On Lazy Randomized Incremental Construction

Mark de Berg   Katrin Dobrindt   Otfried Schwarzkopf

UU–CS–1994–12
February 1994

Utrecht University
Department of Computer Science
Padualaan 14, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands,
Tel. : + 31 - 30 - 531454

On Lazy Randomized Incremental Construction

Mark de Berg   Katrin Dobrindt   Otfried Schwarzkopf

Technical Report UU–CS–1994–12
February 1994

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

ISSN: 0924–3275

On Lazy Randomized Incremental Construction*

Mark de Berg Katrin Dobrindt Otfried Schwarzkopf

Abstract

We introduce a new type of randomized incremental algorithms. Contrary to standard randomized incremental algorithms, these *lazy* randomized incremental algorithms are suited for computing structures that have a 'non-local' definition. In order to analyze these algorithms we generalize some results on random sampling to such situations.

We apply our techniques to obtain efficient algorithms for the computation of single cells in arrangements of segments in the plane, single cells in arrangements of triangles in space, and zones in arrangements of hyperplanes. We also prove combinatorial bounds on the complexity of what we call the ($\leqslant k$)-cell in arrangements of segments in the plane or triangles in space; this is the set of all points on the segments (triangles) that can reach the origin with a path that crosses at most $k-1$ segments (triangles).

1 Introduction

Since randomization first appeared in the computational geometry literature in the late eighties, it has had great impact on the field. Currently the best known algorithms for many problems are randomized. Even when there are deterministic algorithms with the same or a better asymptotic running time, the randomized algorithms are so much simpler and easier to implement that they seem to be the method of choice in practice.

A popular type of randomized algorithms are *randomized incremental algorithms*. Here a geometric structure induced by a set S of certain geometric objects is computed by adding the objects in S in random order, meanwhile maintaining the structure induced by the current subset. Examples are algorithms for computing arrangements of line segments [CS89, Mul88], convex hulls [Sei91, CMS93], Voronoi diagrams [GKS92, BT93, MMÓ91], and the union of fat triangles or pseudo-discs [MS91].

Most of these algorithms are based on the theory by Mulmuley [Mul88] and by Clarkson and Shor [CS89]. They have analyzed random sampling in an abstract setting that applies to a number of geometric problems. Although the settings by Mulmuley and by Clarkson and Shor are formulated in a different way, they both rely on the same basic framework. They consider a set of geometric objects (a set of line segments, say), and define some geometric structure on it (the planar arrangement defined by these segments). This structure is built up from simple elements of constant complexity called *regions* (trapezoids of the trapezoidal map defined by the segments). To analyze the behavior of the structure for a randomly drawn subset of the objects, they define for every region a set of objects

*This research was supported by the Netherlands' Organization for Scientific Research (NWO) and partially by the ESPRIT III Basic Research Action 6546 (PROMotion). The research by K.D. was done while she was working at INRIA, Sophia-Antipolis, France.

defining the region and a set of objects *in conflict with* the region. The assumption they need for their analysis is that a region is present in a geometric structure exactly if all of the defining objects are present, but none of the conflicting objects are. This assumption makes it possible to exactly specify the probability that a certain region appears in the structure induced by a random subset of the given set of objects; it lies at the basis of Mulmuley's as well as Clarkson and Shor's analysis.

Unfortunately, there are geometric structures that do not fit into the framework of Mulmuley and of Clarkson and Shor. One of the most important such structures is a single face in an arrangement of segments in the plane or, in general, a single cell in an arrangement of $(d-1)$-dimensional simplices in d-space. Single cells do not fit into their frameworks because the question whether a trapezoid defined by some segments in the plane is part of the single cell cannot be answered locally: it depends on the complete configuration of the segments. Therefore it is impossible to properly define a set of conflicting objects.

If we want to develop a randomized incremental algorithm for computing a structure with non-local definition—a single cell, for example—then we face two problems. First of all, the analysis of Mulmuley, and of Clarkson and Shor does not apply. Second, because it cannot be decided locally whether a region is part of the structure, it is difficult to maintain the structure efficiently.

Chazelle et al. [CEG$^+$91] overcame these problems for the computation of a single face in an arrangement of line segments in the plane. They have extended part of Clarkson and Shor's analysis to that case (in fact, our more general analysis borrows several ideas from theirs). They also showed how to find the parts of the single cell that are cut off after the insertion of a new segment, using some auxiliary data structures. (See Section 4 for more details.) This, however, seems to be very difficult in higher dimensions.

This was the motivation for our research: we wanted to devise a randomized incremental algorithm for the computation of a single cell in an arrangement of triangles in 3-space. Because we could not find a way to maintain the single cell efficiently after each insertion, we took the following lazy approach, which appears to be new. When inserting a triangle we do not attempt to identify and discard the parts of the cell which are cut off; we simply keep everything around, including the parts that are no longer part of the single cell. This way we would, of course, end up constructing the full arrangement of triangles. Therefore we clean up the structure at certain stages (after inserting the 2^i-th triangle, for $i = 1, \ldots, \log n$, to be precise). To perform this clean up we just visit everything in the structure that we have constructed so far, discarding the parts outside the single cell. To analyze this algorithm we needed a generalization of Clarkson and Shor's results on random sampling to structures with a non-local definition. The result is a new framework and analysis for randomized algorithms, which allows for structures with a 'non-local' definition. Our analysis shows that the bounds obtained by Mulmuley and by Clarkson and Shor remain valid in our more general framework.[1]

Using this analysis we show that the lazy algorithm for computing single cells in 3-space runs in expected time $O(\psi(n) \log n)$, where $\psi(n)$ is the maximum number of boxes in the vertical decomposition of a single cell in an arrangement of n triangles. Recently

[1]We recently learned that Clarkson and Matoušek independently from each other or us proved similar generalisations of Clarkson and Shor's random sampling results.

de Berg et al. [dBGH94] proved that, for any fixed $\varepsilon > 0$, $\psi(n) = O(n^{2+\varepsilon})$. We thus achieve a running time of $O(n^{2+\varepsilon})$. Aronov and Sharir [AS90] give a different algorithm for the single cell problem with the same running time. The best known lower bound on $\psi(n)$, however, is $\Omega(n^2\alpha^2(n))$, and we believe that this bound is closer to the truth than the $O(n^{2+\varepsilon})$ upper bound. If our belief is true, then our algorithm has a better running time than that of Aronov and Sharir. In the two-dimensional case our algorithm for computing a single face runs in time $O(n\alpha(n)\log n)$. (Here and in the rest of the paper $\alpha(n)$ denotes the extremely slowly growing functional inverse of Ackermann's function.) Thus we achieve the same running time as Chazelle et al. [CEG+91], but with a simpler algorithm.

Our technique extends to various other problems. We illustrate this by giving a simple algorithm for computing zones in arrangements of hyperplanes in d-dimensional space. The algorithm computes the zone of a hyperplane in an arrangement of hyperplanes in d-dimensional space in $O(n^{d-1} + n\log n)$ time. It can also compute zones of curves, surfaces, and so on. To our knowledge no efficient algorithms were known for these problems.

Our general random sampling result has a nice combinatorial application. Given a set S of line segments in the plane, define the $(\leqslant k)$-cell of S as the set of points on the segments in S that can reach the origin with a path that crosses at most $k-1$ segments in S. Thus the $(\leqslant 1)$-cell is the boundary of the single cell defined by S containing the origin. The $(\leqslant k)$-cell is also a generalization of the $(\leqslant k)$-level in arrangements of lines. (This is the set of points p on the lines such that there are at most k lines strictly above p.) Using our results on random sampling with non-local definition we prove that the maximum complexity of the $(\leqslant k)$-cell in an arrangement of line segments is $\Theta(nk\alpha(\frac{n}{k}))$. The result extends to 3-space, where we obtain an almost optimal bound on complexity of the $(\leqslant k)$-cell in an arrangement of triangles.

2 Random sampling with non-local definition

We present our analysis of random sampling with non-local definition in an abstract framework, following the spirit of Clarkson and Shor [CS89]. This will permit us to apply it to different situations.

Let S be a set of n objects. (To keep some life in the following presentation, the reader would be well advised to imagine a concrete example. Assume, for instance, that S is a set of n line segments in the plane.) For every subset $R \subseteq S$, define a collection of "regions" $\mathcal{C}(R)$. (The set of line segments R partitions the plane into faces. Consider the face containing the origin. We partition this face into trapezoids by the usual vertical trapezoidation scheme—see Section 3 for the precise definition. The collection of trapezoids thus obtained is $\mathcal{C}(R)$.) Elements of $\mathcal{C}(R)$ are denoted by Δ. Let $\mathcal{F}$ be the set of all possible regions Δ, that is, all Δ that arise in $\mathcal{C}(R)$ for some subset $R \subseteq S$.

To every $\Delta \in \mathcal{F}$ we assign subsets $\mathcal{D}(\Delta)$ and $\mathcal{K}(\Delta)$ of S. We call the set $\mathcal{D}(\Delta)$ the *defining set* of Δ and we call the set $\mathcal{K}(\Delta)$ the *killing set* of Δ. The elements of $\mathcal{K}(\Delta)$ are said to be *in conflict with* Δ. Let $b(\Delta) := |\mathcal{D}(\Delta)|$ and $\omega(\Delta) := |\mathcal{K}(\Delta)|$. First of all, we require the following.

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

We also require that the following conditions hold for all $\Delta \in \mathcal{F}$:

(ii) if $\Delta \in \mathcal{C}(R)$ then $\mathcal{D}(\Delta) \subseteq R$.

(iii) if $\Delta \in \mathcal{C}(R)$ then $\mathcal{K}(\Delta) \cap R = \emptyset$.

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

These conditions are weaker than the conditions of Clarkson and Shor, which require that $\Delta \in \mathcal{C}(R)$ if and only if $\mathcal{D}(\Delta) \in R$ and $\mathcal{K}(\Delta) \cap R = \emptyset$. (The reader should now check that our conditions hold for the trapezoidation of a single face in an arrangement of line segments, while there seems to be no way to make it fit the requirements of Clarkson and Shor.)

Although we want to make statements about random samples, it turns out to be useful to argue about random permutations. When we need a random sample of S of size r, we will use the first r elements in a random permutation of S. Thus, let $s_1, \ldots, s_n$ be a random permutation of S. All the probabilities and expectancies studied in the following will be with respect to this random permutation. Let $S_r := \{s_1, \ldots, s_r\}$, and $\mathcal{C}_r := \mathcal{C}(S_r)$.

