Derandomization in computational geometry

Jiří Matoušek*
Department of Applied Mathematics
Charles University
Malostranské nám. 25, 118 00 Praha 1, Czech Republic

Abstract

We survey techniques for replacing randomized algorithms in computational geometry by deterministic ones with a similar asymptotic running time.

1 Randomized algorithms and derandomization

A rapid growth of knowledge about randomized algorithms stimulates research in derandomization, that is, replacing randomized algorithms by deterministic ones with as small decrease of efficiency as possible. Related to the problem of derandomization is the question of reducing the amount of random bits needed by a randomized algorithm while retaining its efficiency; the derandomization can be viewed as an ultimate case. Randomized algorithms are also related to probabilistic proofs and constructions in combinatorics (which came first historically), whose development has similarly been accompanied by the effort to replace them by explicit, non-random constructions whenever possible.

Derandomization of algorithms can be seen as a part of an effort to map the power of randomness and explain its role. Another, more practical motivation is that contemporary computers include no equipment for generating truly random bits. The random bits are simulated by pseudo-random generators; these can produce long sequences of bits which satisfy various statistical criteria of randomness, but the real randomness in the computation is in fact restricted to the initial setting of the generator. It is thus important to gain theoretical results on limiting the amount of randomness in the algorithm.

In the sequel, the word 'derandomization' will usually be used in a narrower sense, where the deterministic algorithm arises by closely imitating the randomized one, by adding special subroutines that replace the random resources in a manner sufficient for that particular algorithm. Although it is difficult to strictly distinguish when a deterministic algorithm imitates the randomized one and when it becomes a brand new algorithm, in most specific cases the distinction appears quite clearly. One typical feature of derandomized algorithms is that they can hardly be understood without appealing to their underlying randomized counterpart; taken alone as deterministic algorithms they would appear completely mysterious.

Derandomization turns out to be a surprisingly powerful methodology for designing deterministic algorithms. For such a basic problem as computing the convex hull of n points in a fixed dimension d, the only known worst-case optimal deterministic algorithm arises by a (quite complicated) derandomization of a simple randomized algorithm (Chazelle [Cha93b]).

*Supported by grants GAČR 0194 and GAUK 193,194. Part of this survey was written during a visit at ETH Zürich.

Bibliography and remarks. Excellent books on probabilistic methods in combinatorics are Spencer [Spe87] and Alon and Spencer [AS93]; the latter one has a lot of material concerning explicit constructions and derandomization.

The present survey was submitted for publication in the end of 1995. Some small updates and additions were made in June 1997, but certainly not as thorough as the progress in the field would perhaps deserve.

Many of the cited papers first appeared in conference proceedings and, sometimes many years later, in a journal. Where available, we refer to the journal version (presumably more polished and containing fewer mistakes), although the chronology of the results may be considerably distorted by this.

2 Basic types of randomized algorithms in computational geometry

The derandomization in computational geometry can be declared relatively successful compared to other fields; for most randomized algorithms one can produce deterministic ones with only a small loss in asymptotic efficiency, and the open problems usually concern further relatively small improvements. The main reason for this is probably that the space dimension is assumed to be fixed and the constants of proportionality in the asymptotic notation are ignored. These constants grow considerably by derandomization, and currently most of the derandomized algorithms seem unsuitable for a practical use, especially if the dimension is not quite small.

Before we start discussing derandomization, let us recall two basic paradigms for designing randomized algorithms in computational geometry: the *randomized divide-and-conquer* and the *randomized incremental construction*. Our presentation is sketchy and is illustrated on a very simple artificial example, so simple that both approaches are a clear overkill for solving the problem. We assume that the reader can learn more about randomized geometric algorithms from other sources (for instance, in the chapter by Ketan Mulmuley). We consider the following problem:

Problem 2.1 *Given a set H of n lines in the plane, compute the intersection $I(H)$ of the halfplanes determined by the lines of H and containing the origin.*

Randomized divide-and-conquer. For this paradigm, we choose a random sample $S \subset H$ of r lines, where r is a suitable parameter; in a simple version, it is a large constant. We construct the intersection polygon $I(S)$ by some straightforward method, say in time $O(r^2)$. Then we triangulate the polygon $I(S)$ by connecting each of its vertices to the origin; see fig. 1. Let $\mathcal{T}(S)$ denote the set of the triangles of the resulting triangulation. For each triangle $\Delta \in \mathcal{T}(S)$, we compute the list H_Δ of lines of H intersecting Δ. The portion of $I(H)$ within each Δ is equal to the portion of $I(H_\Delta)$ within Δ. We may thus construct each $I(H_\Delta)$ recursively by the same method, clip it by Δ and finally glue these pieces together. The recursion stops when the current set of lines is small (smaller than r^2, say); then we may construct the intersection by some inefficient direct method.

What can be said about the running time of this algorithm? We may assume that the construction of $I(S)$ and $\mathcal{T}(S)$ takes time $O(r^2)$. If we test every line against each triangle, the computation of the lists H_Δ needs $O(nr)$ time. This time is also amply sufficient for

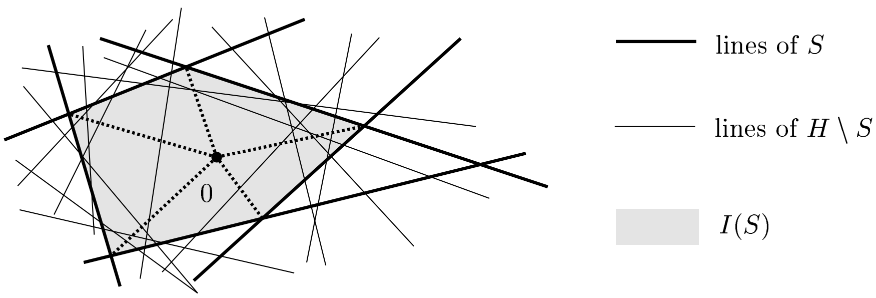


Figure 1: Randomized divide-and-conquer for the intersection of halfplanes.

gluing the polygon $I(H)$ together from the portions of the recursively computed polygons $I(H_\Delta)$.

Let n_Δ denote the size of H_Δ; the key question is how large are the n_Δ's. It turns out that one gains the right feeling about the problem by assuming that each n_Δ is roughly Cn/r, where $C > 1$ is some absolute constant independent of r. If we assume e.g., $n_\Delta \le 2n/r$ for each Δ, we get the recursion

$$T(n) \le O(r^2 + nr) + r\,T(2n/r)\,,$$

where $T(n)$ denotes the running time for n lines. Together with the initial condition $T(n) = O(1)$ for $n \le r^2$ this yields a bound of $T(n) = O(n^{1+\delta})$, where $\delta > 0$ is a constant which tends to 0 as r increases (but the multiplicative constant increases with r). Indeed, the total size of subproblems doubles with each level of the recursion. Hence the total work done at the bottom of the recursion (which turns out to dominate the overall work) will be $n\,2^\ell$, where $\ell \approx \log_r n$ is the depth of the recursion, so $n\,2^\ell \approx n^{1+1/\log_2 r}$, and we have $\delta \approx 1/\log_2 r$.

Unfortunately we cannot assume that $n_\Delta \le Cn/r$ for *each* Δ. First of all, the n_Δ's are determined by the random choice of S, and some choices apparently give much worse values. One could still hope that the bounds are true at least with probability 1/2, say, but even this need not be the case. Valid bounds are the following:

- (*"Pointwise" bound.*) For a randomly chosen S, the following holds with high probability: For all $\Delta \in \mathcal{T}(S)$,
$$n_\Delta \le C\,\frac{n}{r}\,\log r \tag{1}$$
with an absolute constant C (independent of r).

- (*"Higher moments" bound.*) For any constant $c \ge 1$ there exists a constant $C = C(c)$ (independent of r) such that the expectation
$$\mathbf{E}\left[\sum_{\Delta\in\mathcal{T}(S)} n_\Delta^c\right] \le C\left(\frac{n}{r}\right)^c \mathbf{E}\Big[|\mathcal{T}(S)|\Big]\,. \tag{2}$$
(In our case clearly $\mathbf{E}\left[|\mathcal{T}(S)|\right] \le r$.)

The higher moments estimate says that as far as the cth degree average is concerned, the quantities n_Δ behave as if they were $O(n/r)$.

The bound $T(n) = O(n^{1+\delta})$ for the worst-case expected running time can be established using either of these two estimates, only the dependence of δ on r is better for the second one. The difference appears more significantly in other algorithms of this type, where we don't want to choose r just a constant but rather a suitable power of n, say $n^{1/10}$. Then the pointwise (or "trianglewise") bound (1) brings in an extra polylogarithmic factor with each level of recursion, while (2) only a multiplicative constant. Why should a larger value of r help? With larger r, the problem is subdivided into a larger number of subproblems of appropriately smaller size, and thus the depth of the recursion is smaller, sometimes only constant. Since each level of recursion brings a multiplicative "excess" factor into the running time bound, this means that a smaller extra factor accumulates. Using a larger r usually requires that various auxiliary computations are done in a more clever way, however. In our simple example, we cannot increase r to $n^{1/10}$, since then the nr term (coming from a straightforward computation of the lists H_Δ) would already become too large. But if we could compute these lists more cleverly, we could afford to increase r and would get a faster algorithm.

In an algorithm of this type for a more general problem, we usually deal with some collection H of hyperplanes or more general surfaces. In the divide step, we choose a random sample $S \subset H$ of size r and we partition all relevant regions of the arrangement of S into some simple cells like triangles, simplices, trapezoids etc., the important feature is that each such cell Δ can be described by a constant-bounded number of parameters. We let $\mathcal{T}(S)$ denote the set of the resulting cells. Each $\Delta \in \mathcal{T}(S)$ defines one subproblem, dealing with the set H_Δ of surfaces intersecting Δ. These subproblems are solved independently and from their solutions a global solution for H is recovered. This scheme of course cannot include all subtleties in specific applications.

This scheme of algorithm with a constant r usually gives complexity at least by an n^δ factor off from optimal, but it is extremely robust and it has been efficiently derandomized. Playing with larger values of r usually brings improvements (up to optimal algorithms sometimes), but the derandomization becomes more delicate. Let us now move to the second paradigm.

Randomized incremental construction. To solve Problem 2.1 with this approach, we insert the lines one by one in a random order and maintain the current intersection. We start with the whole plane as the current polygon and with every inserted line we cut off an appropriate portion so that when we finish we have the desired intersection.

How do we find which portion should be cut off for the newly inserted line? One seemingly sloppy but in fact quite efficient method is the maintenance of so-called conflict lists. (This method is also easy to generalize for more complicated problems.) Let S_r denote the set of the first r already inserted lines; together with the polygon $I(S_r)$ we also maintain the triangulation $\mathcal{T}(S_r)$ (defined as above) and for each $\Delta \in \mathcal{T}(S_r)$, we store the set H_Δ of lines intersecting it; it is called the *conflict list* of Δ in this context. Finally we maintain the list of intersected triangles for every line not yet inserted. When inserting the $(r+1)$st line we thus know the triangles it intersects and we remove these from the current triangulation (if no triangle is intersected we do nothing); see fig. 2. In order to complete the triangulation $\mathcal{T}(S_{r+1})$ of the new polygon, it suffices to add 3 new triangles.The conflict lists of these new triangles are found by inspecting the conflict lists of the deleted triangles, and then also the conflict lists of lines are updated accordingly. The total update time is proportional to the sum of conflict list sizes for the deleted triangles. Since each triangle must be created before

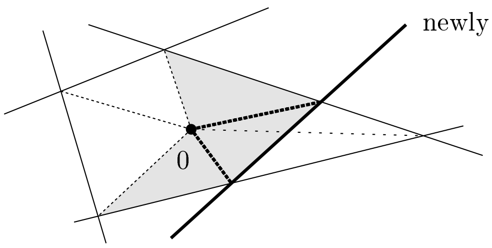


Figure 2: Retriangulating after the insertion of the 5th line.

it can be deleted, the total running time is proportional to the sum of sizes of the conflict lists of all triangles ever created.

The initial intuition for analyzing this quantity is similar to that for the previous paradigm. Namely, the first r inserted lines form a random sample S_r and we expect that each $\Delta \in \mathcal{T}(S_r)$ has a conflict list of size about Cn/r. Therefore, the 3 newly created triangles should contribute about $O(n/r)$ to the running time, and the total should be

$$\sum_{r=1}^{n} O(n/r) = O(n \log n)\,.$$

Using the bound (1), we get only a weaker result $O(n \log^2 n)$. A more sophisticated analysis not relying on this bound shows that the expected contribution of the rth step of the algorithm is indeed only $O(n/r)$ and thus we get an optimal algorithm in this way.

Let us sketch a general scheme of a geometric randomized incremental algorithm. We are given a set H of some objects (hyperplanes, surfaces, but perhaps also points for the case of computing a Voronoi diagram). Our goal is to compute a certain geometric partition $\mathcal{T}(H)$ of space induced by these objects. In order for the analysis to work properly, one needs (similarly as in the divide-and-conquer scheme) that the cells in this partition have a constant description complexity. If the desired partition does not have this property we define a suitable refinement first and work with this refinement. The algorithm inserts the objects in a random order, maintaining the partition $\mathcal{T}(S_r)$ for the set S_r of the already inserted objects. In order to perform the updates quickly, it usually maintains some auxiliary information, such as the conflict lists of the cells of $\mathcal{T}(S_r)$.

The randomized incremental construction has also a vast potential for generalization, and if applicable, it usually gives better and simpler algorithms than the divide-and-conquer approach, mainly because there is no error accumulation phenomenon present. It is much more difficult to derandomize, and only few special results are known. An intuitive explanation for this difference is as follows: The basic form of the divide-and-conquer approach described above requires only a small random sample, for which we can at least easily verify that it has the required properties. On the other hand, the randomized incremental method needs a whole random permutation, and there does not seem to be any easy method of checking whether a given permutation is good other than actually running the considered incremental algorithm with that permutation. Or, put another way, the divide-and-conquer approach creates subproblems and then operates locally on each of them (until a final merging phase, which is usually easy and involves no randomization anymore), while the incremental method always remains on a global level. It seems that in many cases the most useful strategy for derandomization is to merge the local and global approaches: one uses a randomized algorithm of the divide-and-conquer type, but complemented with some global mechanism to prevent the accumulation of excess factors (more concrete examples will be discussed later).

Bibliography and remarks. A survey on randomized algorithms is Karp [Kar91]; a recent book is Motwani and Raghavan [MR95]. Randomized algorithms in computational geometry (mainly incremental ones) are treated extensively in Mulmuley [Mul94]; other good sources are Guibas and Sharir [GS93], Seidel [Sei93], Agarwal [Aga91], Clarkson [Cla92].

