.*

6 Concluding remarks

We presented randomized incremental algorithms for computing $\leq k$-levels or k-levels in arrangements. The main difference with previous algorithms is that we use estimates of the level in the final arrangement to decide which part of the current arrangement to keep, whereas previous algorithms used the level in the current arrangement. Consequently, our algorithm maintains less information (unless k is very small) and therefore it is faster. For most values of k we have shown that our algorithm is optimal. We think that it is also optimal for small k, but we have not been able to prove this.

This is one of the questions we leave open: is it possible to tighten the analysis of the basic algorithm and show that it is optimal for any k (both for computing the $\leq k$-level and for computing the kth order Voronoi diagram in the plane)? If one cannot do this for the basic algorithm, then maybe it is possible for the modified algorithm. Here one has a purely combinatorial question: what is the expected complexity of the zone of $L_{k'}$ in the arrangement of a random sample of size r, where $L_{k'}$ is the k'-level in the full arrangement? (Here k' can be chosen suitably, so that the k'-level has no more than the average complexity.) It seems quite likely that this expectation should be $O(r)$ in the plane and $O(r^2k'/n)$ in 3-space.

References

[ABFK92] N. Alon, I. Bárány, Z. Füredi, and D. Kleitman. Point selections and weak ε-nets for convex hulls. *Combin., Probab. Comput.*, 1(3):189–200, 1992.

[AMS94] P. Agarwal, J. Matoušek, and O. Schwarzkopf. Computing many faces in arrangements of lines and segments. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 76–84, 1994.

[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.

[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.

[CEG+93] B. Chazelle, H. Edelsbrunner, L. Guibas, M. Sharir, and J. Snoeyink. Computing a face in an arrangement of line segments. *SIAM J. Comput.*, 22, 1993.

[CF90] B. Chazelle and J. Friedman. A deterministic view of random sampling and its use in geometry. *Combinatorica*, 10(3):229–249, 1990.

[Cha93] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geom.*, 9(2):145–158, 1993.

[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.*, pages 105–114, 1994.

[DE93] T. Dey and H. Edelsbrunner. Counting triangle crossings and halving planes. *Discrete Comput. Geom.*, 12:281–289, 1994.

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

[ELSS73] P. Erdős, L. Lovász, A. Simmons, and E. Straus. Dissection graphs of planar point sets. In J. N. Srivastava, editor, *A Survey of Combinatorial Theory*, pages 139–154. North-Holland, Amsterdam, Netherlands, 1973.

[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.

[EW86] H. Edelsbrunner and E. Welzl. Constructing belts in two-dimensional arrangements with applications. *SIAM J. Comput.*, 15:271–284, 1986.

[GSS89] L. J. Guibas, M. Sharir, and S. Sifrony. On the general motion planning problem with two degrees of freedom. *Discrete Comput. Geom.*, 4:491–521, 1989.

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

[Ma91] J. Matoušek. Cutting hyperplane arrangements. *Discrete Comput. Geom.*, 6:385–406, 1991.

[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.

[PSS87] R. Pollack, M. Sharir, and S. Sifrony. Separating two simple polygons by a sequence of translations. *Discrete Comput. Geom.*, 3:123–136, 1987.

[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.

[Wel86] E. Welzl. More on k-sets of finite sets in the plane. *Discrete Comput. Geom.*, 1:95–100, 1986.

[ZV92] R. Živaljević and S. Vrećica. The colored Tverberg's problem and complexes of injective functions. J. Combin. Theory Ser. A, 61:309–318, 1992.