Following Clarkson and Shor we define a function $\tau(r)$ that describes the expected number of regions for a sample of size at most r:

$$\tau(r) := \max_{1 \leqslant t \leqslant r} E\big[|\mathcal{C}_t|\big].$$

What we want to bound are the higher moments $E[\sum_{\Delta \in \mathcal{C}_r} \omega(\Delta)^d]$ for constant $d > 0$. We first prove the following simple lemma. We use the *falling factorial* notation $n^{\underline{a}} := n \cdot (n-1) \cdots (n-a+1)$, for integers $a \geqslant 0$.

Lemma 1 *Let $\Delta \in \mathcal{F}$. For $b \leqslant t \leqslant r \leqslant n$ we have $P[\Delta \in \mathcal{C}_r] \leqslant \frac{r^{\underline{b}}}{t^{\underline{b}}} P[\Delta \in \mathcal{C}_t]$.*

Proof: Let $a := b(\Delta)$. We have

$$\begin{aligned} P[\Delta \in \mathcal{C}_r] &= P[\mathcal{D}(\Delta) \subseteq S_r] \cdot P[\Delta \in \mathcal{C}_r \mid \mathcal{D}(\Delta) \subseteq S_r] \\ &= \frac{r^{\underline{a}}}{n^{\underline{a}}} P[\Delta \in \mathcal{C}_r \mid \mathcal{D}(\Delta) \subseteq S_r]. \end{aligned}$$

To bound the second probability, define the sequence $s'_1, s'_2, \ldots, s'_{n-a}$ by taking the sequence $s_1, \ldots, s_n$ and removing the elements of $\mathcal{D}(\Delta)$. We define $S'_r := \{s'_1, s'_2, \ldots, s'_r\}$. Let $\mathcal{S}^\Delta$ be the family of all subsets of S that create Δ in connection with $\mathcal{D}(\Delta)$, that is,

$$\mathcal{S}^\Delta := \Big\{ R \subseteq S \setminus \mathcal{D}(\Delta) \;\Big|\; \Delta \in \mathcal{C}(R \cup \mathcal{D}(\Delta)) \Big\}.$$

Note that property *(iv)* implies that any subset of a set in $\mathcal{S}^\Delta$ is also in $\mathcal{S}^\Delta$. Since $S'_{t-a} \subseteq S'_{r-a}$, this implies

$$P[\Delta \in \mathcal{C}_r \mid \mathcal{D}(\Delta) \subseteq S_r] \;=\; P[S'_{r-a} \in \mathcal{S}^\Delta] \leqslant P[S'_{t-a} \in \mathcal{S}^\Delta].$$

Putting everything together, we get

$$P[\Delta \in \mathcal{C}_r] \;=\; \frac{r^{\underline{a}}}{n^{\underline{a}}} P[S'_{r-a} \in \mathcal{S}^\Delta]$$

$$
\begin{aligned}
&\leqslant \frac{r^{\underline{a}}}{n^{\underline{a}}} P[S'_{t-a} \in \mathcal{S}^{\Delta}] \\
&= \frac{r^{\underline{a}}}{t^{\underline{a}}} \cdot \frac{t^{\underline{a}}}{n^{\underline{a}}} P[S'_{t-a} \in \mathcal{S}^{\Delta}] \qquad \text{(valid since } a \leqslant b \leqslant t\text{)} \\
&= \frac{r^{\underline{a}}}{t^{\underline{a}}} P[\Delta \in \mathcal{C}_t] \\
&\leqslant \frac{r^{\underline{b}}}{t^{\underline{b}}} P[\Delta \in \mathcal{C}_t]
\end{aligned}
$$

□

Before we proceed we need a few definitions, motivated by Chazelle et al. [CEG+91]. We define the following events.

$$
\begin{aligned}
X_r^{\Delta} &: \Delta \in \mathcal{C}_r \\
Y_r^{\Delta} &: \Delta \in \mathcal{C}_r \text{ and } s_{r+1} \in \mathcal{K}(\Delta)
\end{aligned}
$$

Note that

$$
P[Y_r^{\Delta}] = P[Y_r^{\Delta} \mid X_r^{\Delta}] \cdot P[X_r^{\Delta}] = \frac{\omega(\Delta)}{n-r} P[X_r^{\Delta}]. \tag{1}
$$

Furthermore, from the definition of Y_r^{Δ} and property *(iii)* we derive

$$
Y_r^{\Delta} \subseteq X_r^{\Delta} \cap \overline{X_{r+1}^{\Delta}}. \tag{2}
$$

The inclusion can be proper, since there can be other reasons why Δ does not show up in $\mathcal{C}_{r+1}$.

We observe that the sum $\sum_{t=A}^{B} P[Y_t^{\Delta}]$, for $1 \leqslant A < B \leqslant n$, is bounded by the probability that Δ appears in at least one of the sets $\mathcal{C}_A, \mathcal{C}_{A+1}, \ldots, \mathcal{C}_B$. The latter event can be expressed as

$$
X_A^{\Delta} \cup \bigcup_{t=A+1}^{B} (\overline{X_{t-1}^{\Delta}} \cap X_t^{\Delta}).
$$

But what is the probability $P[\overline{X_{t-1}^{\Delta}} \cap X_t^{\Delta}]$ that Δ gets "inserted" in step t? By *(iv)*, the only reason for some $\Delta \in \mathcal{C}_t$ not to be in $\mathcal{C}_{t-1}$ is that $s_t \in \mathcal{D}(\Delta)$. Hence, we have

$$
\begin{aligned}
P[\overline{X_{t-1}^{\Delta}} \cap X_t^{\Delta}] &= P[\overline{X_{t-1}^{\Delta}} \cap X_t^{\Delta} \mid X_t^{\Delta}] \cdot P[X_t^{\Delta}] \\
&= P[s_t \in \mathcal{D}(\Delta)] \cdot P[X_t^{\Delta}] \leqslant \frac{b}{t} P[X_t^{\Delta}].
\end{aligned} \tag{3}
$$

Here we used a technique called *backwards analysis*, introduced by Chew [Che85] and made popular by Seidel [Sei91, Sei93].

Putting everything together, we have

$$
\sum_{t=A}^{B} P[Y_t^{\Delta}] \leqslant P[X_A^{\Delta}] + \sum_{t=A+1}^{B} P[\overline{X_{t-1}^{\Delta}} \cap X_t^{\Delta}] \leqslant P[X_A^{\Delta}] + \frac{b}{A} \sum_{t=A+1}^{B} P[X_t^{\Delta}]. \tag{4}
$$

We can now prove the main theorem of this section.

Theorem 2 *For $1 \leqslant r \leqslant n$ and constant $d \geqslant 0$ there is a constant $C_d > 0$ such that*

$$
E[\sum_{\Delta \in \mathcal{C}_r} \omega(\Delta)^d] \leqslant C_d (\frac{n}{r})^d \tau(r).
$$

Proof: For $r < 2b$ the theorem holds with $C_d = (2b)^d$, so we assume that $r \geqslant 2b$. We proceed by induction on d. For $d = 0$ the sum whose expected value we want to bound is just the complexity of $\mathcal{C}_r$; by the definition of $\tau(r)$ the claim holds with $C_0 = 1$.

Now assume that the claim holds for powers less than d. We first prove by (a second level of) induction that for $r/2 \leqslant r_0 \leqslant r$ and $0 \leqslant j \leqslant d$ there exists a constant $C'_j > 0$ such that

$$\sum_{\Delta \in \mathcal{F}} \Big\{ \omega(\Delta)^j \sum_{t=r_0}^{r} P[X_t^\Delta] \Big\} \leqslant C'_j r (\frac{n}{r})^j \tau(r). \tag{5}$$

To do so, we define

$$F_j := \sum_{\Delta \in \mathcal{F}} \Big\{ \omega(\Delta)^j \sum_{t=r_0}^{r} P[X_t^\Delta] \Big\}.$$

Clearly, $F_0 \leqslant \sum_{t=r_0}^{r} \tau(t) \leqslant r\tau(r)$. Assume now that (5) holds for $j-1$, and consider F_j. We have

$$\begin{aligned}
F_j &= \sum_{\Delta \in \mathcal{F}} \Big\{ \omega(\Delta)^j \sum_{t=r_0}^{r} P[X_t^\Delta] \Big\} \\
&\overset{(1)}{\leqslant} \sum_{\Delta \in \mathcal{F}} \Big\{ \omega(\Delta)^{j-1} \sum_{t=r_0}^{r} (n-t)\, P[Y_t^\Delta] \Big\} \\
&\leqslant n \sum_{\Delta \in \mathcal{F}} \Big\{ \omega(\Delta)^{j-1} \sum_{t=r_0}^{r} P[Y_t^\Delta] \Big\} \\
&\overset{(4)}{\leqslant} n \sum_{\Delta \in \mathcal{F}} \omega(\Delta)^{j-1} P[X_{r_0}^\Delta] + \frac{bn}{r_0} \sum_{\Delta \in \mathcal{F}} \Big\{ \omega(\Delta)^{j-1} \sum_{t=r_0+1}^{r} P[X_t^\Delta] \Big\} \\
&\leqslant nC_{j-1}(\frac{n}{r_0})^{j-1}\tau(r_0) + b\frac{n}{r_0}F_{j-1} \\
&\leqslant r_0 C_{j-1}(\frac{n}{r_0})^{j}\tau(r_0) + b\frac{n}{r_0}C'_{j-1} r(\frac{n}{r})^{j-1}\tau(r) \\
&\leqslant (2^{j-1}C_{j-1} + 2bC'_{j-1}) r (\frac{n}{r})^j \tau(r) \\
&\leqslant C'_j r(\frac{n}{r})^j \tau(r)
\end{aligned}$$

for a $C'_j = 2^{j-1}C_{j-1} + 2bC'_{j-1}$. This proves claim (5). We can now consider $E[\sum_{\Delta \in \mathcal{C}_r} \omega(\Delta)^d]$. Let $r_0 := \lfloor r/2 \rfloor + 1$ and note that $r_0 \geqslant b$.

$$\begin{aligned}
\sum_{\Delta} \{\omega(\Delta)^d P[X_r^\Delta]\} &\leqslant \sum_{\Delta} \Big\{ \omega(\Delta)^d \frac{2}{r} \sum_{t=r_0}^{r} P[X_r^\Delta] \Big\} \\
&\leqslant C\frac{1}{r} \sum_{\Delta} \Big\{ \omega(\Delta)^d \sum_{t=r_0}^{r} P[X_t^\Delta] \Big\} = C\frac{1}{r}F_d
\end{aligned}$$

for some $C > 0$ by Lemma 1. Using (5), the theorem follows with $C_d = CC'_d$. $\square$

3 The complexity of $(\leqslant k)$-cells

The bounds on random sampling proved in the previous section can be used to obtain a nice combinatorial result on $(\leqslant k)$-cells in arrangements of line segments in the plane and arrangements of triangles in space. (See below for the definition of $(\leqslant k)$-cell.) Note that we are proceeding in the opposite direction as in the traditional work on random sampling. Clarkson and Shor [CS89] first prove a lemma on random sampling, based on the probability that a certain region with a given number of conflicts appears in a sample. They use this to give a bound on the complexity of $(\leqslant k)$-levels in arrangements of lines. Using these bounds, they finally obtain their bound for higher moments. We proceed in reverse: we first proved a theorem bounding the higher moments, and now use it to bound the complexity of $(\leqslant k)$-cells in arrangements of line segments. Note that Clarkson and Shor's proof doesn't work for $(\leqslant k)$-cells, for the reasons discussed earlier.