As was mentioned above, most algorithms in computational geometry are designed under the assumption that the space dimension is a (small) constant. One problem where the dependence of the running time on the dimension has been studied intensively is the linear programming problem, and here the inadequacy of the current derandomization methods for coping with large dimension stands out clearly: Randomized algorithms of the incremental type with a subexponential dependence on the dimension were found[1] (Kalai [Kal92], Matoušek *et al.* [MSW96]), but the best known deterministic algorithms remain exponential.

The randomized geometric divide-and-conquer strategy appears in the pioneering works of both Clarkson (e.g., [Cla88a]) and of Haussler and Welzl [HW87]. It has been elaborated in numerous subsequent works, mainly concerning computations with arrangements of lines, segments etc., see e.g., Agarwal and Sharir [AS90], Clarkson et al. [CEG+90], Chazelle et al. [CEGS89b].

The bounds of type (1) appeared in Clarkson [Cla87] and Haussler and Welzl [HW87]. The observation that one can get rid of the $\log r$ factor by using the bounds on higher moments is heavily used in Clarkson [Cla88a]. A technically different approach which effectively brings similar results as an application of a bound of type (2) is due to Chazelle and Friedman [CF90]. A more detailed exposition of the randomized geometric divide-and-conquer paradigm with examples can be found in Agarwal's survey paper [Aga91].

A randomized incremental construction is used in Chew's early paper [Che86], together with an elegant technique nowadays called *backwards analysis*. Clarkson and Shor [CS88] give several simple optimal randomized incremental algorithms for problems considered very difficult at that time. Mulmuley [Mul90], [Mul91a], [Mul91b] solves numerous other problems by algorithms of this type; his analysis is relatively complicated (using probabilistic games). The methodology of backwards analysis was elaborated by Seidel [Sei91a], [Sei91b] and it allowed him to give very simple proofs e.g., for Mulmuley's algorithms.

3 General derandomization techniques

Before explaining the specific methods developed for computational geometry algorithms, let us briefly summarize the general approaches to derandomization.

The method of conditional probabilities. This is perhaps the most significant general derandomization method (implicitly used by Erdős and Selfridge [ES73], formulated and popularized by Spencer [Spe87], and further enriched by Raghavan [Rag88]). It is so well known and nicely explained in many places that we allow ourselves to be very brief.

To be specific, consider the probability space Ω of n-component 0/1 vectors, where the entries of a random vector are set to 1 with probability p, the choices being independent

[1] These algorithms work in the infinite precision computation model common in computational geometry, in contrast to known (weakly) polynomial algorithms for linear programming in the bit model.

for distinct entries. Let $\Phi : \Omega \to \mathbb{R}$ be some function (that is, a random variable on Ω), and suppose that our goal is to find a specific vector $X \in \Omega$ such that $\Phi(X) \le \mathbf{E}[\Phi]$. The derandomization of many algorithms can indeed be formulated in this way; the vector X may stand for a sequence of random bits used by the algorithm, and $\Phi(X)$ may be the running time, or the space, or another performance measure of the algorithm (for one fixed input and various values of X).

For the basic version of the method of conditional probabilities, we first define a sequence of functions $F_0()$, $F_1(v_1)$, $F_2(v_1, v_2)$,..., $F_n(v_1, v_2, \dots, v_n)$ by setting

$$F_k(v_1, \dots, v_k) := \mathbf{E}\left[\Phi(X) \,|\, x_1 = v_1, \dots, x_k = v_k\right] .$$

Here $\mathbf{E}$ denotes the conditional expectation operator; in words, $F_k(v_1, \dots, v_k)$ is the expected value of $\Phi(X)$ when the components x_1 through x_k have been fixed to the values $v_1, \dots, v_k$, respectively, and the remaining $n-k$ components are chosen independently at random. So, in particular, $F_0()$ is the expectation of Φ for a random X, and $F_n(v_1, \dots, v_n) = \Phi(v_1, \dots, v_n)$.

The algorithm for finding a vector X with $\Phi(X) \le \mathbf{E}[\Phi]$ fixes the components of X one by one. We inductively assume that the entries $x_1, \dots, x_k$ have already been fixed to specific values $v_1, \dots, v_k$ respectively. To determine x_{k+1}, we compute the two conditional expectations $E_0 = F_{k+1}(v_1, \dots, v_k, 0)$, $E_1 = F_{k+1}(v_1, \dots, v_k, 1)$ and we fix x_{k+1} to 0 if $E_0 \le E_1$ and to 1 otherwise.

The correctness of the method is based on the inequality

$$F_k(v_1, \dots, v_k) \ge \min(F_{k+1}(v_1, \dots, v_k, 0), F_{k+1}(v_1, \dots, v_k, 1)) , \qquad (3)$$

which guarantees that the current conditional expectation never increases and thus that the final conditional expectation $F_n(v_1, \dots, v_n) = \Phi(v_1, \dots, v_n)$ does not exceed $\mathbf{E}[\Phi]$ as desired. The inequality (3) in turn follows from a basic property of conditional expectation, which in our case can be expressed as

$$F_k(v_1, \dots, v_k) = (1-p)\, F_{k+1}(v_1, \dots, v_k, 0) + p\, F_{k+1}(v_1, \dots, v_k, 1) .$$

The potential problem with this method lies in evaluating the required conditional expectations. This may often be hopelessly difficult to do exactly. As observed by Raghavan, instead of exact conditional expectations we can in fact use any sequence of functions $F_0()$,$F_1(v_1)$,$F_2(v_1, v_2)$,..., $F_n(v_1, \dots, v_n)$ as long as they satisfy (3), $F_0() \le \mathbf{E}[\Phi]$, and $F_n(v_1, \dots, v_n) \ge \Phi(v_1, \dots, v_n)$ for any $v_1, \dots, v_n$ (if we can afford to exceed the expectation C times, then the last bound can only hold up to a factor of C). Sometimes one succeeds in choosing such functions which are much easier to evaluate than the actual conditional expectations. This modification is called the *method of pessimistic estimators.*

By a direct application of this method in computational geometry, one can derandomize most algorithms. The running time usually remains polynomial, but without further techniques it increases significantly compared to the randomized case, and so the resulting algorithm is often superseded by a quite naive deterministic algorithm.

Small probability spaces. The method of conditional probabilities can be viewed as a binary search in the original probability space. Another approach for derandomization replaces the probability space by a smaller one, which can either be searched exhaustively or in combination with the method of conditional probabilities.

An extreme case of this second approach is an explicit (and efficient) construction of a single "quasi-random" object, for instance of a graph that enjoys many of the typical properties of a random graph (the most often used such graphs are *expanders*). Such an object is then used in the algorithm instead of an object generated randomly. This approach is quite common in other fields, and there are also few works applying it in computational geometry, although other methods seem more convenient there.

Another powerful set of tools is subsumed under the heading *(approximately) k-wise independent distributions*. Let $X = (x_1, \ldots, x_n)$ be the vector of random bits used by a considered randomized algorithm; for simplicity let us assume that the probabilities of 0 and 1 are equal. True random bits should be mutually independent, but often it suffices for the algorithm to have them *k-wise independent*. This means that for any $i_1, i_2, \ldots, i_k$, $1 \leq i_1 < i_2 < \ldots < i_k \leq n$ and for any $b_1, b_2, \ldots, b_k \in \{0, 1\}$, we have

$$\text{Prob}\,(x_{i_1} = b_1, \ldots, x_{i_k} = b_k) = 2^{-k} \tag{4}$$

(any k positions in X behave as k truly random independent bits). While a full (n-wise) independence requires a probability space consisting of all the 2^n possible n-component 0/1 vectors, for the k-wise independence with $k \ll n$ it may suffice to pick X at random from a much smaller set Ω of n-bit vectors. For example, if we pick a 3-bit vector at random among the vectors 001, 010, 100, and 111, any two positions in this random vector are independent. It turns out that if k is a constant, roughly $n^{k/2}$ vectors suffice and are also necessary. The construction of such probability spaces is nontrivial although not too complicated.

Hence, if the analysis of some algorithm can be made to work assuming k-wise independence for a constant k only, we can derandomize such algorithm by simply executing it for each of the roughly $n^{k/2}$ vectors in the smaller probability space. For at least one vector it must execute satisfactorily. Hence the algorithm has been derandomized with running time increased $O(n^{k/2})$ times. This may look not particularly impressive, but there is an important advantage over the method of conditional probabilities: While the latter one requires a sequential search, the former one can be parallelized easily, simply by executing the randomized algorithm for all vectors from the small probability space in parallel. Hence this method of derandomization essentially preserves any parallelism present in the randomized algorithm.

Some algorithms require a larger degree of independence, say $k = \log n$ or $k = \log^c n$. Then searching all the vectors in a k-wise independent probability space does not yield a polynomial algorithm anymore. In such case, one can sometimes combine the k-wise independence method with a binary search in the probability space. Another important observation is that one often need not insist on satisfying (4) exactly, but can allow an absolute error of ε for a suitable $\varepsilon > 0$ (say $\varepsilon = 2^{-k}/100$). In such case, we speak of (ε, k)-independence. Compared to the exact k-wise independence, the size of the probability space can be significantly reduced. There are relatively simple algebraic constructions of n-bit random (ε, k)-independent vectors on probability spaces of size roughly $k^2 \log^2 n/\varepsilon^2$. These results belong among the most powerful tools for derandomization. For instance, if $k = O(1)$ and $1/\varepsilon = O(\log n)$ suffices to make the algorithm work, then it can be derandomized with a polylogarithmic overhead only by running it on all points of the small probability space.

How to use k-wise independence instead of full independence in algorithm analysis? For instance, suppose that the running time $T = T(X)$ can be expressed as a sum of several terms of the form $f(x_{i_1}, \ldots, x_{i_k})$, where f is some function depending on at most k of the n variables x_i. Then the expected running time $\mathbf{E}[T]$ remains is the same when X is chosen from the fully independent distribution ("usual" random bits) and when X is chosen from any k-wise independent distribution.

There are various more sophisticated ways of applying k-wise independence; let us mention one more useful tool — a *"weak Chernoff inequality"*. Recall that if $S = x_1 + \cdots + x_n$ is the sum of n independent 0/1 random variables, then the probability of S deviating from its expected value $\mathbf{E}[S]$ can be bounded by

$$\text{Prob}\ (|S - \mathbf{E}[S]| > a) < 2e^{-2a^2/n}. \tag{5}$$

If we only assume that the x_i's are k-wise independent, we have a tail estimate decreasing as a k-th power (instead of decreasing exponentially as for the full independence):

$$\text{Prob}\ (|S - \mathbf{E}[S]| > a) < C \left(\frac{k\mathbf{E}[S] + k^2}{a^2} \right)^{k/2}, \tag{6}$$

C an absolute constant.

Another interesting approach is to produce a small probability space tailored for a given randomized algorithm. The idea is to look what constraints are put by the analysis on the joint distribution of the random bits vector X, and then compute a small probability space satisfying these constraints. A potential advantage is that the algorithm may actually make use of much fewer constraints than are imposed by k-wise independence. The computation of a suitable space can be difficult, but for many practically important types of constraints it can be done in polynomial time by linear programming or by other methods.

Bibliography and remarks. Construction of "quasi-random" objects, mainly graphs, is discussed in Alon and Spencer [AS93]. This subject is very wide and has many often unexpected connections; another long survey would be needed to cover just the basics. A recent work in this area is Wigderson and Zuckerman [WZ93], where also more references can be found. Explicit quasi-random graph (expanders) were applied in geometric algorithms by Ajtai and Megiddo [AM92] and by Katz and Sharir [KS93a], [KS93b].

Small k-wise independent probability spaces were constructed by Joffe [Jof74]; an asymptotically optimal size was achieved by Alon et al. [ABI86] (see also [KM94a]). Constructions of (exactly) k-wise independent probability spaces are also covered in [AS93]. A nice survey of applications of 2-wise independence is Luby and Wigderson [LW95]. Bounded independence was used for derandomization e.g., in Karp and Wigderson [KW85]. A combination of bounded independence with a search in the Raghavan-Spencer spirit appears in Luby [Lub93] for 2-wise independence. The possibility of handling $(\log n)$-wise independence or even sometimes $(\log^{const} n)$-wise independence in this way was discovered by Berger and Rompel [BR91] and independently by Motwani et al. [MNN94]. The idea of allowing an approximate independence only and thus reducing the space size is due to Naor and Naor [NN93]; simplified and elaborated constructions of approximately independent probability spaces of 0/1 vectors were given by Alon et al. [AGHP92]. For constructions for more general distributions than uniform 0/1 vectors see e.g., Even et al. [EGL$^+$92] (in general, random variables $x_1, \ldots, x_n$ are called k-wise independent if $\text{Prob}\,(x_{i_1} \in A_1, \ldots, x_{i_k} \in A_k) = \prod_{j=1}^{k} \text{Prob}\,(x_{i_j} \in A_j)$ for any $i_1, \ldots, i_k$, $1 \le i_1 < \ldots < i_k \le n$, and any $A_1, \ldots, A_k$, A_j a measurable subset of the range of the variable x_{i_j}). More general forms of the Chernoff inequality (5) are discussed e.g., in Alon and Spencer [AS93]. Inequality (6) was proved by Rompel [Rom90] (it holds for any k-wise independent random variables $x_1, \ldots, x_n$ with values in $[0, 1]$); see also Schmidt et al. [SSS95]. Generating sample spaces tailored for a given algorithm was started by Koller and Megiddo [KM94b] and further developed by Karger and Koller [KK94]. Our bibliography is by no means complete here; the reader may consult some of the recent papers mentioned here for references to other important works in this area.

The k-wise independence methods were applied by Berger *et al.* [BRS94] and later by Goodrich [Goo93], [Goo96], by Amato et al. [AGR95], and by Mahajan et al. [MRS97] for parallelization of derandomized geometric algorithms. Mulmuley [Mul96] extends earlier work of Karloff and Raghavan [KR93] on limiting random resources using bounded independence distributions and demonstrates that a polylogarithmic number of random bits is sufficient for guaranteeing a good expected performance of many randomized incremental algorithms.

4 Deterministic sampling in range spaces

In this section we discuss methods developed for derandomizing algorithms of the divide-and-conquer type. There were essentially two abstract frameworks for such randomized algorithms proposed in the literature: the *range spaces* of Haussler and Welzl [HW87] and the framework of Clarkson, see [CS89]. We will mainly consider the first framework.