Let S be a set of n line segments in the plane. The $(\leqslant k)$-cell of S is defined as the set of points on the segments in S that can reach the origin with a path σ whose interior intersects at most $k-1$ segments in S. We denote the complexity of the $(\leqslant k)$-cell of S by $L_k(S)$. Notice that the $(\leqslant 1)$-cell is the boundary of the single cell defined by S containing the origin. Hence, for any set R of r segments we have $L_1(R) = O(r\alpha(r))$ [GSS89, PSS87].

The $(\leqslant k)$-cell of a set S of triangles in space is defined similarly: it is the set of points on the triangles in S that can reach the origin with a path σ whose interior intersects at most $k-1$ triangles in S. Again we denote the complexity of the $(\leqslant k)$-cell of S by $L_k(S)$. For a set R of n triangles in space we have $L_1(R) = O(r^2 \log r)$ [AS92].

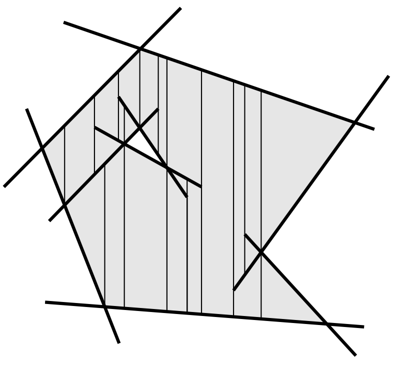

Figure 1: The vertical decomposition of a single cell.

Our proof technique requires a decomposition scheme for single cells.

For an arrangement of line segments in the plane we use the vertical decomposition, or trapezoidation.

This decomposition is obtained by extending vertical segments from any vertex of the single cell upward and/or downward, as illustrated in Figure 1. The number of trapezoids in the vertical decomposition of a single cell in the plane it linear in the complexity of the cell. Hence, for a single cell defined by m segments it is $O(m\alpha(m))$.

The vertical decomposition scheme that we use for arrangements of triangles in 3-space is defined analogously to its planar counterpart—see Section 6.1 for details. We denote the maximum complexity of the vertical decomposition of a single cell in an arrangement of m triangles by $\psi(m)$. De Berg et al. [dBGH94] proved that $\psi(n)$ is lower bounded by $O(n^2\alpha^2(n))$ and upper bounded by $O(n^{2+\varepsilon})$, for any constant $\varepsilon > 0$.

Theorem 3

(i) The maximum complexity of the $(\leqslant k)$-cell of a set of n line segments in the plane is $\Theta(nk\alpha(\frac{n}{k}))$.

(ii) The maximum complexity of the $(\leqslant k)$-cell of a set of n triangles in 3-space is $O(\psi(n)k)$ and $\Omega(n^2k\alpha(\frac{n}{k}))$, where $\psi(m) = O(m^{2+\varepsilon})$ is the maximum complexity of the vertical decomposition of a single cell in an arrangement of m triangles.

Proof: *(i)* To simplify the notation and without loss of generality we assume that n is a multiple of $2k+1$. Let $s_1, s_2, \ldots, s_n$ be a random permutation of S. Partition $s_1, s_2, \ldots, s_n$ into $2k+1$ subsets R_i of equal size, where $R_i := \{s_{\frac{(i-1)n}{2k+1}+1}, \ldots, s_{\frac{in}{2k+1}}\}$. For each subset R_i we decompose the single cell defined by R_i and containing the origin using a vertical decomposition. This decomposes the single cell into $O(L_1(R_i))$ trapezoids, each defined by at most 4 segments. The collection of trapezoids of the single cell of R_i is denoted $\mathcal{C}(R_i)$; it fulfills the conditions *(i)* to *(iv)* of Section 2.

To bound $L_k(S)$ it suffices to bound the number of vertices of the $(\leqslant k)$-cell. Such a vertex is either an endpoint of a segment—there are only $2n$ of these—or an intersection point between two segments s and s'. Let q denote such an intersection point. By definition, there is a path σ connecting q to the origin that intersects at most $k-1$ segments in S. Hence, there are at least k indices $i_1, \ldots, i_k$ such that σ does not cross any of the segments in R_{i_j} and $s, s' \notin R_{i_j}$. In other words, q lies in a trapezoid of $\mathcal{C}(R_{i_j})$ for at least k different indices i_j. For every trapezoid Δ of such a $\mathcal{C}(R_{i_j})$, there are $\omega(\Delta)$ segments which intersect it, and there are at most $\omega(\Delta)^2$ intersection points between pairs of these segments. Thus, the number of vertices of the $(\leqslant k)$-cell inside Δ is at most $\omega(\Delta)^2$, and the total number of vertices of the $(\leqslant k)$-cell is bounded by:

$$2n + \frac{1}{k} E[\sum_{i=1}^{2k+1} \sum_{\Delta \in \mathcal{C}(R_i)} \omega(\Delta)^2] = 2n + \frac{1}{k} \sum_{i=1}^{2k+1} E[\sum_{\Delta \in \mathcal{C}(R_i)} \omega(\Delta)^2].$$

Each R_i is a random sample of S of size $n/(2k+1)$, so from Theorem 2 we can conclude that there is a constant C_2 such that

$$E[\sum_{\Delta \in \mathcal{C}(R_i)} \omega(\Delta)^2] \leqslant C_2 (2k+1)^2 L_1(R_i).$$

The upper bound now follows from the fact that $L_1(R_i) = O(\frac{n}{2k+1}\alpha(\frac{n}{2k+1}))$. For the lower bound we use an example from Sharir [Sha91]. Let S' be a set of $n/(k/2+1)$ segments that define a single cell of complexity $\Theta(\frac{n}{k}\alpha(\frac{n}{k}))$. If we replace every segment $s \in S'$ by $k/2+1$ parallel segments which are close enough to s, then we obtain a set S of n segments that define a $(\leqslant k)$-cell of complexity $\Theta(nk\alpha(n/k))$.

(ii) We use the same proof technique as in the planar case. Thus we partition S into $2k+1$ random subsets R_i. We decompose the single cell defined by each subset R_i into boxes, using the vertical decomposition. This results in a set $\mathcal{C}(R_i)$ consisting of $O(\psi(n/(2k+1)))$ boxes. See Section 6.1 for details on the vertical decomposition.

Observe that the complexity of the $(\leqslant k)$-cell is determined, up to an additive $O(n^2)$ term, by the number of vertices that are the intersection of three triangles in S. Arguing in the same way as in the planar case we find that there is a constant C_3 such that the complexity of the $(\leqslant k)$-cell can be bounded as follows:

$$\begin{aligned} & O(n^2) + \frac{1}{k} E[\sum_{i=1}^{2k+1} \sum_{\Delta \in \mathcal{C}(R_i)} \omega(\Delta)^3] = \\ & = \quad O(n^2) + \frac{1}{k} \sum_{i=1}^{2k+1} E[\sum_{\Delta \in \mathcal{C}(R_i)} \omega(\Delta)^3] \end{aligned}$$

$$
\begin{aligned}
&= O(n^2) + \frac{1}{k}\sum_{i=1}^{2k+1} C_3(2k+1)^3 O(\psi(\frac{n}{2k+1})) \\
&= O(\psi(n)k).
\end{aligned}
$$

(The last step uses the fact that $\psi(n)$ is at least quadratic.) The lower bound example is similar to the planar case: take a set of $n/(k/3+1)$ triangles that define a single cell of complexity $\Theta((n/k)^2\alpha(n/k))$, and replace every triangle by $k/3+1$ parallel copies. $\square$

Remark. Sharir [Sha91] proved bounds on ($\leqslant k$)-sets in arrangements of curves and surfaces. When the curves are infinite then his ($\leqslant k$)-sets are the same as our ($\leqslant k$)-cells. Sharir's proof is based on the theory of Clarkson and Shor, so it cannot deal with ($\leqslant k$)-cells of Jordan arcs or surface patches. Our proof, on the other hand, also applies to such situations. As an example, consider a set of n arcs of bounded degree in the plane such that any pair intersects at most s times. The complexity of a single cell in such an arrangement is $O(\lambda_{s+2}(n))$ [GSS89], where $\lambda_q(m)$ is the maximum length of a (m,q)-Davenport-Schinzel sequence. Our proof techniques immediately gives us that the ($\leqslant k$)-cell has complexity $O(\lambda_{s+2}(n/k)k)$.

4 A lazy randomized incremental algorithm: computing a single face in an arrangement of line segments

The computation of single cells in two- and higher-dimensional spaces is important for motion planning problems: a single cell in the arrangement of constraint surfaces in configuration space corresponds to the reachable region for the robot in the workspace. The two-dimensional problem has already been solved quite satisfactorily by Chazelle et al. [CEG+91]. The algorithm that we present has the same running time as theirs and is slightly simpler. More importantly, it introduces a new technique called *lazy randomized incremental construction*, which can be used to solve a variety of other problems—see Section 6.