4.1 ε-nets and ε-approximations

We begin by considerations leading to a proof of the bound (1), $n_\Delta \leq C(n/r)\log r$. This bound says that if we draw a sample S of r lines out of n at random, certain specific triangles (dependent on S) will only be intersected by $O((n/r)\log r)$ lines. The first step in the Haussler-Welzl proof is to observe that the relevant triangles are not properly intersected by any line of S. Then this condition is strengthened by requiring that *any* triangle which is not properly intersected by a line of S is only intersected by that few lines of H. The next very fruitful step is to abstract oneself from the specific geometric situations and formulate the problem in terms of set systems. In our case, we have H as a ground set, and we consider all subsets of H of the form H_Δ, where Δ is some triangle and H_Δ are the lines of H properly intersecting it; this defines a set system $(H, \mathcal{R})$. The property of S we want to establish now becomes that with high probability, any set in $\mathcal{R}$ of size at least $C(n/r)\log r$ contains an element of the sample S.

With this motivating example, we can proceed to the general framework. One considers a ground set X and a system $\mathcal{R}$ of its subsets. (The pair $\Sigma = (X, \mathcal{R})$ is sometimes called a *range space* in this context; the sets of $\mathcal{R}$ are called the *ranges* of Σ.) One may consider infinite set systems, such as the set of all points in the plane with all triangles as ranges, or finite ones, such as the set of all subsets of a finite point set X in the plane definable as intersections of X with some triangle. In general, if Y is a subset of X, we denote by $\mathcal{R}|_Y$ the set system *induced by* $\mathcal{R}$ *on* Y, i.e. $\{R \cap Y;\ R \in \mathcal{R}\}$ (let us emphasize that although many sets of $\mathcal{R}$ may intersect Y in the same subset, this intersection only appears once in $\mathcal{R}|_Y$).

A subset $S \subseteq X$ is called an ε*-net for* $(X, \mathcal{R})$ provided that $S \cap R \neq \emptyset$ for every $R \in \mathcal{R}$ with $|R|/|X| > \varepsilon$ (in this definition, $\varepsilon \in [0, 1)$ is a real number, and we assume that X is finite), i.e. S intersects all "large" ranges. For an efficient geometric divide-and-conquer, we are interested in finding small ε-nets: For proving a bound of type (1), we want to show that a random sample of a suitable size forms an ε-net with high probability, for derandomization we would like to compute ε-nets by an efficient deterministic algorithm. In the sequel, it will sometimes be more convenient to write $1/r$ for ε, with $r > 1$.

We cannot expect the existence of small ε-nets for an arbitrary set system. A property guaranteeing small ε-nets and satisfied by many natural geometric set systems is a *bounded VC-dimension*; a related concept is a *polynomially bounded shatter function*.

For a set system $(X, \mathcal{R})$ and a natural number $m \leq |X|$, let $\pi_{\mathcal{R}}(m)$ denote the maximum possible number of sets in a set system of the form $\mathcal{R}|_A$, for an m-point subset $A \subseteq X$. To understand this concept, the reader is invited to check that if $X = \mathbb{R}^2$ is the plane and $\mathcal{H}$ is the system of all (closed) halfplanes, then $\pi_{\mathcal{H}}(m) \leq Cm^2$ for an absolute constant C and for all m. Shatter functions of set systems arising in computational geometry are usually polynomially bounded, and this implies the existence of small ε-nets, as we will see below.

First we introduce one more concept, the *Vapnik-Chervonenkis dimension* (or VC-dimension for short). The VC-dimension of $(X, \mathcal{R})$ can be defined as $\sup\{d;\ \pi_{\mathcal{R}}(d) = 2^d\}$. In other words, it is the maximum d such that there exists a d-element *shattered subset* $A \subseteq X$, i.e. a subset such that each $B \subseteq A$ can be expressed as $B = A \cap R$ for some $R \in \mathcal{R}$. It turns out that the shatter function of a set system of VC-dimension d is bounded by $\binom{m}{0} + \binom{m}{1} + \cdots + \binom{m}{d}$, and this estimate is tight in the worst case (hence the shatter function of any set system is either exponential or polynomially bounded). Conversely, if the shatter function is bounded by a fixed polynomial, then the VC-dimension is bounded by a constant, but there are many natural geometric set systems of VC-dimension d whose shatter function has a considerably smaller order of magnitude than m^d (for instance, the set system defined by halfplanes in the plane has VC-dimension 3, while the shatter function is quadratic).

With these definitions, a basic theorem on existence of ε-nets can be stated as follows.

Theorem 4.1 *For any $d \geq 1$ there exists a $C(d)$ such that for any $r > 1$ and for any set system $(X, \mathcal{R})$ with X finite and of VC-dimension at most d there exists a $(1/r)$-net of size at most $C(d) r \log r$. In fact, a random sample $S \subseteq X$ of this size is a $(1/r)$-net with a positive probability (even with probability whose complement to 1 is exponentially small in r). This size is in general best possible up to the value of $C(d)$.*

This implies, among others, the bound (1) stated in section 2. As the theorem says, the $\log r$ factor in general cannot be removed from the bound. A challenging and probably very difficult open problem is whether some improvement in the $(1/r)$-net size is possible for set systems defined geometrically, such as the one defined by triangles in the plane.

A related notion to the ε-net is an ε-approximation. A subset $A \subseteq X$ is an *ε-approximation* for $(X, \mathcal{R})$, provided that

$$\left| \frac{|A \cap R|}{|A|} - \frac{|R|}{|X|} \right| \leq \varepsilon$$

for every set $R \in \mathcal{R}$. Clearly, an ε-approximation is also an ε-net, but not conversely. It can be shown that for a set system of VC-dimension bounded by a constant d, a random sample A of size $Cr^2 \log r$ with a suitable constant $C = C(d)$ is a $(1/r)$-approximation for $(X, \mathcal{R})$ with high probability. The size of the $(1/r)$-approximation obtained in this way is thus roughly quadratic compared to a $(1/r)$-net.

For the basic derandomization, one needs ε-nets more often than ε-approximations. The ε-approximations, however, have some pleasant properties of algebraic nature not shared by ε-nets which make them more suitable for deterministic computing, and ε-nets are essentially computed via ε-approximations. On the other hand, ε-approximations may be useful in themselves, not only as an auxiliary device for computing ε-nets. In general one can say that performing some geometric construction for a $(1/r)$-approximation amounts to performing the construction approximately for the original point set. Let us illustrate it on a well-known geometric notion — the ham-sandwich cut (more examples come in the next sections). Let $P_1, \ldots, P_d$ be d finite sets in $\mathbb{R}^d$. A hyperplane h is said to *bisect* P_i if each of the open halfspaces defined by h contains at most $\lfloor |P_i|/2 \rfloor$ points of P_i. A *ham-sandwich cut* for $P_1, \ldots, P_d$ is a hyperplane simultaneously bisecting all the P_i's. It is well-known that a ham-sandwich cut always exists, but no reasonably efficient algorithm for finding it is known in higher dimensions. Let A_i be an ε-approximation for P_i with respect to halfspaces[2], with $\varepsilon > 0$ being a small constant (thus also the size of A_i can be bounded by a constant). We can find a ham-sandwich cut h for the A_i's by some brute force method; since the A_i's have a constant-bounded size this is a maybe complicated but constant-time operation. By the definition of ε-approximation, we get that h is also an ε-approximate ham-sandwich cut for the P_i, which means that each P_i is divided in ratio between $1/2 - \varepsilon$ and $1/2 + \varepsilon$. For applications of ham-sandwich cuts, such approximate cuts often suffice, and these can be constructed in linear time (this time is sufficient for finding the A_i; see below).

Bibliography and remarks. Range spaces and ε-nets were introduced by Haussler and Welzl [HW87]. They were inspired by a previous work by Vapnik and Chervonenkis [VC71], which contains the concept of VC-dimension and ε-approximation (under different names) and also a proof of existence for $(1/r)$-approximations of size $O(r^2 \log r)$ for a fixed VC-dimension. The upper bound in Theorem 4.1 was proved by Haussler and Welzl, by a modification of the Vapnik-Chervonenkis method. The best value of $C(d)$ is $d + o(1)$; this was shown by Komlós et al. [KPW92] as well as the lower bound.

While the $O(r \log r)$ size for $(1/r)$-nets is optimal in general, the $O(r^2 \log r)$ bound for $(1/r)$-approximations is not. It was shown by Matoušek et al. [MWW93] that if the shatter function is bounded by $O(m^d)$ for a fixed d, then there exist $(1/r)$-approximations of size $O(r^{2-2/(d+1)}(\log r)^{2-1/(d+1)})$; a similar bound is given there in terms of the so-called dual shatter function. Here also a close relation of ε-approximations to the notion of *discrepancy* is observed, which is a classical object of study in combinatorics, see e.g, [AS93], [BC87].

There are few geometric situations where the existence of $(1/r)$-nets of size $O(r)$ has been established (improving the general $O(r \log r)$ bound), most notably for halfspaces in $\mathbb{R}^3$ [MSW90]. This has a nice algorithmic application in a polytope approximation problem; see Brönniman and Goodrich [BG95].

The VC-dimension and ε-nets are frequently used also in other fields, in particular in statistics (this is where they come from, after all) and in learning theory.

The applicability of ε-approximation to approximate ham-sandwich cuts is noted in Lo et al. [LMS94]; a similar trick can be used also for centerpoints (see e.g., [Ede87] for definition, and Clarkson et al. [CEM+96] for an interesting randomized algorithm for finding approximate centerpoints).

4.2 A deterministic algorithm for ε-approximations

We consider the deterministic computation of a $(1/r)$-net for a set system of a constant-bounded VC-dimension, such as the set system defined by triangles on an n-point set in the plane. If the set system were given to us by listing the elements of each set, then already the input size would be quite large compared to the size of the underlying geometric problem (e.g., triangles may define about n^6 subsets of an n-point set in the plane). In order to get efficient algorithms for computing ε-nets etc., we need to assume some more compact

[2] To be quite precise we should say "for the set system $(P_i, \mathcal{R})$, where the sets in $\mathcal{R}$ are subsets of P_i defined by halfspaces". In the sequel we will mostly use the abbreviated form.

representation. A suitable way is to assume the existence of a *subsystem oracle* for $(X, \mathcal{R})$. This is an algorithm (depending on the specific geometric application) that, given any subset $Y \subseteq X$, lists all sets of $\mathcal{R}|_Y$. Note that the maximum number of such sets is bounded by $\pi_{\mathcal{R}}(|Y|)$. We say that the subsystem oracle is *of dimension at most* d if it lists all sets in time $O(|Y|)^{d+1}$ (the "+1" in the exponent accounts for the fact that each output set is given by a list of elements of size up to $|Y|$). For geometric applications, the construction of the oracle is usually easy and most often it amounts to constructing some arrangement of hyperplanes or surfaces.

A basic result on deterministic computation of $(1/r)$-nets and $(1/r)$-approximations is as follows:

Theorem 4.2 *Let $(X, \mathcal{R})$ be a set system with a subsystem oracle of dimension d, d a constant. Given any $r > 1$, one can compute a $(1/r)$-approximation of size $O(r^2 \log r)$ and a $(1/r)$-net of size $O(r \log r)$ in time $O(nr^c)$, $c = c(d)$ a constant, $n = |X|$.*

In particular, if r is a constant, both the $(1/r)$-net and the $(1/r)$-approximation can be computed in time $O(|X|)$. This in itself suffices for derandomizing a large number of geometric algorithms (where constant-size samples are used) in a quite straightforward manner. Most of the other divide-and-conquer algorithms can be modified to use constant-size samples with some decrease in efficiency, typically of n^δ. Finally, for many problems where one finds mainly randomized incremental algorithms in the literature, it is not too difficult to design divide-and-conquer type solutions, again with a somewhat worse efficiency.

We describe the basic ingredients of an algorithm for deterministic computation of ε-approximations.

Polynomial-time sampling. As a first ingredient, we need a polynomial time algorithm for the following problem: Given an n-element set Y, a system $\mathcal{R}$ of subsets of Y, and a parameter r, we want to compute a $(1/r)$-approximation of size $s = O(r^2 \log r)$. This is much weaker than Theorem 4.2, since we only want time polynomial in r and n, with possibly large exponents. ($\mathcal{R}$ may be given by the list of its sets here.) As we know (by a result of Vapnik and Chervonenkis), a s-element random subset of Y is a $(1/r)$-approximation with a positive probability. This gives a straightforward randomized algorithm for our problem, which can be derandomized by the method of conditional probabilities. This yields the required polynomial time algorithm.

A merge-reduce scheme. The algorithm for Theorem 4.2 is based on the above described polynomial-time sampling subroutine and on the following two easy observations:

Observation 4.3 *Let $X_1, \ldots, X_m \subseteq X$ be disjoint subsets of equal cardinality and let A_i be an ε-approximation of cardinality s for $(X_i, \mathcal{R}|_{X_i})$, $i = 1, 2, \ldots, m$. Then $A_1 \dot\cup \ldots \dot\cup A_m$ is an ε-approximation for the subsystem induced by $\mathcal{R}$ on $X_1 \dot\cup \ldots \dot\cup X_m$.*

Observation 4.4 *Let A be an ε-approximation for $(X, \mathcal{R})$ and let A' be a δ-approximation for $(A, \mathcal{R}|_A)$. Then A' is an $(\varepsilon + \delta)$-approximation for $(X, \mathcal{R})$.*

The algorithm starts by partitioning the point set X into small pieces of equal size (arbitrarily), and then it alternates between steps of two types — merging and reduction.

On the beginning of the ith step, we have a partition Π_i of X into pieces of equal size, and for each such piece P we have an ε_i-approximation of size s_i for $(P, \mathcal{R}|_P)$.

If the ith step is a merging step, we group the pieces of the partition Π_{i-1} into groups by g_i pieces each (g_i a suitable parameter), and we merge the pieces in each group into a single new piece. (The simplest sequential version has $g_i = 2$.) The ε_i-approximation for the merged piece is obtained by simply merging the ε_{i-1}-approximations for the pieces being merged; we thus get $s_i = g_i s_{i-1}$, and by Observation 4.3 we may take $\varepsilon_i = \varepsilon_{i-1}$.

If the ith step is a reduction step, we leave the partition of X unchanged (i.e. $\Pi_i = \Pi_{i-1}$), but we replace each ε_{i-1}-approximation A for a piece P in Π_{i-1} by a smaller ε_i-approximation A'. To this end, we use polynomial-time sampling to obtain a δ_i-approximation A' of size s_i for the set system $(A, \mathcal{R}|_A)$, with δ_i chosen suitably. By Observation 4.4, such an A' is also an ε_i-approximation for the piece P, with $\varepsilon_i = \varepsilon_{i-1} + \delta_i$.