Let S be a set of n line segments in the plane. We wish to compute the face $\mathcal{C}(S)$ in the arrangement of S which contains the origin. Actually, we shall compute the vertical decomposition of this face. Our basic algorithm follows the general framework for randomized incremental algorithms described by Boissonnat et al. [BDS+92], which was also followed by Chazelle et al. These algorithms maintain a data-structure, the so-called *history graph*. The history graph for an insertion sequence $s_1, s_2, \ldots, s_n$ of the segments of S is a rooted, directed, and acyclic graph, whose nodes correspond to trapezoids that have been created during the incremental construction. In the following, we often do not distinguish between a node and its corresponding trapezoid. The trapezoids of the current single cell are leaf nodes in the history graph. The addition of a segment s_r is done as follows. The trapezoids of the current face that are intersected by s_r are determined by searching in the history graph. The new trapezoids that arise because of the insertion of s_r are computed, and their corresponding nodes are added to the history graph as leaves below the nodes (which are now no longer leaves) corresponding to the intersected trapezoids—see below for details.

In a standard randomized incremental algorithm this is basically all one has to do: update the history graph locally where regions are destroyed by the newly inserted object. But since single cells are not defined locally, this is not sufficient here: one should determine those trapezoids that are no longer in the single face, because they lie in a part that is cut off by s_r.

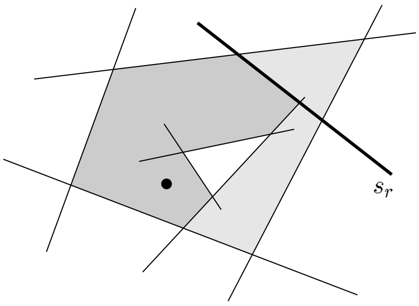

Figure 2: A new segment cuts off part of the single face.

Figure 2 illustrates this. The dot in this figure indicates the origin. Trapezoids in the lightly shaded region are cut off when s_r is inserted.

The leaves in the history graph corresponding to these trapezoids then have to be marked—we say that they are colored red—so that they will not be refined in future steps. This is the approach taken by Chazelle et al. In order to "clean up" the structure like this they need some extra tricks: they maintain a union-find structure on the boundary components of the single face so that they can detect when s_r cuts off part of the single face, and they apply a tandem search on the adjacency graph of the trapezoids in the current face to determine the parts that are cut off. Unfortunately, this approach does not seem to work in 3-space, where it is difficult to detect whether a new triangle cuts off a portion of the single cell.

We propose a much simpler strategy, which also works in 3-space: we simply skip the clean-up step, that is, after an insertion we do not attempt to identify the trapezoids that are no longer part of the single cell. Of course, we would end up constructing the full arrangement of line segments if we always keep all the trapezoids around. So we do perform a clean-up step, but only a small number of times. In particular, we perform a clean-up step after inserting the 2^i-th segment, for $1 \leqslant i \leqslant \lfloor \log n \rfloor$. The rationale behind this is that we are now allowed to perform the clean-up in a brute-force way, by visiting all the leaves in the history graph.

The global framework of a lazy randomized incremental algorithm is thus as follows.

> Let $s_1, s_2, \ldots, s_n$ be a random permutation of S.
> *last_cleanup* := 1
> **for** $r := 1$ **to** n
> **do** perform an update step to insert object s_r;
>   **if** $r = 2 \cdot$ *last_cleanup*
>   **then** perform a clean-up step; *last_cleanup* := r.

Figure 3 illustrates the lazy approach for the single face algorithm. The shaded trapezoids are the ones corresponding to green leaves. After we have inserted the first four segments and we have done the clean-up step, the shaded trapezoids are exactly the ones in the single face defined by the origin (the small dot in the figure). After inserting some more segments this is no longer true. But after inserting the eighth segment we do another clean-up, where we color the trapezoids outside the single face red. For readers who are not familiar with the algorithm of Chazelle et al. we shall give a more detailed description of the update step and the clean-up step for the computation of a single face in an arrangement of line segments.

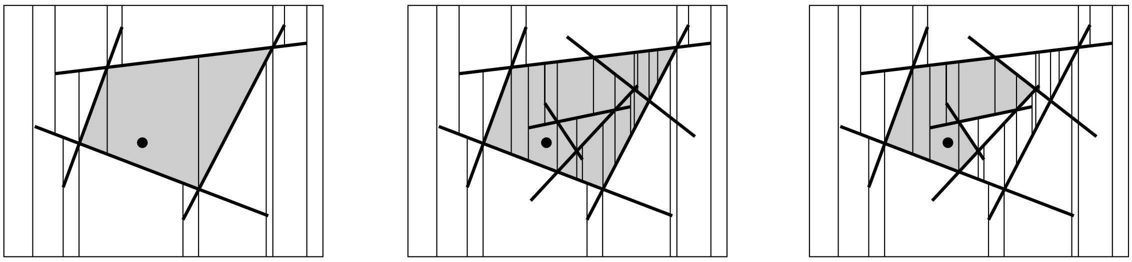

Figure 3: Computing a single face lazily.

We maintain two data structures: the history graph and an adjacency graph $\mathcal{G}$. The leaves of the history graph correspond to trapezoids. Leaves whose trapezoids need no further refinement because they are already known to be outside the current face are *red*; the other leaves are *green*. The nodes of the graph $\mathcal{G}$ correspond to trapezoids in the current structure that can still be in the single face. In other words, they correspond to green leaves of the history graph. There is an arc between two nodes in $\mathcal{G}$ if and only if the corresponding trapezoids share a vertical edge. This graph makes it possible to go from a trapezoid to its neighbors in the same face of the decomposition. We maintain cross-pointers between nodes in $\mathcal{G}$ and the corresponding green leaves in the history graph. We denote the trapezoid corresponding to a node ν in the history graph or the adjacency graph by $\Delta(\nu)$.

The update step. Starting at the root we propagate the segment s_r that has to be inserted down the history graph: from a node ν we proceed recursively to all children of ν whose trapezoids are intersected by s_r and that have not been visited before in this update step. This way we find all the leaves γ in the history graph such that the trapezoid $\Delta(\gamma)$ is intersected by s_r. The red leaves we have found need not be refined. For each green leaf γ we have found we split $\Delta(\gamma)$ with segment s_r to obtain a number of new trapezoids. The number of new trapezoids is at most four; Chazelle et al. give a detailed description of the different cases that can occur when a trapezoid is split. Some (at most two) of these new trapezoids are not properly defined trapezoids yet. More precisely, each vertical edge which does not contain a vertex of the new arrangement must be removed, and the two trapezoids that share this edge must be merged. In the history graph a green leaf is created for each new trapezoid, and there is an arc from a trapezoid $\Delta(\gamma)$ that has been split to a new trapezoid Δ if the interiors of $\Delta(\gamma)$ and Δ intersect. Note that a node γ has at most four out-going arcs. Finally, we update the adjacency graph $\mathcal{G}$: we remove the nodes corresponding to intersected trapezoids, and we add the newly created trapezoids with the appropriate adjacency relations.

The clean-up step. Let ν_o be the node in $\mathcal{G}$ whose trapezoid contains the origin. (This trapezoid can either be maintained during the update steps, or we can simply check all nodes of $\mathcal{G}$ to find it.) Perform a graph traversal on $\mathcal{G}$ to identify the connected component of $\mathcal{G}$ containing ν_o, and delete the other components from $\mathcal{G}$. Color all the nodes of the history graph red that correspond to nodes of $\mathcal{G}$ which are not in the same component as ν_o.

In the next section we give a general analysis of lazy randomized algorithms. This analysis is then used to prove that the expected running time of the above single face algorithm is $O(n\alpha(n)\log n)$. Anticipating this result we state the following theorem.

Theorem 4 *Given a set of n line segments in $\mathbb{R}^2$, the face in the arrangement containing the origin can be computed in $O(n\alpha(n)\log n)$ expected time using $O(n\alpha(n))$ storage.*

Remarks. The algorithm of Chazelle et al. is 'on-line' in the sense that at any point during the construction, they can use the current history graph to determine whether a query point lies in the current single cell. Our algorithm does not give us this possibility, since we have no way of knowing which trapezoids of the current subdivision are part of the single cell that we want to maintain. However, we can always report the current single face by doing a graph traversal, as in a clean-up step.

Our algorithm as well as the algorithm by Chazelle et al. also works for curves of bounded degree. The time bound becomes $O(\lambda_{s+2}(n)\log n)$. This bound is asymptotically the same as for the worst-case size of a single cell times a logarithmic factor.

Finally, note that our algorithm departs from previous work on randomized incremental algorithms as the structure maintained at time r is *not* independent of the order in which the segments have been inserted. More precisely, it depends on which of the objects were inserted before the most recent clean-up step and which objects were inserted after the most recent clean-up step.

5 The analysis of lazy randomized incremental algorithms

Below we give a general analysis of lazy randomized incremental algorithms. We then use this general result to analyze the example algorithm that we gave in the previous section, namely the computation of a face in an arrangement of line segments.

5.1 A general analysis

As in Section 2, we formulate the analysis in an abstract framework. This will allow us to apply it to several different cases. Again we advise the reader to keep a concrete example in mind, such as the computation of a face in an arrangement of line segments.

Let S be a set of n objects (a set of line segments, say), and let $\mathcal{F}$ be a set of *regions* (trapezoids defined by at most four of the segments). For every region $\Delta \in \mathcal{F}$, we assign subsets $\mathcal{D}(\Delta)$ and $\mathcal{K}(\Delta)$ of S, called the *defining set* and the *killing set*. Furthermore, for every $R \subseteq S$ we define two subsets $\mathcal{T}(R) \subset \mathcal{F}$ and $\mathcal{M}(R) \subset \mathcal{F}$. Finally, for $R' \subseteq R \subseteq S$ we define $\mathcal{C}(R, R') := \mathcal{T}(R) \cap \mathcal{M}(R')$. The idea is that the set $\mathcal{C}(R, R')$ 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 last clean-up was performed.

To illustrate these definitions, we apply them to our line segments example. Here $\mathcal{T}(R)$ is the set of *all* trapezoids in the full arrangement $\mathcal{A}(R)$, while $\mathcal{M}(R)$ is the set of trapezoids in $\mathcal{F}$ that are contained in the face of $\mathcal{A}(R)$ which contains the origin—note that a region $\Delta \in \mathcal{M}(R)$ does not have to be defined by segments in R, so $\mathcal{M}(R)$ is *not* a subset of $\mathcal{T}(R)$. The set $\mathcal{C}(R, R)$, however, corresponds exactly to the trapezoidation of the face containing the origin.

As in Section 2, we let $b(\Delta) := |\mathcal{D}(\Delta)|$ and $\omega(\Delta) := |\mathcal{K}(\Delta)|$. We require that

(i') there is a constant $b > 0$ such that $b(\Delta) \leqslant b$ 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 $R' \subseteq R \subseteq S$ we have $\mathcal{M}(R) \subseteq \mathcal{M}(R')$.

Note that conditions *(i')* and *(ii')* are exactly Clarkson's condition for the set $\mathcal{T}(R)$.

Let $s_1, \ldots, s_n$ be a random permutation of S, and let $S_r := \{s_1, \ldots, s_r\}$. Define $\mathcal{C}_r := \mathcal{C}(S_r, S_p)$, where p is the largest power of two that is less than r. Furthermore, for $1 \leqslant q < r \leqslant n$, we define the function

$$\tau(r, q) := \max_{q<t\leqslant r} E[|\mathcal{C}(S_t, S_q)|]. \tag{6}$$

Theorem 5 *Let $3 \leqslant r \leqslant n$ and let p be the largest power of two that is less than r. The expected size of $\mathcal{C}_r$ is at most $\tau(r, p)$. The expected number of regions in $\mathcal{C}_r \setminus \mathcal{C}_{r-1}$ is bounded by $O(\frac{1}{r}\tau(r, p/2))$. The expected total conflict size of these regions is bounded as*

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

Proof: The first claim follows immediately from the definition of $\tau(r, p)$. The other two results follow from the following bound, which holds for constant $d \geqslant 0$:

$$E[\textstyle\sum_{\Delta\in\mathcal{C}_r\setminus\mathcal{C}_{r-1}} \omega(\Delta)^d] \leqslant O(\frac{1}{r}(\frac{n}{r})^d\tau(r, p/2)) \tag{7}$$

To prove (7), let $q := p/2$, and fix a set $S^* \subset S$ of size q. We shall first restrict ourselves to those random permutations $s_1, \ldots, s_n$ of S with $S_q = S^*$. We denote by $P^*[X]$ the probability of an event X with respect to this restricted sample space.

$$\begin{aligned}
E[\textstyle\sum_{\Delta\in\mathcal{C}_r\setminus\mathcal{C}_{r-1}} \omega(\Delta)^d \mid S_q = S^*] &= \sum_{\Delta\in\mathcal{F}} \omega(\Delta)^d\, P^*[\Delta \in \mathcal{C}_r \wedge \Delta \notin \mathcal{C}_{r-1}] \\
&\overset{(iii')}{\leqslant} \sum_{\Delta\in\mathcal{F}} \omega(\Delta)^d\, P^*[\Delta \in \mathcal{C}(S_r, S_p) \wedge \Delta \notin \mathcal{T}(S_{r-1})] \\
&\overset{(iii')}{\leqslant} \sum_{\Delta\in\mathcal{F}} \omega(\Delta)^d\, P^*[\Delta \in \mathcal{C}(S_r, S_q) \wedge \Delta \notin \mathcal{T}(S_{r-1})] \\
&= \sum_{\Delta\in\mathcal{M}(S^*)} \omega(\Delta)^d\, P^*[\Delta \in \mathcal{T}(S_r) \wedge \Delta \notin \mathcal{T}(S_{r-1})] \\
&\overset{(ii')}{\leqslant} \sum_{\Delta\in\mathcal{M}(S^*)} \omega(\Delta)^d\, P^*[\Delta \in \mathcal{T}(S_r) \wedge s_r \in \mathcal{D}(\Delta)] \\
&\overset{(i')}{\leqslant} \frac{b}{r-q} \sum_{\Delta\in\mathcal{M}(S^*)} \omega(\Delta)^d\, P^*[\Delta \in \mathcal{T}(S_r)] \\
&\leqslant \frac{2b}{r}\, E[\textstyle\sum_{\Delta\in\mathcal{C}(S_r,S_q)} \omega(\Delta)^d \mid S_q = S^*].
\end{aligned}$$

To bound the expected value in the last expression, we apply Theorem 2 to the restricted sample space where $S_q = S^*$. Let $\overline{S} := S \setminus S^*$. For $\overline{R} \subseteq \overline{S}$, we define a set of regions as

$$\overline{\mathcal{C}}(\overline{R}) := \mathcal{C}(\overline{R} \cup S^*, S^*) = \mathcal{M}(S^*) \cap \mathcal{T}(S^* \cup \overline{R}).$$

The defining and killing sets of a region Δ are given as $\overline{\mathcal{D}(\Delta)} := \mathcal{D}(\Delta) \setminus S^*$ and $\overline{\mathcal{K}(\Delta)} := \mathcal{K}(\Delta) \setminus S^*$. It is not difficult to check that the conditions *(i)* to *(iv)* of Section 2 hold for the space defined in this way: condition *(i)* follows from our condition *(i')* and, since S^* is fixed, conditions *(ii)*, *(iii)* and *(iv)* follow from *(ii')*. We can thus use Theorem 2 to conclude that there is a constant $C_d > 0$ such that

$$\begin{aligned} E[\textstyle\sum_{\Delta \in \mathcal{C}(S_r, S_q)} \omega(\Delta)^d \mid S_q = S^*] &= E[\textstyle\sum_{\Delta \in \mathcal{C}(\overline{R} \cup S^*, S^*)} \omega(\Delta)^d \mid \overline{R} \subseteq \overline{S},\, |\overline{R}| = r - q] \\ &\leqslant C_d (\frac{n-q}{r-q})^d \max_{1 < t \leqslant r-q} E[|\mathcal{C}(\overline{R} \cup S^*, S^*)| \mid \overline{R} \subseteq \overline{S},\, |\overline{R}| = t] \\ &= C_d (\frac{n-q}{r-q})^d \max_{q < t \leqslant r} E[|\mathcal{C}(R, S^*)| \mid S^* \subseteq R \subseteq S,\, |R| = t]. \end{aligned}$$

The expectancy on the left hand side is still with respect to all random permutations of S with $S_q = S^*$. Taking the average over all possible choices of $S^* \subset S$ proves equation (7). ☐

5.2 The analysis of the single face algorithm

We apply the abstract analysis given above to bound the expected running time of the lazy randomized incremental algorithm for computing a single face that we presented in the previous section.

Here S is the given set of n line segments and $\mathcal{F}$ is the set of trapezoids that can be defined by any four of them. Thus condition *(i')* holds with $b = 4$. As mentioned above, we define $\mathcal{T}(R)$, for $R \subseteq S$, to be the set of all trapezoids in the vertical decomposition of the full arrangement $\mathcal{A}(R)$. Clearly, condition *(ii')* holds for $\mathcal{T}(R)$. Our clean-up steps are modeled using the set $\mathcal{M}(R)$, which we define to contain all trapezoids that lie in the single face of $\mathcal{A}(R)$ containing the origin. Then $\mathcal{C}_r = \mathcal{T}(S_r) \cap \mathcal{M}(S_p)$, where p is the largest power of two smaller than r, is exactly the set of trapezoids present after the insertion of s_r (but before doing any clean-up step). Notice that the single face shrinks when extra segments are added, so condition *(iii')* is satisfied.

We first bound the function $\tau(r, q)$, for $1 \leqslant q \leqslant n$ and $q < r \leqslant 4q$. Thus we have to bound the expected number of trapezoids in the trapezoidation $\mathcal{T}(S_t)$ that lie in the single face of $\mathcal{A}(S_q)$ containing the origin, for $q < t \leqslant r \leqslant 4q$. For this it is sufficient to bound the expected number of vertices of $\mathcal{A}(S_t)$ that lie in that face or on its boundary. Recall that $\mathcal{C}(S_q, S_q)$ denotes the trapezoids of the trapezoidation of the single face of $\mathcal{A}(S_q)$, and let $\omega(\Delta)$ be the number of segments in S_t intersecting trapezoid Δ. Then it is easy to see that the number of vertices we have to consider is bounded by

$$O\Big(\sum_{\Delta \in \mathcal{C}(S_q, S_q)} (1 + \omega(\Delta) + \omega(\Delta)^2) \Big). \tag{8}$$

To bound the expected value of this sum we note that the complexity of the single face is $O(q\alpha(q))$ [GSS89, PSS87], and we use that S_q is a random sample of S_t of size q with

$q \geqslant t/4$. We have seen in Section 2 that the set $\mathcal{C}(S_q, S_q)$ fulfills conditions *(i), (ii), (iii),* and *(iv)*. Hence, Theorem 2 tells us that

$$E[\sum_{\Delta \in \mathcal{C}(S_q,S_q)} (1 + \omega(\Delta) + \omega(\Delta)^2)] = O((1 + \frac{t}{q} + (\frac{t}{q})^2) q\alpha(q)) = O(t\alpha(t)).$$

It follows that the expected size of $\mathcal{C}(S_t, S_q)$ is $O(t\alpha(t))$. Hence, $\tau(r,q) = O(r\alpha(r))$ for $q \leqslant r < 4q$. By Theorem 5, this implies that the expected size of $\mathcal{C}_r$ is $O(r\alpha(r))$, so the structure we are maintaining is asymptotically not larger than the face we actually want to maintain.

The history graph constructed by the algorithm contains a node for every created trapezoid. The number of out-going edges for every node is at most four, so it suffices to bound the total number of trapezoids created by the algorithm to bound the expected size of the history graph. By Theorem 5, this is

$$\begin{aligned} \sum_{r=1}^{n} E[|\mathcal{C}_r \setminus \mathcal{C}_{r-1}|] &= \sum_{r=1}^{n} O(\tau(r, p/2)/r) \\ &= \sum_{r=1}^{n} O(r\alpha(r)/r) \\ &= O(n\alpha(n)). \end{aligned}$$

This bounds the expected amount of storage used by the algorithm. We can turn this into a worst-case bound by changing the algorithm slightly: as soon as we use too much storage we stop the algorithm and start with a new random permutation. We expect to succeed in a constant number of trials.