As we can see, this algorithm has a number of parameters which must be adjusted suitably to make everything work. Roughly, one should keep the sizes of the current ε_i-approximations, s_i, sufficiently small — not much larger than the final size of the desired $(1/r)$-approximation; then all the polynomial-time samplings together can be done in $O(nr^{const})$ time. Details and variants can be found in [Mat95a], [CM96], [Goo93], [Sri95].

Bibliography and remarks. A deterministic algorithm for computing ε-nets and ε-approximations as in Theorem 4.2 was given by Matoušek [Mat95a]. Predecessors of this work are [Mat90], Agarwal [Aga90], Chazelle and Friedman [CF90], [Mat91a]. Chazelle and Matoušek [CM96] present a cleaner exposition of the algorithm and give an explicit dependence of the constants on the dimension d. Srivastav [Sri95] gives a somewhat more efficient implementation of the polynomial sampling subroutine.

Brönnimann et al. [BCM93] modify the algorithm to work with so-called sensitive ε-approximations instead of ε-approximations. A subset $A \subseteq X$ is a *sensitive ε-approximation* for $(X, \mathcal{R})$ if

$$\left| \frac{|R|}{|X|} - \frac{|R \cap A|}{|A|} \right| \leq \frac{\varepsilon}{2}\Big(\sqrt{\frac{|R|}{|X|}} + \varepsilon\Big)$$

for every set $R \in \mathcal{R}$ (this definition may appear somewhat technical but it arises naturally by looking at what properties can one expect from a random sample). For a constant-bounded dimension set system, one can show the existence of a sensitive $(1/r)$-approximation of a similar size as for a $(1/r)$-approximation, that is, $O(r^2 \log r)$. A sensitive ε-approximation is at once an ε-approximation and an ε^2-net; this leads to a somewhat more efficient computation of ε-nets. The parameters of the resulting algorithm, with dependence on d made explicit, are as follows: A (sensitive) $(1/r)$-approximation of size $O(dr^2 \log(dr))$ can be computed in time $O(d)^{3d} r^{2d} \log^d(dr)|X|$, and a $(1/r)$-net of size $O(dr \log(dr))$ can be computed in time $O(d)^{3d} r^d \log^d(dr)|X|$.

A parallel implementation of the algorithm of [Mat95a] with a polylogarithmic running time was outlined in [CM96]. More efficient parallel versions were developed by Goodrich, by Srivastav [Sri95], and by Mahajan et al. [MRS97]. Goodrich [Goo93] gave algorithms for the Exclusive Read, Exclusive Write (EREW) PRAM model that compute a $(1/r)$-approximation of size $O(r^{2+\delta})$ in time $O(\log^2 n / \log r)$, resp. of size $O(n^\delta r^2)$ in time $O(\log n)$, both with $O(nr^{const})$ work. For a parallel implementation of polynomial-time sampling, a k-wise independent random choice of the subset is used instead of full independence (k a constant). Then the tail estimate (6) is applied instead of the Chernoff tail estimate, and the size of the sample is taken slightly larger — $r^{2+\delta}$. The merge-reduce scheme is also adjusted somewhat differently from [Mat95a]. Mahajan et al. [MRS97] combine techniques of Nissan [Nis92] and of Karger and Koller [KK94] to improve the parallel version of the polynomial-time sampling step; they can achieve $O(r^2 \log r)$ size of the $(1/r)$-approximation in NC, while the previous approaches only gave size $O(r^{2+\delta})$.

Goodrich [Goo96] considers very fast parallel algorithms (with $(\log\log n)^{const}$ time) in the Concurrent Read, Concurrent Write (CRCW) PRAM model. In this case, polynomial-time sampling is based on k-wise independent sample again, but it presents new difficulties, since e.g., one cannot check exactly whether a given sample is a $(1/r)$-approximation. The following weakening of an ε-approximation is introduced: a set $A \subseteq X$ is a *δ-relative ε-approximation* for $(X, \mathcal{R})$, if

$$\left| \frac{|R|}{|X|} - \frac{|R \cap A|}{|A|} \right| \leq \delta \frac{|R|}{|X|} + \varepsilon$$

for all $R \in \mathcal{R}$. Using this tool, [Goo96] shows that a $(1/r)$-net of size $O(r^{1+\delta})$ and a $(1/4)$-relative $(1/r)$-approximation of size $O(r^{2+\delta})$ can be computed in $O((\log\log n)^3)$ time using $O(nr^{const})$ work (this disregards the time needed for the subsystem oracle, whose implementation with a comparable speed may be quite nontrivial).

The use of the linear-time construction of ε-nets with a constant ε for derandomizing divide-and-conquer type algorithms is often quite routine and in many works it is just a small remark.

4.3 ε-approximations via geometric partitions

We describe an alternative way for computing ε-approximations in geometrically defined set systems. These ε-approximations are larger than the ones guaranteed by Theorem 4.2 but can be computed much faster.

Let P be an n-point set in $\mathbb{R}^d$ and let $\Pi = (P_1, P_2, \ldots, P_m)$ be a partition of P into disjoint sets (called *classes* of Π). For simplicity let us assume that n is divisible by m and that all the classes P_i have equal cardinality, and also that the set P is in general position. For a hyperplane h, let us say that *h crosses P_i* if it intersects the convex hull of P_i. We let the *crossing number* of the partition Π be the maximum number of its classes which can be simultaneously crossed by a hyperplane. Our main device here are partitions with crossing number substantially smaller than m.

Suppose that we have a partition Π with crossing number $\kappa << m$. Let us form an m-point set $A \subset P$ by selecting one point from each class (it does not matter which one). We claim that such an A is an ε-approximation for P with respect to simplices, where $\varepsilon = (d+1)\kappa/m$. To see this, consider a simplex σ, and call a class P_i *homogeneous* if its convex hull lies completely outside σ or completely inside σ. Clearly the number of points of the homogeneous classes inside σ is precisely reflected by the number of their representatives in A inside σ; an error may arise for classes that are not homogeneous (see fig. 3). The number of such classes is at most $\kappa(d+1)$, since a non-homogeneous class must be crossed by at least one of the $d+1$ hyperplanes bounding the simplex σ. Therefore the error with which $|A \cap \sigma|/|A|$ approximates $|P \cap \sigma|/|P|$ is at most $\kappa(d+1)/m$ as claimed.

Partitions with small crossing numbers are of great interest in another area of computational geometry, so-called *range searching problems*. One of the products of the research in that area is the following result:

Theorem 4.5 *Let P be an n-point set in $\mathbb{R}^d$ ($d \geq 2$), let s be an integer parameter, $2 \leq s < n$, and set $r = n/s$. There exists a partition $\Pi = (P_1, P_2, \ldots, P_m)$ of P whose classes satisfy $s \leq |P_i| < 2s$ (thus $m = \Theta(r)$) and whose crossing number is $O(r^{1-1/d})$. This bound is asymptotically tight in the worst case. If $s > n^\delta$ for any positive constant δ, then such a*

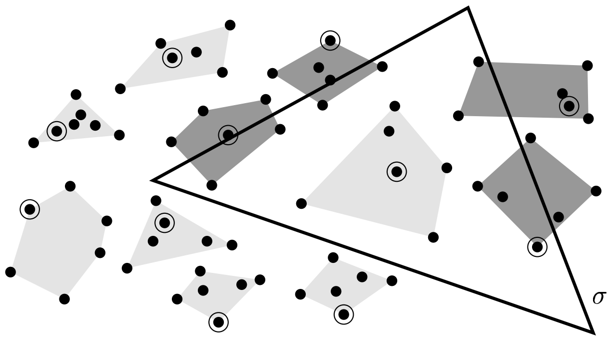


Figure 3: Constructing $(1/r)$-approximations via partitions with low crossing number.

partition can be computed deterministically in $O(n \log r)$ time. For smaller values of s an $O(n^{1+\varepsilon})$ algorithm exists, for an arbitrarily small constant $\varepsilon > 0$.

This result does not give exactly equal-size classes, but this is only a minor technical problem and one can use such partitions for computing ε-approximations similarly as indicated above. Here we get $\varepsilon \approx r^{1-1/d}/r = r^{-1/d}$; in other words, if we want a $(1/t)$-approximation, we need to choose $r \approx t^d$, and this is also the size of the resulting approximation.

Hence we have

Corollary 4.6 *Given an n-point set $P \subset \mathbb{R}^d$ and a parameter r, $r < n^{1/d-\delta}$ for some positive constant δ, a $(1/r)$-approximation of size $O(r^d)$ for P with respect to simplices can be computed deterministically in $O(n \log r)$ time.*

This result can be combined with Theorem 4.2 to reduce the size of the $(1/r)$-approximation. Namely, we first compute a $(1/2r)$-approximation A_1 using Corollary 4.6, and then we compute a $(1/2r)$-approximation A_2 for A_1 using Theorem 4.2. This A_2 is easily seen to be a $(1/r)$-approximation for the original set P, and it has size $O(r^2 \log r)$ only.

Linearization. Can Corollary 4.6 help if we want to compute an ε-approximation for other set systems than those defined by simplices on point sets in $\mathbb{R}^d$? No general result of this type encompassing all set systems of bounded VC-dimension is known, but the above result can be applied in many other geometric situations.

As a basic and well-known example (see [Ede87]), let us consider the set system defined by circular disks in the plane. A disk $C = C(a, b, r) \subset \mathbb{R}^2$ with center (a, b) and radius r is described by the inequality $(x-a)^2 + (y-b)^2 \leq r^2$. We define a mapping $\varphi : \mathbb{R}^2 \to \mathbb{R}^3$ by $\varphi(x, y) = (x, y, x^2 + y^2)$. This mapping maps the plane onto the unit paraboloid $z = x^2 + y^2$. The property of this mapping important for us is that for any disk $C(a, b, r) \subset \mathbb{R}^2$, there exists a halfspace $H = H(a, b, r)$ in $\mathbb{R}^3$ such that the points of the plane mapped into the halfspace H by φ are exactly those of the disk C, or $C = \varphi^{-1}(H)$. It is easily seen that a suitable halfspace H is $\{(x, y, z) \in \mathbb{R}^3;\ 2ax + 2by - z + r^2 - a^2 - b^2 \geq 0\}$. Geometrically this means that the image of any circle can be cut off from the unit paraboloid by a halfspace.

Given a point set $P \subset \mathbb{R}^2$, consider its image $\varphi(P) \subset \mathbb{R}^3$ and let $S \subset \varphi(P)$ be an ε-net for the point set $\varphi(P)$ with respect to halfspaces. By the above, the preimage $\varphi^{-1}(S)$ is an ε-net for P with respect to disks. On the abstract level, what happens is that φ embeds the set system $(X, \mathcal{R})$ of our interest (in our case the one with disk ranges in the plane) into a set system defined by halfspaces in higher dimension, in such a way that each $R \in \mathcal{R}$ is a preimage of a halfspace. Such a mapping φ is called a *linearization* of $(X, \mathcal{R})$. It turns out that for any set system of the form $(\mathbb{R}^d, \mathcal{R})$, where each $R \in \mathcal{R}$ is defined by a single polynomial inequality of degree at most D, there exists a linearization into some $\mathbb{R}^k$. In fact, there is a very easy construction of such a linearization (not necessarily yielding the smallest possible dimension of the image space) — the idea is to replace each monomial in the defining inequality by one new coordinate in the higher dimensional space into which we are embedding. Sets defined by a conjunction of several polynomial inequalities can similarly be embedded into set systems defined by intersections of several halfspaces in a higher dimension. This implies that ε-nets and ε-approximations for set systems with sets defined by a constant number of bounded degree inequalities can be computed using the techniques for simplices.

Bibliography and remarks. Literature on geometric range searching is extensive; recent survey papers are Agarwal and Erickson [AE97] and Matoušek [Mat95b]. The original idea for producing geometric partitions is due to Willard [Wil82]. Theorem 4.5 was proved by Matoušek [Mat92a] by generalizing the ideas of Chazelle and Welzl [CW89]. Interestingly, the $s = 2$ case has a version for general set systems, while for $s = n/O(1)$ a similar result cannot be expected to hold in such a general setting, as an example by Alon et al. [AHW87] shows.

Partitions with crossing number only slightly worse than in Theorem 4.5 can be computed in parallel quickly, and this gives also a fast parallel algorithm for ε-approximations (Goodrich [Goo93]).

The observation that partition trees can produce ε-approximations in geometric situations quickly is also in [Mat92a], together with some applications; more applications can be found in Chazelle and Matoušek [CM94], Matoušek and Schwarzkopf [MS96], and Amato et al.[AGR94],[AGR95].

The linearization produced by assigning a new coordinate to each monomial is well known in algebraic geometry (the so-called *Veronese map*, see e.g., [Har92]). It has been used by Yao and Yao [YY85] to show that various geometrically defined set systems can be embedded into the set systems defined by simplices on point sets in some $\mathbb{R}^k$; see also [AM94] for more information. Linearization has been used for derandomization in an essential way in [MS96], [AGR94], and [AGR95].

4.4 Higher moment bounds and seminets

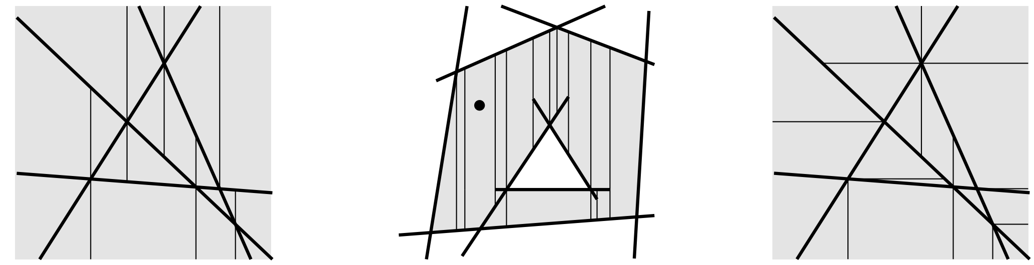
First we give three examples for a future reference.

Example A. Let X be a set of lines in the plane. For any subset $S \subseteq X$, define $\mathcal{T}(S)$ as the set of all trapezoids in the *vertical decomposition* of the arrangement[3] of S.

Example B. Let X be a set of line segments in the plane. For any subset $S \subseteq X$, define $\mathcal{T}(S)$ as the set of all trapezoids in the vertical decomposition of the cell containing the origin of coordinates in the arrangement of S.