To bound the running time, we have to consider two different steps. First, consider the update steps. During update step r we spend time to find the trapezoids of $\mathcal{C}_{r-1}$ that are intersected by s_r, and we spend time to split certain trapezoids, update the history graph and update the graph $\mathcal{G}$. More precisely, we spend constant time for each node of the history graph that we visit. (Here we use that a node in the history graph has constant degree, so that we can check all its children to see which ones to visit.) Observe that we visit a node ν of the history graph only if s_r intersects the trapezoid $\Delta(\nu)$. It follows that we can bound the time necessary for *all* history searches by

$$\sum_{\Delta \in \mathcal{F}} \omega(\Delta)\, P[\Delta \text{ is created by the algorithm}].$$

This we can rewrite and bound using Theorem 5.

$$\begin{aligned} \sum_{r=1}^{n} E[\textstyle\sum_{\Delta \in \mathcal{C}_r \setminus \mathcal{C}_{r-1}} \omega(\Delta)] &= \sum_{r=1}^{n} O(\frac{n}{r^2} \tau(r, p/2)) \\ &= \sum_{r=1}^{n} O(\frac{n}{r^2} r\alpha(r)) \\ &= O(n\alpha(n) \log n). \end{aligned}$$

It remains to bound the time needed for the clean-up steps. A clean-up in step $p = 2^i$ takes time proportional to the number of trapezoids in $\mathcal{C}_p$. By the above, the expected number of trapezoids in $\mathcal{C}_p$ is $O(p\alpha(p))$ and, hence, the total expected work for the clean-up steps is

$$\sum_{i=1}^{\lfloor \log n \rfloor} O(2^i \alpha(2^i)) = O(n\alpha(n)).$$

This finishes the proof of Theorem 4.

6 More lazy randomized algorithms: computing single cells and zones

We now apply our technique to solve the problem that initiated our research: the computation of a single cell in an arrangement of triangles in 3-space. We also describe an algorithm for computing zones in arrangements of lines in the plane and arrangements of hyperplanes in d-dimensional space.

6.1 Vertical decompositions in 3-space

To be able to use a randomized incremental algorithm for computing single cells in three-space we need a decomposition scheme for arrangements of triangles. Therefore we first recall the notion of vertical decompositions in three-space introduced by Clarkson et al. [CEG$^+$90], before we present our algorithm and its analysis.

Let $S = \{s_1, \ldots, s_n\}$ be a set of n possibly intersecting triangles in three-space. The *vertical decomposition* of the arrangement $\mathcal{A}(S)$ decomposes each cell of $\mathcal{A}(S)$ into subcells, the *boxes*, and is defined as follows: from every point on an edge of $\mathcal{A}(S)$—this can be a part of a triangle edge or of the intersection of two triangles—we extend a vertical ray in positive and negative x_3-direction to the first triangle above and the first triangle below this point. This way we create for every edge a vertical wall, which we call a *primary wall*. We have now obtained a *cylindrical decomposition* of $\mathcal{A}(S)$ into cells with a unique bottom and top face; the vertical projections of both faces are exactly the same. However, the number of vertical walls of an obtained cell need not be constant and the cell may not be simply connected. So we decompose the vertical projection of its bottom face into trapezoids as in the planar case, that is, we draw segments parallel to the x_2-axis from the vertices of the face. These segments are extended vertically upward until they meet the top face; the walls thus erected are the *secondary walls*. Each cell of the vertical decomposition is now a box with a trapezoidal base and top—which may degenerate to a triangle—that are connected by vertical walls. De Berg et al. [dBGH94] proved the following bounds on the complexity of vertical decompositions.

Theorem 6 [dBGH94]
(i) The maximum combinatorial complexity of the vertical decomposition of an arrangement of n triangles with complexity A is lower bounded by $\Omega(n^2\alpha^2(n) + A)$ and upper bounded by $O(n^{2+\varepsilon} + A)$.
(ii) The maximum combinatorial complexity $\psi(n)$ of the vertical decomposition of a single cell in an arrangement of n triangles is lower bounded by $\Omega(n^2\alpha^2(n))$ and upper bounded by $O(n^{2+\varepsilon})$.

6.2 Computing single cells in arrangements of triangles

Our algorithm for the computation of the vertical decomposition of a single cell in an arrangement of n triangles in 3-space follows the lazy approach described in Section 4. When adding a triangle we do not seek to discard the boxes which are no longer in the cell; instead we keep even boxes that are no longer part of the single cell, and clean up the decomposition only after inserting the 2^i-th triangle, for $i = 1, \ldots, \lfloor \log n \rfloor$.

The algorithm we describe is a natural generalization of the algorithm of Section 4 to three dimensions. Thus it maintains a history graph and an adjacency graph. The red leaves of the history graph correspond to boxes that need no further refinement because they are already *known* to be outside the current cell; the other leaves are green. The boxes of the current decomposition are the green leaves of the history graph. The nodes of the adjacency graph $\mathcal{G}$ correspond to these boxes. There is an arc between two nodes in $\mathcal{G}$ if and only if the corresponding boxes share a vertical wall; this graph makes it possible to go from a box to its neighbors in the same cell of the decomposition. We maintain cross-pointers between nodes in $\mathcal{G}$ and the corresponding green leaves in the history graph. Next we describe the actions we have to take when adding a new triangle s_r. We start with the description of the update step, and then we describe the clean-up step.

The update step. We propagate the segment s_r down the history graph to determine the leaves whose corresponding boxes intersect s_r, as in Section 4. The red leaves we find need not be refined. For all green leaves γ that we find we decompose the box $\Delta(\gamma)$ into a constant number of new boxes (at most nineteen, to be precise). A green leaf of the history graph is created for each new box and linked to γ.

As in the planar case, the new set of boxes need not be a proper (vertical) decomposition. So our next step should be to merge the boxes which are separated by wall portions that must be removed. In the planar case the merging was a straightforward operation, but now it imposes a technical difficulty that requires some care.

Recall that there are two types of walls in the vertical decomposition: primary walls and secondary walls. Primary walls are erected from edges of the arrangement. Secondary walls are obtained as follows. After adding the primary walls, the bottom faces of the cells in the cylindrical decomposition are decomposed into trapezoids; the segments added in the trapezoidation are extended vertically upward until they meet the top face to obtain the secondary walls. Observe that any secondary wall must contain either a vertex of the three-dimensional arrangement of triangles or a vertical edge which is the intersection of two primary walls. Consider a wall intersected by s_r. The wall is cut into at most four pieces by s_r and the primary walls from its edges: a portion below s_r, a portion above s_r, and two other portions, one on either side of s_r. If the wall is a primary wall erected from an edge e of the arrangement, we have to remove the portion of the wall lying on the opposite side of s_r (as seen from e). Figure 4(i) illustrates this. The lightly shaded part of the wall erected for edge e must be removed when s_r is inserted. If the wall is a secondary wall, as in Figure 4(ii), then the only portion that remains is the portion containing either a vertex of the new arrangement or a vertical edge which is the intersection of two new primary walls. Note that a secondary wall may even disappear completely. Now that we have seen which walls to remove, let us examine the merging in more detail.

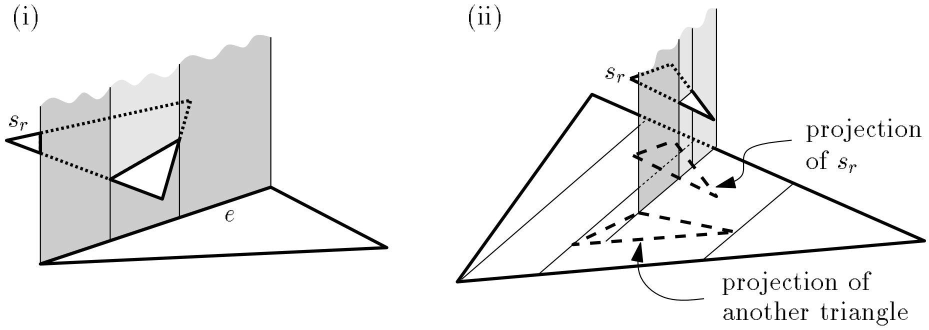

Figure 4: The lightly shaded wall portions have to be removed because of the new triangle s_r.

We distinguish two types of boxes in the merging step: the boxes that have neither a bottom nor a top face contained in s_r, and the boxes that do have their top or bottom face contained in s_r. Notice that two boxes of a different type never have to be merged, since any box has its top and bottom face contained in a single triangle.

It is not difficult to see that the merging of the boxes of the first type only involves the removal of portions of secondary walls. This makes the merging easy. We simply check which portions contain either a vertex of the new arrangement or a vertical edge which is the intersection of two new primary walls (this information is easy to maintain). All other wall portions are removed, that is, we merge the corresponding boxes and we update the links in the history graph and the adjacency graph accordingly. Note that the boxes only grow, so the out-degree of the nodes in the history graph does not increase.

Next, we consider the boxes of the second type, whose bottom or top face is contained in s_r. Here we have to remove portions of primary walls as well as portions of secondary walls. The removal of portions of primary walls is problematic: it may happen that some of the secondary walls were 'stopped' by the portions of primary walls that we are going to remove. This means that these secondary walls now have to be extended. But then an old box can be cut into many pieces by the extended secondary walls, increasing the out-degree of the corresponding node in the history graph. In other words, we are not allowed to link an old box to all the boxes in the new vertical decomposition that it intersects. Fortunately, this problem can be solved using a trick proposed by Boissonnat and Dobrindt [BD92]. The idea is to 'encode' the links from an old box to all the new boxes it intersects in a rooted graph—the *secondary history graph*. The leaves of the secondary history graph correspond to the new boxes. Instead of having links to all the new boxes it intersects, an old box has an out-going arc to the root of the secondary history graph. This way the out-degree of the nodes is kept constant. Next we describe the construction of this secondary history graph in more detail.

Let B_b (B_t) denote the set of boxes whose bottom (top) face is contained in s_r, after we have splitted the boxes with s_r, but before we have started the merging process. We will construct two secondary history graphs, one for the boxes in B_b and one for the boxes in B_t. We describe how to construct the secondary history graph for B_b; the construction for B_t is similar.