[3] The vertical decomposition is obtained by extending a vertical segment from each vertex of the arrangement in both directions until it hits another line of S.

Example C. Let X and S be as in Example A. Decompose a cell of the arrangement of S vertically if it has an even number of edges and horizontally if it has an odd number of edges; let $\mathcal{T}(S)$ be the set of trapezoids in the resulting decomposition.



A B C

Figure 4: $\mathcal{T}(S)$ in Examples A–C.

Configuration spaces. Now we present an abstract framework, which is somewhat more complicated than range spaces but allows one to prove higher moment bounds of type (2). Formally, a *configuration space*[4] can be defined as a 4-tuple $(X, \mathcal{T}, D, I)$. Here X is a finite set whose elements are called the *objects* (we put $n = |X|$). $\mathcal{T}$ is a mapping assigning to each $S \subseteq X$ a set $\mathcal{T}(S)$; the elements of $\mathcal{T}(S)$ are usually called *regions*. In applications, X is typically a set of hyperplanes or surfaces in some $\mathbb{R}^d$ and $\mathcal{T}(S)$ is some kind of decomposition of the arrangement (or its part) of objects from S into "simple" cells. Let $Reg = \bigcup_{S\subseteq X} \mathcal{T}(S)$ be the set of all regions occurring in decompositions for subsets of X. D is a mapping assigning to each region $\Delta \in Reg$ a set $D(\Delta) \subseteq X$. The set $D(\Delta)$ is called the *defining set* (or the set of *triggers*) of Δ. Finally I is another map assigning to each region $\Delta \in Reg$ a set $I(\Delta) \subseteq X$, called the *objects incident to* Δ (or *stoppers* or *objects killing* Δ).

Examples A,B,C readily supply configuration spaces. In all of them, X and $\mathcal{T}$ have been described (so Reg is a set of trapezoids), $I(\Delta)$ is the set of objects of X intersecting the interior of Δ, and $D(\Delta)$ is the set of lines or segments defining the trapezoid Δ in a suitable geometric sense: e.g., for Example A, $D(\Delta)$ typically consists of the two lines containing the top and bottom sides of Δ, and two more lines incident to the two vertices of the arrangement from which the vertical sides of Δ emanate. (A problem occurs if e.g., the lines in Example A are not in general position; then a trapezoid may have several minimal defining sets. This can be resolved by formally taking an appropriate number of copies of such a trapezoid, one for each minimal defining set.)

There are several axioms we may put on a configuration space. We list them first and then comment on them.

0. The number $d = \max\{|D(\Delta)|; \Delta \in Reg\}$, called the *dimension* of the configuration space, is a constant. Moreover, for any $S \subseteq X$ with $|S| \leq d$, the number of regions in $\mathcal{T}(S)$ is bounded by a constant, C.

[4]The terminology and axiomatics is far from being stabilized in the literature. We took the name "configuration space" from [Mul94] but the axiomatics we present is essentially one from [AMS94]. Similar objects are also called *multi-hypergraphs with bounded vertex dependency* [CF90], $\mathcal{T}$*-generated set systems* [AGR94], or in some papers they remain nameless.

1. For any $\Delta \in \mathcal{T}(S)$, $D(\Delta) \subseteq S$ and $S \cap I(\Delta) = \emptyset$.

2. If $\Delta \in \mathcal{T}(S)$ and S' is a subset of S with $D(\Delta) \subseteq S'$, then $\Delta \in \mathcal{T}(S')$.

$2'$. If $D(\Delta) \subseteq S$ and $I(\Delta) \cap S = \emptyset$, then $\Delta \in \mathcal{T}(S)$.

3. For $r \in \{0, 1, \ldots, n\}$, let $f(r)$ be the expected number of regions in $\mathcal{T}(S)$ for S being a random r-element subset of X. We require that f is nondecreasing and that it never grows too fast, say $f(2r) \leq C_1 f(r)$ for some constant C_1.

Axioms 0 and 1 are very natural and they are always imposed. Axiom 3 is a little technical but it is naturally satisfied in most applications. The axioms to look at more carefully are 2 and $2'$. Axiom $2'$ essentially says that the decomposition is defined locally, i.e. Δ occurs iff all its defining objects occur and none of the incident ones does. The weaker axiom 2, a kind of monotonicity requirement, is perhaps best remembered as follows: If Δ occurs for some S, we cannot destroy it by removing objects from S unless we remove some of the defining objects of Δ. The reader is invited to check that in Example A axiom $2'$ (thus also 2) holds, in B axiom 2 holds but $2'$ doesn't, and in the rather artificial Example C none of $2, 2'$ holds.

Sampling in configuration spaces. For any configuration space as above, we may consider a set system $\mathcal{I}$ on X given by $\mathcal{I} = \{I(\Delta);\ \Delta \in \mathit{Reg}\}$. It is not difficult to check that under axioms $0, 1$, and 2, the shatter function $\pi_{\mathcal{I}}$ is polynomially bounded (and thus the VC-dimension is finite). Hence the ε-net theorem 4.1 implies that for a random r-element sample $S \subseteq X$ we have, with high probability, $|I(\Delta)| \leq \mathit{const.}(n/r)\log r$ for all $\Delta \in \mathcal{T}(S)$. There may be some regions Δ in $\mathcal{T}(S)$ for which $|I(\Delta)|$ exceeds n/r substantially, but typically they are few. The following theorem expresses this quantitatively in two ways.

Theorem 4.7 *Let $(X, \mathcal{T}, D, I)$ be a configuration space satisfying axioms 0,1,2, and 3, and let S be a r-element random subset of X.*

(i) *Let t be a parameter, $1 \leq t \leq r/d$, and let $\mathcal{T}_t(S)$ stand for the set of all regions in $\mathcal{T}(S)$ with $|I(\Delta)| \geq t(n/r)$. Then*

$$\mathbf{E}\Big[|\mathcal{T}_t(S)|\Big] \leq O(2^{-t})\mathbf{E}\Big[|\mathcal{T}(S)|\Big] .$$

(ii) *For any constant c,*

$$\mathbf{E}\Big[\sum_{\Delta \in \mathcal{T}(S)} |I(\Delta)|^c\Big] = O((n/r)^c)\mathbf{E}\Big[|\mathcal{T}(S)|\Big] .$$

Results of this type have been applied in a number of randomized algorithms (both incremental and divide-and-conquer), and for derandomization it is important to compute suitable samples S deterministically. Following the terminology of Amato et al. [AGR94], let us call a set $S \subseteq X$ a $(1/r)$-*seminet of order* c for the considered configuration space if

$$\sum_{\Delta \in \mathcal{T}(S)} |I(\Delta)|^c \leq C_0 (n/r)^c f(r) ,$$

where c, C_0 are positive constants[5] and $f(r)$ is as in Axiom 3 (or, if the expectation $\mathbf{E}[|\mathcal{T}(S)|]$ is difficult to determine, $f(r)$ may be a suitable upper bound on this expectation). Hence

[5]Hence, to be quite precise, we should say something like "$(1/r)$-seminet of order c with factor C_0".

Theorem 4.7(ii) can be re-stated by saying that a r-element random sample is a $(1/r)$-seminet of any fixed order c (with a suitable $C_0 = C_0(c)$) with high probability.

Under the stronger Axiom $2'$, a $(1/r)$-seminet for a given configuration space can be computed in time polynomial in n and r (that is, if the mappings $\mathcal{T}$, D and I are polynomially computable). A straightforward application of the method of conditional probabilities works here. On the other hand, currently it is not clear if a polynomial-time derandomization is possible under Axiom 2 only.

To make the computation (in the situation Axiom $2'$ holds) faster, one may use the following observation: if A is a $(1/r)$-approximation for the set system $\mathcal{I}$ induced by the considered configuration space and $S \subseteq A$ is a $(1/r)$-seminet of order c for the configuration space induced by A (in an obvious sense), then S is a $(1/r)$-seminet of order c for the original configuration space (with a somewhat worse C_0). Hence, a $(1/r)$-seminet can be computed in time $T_A(n,r) + r^{const}$, where $T_A(n,r)$ is time needed to compute a $(1/r)$-approximation for $(X, \mathcal{I})$ of size polynomial in r.

This trick with ε-approximation we used here is also useful in other situations; we could call it *sampling from an ε-approximation*. A general scheme is the following: Given a set X of objects, we want to compute a sample S with some properties more complicated than, say, being an ε-net, so that no fast deterministic algorithm is available directly. We compute an ε-approximation A for some appropriate set system on X with a suitable ε quickly and compute a good sample S with respect to A; we can use a slow method (e.g., conditional probabilities in raw form) since A is small. Then, for many properties of the sample, S will be also good for X, perhaps only with slightly worse constants.

Bibliography and remarks. The described framework goes back to Clarkson [Cla88a] (a journal version is [CS89]), who essentially proved Theorem 4.7(ii) under Axiom $2'$ instead of 2. Theorem 4.7(i) under Axiom $2'$ is due to Chazelle and Friedman [CF90]. Theorem 4.7(ii) in our formulation goes back to Chazelle et al.[CEG+93] who formulate their results in a "dynamic" form, as a time bound for a randomized incremental algorithm (see also De Berg et al. [dBDS95]); part (i) appears in Agarwal et al. [AMS94]. The idea of sampling from an ε-approximation is implicit in [Mat91a], and it has been used in several other papers ([CM94], [MS96], [AGR94], [AGR95]). The notion of a $(1/r)$-seminet was introduced by Amato et al. [AGR94], inspired by a paper of Chazelle [Cha93b].

As was observed by Ramos [Ram96], the problem with computing a $(1/r)$-seminet in polynomial time under Axiom 2 (without "locality") can be often circumvented in specific applications. For instance, for computing a single face in an arrangement of segments (this is one of the problems considered in [AGR95], where the derandomization described in the proceedings version of that paper doesn't quite work), one need not work with $\mathcal{T}(S)$ as in Example B above. Instead, the computation can be done with $\mathcal{T}'(S)$, defined as the set of all trapezoids of $\mathcal{T}(S)$ *that intersect the face in the arrangement of X containing in the origin* . For this new $\mathcal{T}'$, the "locality" axiom $2'$ holds. This may seem a strange remedy since the face in the arrangement of X is what we ultimately want to compute, but here sampling from an ε-approximation comes to rescue — we sample in the situation X is a suitable $(1/r)$-approximation to the original set of segments.

5 Derandomization of basic computational geometry algorithms

5.1 Cuttings

Let us return to the geometric divide-and-conquer, and suppose that we deal with a collection H of n hyperplanes in $\mathbb{R}^d$. Similarly to our introductory example in section 2, we want to partition the problem involving H into smaller subproblems, but suppose that this time we are interested in the whole arrangement of H (not only in its cell containing the origin as in Problem 2.1).

The basic strategy dictates that we choose a random sample $S \subset H$ of a suitably selected size r and partition the cells of the arrangement of S into some constant complexity subcells. In this situation, a suitable way is for instance the so-called *bottom-vertex* (or *canonical*) *triangulation*[6]. This yields a partition $\mathcal{T}(S)$ consisting of $O(r^d)$ simplices. Each simplex $\Delta \in \mathcal{T}(S)$ defines one subproblem, in which only the hyperplanes of H_Δ appear.

The set system with H as a point set and with sets defined by simplices has a bounded VC-dimension, and thus by Theorem 4.1, a random sample S is an ε-net with high probability, where $\varepsilon = \text{const.} \log r/r$. This in turn implies that the bound (1) holds; we are thus subdividing into $O(r^d)$ subproblems of size $O((n/r)\log r)$ each.

The efficiency of the resulting algorithm depends on the number of levels of recursion and on the factor we lose at each level; we thus want to obtain small subproblems using as few simplices as possible. What result can we expect at best? The arrangement of H has $\approx n^d$ vertices, while a simplex Δ intersected by $n_\Delta = |H_\Delta|$ hyperplanes may only contain $\approx n_\Delta^d$ vertices[7], hence for a partition with $O(r^d)$ simplices, the average n_Δ has to be at least of the order n/r. We thus see that the construction using ε-nets is not bad; it turns out, however, that one can do a little better and match the lower bound up to a constant factor.

First we state the desired property of the partition as a definition. A *cutting* is a finite set of closed simplices[8] covering $\mathbb{R}^d$. In known constructions the simplices have disjoint interiors. Let H be a set of n hyperplanes in $\mathbb{R}^d$ and $\varepsilon > 0$ a parameter. A cutting Ξ is called an *ε-cutting for H* provided that $|H_\Delta| \le \varepsilon n$ for every $\Delta \in \Xi$, that is, the interior of no simplex of Ξ is intersected by more than εn hyperplanes of H.

The above considerations can be restated by saying that a $(1/r)$-cutting for H must have $\Omega(r^d)$ simplices, and that $O((r\log r)^d)$ simplices can actually be achieved. The promised stronger result is

Theorem 5.1 *For any collection H of n hyperplanes in $\mathbb{R}^d$ and a parameter r, $1 < r \le n$, a $(1/r)$-cutting consisting of $O(r^d)$ simplices exists, and it can be computed deterministically in $O(nr^{d-1})$ time, together with the list of hyperplanes intersecting each simplex of the cutting. The algorithm can be implemented in parallel on an EREW PRAM, with $O(\log n)$ running time and $O(nr^{d-1})$ work. For certain fixed $\alpha = \alpha(d) > 0$ and $r < n^\alpha$, a $(1/r)$-cutting of size $O(r^d)$ can be computed in $O(n \log r)$ time.*

[6] For a convex polygon, a bottom vertex triangulation means drawing all diagonals incident to the bottom vertex (the one with lexicographically smallest coordinate vector). For convex polytopes of higher dimension, it is defined inductively, by first bottom-vertex triangulating all lower-dimensional faces and then lifting each simplex in these triangulations towards the bottommost vertex of the polytope.

[7] There may be also some vertices on simplex boundaries, but it turns out that these can't play an important role.

[8] Here a simplex means an intersection of $d+1$ halfspaces, hence also "simplices" going to infinity are allowed.

We have already remarked that the $O(r^d)$ size is asymptotically optimal. The (sequential) running time $O(nr^{d-1})$ is also optimal in the following sense: if we want also the lists H_Δ to be output, then already the output size is of order nr^{d-1}, as the lower bound argument shows.

Cuttings via seminets. One way of constructing a $(1/r)$-cutting of size $O(r^d)$ is as follows: Take a random subset $S \subseteq H$ of size r, and let $\mathcal{T}(S)$ be the bottom-vertex triangulation of its arrangement. Consider a simplex $\Delta \in \mathcal{T}(S)$, and put $t_\Delta = |H_\Delta|(r/n)$ (the *excess* of Δ, i.e. how many times does $|H_\Delta|$ exceed the "right" value n/r). If $t_\Delta \le 1$, leave Δ untouched; otherwise construct some $(1/t_\Delta)$-cutting Ξ_Δ for the collection H_Δ of hyperplanes. Intersect all simplices of Ξ_Δ with the simplex Δ. Discard all intersections that are empty, and triangulate the intersections that are not simplices. Denote by $\tilde{\Xi}_\Delta$ the resulting collection of simplices. The promised $(1/r)$-cutting is the union of all $\tilde{\Xi}_\Delta$ over $\Delta \in \Xi$.

Using the ε-net argument as above, we can guarantee that the size of each Ξ_Δ is $O(t_\Delta^d \log^d t_\Delta) = O(t_\Delta^{d+1})$. Then we get that the total number of simplices in Ξ is bounded by

$$\sum_{\Delta \in \mathcal{T}(S)} O(t_\Delta^{d+1}),$$

and the expectation of this sum is $O(r^d)$ by Theorem 4.7(ii) (here it is important that the triangulation $\mathcal{T}(S)$ is not arbitrary but the bottom-vertex one — from this the validity of Axiom 2$'$ can be derived). This provides a simple $O(nr^{d-1})$ randomized algorithm, which can be derandomized in polynomial time. Using sampling from an ε-approximation, one can get an $O(nr^{d-1})$ deterministic algorithm by this approach if r is not too large, namely if $r < n^{1-\delta}$ for some fixed $\delta > 0$ (see [Mat95a]). The part of Theorem 5.1 with $O(n \log r)$ time for $r < n^\alpha$ follows by using Corollary 4.6 [Mat92a]. An $O(nr^{d-1})$ deterministic algorithm for all r seems to require a different approach, which we sketch next.

Optimal cuttings via vertex accounting. Supposing that we have a way of efficiently computing good $(1/r_0)$-cuttings for some (say constant) r_0, we can compute $(1/r_0^k)$-cuttings for $k = 1, 2, \ldots$ by repeatedly refining the cutting. This is similar to the procedure described before, but there the refinement was made only once. If this approach is applied directly, the quality of the cuttings deteriorates with each level of refinement. A way to control the size of the refined cuttings is to keep track of the number of vertices of the arrangement of H within the simplices of the current cutting.

We introduce one piece of notation: For a full-dimensional simplex Δ and a collection H of hyperplanes, let $V(H, \Delta)$ denote the set of vertices of the arrangement of H within Δ.

Let Δ be a simplex with $|H_\Delta| = m$. The whole arrangement of H_Δ has $\binom{m}{d}$ vertices. The basic idea is that if only a small fraction of these vertices is contained within Δ then we can compute a partition $\tilde{\Xi}_\Delta$ of Δ into simplices such that each simplex of this partition is intersected only by at most m/r_0 hyperplanes of H_Δ, and at the same time $|\tilde{\Xi}_\Delta|$ is substantially smaller than r_0^d.

To see why this should be so consider the case when Δ contains *no* vertices defined by H_Δ at all. Then the hyperplanes of H_Δ form "tunnels" penetrating the whole Δ from one facet to another, and the combinatorial complexity of their arrangement within Δ is only $O(m^{d-1})$; essentially we are dealing with a $(d-1)$-dimensional problem (these considerations make no claim at exactness).

The requirement that H_Δ defines no vertices inside Δ is too strong (and computationally difficult even to check). Rather we distinguish two cases in the refinement procedure:

(i) (*"Rich" simplex*)

$$|V(H_\Delta, \Delta)| \geq \delta \binom{m}{d},$$

where $\delta > 0$ is a suitable small constant, considerably smaller than $1/r_0$. Then we compute a $(1/r_0)$-cutting Ξ_Δ of size $Cr_0^d \log^d r_0$ for H_Δ by the ε-net method and form the collection $\tilde{\Xi}_\Delta$ as in the basic refinement method.

(ii) (*"Poor" simplex*)

$$|V(H_\Delta, \Delta)| < \delta \binom{m}{d}.$$

Then we choose a random sample $S_\Delta \subset H_\Delta$ of size $Cr_0 \log r_0$ and we triangulate the portion of its arrangement within Δ, forming a collection $\tilde{\Xi}_\Delta$. We check two conditions:

(a) $|\tilde{\Xi}_\Delta| \leq r_0^d/2$, and

(b) each simplex of $\tilde{\Xi}_\Delta$ is intersected by at most m/r_0 hyperplanes of H_Δ.

We repeat the random choice of S_Δ until both these conditions are satisfied.

There are several things to be clarified. First, why should a random sample S_Δ in the "poor" case satisfy (a) and (b) with a positive probability? As for (b), this follows from the general ε-net theorem. Concerning (a), one observes that for a vertex $v \in V(H_\Delta, \Delta)$, the probability that it becomes a vertex of $V(S_\Delta, \Delta)$ is approximately $|S_\Delta|^d/m^d$ and thus

$$\mathbf{E}\Big[|V(S_\Delta, \Delta)|\Big] \approx \frac{|S_\Delta|^d}{m^d}\, \delta \binom{m}{d} < \delta (Cr_0 \log r_0)^d.$$

Geometric considerations show that the number of simplices formed by triangulating the arrangement of S_Δ within Δ is proportional to $|S_\Delta|^{d-1} + |V(S_\Delta, \Delta)|$; so if we set δ small enough then the expected number of these simplices will be smaller than, say, $r_0^d/4$ and so (a) holds with probability at least $1/2$.

If we apply this procedure and keep refining the cutting, we obtain a tree-like hierarchy of simplices. We want to bound the total number of simplices produced. The basic intuition is that the poor simplices have few children, and the rich simplices may have many children but they must distribute their wealth among them and so not too many of their children may become rich. More precisely, suppose that we stop building the tree at the level where the simplices are intersected by at most n/r hyperplanes, thus forming a $(1/r)$-cutting. Let us consider a rich simplex Δ intersected by m hyperplanes. Such Δ has at most $j = \lfloor \log_{r_0}(mr/n) \rfloor$ levels of refinement below it. We charge the simplex Δ plus all poor simplices for which Δ is the nearest rich ancestor to the vertices contained in Δ, whose number is $\Omega(m^d)$. The number of poor simplices charged in this way is at most $r_0^d/2 + (r_0^d/2)^2 + \cdots + (r_0^d/2)^j = O((mr/n)^d/2^j)$, thus a single vertex inside Δ is charged $O((r/n)^d/2^j)$ by Δ and its poor successors. From this we find that the total charge made to any single vertex in the arrangement is $O((r/n)^d)$, hence the overall number of all simplices is $O(r^d)$.

For a derandomization of this method one needs to distinguish the poor and rich cases and find the samples S_Δ deterministically. Both these problems can be resolved using ε-approximations. The following result of independent interest is used:

Lemma 5.2 *Let H be a collection of n hyperplanes intersecting a d-dimensional simplex Δ, and let A be an ε-approximation for H with respect to segments. Then*

$$\left|\frac{|V(H,\Delta)|}{|H|^d} - \frac{|V(A,\Delta)|}{|A|^d}\right| \leq \varepsilon\,.$$

The strategy for finding S_Δ is a sampling from an ε-approximation: First we find A_Δ, a $(1/t)$-approximation for H_Δ with respect to simplices, where t is a constant, $t >> 1/\delta$. We count the vertices of $V(A_\Delta,\Delta)$, and by Lemma 5.2 this tells us (with relative error $1/t$, which is insignificant) if we deal with the rich case or the poor case. In the poor case, we choose S_Δ as a sample *from* A_Δ; we require the condition (a) plus the condition

(b′) S_Δ is a $(1/2r_0)$-net for A_Δ (with respect to simplices).

It is easy to see that such an S_Δ will also satisfy (b). The existence of such S_Δ follows as before by a probabilistic proof, and a particular S_Δ can be found by trying all possibilities (remember that $|A_\Delta| = O(1)$). Alternatively we may compute it in time polynomial in $|A_\Delta|$ by the method of conditional probabilities.

Bibliography and remarks. The ideas leading to the construction of cuttings via ε-nets appeared in the early works of Clarkson [Cla88a] and Haussler and Welzl [HW87]. The bottom-vertex triangulation is from Clarkson [Cla88b]. The existence of asymptotically optimal cuttings was first established by Chazelle and Friedman [CF90] by the method with seminets; derandomization of their method was elaborated in [Mat91a],[Mat95a]. In the plane, a deterministic construction of a $(1/r)$-cutting of size $O(r^2)$ was independently found by Matoušek [Mat90], with $O(nr^2 \log r)$ running time. This running time was improved by Agarwal [Aga90] to $O(nr(\log r)^\omega)$, for a constant $\omega < 3.3$. This paper contains the idea of using vertices as an accounting device.

An $O(nr^{d-1})$ sequential deterministic algorithm for computing a $(1/r)$-cutting was given by Chazelle [Cha93a] by the indicated method (including Lemma 5.2). The parallelization is due to Amato et al. [AGR94]. Chazelle's construction has the advantage of providing a hierarchy of coarser cuttings, which for example gives an optimal data structure for point location in the cutting for free. It also gave a new existence proof for the asymptotically optimal cuttings.

Various generalizations of cuttings were used to solve other problems. One direction of generalization is to decompose not the whole arrangement, but only some portion of it, for instance one cell (this is what we did in the algorithm for Problem 2.1). A 0-*shallow ε-cutting* for a collection H of n hyperplanes in $\mathbb{R}^d$ is a collection Ξ of simplices which completely cover the cell of the arrangement of H containing the origin (and perhaps something more), such that no $\Delta \in \Xi$ is intersected by more than εn hyperplanes of H. Existence of 0-shallow $(1/r)$-cuttings of size $O(r^{\lfloor d/2 \rfloor})$ for any H and r is established in [Mat92b] by the method of [CF90], and a deterministic algorithm is given for constructing them in $O(nr^{\lfloor d/2 \rfloor -1})$ time, but only for $r < n^{1-\delta}$ with an arbitrary positive constant $\delta > 0$. For even dimension d, one can do better using the vertex-accounting mechanism (see [AGR94]), but in odd dimension such an approach fails: the reason is (roughly) that the maximum complexity of the intersection of n halfspaces in dimensions d (d odd) and $d-1$ has the same order of magnitude, and hence 'boundary effects' can be too large. Ramos [Ram97a] observed that by combining the methods of Brönniman et al. [BCM93] and of Matoušek and Schwarzkopf [MS93], one can get an $O(nr^{\lfloor d/2 \rfloor -1})$ deterministic algorithm for computing 0-shallow $(1/r)$-cuttings in the full range of r's.

Another direction of generalization is to consider other surfaces than hyperplanes. In this case also the simplices of the cutting must be replaced by some more general (curved) cells. Here the situation is much less satisfactory than for hyperplanes because it is not known how to decompose cells in arrangements of surfaces into constant-complexity subcells in an optimal or near-optimal way; see Chazelle et al. [CEGS89a] for current best results and Sharir and Agarwal [SA95] for more background information. Improvement in this would bring asymptotic improvements in many algorithms in computational geometry. An abstract treatment of cuttings is pursued in Agarwal and Matoušek [AM94].

5.2 Convex hull, diameter, and other problems

Convex hulls. In order to go into derandomization of sequential algorithms for computing the convex hull of n points in $\mathbb{R}^d$, one must have quite strict requirements on the asymptotic complexity of the algorithm. We recall that a worst-case lower bound for this problem is $\Omega(n \log n + n^{\lfloor d/2 \rfloor})$. Optimal deterministic algorithms have been long known for all even dimensions and also for dimension 3, and for odd dimensions there is a deterministic algorithm with $O(n^{\lfloor d/2 \rfloor} \log n)$ running time, so the main challenge was to get rid of the $\log n$ factor for odd dimensions $d \geq 5$. This was achieved by Chazelle in a *tour de force* of derandomization [Cha93b].

Chazelle's algorithm is fairly complicated, so we can only indicate the main ingredients. First of all, one does not really derandomize the strictly incremental approach; rather new points are inserted in $O(\log n)$ batches, each batch having about the same number of points as all the previous batches together. The derandomization is done by the method of conditional probabilities, which is used to figure out the objects to be inserted in the next batch (otherwise one follows the randomized counterpart). The conditional expectations must be suitably approximated, and the largest part of the analysis of the algorithm aims at showing that the error made by this approximation does not destroy the properties of the randomized algorithm. The required quantities are computed using ε-approximations, and Lemma 5.2 is used heavily for estimating the errors.

The derandomization technique developed for convex hulls does not seem to be immediately applicable to other problems solved by randomized incremental constructions. The other problems have a still larger degree of inherent "nonlocality" compared to the convex hull computation, and approximating the required conditional expectations is thus harder.

More opportunities for derandomization of convex hull algorithm remained in the realm of parallel algorithms and for output-sensitive algorithms in $\mathbb{R}^3$, and here indeed the best known deterministic algorithm were obtained in this way (see Remarks and bibliography).

Diameter in $\mathbb{R}^3$. Given an n-point set $P \subset \mathbb{R}^3$, the problem is to determine $\operatorname{diam}(P)$, the diameter of P. Clarkson and Shor [CS89] gave a simple optimal $O(n \log n)$ randomized algorithm; obtaining a comparably efficient deterministic algorithm seems surprisingly difficult. The current record for deterministic running time is $O(n \log^2 n)$, and algorithms approaching this bound are all quite complicated.

All efficient algorithms for computing diameter in $\mathbb{R}^3$ use as a subroutine an algorithm for computing the intersection of n balls of equal radius in $\mathbb{R}^3$. The connection between these two problems is quite simple: The diameter of the set P is at most r iff the intersection $I(P, r)$ of balls of radius r centered at the points of P contains all points of P; hence if we can compute $I(P, r)$ and decide membership of points in it quickly, we can compare the (unknown) value of $\operatorname{diam}(P)$ with any given number r. Known deterministic algorithms use this comparison method plus the so-called parametric search to find the diameter (the parametric search technique is briefly outlined in the Appendix). Convex hull algorithms can often be adapted to compute the intersection of equal-radius balls, since this is rather similar to computing the intersection of halfspaces, which is a dual version of convex hull computation (however, the problems are not quite the same and additional technical complications may arise). For computing the diameter via parametric search, the ball intersection algorithms has to be implemented in parallel (in a very weak computational model), so that one has to mimic some parallel convex hull algorithm, and all such (fast) 3-dimensional algorithms are fairly complicated.