The union of the bottom faces of the boxes in B_b is the part of the top side of s_r that is inside the current single cell (that is, the single cell of the latest clean-up step). Notice that this union is not necessarily connected, and that its components are not necessarily simply connected. Consider all faces of the boxes in B_b which are contained in a still existing primary wall,[2] and let S_b be the set of bottom edges of these faces. Such a bottom edge is the intersection of the face with the triangle s_r, since the boxes have s_r as their bottom face. We merge collinear segments of S_b if they share an endpoint and there is no third segment with this endpoint. Observe that the segments in S_b have disjoint interiors. Also observe that the cells of the arrangement $\mathcal{A}(S_b)$ on s_r defined by S_b are exactly the bottom faces of the cells of the cylindrical decomposition of the current arrangement that are contained in s_r. Now, in order to obtain a correct vertical decomposition we have to compute a (two-dimensional) trapezoidation of $\mathcal{A}(S_b)$, and extend the extra segments thus created vertically upward to obtain the new secondary walls. This trapezoidation is computed using a two-dimensional randomized incremental algorithm described in [BDS$^+$92]. The history graph that results from this construction is our secondary history graph for the set B_b. We also store the adjacency graph for the trapezoidal decomposition of $\mathcal{A}(S_b)$. The leaves of the secondary history graph correspond to trapezoids in the trapezoidation of $\mathcal{A}(S_b)$, which, in turn, correspond to the new boxes in the tree-dimensional vertical decomposition. In general, the union of the bottom faces of the boxes in B_b will not be the entire triangle s_r, since parts of s_r can lie outside the current single cell. The leaves that correspond to trapezoids lying outside this union are of no interest to us, so we color them red. Finally, we link all the boxes in B_b to the root of the secondary history graph. Note that the number of out-going arcs for every node is bounded by a constant.

Now consider a later step in our algorithm, where we insert a triangle s_j, $j > r$. We have to determine the boxes in the current decomposition which are intersected by s_j. This we can still do by traversing the history graph, as follows.

The search in the history graph proceeds in the normal way as long as we do not encounter any secondary history graphs. Now suppose that we reach a node ν in the primary history graph that has an out-going arc to the root of a secondary history graph. Assume that this secondary history graph was constructed when we inserted triangle s_r, and assume that $\Delta(\nu)$ is a box in B_b. (The set B_b is defined with respect to the triangle s_r, as above.) We compute a point $q \in s_j \cap \Delta(\nu)$, project this point onto s_r, and continue the search in the secondary history graph with the projected point $\overline{q}$. This way we find in $O(\log n)$ time the trapezoid in the trapezoidation of $\mathcal{A}(S_b)$ that contains $\overline{q}$. We now switch to the adjacency graph of the trapezoidation of $\mathcal{A}(S_b)$. Using a graph traversal we determine all the trapezoids that are in the single cell of $\mathcal{A}(S_b)$ defined by $\overline{q}$ and whose corresponding boxes are intersected by s_j. These boxes correspond to nodes in the primary history graph from which we continue our search.

Observe that s_j can intersect several of the boxes in B_b. For each of these boxes we compute a point to search with in the secondary history graph. Hence, we may visit trapezoids in $\mathcal{A}(S_b)$ more than once. To avoid traversing the same part of the adjacency graph too often, we mark every trapezoid when we visit it. When we reach this trapezoid again while handling s_j, we know that we already visited it and we don't start a graph traversal.

This finishes the description of the search in the secondary history graph. But what

[2]Some faces of the boxes in B_b can be contained in a portion of a primary wall that we must remove.

about the correctness of the search, and what about the extra time we spend?

Lemma 7 *The searches in the secondary history graph for $\mathcal{A}(S_b)$ give us exactly those trapezoids whose corresponding box is intersected by s_j.*

Proof: We explicitly test boxes for intersection when we traverse the adjacency graph, so it remains to prove that we find all intersected boxes. Let Δ_1 be an intersected box corresponding to a trapezoid in $\mathcal{A}(S_b)$, and let q_1 be a point in $\Delta_1 \cap s_j$. Let Δ_2 be the box in B_b that contains q_1. The box Δ_2 intersects s_j, so we have generated some point $q_2 \in \Delta_2 \cap s_j$ to search in the history graph. It is not necessarily true that $\overline{q_2}$ and $\overline{q_1}$ are in the same trapezoid of $\mathcal{A}(S_b)$: some new secondary walls that arose when we constructed the trapezoidation for $\mathcal{A}(S_b)$ may prevent this. But we claim that the two trapezoids containing $\overline{q_1}$ and $\overline{q_2}$ are connected in the adjacency graph of $\mathcal{A}(S_b)$. To see this, note that q_1q_2, the line segment connecting q_1 and q_2, is contained in Δ_2 and does not intersect any primary walls. When constructing the secondary history graph for $\mathcal{A}(S_b)$ we only add secondary walls. Hence, the segment $\overline{q_1q_2}$ does not intersect primary walls either, and the trapezoids containing $\overline{q_1}$ and $\overline{q_2}$ are connected in the adjacency graph. Moreover, the box corresponding to a trapezoid intersected by $\overline{q_1q_2}$ is intersected by q_1q_2 and, hence, by s_j. We conclude that we will report the box Δ_1 during a traversal of the adjacency graph. $\Box$

The time it takes to go down the secondary history graph with some point $\overline{q}$ is bounded by the depth of the secondary history graph, which is expected to be $O(\log n)$. We charge this extra time to the node ν of the primary history graph that generated the point q. The time for traversing the adjacency graph is linear in the number of nodes in the primary history graph that we discover; we charge the time to these nodes. In total, every node in the primary history graph gets charged an extra $O(\log n)$ each time it is visited.

The clean-up step. We identify the connected component of $\mathcal{G}$ containing the node whose trapezoid contains the origin by a graph traversal. The nodes of $\mathcal{G}$ which are not in this component are deleted from $\mathcal{G}$ and the corresponding leaves of the history graph are colored red.

6.3 Analysis of the single cell algorithm

The analysis combines the analysis of Boissonnat and Dobrindt [BD92] with the results of Section 5.1. We distinguish *principal* nodes of the history graph, corresponding to boxes that have been created during the incremental construction, and *secondary* nodes, which are inner nodes of the secondary history graphs. We express the main results using the function $\psi(n)$, which denotes the maximum number of boxes in the vertical decomposition of a single cell in an arrangement of n triangles. We proceed as in Section 5.2. S is the given set of triangles, $\mathcal{F}$ is the set of all boxes (each defined by at most six triangles), $\mathcal{T}(R)$ is the set of boxes in the vertical decomposition of a subset $R \subseteq S$, and $\mathcal{M}(R)$ is the set of boxes in $\mathcal{F}$ contained in the single cell of $\mathcal{A}(R)$ containing the origin. It follows that $\mathcal{C}_r = \mathcal{T}(S_r) \cap \mathcal{M}(S_p)$, where p is the largest power of two that is less than r, is exactly the set of boxes present at stage r of our algorithm.

As in Section 5.2, we use Theorem 6 to bound the expected size of the structure we maintain. We shall prove that

$$E[|\mathcal{C}(S_t, S_q)|] = O(\psi(t)), \tag{9}$$

for $q < t \leqslant r \leqslant 4q$, which implies that $\tau(r,q) = O(\psi(r))$ for $q \leqslant r \leqslant 4q$.

In Section 5.2 we used the number of vertices of the planar arrangement $\mathcal{A}(S_t)$ lying in a single face of $\mathcal{A}(S_q)$ as an asymptotic bound on the complexity of the structure that we maintained. The number of such vertices was then bounded using Equation (8). The complexity of the structure that we maintain in the three-dimensoinal setting can be analyzed in a similar way. In particular, to prove Equation (9) it is sufficient to bound the expected number of vertices of $\mathcal{A}(S_t)$ that lie in that cell or on its boundary plus the expected number of intersections between two primary walls erected from edges of $\mathcal{A}(S_t)$ that lie in that cell or on its boundary. To bound the first number we have to add a term $\omega(\Delta)^3$ in Equation (8). To bound the second number we note that the expected number of edges in a box $\Delta \in \mathcal{C}(S_q, S_q)$ is $\omega(\Delta)^2$; hence, the expected number of intersections between two primary walls erected from them adds a term $\omega(\Delta)^4$ in Equation (8). By Theorem 5, this implies that the expected complexity of $\mathcal{C}_r$ is only $O(\psi(r))$. Again, the structure we are maintaining is asymptotically not larger than the single cell itself.

We now bound the expected size of the history graph constructed by the algorithm. The expected number of principal nodes can be bounded as

$$\sum_{r=1}^{n} E[|\mathcal{C}_r \setminus \mathcal{C}_{r-1}|] = \sum_{r=1}^{n} O(\frac{1}{r}\psi(r)) = O(\psi(n)),$$

since $\psi(n)$ is at least linear. Next we bound the number of secondary nodes created by the algorithm. Recall that the secondary history graphs are constructed on a set of non-intersecting line segments. This implies that the total number of trapezoids that are created for the secondary history graphs is linear in the number of line segments [BDS+92]. It remains to observe that this number is linear in the number of boxes of the vertical decomposition that are destroyed. Since a destroyed box has to be created first, the total number of secondary nodes is bounded by $O(\psi(n))$.

As for the running time, we observe that the history search for a new triangle s_r visits only principal nodes ν of the current history graph with $s_r \in \mathcal{K}(\Delta(\nu))$. It follows that the expected number of principal nodes visited during all history searches can be bounded by

$$\sum_{r=1}^{n} E[\textstyle\sum_{\Delta \in \mathcal{C}_r \setminus \mathcal{C}_{r-1}} \omega(\Delta)] = \displaystyle\sum_{r=1}^{n} O(\frac{n}{r^2}\psi(r)) = O(\psi(n)),$$

since $\psi(n) = \Omega(n^{1+\varepsilon})$ for a constant $\varepsilon > 0$. As argued above, the searches in the secondary history graphs can be charged to principal nodes in such a way that every principal node gets charged an extra $O(\log n)$ factor each time it is visited.

After we have identified the boxes of the current decomposition that are intersected by the new triangle, we perform the merging step as described before. Creating the secondary history graph takes $O(m \log m)$ time, where m is the number of boxes that are destroyed in this step. By the bound on the total number of boxes created by the algorithm, this results in an expected total update time of $O(\psi(n) \log n)$.

Finally, the total expected work for the clean-up steps is bounded by

$$\sum_{i=1}^{\lfloor \log n \rfloor} \psi(2^i) = O(\psi(n)).$$

This completes the analysis of the algorithm and yields the following theorem.

Theorem 8 *Given a set of n triangles in three-space, the cell in the arrangement containing the origin can be computed in $O(\psi(n) \log n)$ expected time using $O(\psi(n))$ storage, where $\psi(m)$ denotes the maximum number of boxes in the vertical decomposition of a single cell in an arrangement of m triangles.*

By Theorem 6(ii) we obtain the following corollary.