Linear programming. The linear programming problem considered in computational geometry is usually formulated as follows: Given n halfspaces in $\mathbb{R}^d$, compute a point of their intersection maximizing a given linear function. For any fixed d, this problem can be solved in $O(n)$ time; the first such algorithm (a deterministic one) was given by Megiddo [Meg84]. Since then, the dependence of the constant of proportionality on d has been improved several times, from the original 2^{2^d} in Megiddo's paper. As was mentioned in section 2, randomized algorithms are far ahead of deterministic ones in this respect (with roughly $\exp(\sqrt{d})$ for randomized algorithms and $\exp(O(d \log d))$ for deterministic ones).

The best deterministic algorithms were obtained by derandomization. Two basic approaches have been used. One, derived from Megiddo's original algorithm, is built by induction on the dimension. In dimension d, it searches for the position of the optimum by testing its position relative to suitably chosen hyperplanes; an oracle for such tests is implemented using linear programming in dimension $d-1$. After testing a constant number of hyperplanes, the position of the optimum is determined sufficiently precisely so that a constant fraction of the constraints can be thrown away (a prune-and-search method). The most efficient methods for such a search are based on computation of suitable ε-nets. The second approach is by derandomizing an algorithm of Clarkson [Cla95].

Segment arrangements. Given n segments in the plane with a total of k intersections, we want to compute their arrangement. The optimal running time for this problem is $O(n \log n + k)$, which is easy to achieve by a randomized incremental construction but surprisingly difficult deterministically. After the first optimal deterministic solution by Chazelle and Edelsbrunner [CE92] (which did not use derandomization), there remained the problems to give an $O(n)$ space algorithm and to give an efficient parallel version. Both these goals were achieved by Amato et al. [AGR95] by designing a divide-and-conquer type algorithm and derandomizing it. The basic approach is to take a sample of $r = n^\alpha$ segments, compute the vertical decomposition of their arrangement, and recurse in each trapezoid. This in itself (choosing the sample as a suitable ε-seminet) leads to an $O(n \log^{const} n + k)$ running time, since each of the $\log \log n$ levels of recursion costs a constant factor. The next idea is to control this blowup by the vertex-accounting mechanism (described in section 5.1); the most delicate case is when k is small, say $k \approx \log^2 n$, since then "boundary effects" might create too many subproblems. Amato et al. add two more ingredients to the algorithm: If a subproblem becomes very small (having fewer than $\log^2 n$ segments), they solve it directly by the above mentioned $O(n \log^{const} n + k)$ algorithm, and if the current subproblem has very few intersections, they slice it into very many subproblems along a suitable collection of nonintersecting segments. This allows them to keep the total size of all current subproblems bounded by $O(n+k/\log n)$ and leads to an optimal algorithm; we refer to [AGR95] for details (a way to fix a problematic derandomization in that paper was noted by Ramos [Ram96]; see remarks to section 4.4).

Bibliography and remarks. The problems considered in this section all have a rich history which we do not try to cover here. We only mention results obtained by derandomization and directly related randomized algorithms.

Convex hulls. The first worst-case optimal algorithm in all dimensions is a randomized incremental construction due to Clarkson and Shor [CS89]. A similar (but not identical) algorithm was optimally derandomized by Chazelle [Cha93b]. The method is simplified both conceptually and technically in [BCM93]. For $d = 3$, an optimal output-sensitive deterministic convex hull algorithm (of complexity $O(n \log h)$, h being the number of facets of the hull) was obtained by Chazelle and Matoušek [CM94], by derandomizing a divide-and-conquer type algorithm of Clarkson and Shor [CS89]. Finding an optimal output-sensitive algorithm in dimensions $d \geq 4$, randomized or deterministic, remains a challenging open problem (a long-time record in this direction is Seidel's $O(n^2 + h \log n)$ deterministic algorithm [Sei86]).

In the parallel domain, Goodrich [Goo93] obtained an $O(\log^2 n)$ time work-optimal deterministic convex hull algorithm for dimension 3 by derandomization; a simpler version with the same performance is given by Amato et al. [AGR94], who also provide an $O(\log^3 n)$ time, $O(n \log h)$ work output-sensitive algorithm, and for for even dimensions $d \geq 4$ they describe an $O(\log n)$ time, $O(n^{\lfloor d/2 \rfloor})$ work deterministic algorithm (based on a shallow cutting construction and vertex accounting; see remarks to section 5.1). (All algorithms in this paragraph are in the EREW PRAM model.)

Diameter in $\mathbb{R}^3$. The first derandomization of the Clarkson-Shor 3-dimensional diameter algorithm is due to Chazelle et al. [CEGS93] and has $O(n^{1+\delta})$ running time for any fixed $\Delta > 0$. This was improved by Matoušek and Schwarzkopf [MS96] to $O(n \log^{const} n)$, by Ramos [Ram94] to $O(n \log^5 n)$, by Amato et al. [AGR94] to[9] $O(n \log^3 n)$, and finally by Ramos [Ram97b] to the current best time $O(n \log^2 n)$. Among these, the Ramos' $O(n \log^5 n)$ algorithm is the most elementary one.

Parametric search is an ingenious algorithmic technique; roughly speaking, under certain favorable circumstances it allows one to convert decision algorithms into search algorithms. It was formulated by Megiddo [Meg83]. A technical improvement, which sometimes reduces the running time by a logarithmic factor, was suggested by Cole [Col87] (see also Cole et al. [CSY87], Cohen and Megiddo [CM93], Norton et al. [NPT92] for a higher-dimensional generalization). Parametric search is usually not considered a derandomization technique, but sometimes it helps considerably in constructing deterministic algorithms.

Linear programming. A straightforward derandomization using ε-nets is applied to Clarkson's algorithm [Cla95] by Chazelle and Matoušek [CM96], which surprisingly yields a linear time deterministic algorithm with the best known dependence on the dimension. This algorithm is also applicable for other problems similar to linear programming, extending previous results of Dyer [Dye92]. Agarwal et al. [AST93] give another derandomized linear programming algorithm with a similar efficiency but based on Megiddo's original approach; this algorithm is applicable on yet another class of search and optimization problems. Their technique of searching can also speed up some applications of multidimensional parametric search.

A very fast parallel deterministic implementation on CRCW PRAM was first found by Ajtai and Megiddo [AM92] using expanders; their algorithm has $O((\log \log n)^d)$ running time with $O(n(\log \log n)^d)$ work. Goodrich [Goo96] applies their ideas, ideas of Dyer [Dye95], and a fast parallel computation of ε-nets to give algorithms with $O(n)$ work and $O((\log \log n)^{d+2})$ running time on a CRCW PRAM, resp. $O(\log n (\log \log n)^{d-1})$ time on an EREW PRAM (previously, [Dye95] gave an EREW PRAM algorithm with the same

[9] Brönnimann et al. [BCM93]who claimed the $O(n \log^3 n)$ bound earlier have an error in the diameter computation part; this could probably be fixed, but the algorithm is more complicated than the one of [AGR94].

running time and slightly worse work).

Segment arrangements and generalizations. Optimal randomized algorithms for constructing a segment arrangement, with $O(n \log n + k)$ expected running time, were discovered by Clarkson and Shor [CS88] and by Mulmuley [Mul90]. Amato et al. [AGR95] give an $O(\log^2 n)$ time, work-optimal EREW PRAM version of their algorithm. Another deterministic algorithm with $O(n)$ space and optimal time was found also by Balaban [Bal95]; this one doesn't seem to parallelize easily. [AGR95] can also compute a single face in an arrangement of n segments in $O(n\,\alpha^2(n) \log n)$ deterministic time (compared to the best known $O(n\,\alpha(n) \log n)$ randomized algorithm of Chazelle et al. [CEG+93]). They use similar methods to construct a point location structure for an arrangement of n $(d-1)$-dimensional possibly intersecting simplices in $\mathbb{R}^d$, $d \geq 3$, with $O(\log n)$ query time and $O(n^{d-1} \log^{const} n + k)$ deterministic preprocessing time and storage, where k is the complexity of the arrangement; this is based on derandomizing Pellegrini's work [Pel96]. Deterministic computation of the lower envelope of n bounded-degree univariate algebraic functions has been studied by Ramos [Ram97b] and used as a subroutine in his 3-dimensional diameter algorithm.

Acknowledgment. I am grateful to Bernard Chazelle, to Edgar Ramos, and to two anonymous referees for very useful comments to earlier versions of this paper.

References

[AASS90] P. K. Agarwal, B. Aronov, M. Sharir, and S. Suri. Selecting distances in the plane. In *Proc. 6th Annu. ACM Sympos. Comput. Geom.*, pages 321–331, 1990.

[ABI86] N. Alon, L. Babai, and A. Itai. A fast and simple randomized algorithm for the maximal independent set problem. *Journal of Algorithms*, 7:567–583, 1986.

[AE97] P. K. Agarwal and J. Erickson. Geometric range searching and its relatives. Manuscript, 1997.

[Aga90] P. K. Agarwal. Partitioning arrangements of lines: I. An efficient deterministic algorithm. *Discrete Comput. Geom.*, 5:449–483, 1990.

[Aga91] P. K. Agarwal. Geometric partitioning and its applications. In J.E. Goodman, R. Pollack, and W. Steiger, editors, *Computational Geometry: Papers from the DIMACS special year.* Amer. Math. Soc., 1991.

[AGHP92] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple construction of almost k-wise independent random variables. *Random Structures and Algorithms*, 3:289–304, 1992.

[AGR94] N. M. Amato, M. T. Goodrich, and E. A. Ramos. Parallel algorithms for higher-dimensional convex hulls. In *Proc. 35th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 683–694, 1994.

[AGR95] N. M. Amato, M. T. Goodrich, and E. A. Ramos. Computing faces in segment and simplex arrangements. In *Proc. 27th Annu. ACM Sympos. Theory Comput.*, pages 672–682, 1995.

[AHW87] N. Alon, D. Haussler, and E. Welzl. Partitioning and geometric embedding of range spaces of finite Vapnik-Chervonenkis dimension. In *Proc. 3rd Annu. ACM Sympos. Comput. Geom.*, pages 331–340, 1987.

[AM92] M. Ajtai and N. Megiddo. A deterministic $poly(\log \log n)$-time n-processor algorithm for linear programming in fixed dimensions. In *Proc. 24th Annu. ACM Sympos. Theory Comput.*, pages 327–338, 1992.

[AM94] P. K. Agarwal and J. Matoušek. On range searching with semialgebraic sets. *Discr. & Comput. Geom.*, 11:393–418, 1994.

[AMS94] P. K. 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.

[AS90] P. K. Agarwal and M. Sharir. Red-blue intersection detection algorithms, with applications to motion planning and collision detection. *SIAM J. Comput.*, 19(2):297–321, 1990.

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

[AST92] P. K. Agarwal, M. Sharir, and S. Toledo. Applications of parametric searching in geometric optimization. In *Proc. 3rd ACM-SIAM Sympos. Discrete Algorithms*, pages 72–82, 1992.

[AST93] P. K. Agarwal, M. Sharir, and S. Toledo. An efficient multidimensional searching technique and its applications. Tech. Report CS-1993-20, Department Computer Science, Duke University, 1993.

[Bal95] I. Balaban. An optimal algorithm for finding segment intersections. In *Proc. 11th ACM Sympos. Comput. Geom.*, pages 211–219, 1995.

[BC87] J. Beck and W. Chen. *Irregularities of Distribution.* Cambridge University Press, 1987.

[BC94] H. Brönnimann and B. Chazelle. Optimal slope selection via cuttings. In *Proc. 6th Canad. Conf. Comput. Geom.*, pages 99–103, 1994.

[BCM93] H. Brönnimann, B. Chazelle, and J. Matoušek. Product range spaces, sensitive sampling, and derandomization. In *Proc. 34th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 400–409, 1993. Revised version is to appear in *SIAM J. Computing*.

[BG95] H. Brönnimann and M. Goodrich. Almost optimal set covers in finite VC-dimension. *Discr. & Comput. Geom.*, 14:463–484, 1995.

[BR91] B. Berger and J. Rompel. Simulating $(\log n)^c$-wise independence in NC. *Journal of the ACM*, 38(4):1028–1046, 1991.

[BRS94] B. Berger, J. Rompel, and P. W. Shor. Efficient NC algorithms for set cover with applications to learning and geometry. *J. Comput. Syst. Sciences*, 49:454–477, 1994.

[CE92] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *J. ACM*, 39:1–54, 1992.

[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$^+$93] B. Chazelle, H. Edelsbrunner, L. Guibas, M. Sharir, and J. Snoeyink. Computing a face in an arrangement of line segments and related problems. *SIAM J. Comput.*, 22:1286–1302, 1993.

[CEGS89a] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. A singly-exponential stratification scheme for real semi-algebraic varieties and its applications. In *Proc. 16th Internat. Colloq. Automata Lang. Program.*, volume 372 of *Lecture Notes Comput. Sci.*, pages 179–192. Springer-Verlag, 1989.

[CEGS89b] B. Chazelle, H. Edelsbrunner, L. J. Guibas, and M. Sharir. Lines in space: combinatorics, algorithms, and applications. In *Proc. 21st Annu. ACM Sympos. Theory Comput.*, pages 382–393, 1989.

[CEGS93] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. Diameter, width, closest line pair and parametric searching. *Discrete Comput. Geom.*, 10:183–196, 1993.

[CEM+96] K. L. Clarkson, D. Eppstein, G. L. Miller, C. Sturtivant, and S.-H. Teng. Approximating center points with iterated Radon points. *Int. J. Comput. Geom. Appl.*, 6:357–377, 1996.

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

[Cha93a] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geom.*, 9(2):145–158, 1993.

[Cha93b] B. Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete Comput. Geom.*, 10:377–409, 1993.

[Cha96] T. M. Chan. Fixed-dimensional linear programming queries made easy. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages 284–290, 1996.

[Che86] L. P. Chew. Building Voronoi diagrams for convex polygons in linear expected time. Technical Report PCS-TR90-147, Dept. Math. Comput. Sci., Dartmouth College, Hanover, NH, 1986.

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

[Cla88a] K. L. Clarkson. Applications of random sampling in computational geometry, II. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pages 1–11, 1988.

[Cla88b] K. L. Clarkson. A randomized algorithm for closest-point queries. *SIAM J. Comput.*, 17:830–847, 1988.

[Cla92] K. L. Clarkson. Randomized geometric algorithms. In D.-Z. Du and F. K. Hwang, editors, *Computing in Euclidean Geometry*, volume 1 of *Lecture Notes Series on Computing*, pages 117–162. World Scientific, Singapore, 1992.

[Cla95] K. L. Clarkson. Las Vegas algorithms for linear and integer programming. *J. ACM*, 42:488–499, 1995.

[CM93] E. Cohen and N. Megiddo. Strongly polynomial-time and NC algorithms for detecting cycles in dynamic graphs. *Journal of the ACM*, 40:791–832, 1993.

[CM94] B. Chazelle and J. Matoušek. Derandomizing an output-sensitive convex hull algorithm in three dimensions. *Comput. Geom.: Theor. Appl.*, 5:27–32, 1994.

[CM96] B. Chazelle and J. Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. *J. Algorithms*, 21:116–132, 1996.

[Col87] R. Cole. Slowing down sorting networks to obtain faster sorting algorithms. *Journal of the ACM*, 34:200–208, 1987.

[CS88] K. L. Clarkson and P. W. Shor. Algorithms for diametral pairs and convex hulls that are optimal, randomized, and incremental. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pages 12–17, 1988.

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

[CSSS89] R. Cole, J. Salowe, W. Steiger, and E. Szemerédi. An optimal-time algorithm for slope selection. *SIAM J. Comput.*, 18:792–810, 1989.

[CSY87] R. Cole, M. Sharir, and C. Yap. On k-hulls and related problems. *SIAM Journal on Computing*, 16(1):61–67, 1987.

[CW89] B. Chazelle and E. Welzl. Quasi-optimal range searching in spaces of finite VC-dimension. *Discrete Comput. Geom.*, 4:467–489, 1989.

[dBDS95] M. de Berg, K. Dobrindt, and O. Schwarzkopf. On lazy randomized incremental construction. *Discr. Comput. Geom.*, 14:261–286, 1995.

[DMN92] M. B. Dillencourt, D. M. Mount, and N. S. Netanyahu. A randomized algorithm for slope selection. *Internat. J. Comput. Geom. Appl.*, 2:1–27, 1992.

[Dye92] M. E. Dyer. A class of convex programs with applications to computational geometry. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 9–15, 1992.

[Dye95] M. Dyer. A parallel algorithm for linear programming in fixed dimension. In *Proc. 11th ACM Symp. on Comput. Geom.*, pages 345–349, 1995.

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

[EGL+92] G. Even, O. Goldreich, M. Luby, N. Nisan, and B. Veliković. Approximations of general independent distributions. In *Proc. 24th ACM Symp. on Theory of Computing*, pages 10–16, 1992.

[ES73] P. Erdős and J. L. Selfridge. On a combinatorial game. *J. Comb. Theory Ser. A*, 14:298–301, 1973.

[ES76] M. Eisner and D. Severance. Mathematical techniques for efficient record segmentation in large shared database. *Journal of the ACM*, 23:619–635, 1976.

[Goo93] M. T. Goodrich. Geometric partitioning made easier, even in parallel. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 73–82, 1993.

[Goo96] M. Goodrich. Fixed-dimensional parallel linear programming via relative ϵ-approximations. In *Proc. 7th Annual ACM-SIAM Sympos. on Discrete Algorithms*, pages 132–141, 1996.

[GS93] L. Guibas and M. Sharir. Combinatorics and algorithms of arrangements. In J. Pach, editor, *New Trends in Discrete and Computational Geometry*, volume 10 of *Algorithms and Combinatorics*, pages 9–36. Springer-Verlag, 1993.

[Gus83] D. Gusfield. Parametric combinatorial computing and a problem of program module distribution. *Journal of the ACM*, 30:551–563, 1983.

[Har92] J. Harris. *Algebraic Geometry (A First Course)*. Springer-Verlag, Berlin, 1992.

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

[Jof74] A. Joffe. On a set of almost deterministic k-independent random variables. *Annals of Probability*, 2:161–162, 1974.

[Kal92] G. Kalai. A subexponential randomized simplex algorithm. In *Proc. 24th Annu. ACM Sympos. Theory Comput.*, pages 475–482, 1992.

[Kar91] R. Karp. An introduction to randomized algorithms. *Discr. Appl. Math*, 34:165–201, 1991.

[KK94] D. Karger and D. Koller. (De)randomized construction of small sample spaces in $\mathcal{NC}$. In *Proc. 35th IEEE Symp. Foundat. Comput. Sci.*, pages 252–263, 1994.

[KM94a] H. Karloff and Y. Mansour. On construction of k-wise independent random variables. In *Proc. 26th ACM Symp. Theor. Comput.*, 1994.

[KM94b] D. Koller and N. Megiddo. Constructin small sample spaces satisfying given constraints. *SIAM Journal on Discrete Mathematics*, 7:260–274, 1994.

[KPW92] J. Komlós, J. Pach, and G. Woeginger. Almost tight bounds for ϵ-nets. *Discrete Comput. Geom.*, 7:163–173, 1992.

[KR93] H. Karloff and P. Raghavan. Randomized algorithms and pseudorandom numbers. *J. ACM*, 40(3):454–476, 1993.

[KS93a] M. J. Katz and M. Sharir. An expander-based approach to geometric optimization. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 198–207, 1993.

[KS93b] M. J. Katz and M. Sharir. Optimal slope selection via expanders. *Inform. Process. Lett.*, 47:115–122, 1993.

[KW85] R. Karp and M. Wigderson. A fast parallel algorithm for the maximum independent set problem. *Journal of the ACM*, 32:762–773, 1985.

[LMS94] C.-Y. Lo, J. Matoušek, and W. L. Steiger. Algorithms for ham-sandwich cuts. *Discr. & Comput. Geom.*, 11:433, 1994.

[Lub93] M. Luby. Removing randomness in parallel computation without processor penalty. *J. Comput. Syst. Sci.*, 47:250–286, 1993.

[LW95] M. Luby and A. Wigderson. Pairwise independence and derandomization. Tech. Report UCB/CSD-95-880, Univ. of California at Berkeley, 1995. Available electronically at `http://www.icsi.berkeley.edu/~luby/pair_sur.html`.

[Mat90] J. Matoušek. Construction of ϵ-nets. *Discrete Comput. Geom.*, 5:427–448, 1990.

[Mat91a] J. Matoušek. Cutting hyperplane arrangements. *Discrete Comput. Geom.*, 6:385–406, 1991.

[Mat91b] J. Matoušek. Randomized optimal algorithm for slope selection. *Inform. Process. Lett.*, 39:183–187, 1991.

[Mat92a] J. Matoušek. Efficient partition trees. *Discrete Comput. Geom.*, 8:315–334, 1992.

[Mat92b] J. Matoušek. Reporting points in halfspaces. *Comput. Geom. Theory Appl.*, 2(3):169–186, 1992.

[Mat93] J. Matoušek. Linear optimization queries. *J. Algorithms*, 14:432–448, 1993.

[Mat95a] J. Matoušek. Approximations and optimal geometric divide-and-conquer. *J. of Computer and System Sciences*, 50:203–208, 1995.

[Mat95b] J. Matoušek. Geometric range searching. *ACM Comput. Surveys*, 26:421–461, 1995.

[Meg79] N. Megiddo. Combinatorial optimization with rational objective functions. *Math. Oper. Res.*, 4:414–424, 1979.

[Meg83] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *Journal of the ACM*, 30:852–865, 1983.

[Meg84] N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. ACM*, 31:114–127, 1984.

[MNN94] R. Motwani, J. Naor, and M. Naor. The probabilistic method yields deterministic parallel algorithms. *J. Comput.Syst. Sci.*, 49:478–516, 1994.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[MRS97] S. Mahajan, E.A. Ramos, and K.V. Subrahmanyam. Solving some discrepancy problems in NC. Manuscript, 1997.

[MS93] J. Matoušek and O. Schwarzkopf. On ray shooting in convex polytopes. *Discrete & Computational Geometry*, 10(2):215–232, 1993.

[MS96] J. Matoušek and O. Schwarzkopf. A deterministic algorithm for the three-dimensional diameter problem. *Comput. Geom. Theor. Appl.*, 6:253–262, 1996.

[MSW90] J. Matoušek, R. Seidel, and E. Welzl. How to net a lot with little: small ϵ-nets for disks and halfspaces. In *Proc. 6th Annu. ACM Sympos. Comput. Geom.*, pages 16–22, 1990.

[MSW96] J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Algoritmica*, 16:498–516, 1996.

[Mul90] K. Mulmuley. A fast planar partition algorithm, I. *Journal of Symbolic Computation*, 10:253–280, 1990.

[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.

[Mul94] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms.* Prentice Hall, Englewood Cliffs, NJ, 1994.

[Mul96] K. Mulmuley. Randomized geometric algorithms and pseudo-random generators. *Algorithmica*, 16:450–463, 1996.

[MWW93] J. Matoušek, E. Welzl, and L. Wernisch. Discrepancy and ε-approximations for bounded VC-dimension. *Combinatorica*, 13:455–466, 1993.

[Nis92] N. Nissan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12:449–461, 1992.

[NN93] J. Naor and M. Naor. Small-bias probability spaces: Efficient construction and applications. *SIAM J. Computing*, 22:838–856, 1993.

[NPT92] C. H. Norton, S. A. Plotkin, and É. Tardos. Using separation algorithms in fixed dimensions. *J. Algorithms*, 13:79–98, 1992.

[Pel96] M. Pellegrini. On point location and motion planning among simplices. *SIAM J. Comput.*, 25:1061–1081, 1996.

[Rag88] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *Journal of Computer and System Sciences*, 37:130–143, 1988.

[Ram94] E. Ramos. An algorithm for intersecting equal radius balls in R^3. Tech. Report UIUCDS-R-94-1851, Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, 1994.

[Ram96] E. Ramos. Private communication, 1996.

[Ram97a] E. Ramos. Unpublished note, Max-Planck Institut für Informatik, Saarbrucken, 1997.

[Ram97b] E. Ramos. Construction of 1-d lower envelopes and applications. In *Proc. 13th Ann. ACM Sympos. Comput. Geom.*, 1997.

[Rom90] J. T. Rompel. Techniques for computing with low-independence randomness. PhD. thesis, Dept. of EECS, M.I.T., 1990.

[SA95] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications.* Cambridge University Press, Cambridge, 1995.

[Sei86] R. Seidel. Constructing higher-dimensional convex hulls at logarithmic cost per face. In *Proc. 18th Annu. ACM Sympos. Theory Comput.*, pages 404–413, 1986.

[Sei91a] R. Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Comput. Geom. Theory Appl.*, 1:51–64, 1991.

[Sei91b] 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.

[Spe87] J. Spencer. *Ten lectures on the probabilistic method.* CBMS-NSF. SIAM, 1987.

[Sri95] A. Srivastav. Derandomized algorithms in combinatorial optimization. Habilitation-Thesis, Institut für Informatik, Freie Universität Berlin, 1995.

[SSS95] J. Schmidt, A. Siegel, and A. Srinivasan. Chernoff-Hoeffending bounds for applications with limited independence. *SIAM Journal on Discrete Mathematics*, 8:223–250, 1995.

[VC71] V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.*, 16:264–280, 1971.

[Wil82] D. E. Willard. Polygon retrieval. *SIAM J. Comput.*, 11:149–165, 1982.

[WZ93] A. Wigderson and D. Zuckerman. Expanders that beat the eigenvalue bound: Explicit construction and applications. In *Proc. 25th ACM Symposium on Theory of Computing*, pages 245–51, 1993.

[YY85] A. C. Yao and F. F. Yao. A general approach to D-dimensional geometric queries. In *Proc. 17th Annu. ACM Sympos. Theory Comput.*, pages 163–168, 1985.

Appendix: The parametric search technique

Parametric search is a general strategy for algorithm design. Roughly speaking, it produces algorithms for searching from algorithms for verification, under suitable assumptions.

Let us consider a problem in which the goal is to find a particular real number, t^*, which depends on some input objects. We consider these input objects fixed. Suppose that we have two algorithms at our disposal: First, an algorithm O, which for a given number t decides among the possibilities $t < t^*$, $t = t^*$ and $t > t^*$ (although it does not explicitly know t^*, only the input objects); let us call such an algorithm O the *oracle*. Second, an algorithm G (called the *generic* algorithm), whose computation depends on the input objects and on a real parameter t, and for which it is guaranteed that its computation for $t = t^*$ differs from the computation for any other $t \neq t^*$. We can use algorithm O also in the role of G, but often it is possible to employ a simpler algorithm for G. Under certain quite weak assumptions about algorithm G, the parametric search produces an algorithm for finding t^*.

The main idea is to simulate the computation of algorithm G for the (yet unknown) parameter value $t = t^*$. The computation of G of course depends on t, but we assume that all the required information about t is obtained by testing the signs of polynomials of small (constant bounded) degree in t. The coefficients in each such polynomial may depend on the input objects of the algorithm and on the outcomes of the previous tests, but not directly on t. The sign of a particular polynomial can be tested also in the unknown t^*: We find the roots $t_1, \ldots, t_k$ of the polynomial p, we locate t^* among them using the algorithm O and we derive the sign of $p(t^*)$ from it. In this way we can simulate the computation of the algorithm G at t^*.

If we record all tests involving t made by algorithm G during its computation, we can then find the (unique) value t^* giving appropriate results in all these tests, thereby solving the search problem.

In this version we need several calls to the oracle for every test performed by algorithm G. The second idea is to do many tests at once whenever possible. If algorithm G executes a group of mutually independent tests with polynomials $p_1(t), \ldots, p_m(t)$ (meaning that the polynomial p_i does not depend on the outcome of the test involving another polynomial p_j), we can answer all of them by $O(\log n)$ calls of the oracle: We compute the roots of all the polynomials $p_1, \ldots, p_m$ and we locate the position of t^* among them by binary search. Parametric search will thus be particularly efficient for algorithms G implemented in parallel, with a small number of parallel steps, since the tests in one parallel step are necessarily independent in the above mentioned sense.

Parametric search was formulated by Megiddo [Meg83], the idea of simulating an algorithm at a generic value appears in [ES76], [Gus83], [Meg79]. A technical improvement, which sometimes reduces the running time by a logarithmic factor, was suggested by Cole [Col87]. A generalization of parametric search to higher dimension, where the parameter t is a point in $\mathbb{R}^d$ and the oracle can test the position of t^* with respect to a given hyperplane, appears in [CSY87], [CM93], [NPT92], [Mat93]. Currently is parametric search a quite popular technique also in computational geometry; from numerous recent works we select more or less randomly [CSSS89], [AASS90], [AST92], [CEGS93].

Algorithms based on parametric search, although theoretically elegant, appear quite complicated for implementation. In many specific problems, parametric search can be replaced by a randomized algorithm (see [DMN92], [Mat91b], [Cha96]) or by other techniques (e.g., [BC94], [KS93a]) with a similar efficiency.