Corollary 9 *Given a set of n triangles in three-space, the cell in the arrangement containing the origin can be computed in $O(n^{2+\varepsilon})$ expected time using $O(n^{2+\varepsilon})$ storage, for any fixed $\varepsilon > 0$.*

6.4 Computing zones in arrangements of hyperplanes

Given a set S of n hyperplanes in d-dimensional space and a surface γ, the *zone of γ with respect to S* is the set of cells of $\mathcal{A}(S)$ intersected by γ. We assume that γ has constant description complexity.

Our algorithm follows the lazy incremental construction paradigm outlined in Section 4. Thus we add the hyperplanes of the set S in random order, meanwhile maintaining the bottom-vertex decomposition of the arrangement $\mathcal{A}(S)$ (see for instance Mulmuley's book [Mul93, Chapter 6.3] for a definition of the bottom-vertex triangulation). After inserting the hyperplane number 2^i, for $i = 1, \ldots, \lfloor \log n \rfloor$, we do a clean-up step and we discard all cells that are no longer in the zone. We will depart from our previous scheme and describe this algorithm using a *conflict graph* instead of a history graph. Using conflict graphs turns out to be slightly simpler for this problem; if necessary, the algorithm can be modified to use a history graph. It also shows that the lazy paradigm is independent of the nature of the conflict search. So, for every simplex Δ of the bottom-vertex decomposition of $\mathcal{A}(S_r)$, we maintain the conflict list $\mathcal{K}(\Delta)$. For every s_j not yet added we maintain reverse pointers to all Δ with $s_j \in \mathcal{K}(\Delta)$. The update step and the clean-up step are performed as follows.

The update step. Using the reverse pointers to the conflict lists, we first collect the simplices in the current arrangement intersected by the new hyperplane s_r. This tells us which cells of the current arrangement have to be split and retriangulated. This can be done in time proportional to the total number of simplices intersected, since we are allowed to spend time proportional to the size of the subcell that does not contain the bottom-vertex of the cell. See again Mulmuley's book for details.

It remains to create the conflict lists for the newly created simplices. To this end we collect the hyperplanes in all conflict lists of all destroyed simplices of a given cell. For every such hyperplane s_j, $j > r$, we then determine the new simplices Δ intersected by s_j. For the simplices in each of the two new cells this can be done in the following way. First, we find one vertex of the cell that is on the opposite side of s_r than the bottom vertex of the cell. From that vertex we perform a graph search on the boundary of the cell. It is not difficult to show that the total time for this operation is proportional to the total length of the conflict lists of all destroyed and created simplices. Details can be found in Schwarzkopf [Sch92].

The clean-up step. First, we test every simplex of the current arrangement to find those intersected by the surface γ. Then we traverse the triangulation, starting from these simplices, to identify the cells in the zone of γ. All simplices not in the zone of γ are simply discarded.

6.5 Analysis of the zone algorithm

Let $\zeta_\gamma(n,d)$ denote the maximum complexity of the zone of γ in an arrangement of n hyperplanes in d-dimensional space. We want to bound the running time of our algorithm in terms of $\zeta_\gamma(n,d)$.

We employ our tools from Section 5.1 and define $\mathcal{F}$ as the set of all simplices defined by hyperplanes in S. For a box $\Delta \in \mathcal{F}$, $\mathcal{K}(\Delta)$ is the set of hyperplanes intersecting Δ. For $R \subseteq S$, we define $\mathcal{T}(R)$ as the set of simplices in the bottom-vertex triangulation of $\mathcal{A}(R)$, and $\mathcal{M}(R)$ as the set of all simplices in $\mathcal{F}$ that are contained in the zone of γ with respect to R. Then $\mathcal{C}_r$ as defined in Section 5.1 is the set of simplices present in our structure after inserting hyperplane s_r, but before any clean-up step.

We first want to bound the expected size of $\mathcal{C}(S_t, S_q)$, for $q < t \leqslant r \leqslant 4q$. It suffices to bound the number of vertices of $\mathcal{A}(S_t)$ within the zone of γ with respect to S_q. We can bound this using Theorem 2 as

$$E[|\mathcal{C}(S_t, S_q)|] = O(\zeta_\gamma(t,d)).$$

This implies that $\tau(r,q) = O(\zeta_\gamma(r,d))$, for $q > r/4$.

As observed before, the running time in step r is dominated by the total size of the conflict lists of all simplices that are created or destroyed. It follows that the total expected running time is bounded by

$$\sum_{r=1}^{n} E[\textstyle\sum_{\Delta\in\mathcal{C}_r\setminus\mathcal{C}_{r-1}} \omega(\Delta)] = \sum_{r=1}^{n} O(\frac{n}{r^2}\zeta_\gamma(r,d)).$$

To bound the storage requirement of the algorithm, we have to bound the expected total size of all conflict lists at time r, which is

$$E[\sum_{\Delta\in\mathcal{C}_r} \omega(\Delta)].$$

The same proof technique used in Theorem 5 can be applied to show that this sum is bounded by $O((n/r)\zeta_\gamma(r,d))$. This is always bounded by $O(n + \zeta_\gamma(n,d))$. Thus, we obtain the following theorem.

Theorem 10 *Let $\zeta_\gamma(n,d)$ denote the maximum complexity of the zone of a surface γ in an arrangement of n hyperplanes in d-dimensional space. Then the zone of γ with respect* *to an arrangement $\mathcal{A}(S)$ of n hyperplanes in d-dimensional space can be computed in expected time*

$$\sum_{r=1}^{n} O(\frac{n}{r^2}\zeta_\gamma(r,d))$$

and storage $O(n+\zeta_\gamma(n,d))$.

When γ is a hyperplane then tight bounds are known on the maximum zone complexity: the famous Zone Theorem [ESS93] states that in d-dimensional space the complexity is is $O(n^{d-1})$. In this case we obtain an optimal algorithm.

Corollary 11 *The zone of a hyperplane in an arrangement of n hyperplanes in d-dimensional space can be computed in $O(n^{d-1}+n\log n)$ time using $O(n^{d-1})$ storage.*

7 Conclusion

We introduced a new type of randomized incremental algorithms, which are suited for computing structures that have a 'non-local' definition. In order to analyze these lazy randomized algorithms we generalized the results of Clarkson and Shor on random sampling to such situations. Our technique yields efficient algorithms for computing single cells in arrangements of segments in the plane and in arrangements of triangles in 3-space, and for computing zones. We also used our combinatorial results on random sampling to prove bounds on ($\leqslant k$)-cells in such arrangements.

We believe that our technique can be useful for a number of other problems as well. For example, the best known algorithm for computing all non-convex cells in an arrangement of triangles in 3-space runs in $O(n^{8/3}\log^{14/3} n)$ time [AS90], although the total complexity of these cells is known to be $O(n^{7/3}\log n)$. If the maximum complexity of the vertical decomposition of m cells in an arrangement of n triangles is close to the maximum complexity of the cells themselves, which is $O(n^2\log^2 n+nm^{2/3}\log n)$, then we immediately obtain an algorithm with a running time close to $O(n^{7/3})$.

References

[AS90] B. Aronov and M. Sharir. Triangles in space or building (and analyzing) castles in the air. *Combinatorica*, 10(2):137–173, 1990.

[AS92] B. Aronov and M. Sharir. Castles in the air revisited. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 146–156, 1992.

[BD92] J. D. Boissonnat and K. Dobrindt. Randomized construction of the upper envelope of triangles in R^3. In *Proc. 4th Canad. Conf. Comput. Geom.*, pages 311–315, 1992.

[BDS+92] J.-D. Boissonnat, O. Devillers, R. Schott, M. Teillaud, and M. Yvinec. Applications of random sampling to on-line algorithms in computational geometry. *Discrete Comput. Geom.*, 8:51–71, 1992.

[BT93] J.-D. Boissonnat and M. Teillaud. On the randomized construction of the Delaunay tree. *Theoret. Comput. Sci.*, 112:339–354, 1993.

[CEG+90] K. Clarkson, H. Edelsbrunner, L. Guibas, M. Sharir, and E. Welzl. Combinatorial complexity bounds for arrangements of curves and spheres. *Discrete Comput. Geom.*, 5:99–160, 1990.

[CEG+91] B. Chazelle, H. Edelsbrunner, L. Guibas, M. Sharir, and J. Snoeyink. Computing a face in an arrangement of line segments. In *Proc. 2nd ACM-SIAM Sympos. Discrete Algorithms*, pages 441–448, 1991.

[Che85] L. P. Chew. Building Voronoi diagrams for convex polygons in linear expected time. Report, Dept. Math. Comput. Sci., Dartmouth College, Hanover, NH, 1985.

[CMS93] K. L. Clarkson, K. Mehlhorn, and R. Seidel. Four results on randomized incremental constructions. *Comput. Geom. Theory Appl.*, 3(4):185–212, 1993.

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

[dBGH94] M. de Berg, L. J. Guibas, and D. Halperin. Vertical decompositions for triangles in 3-space. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, 1994.

[ESS93] H. Edelsbrunner, R. Seidel, and M. Sharir. On the zone theorem for hyperplane arrangements. *SIAM J. Comput.*, 22(2):418–429, 1993.

[GKS92] L. J. Guibas, D. E. Knuth, and M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7:381–413, 1992.

[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.

[MMÓ91] K. Mehlhorn, S. Meiser, and C. Ó'Dúnlaing. On the construction of abstract Voronoi diagrams. *Discrete Comput. Geom.*, 6:211–224, 1991.

[MS91] N. Miller and M. Sharir. Efficient randomized algorithm for constructing the union of fat triangles and of pseudodiscs. Manuscript, 1991.

[Mul88] K. Mulmuley. A fast planar partition algorithm, I. In *Proc. 29th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 580–589, 1988.

[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.

[Sch92] O. Schwarzkopf. *Dynamic Maintenance of Convex Polytopes and Related Structures.* Ph.D. thesis, Fachbereich Mathematik, Freie Universität Berlin, Berlin, Germany, June 1992.

[Sei91] R. Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete Comput. Geom.*, 6:423–434, 1991.

[Sei93] R. Seidel. Backwards analysis of randomized geometric algorithms. In J. Pach, editor, *New Trends in Discrete and Computational Geometry*, volume 10 of *Algorithms and Combinatorics*, pages 37–68. Springer-Verlag, 1993.

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