A DETERMINISTIC VIEW OF RANDOM SAMPLING AND ITS USE IN GEOMETRY

Bernard Chazelle
Joel Friedman

CS-TR-181-88

September 1988

A Deterministic View of Random Sampling and its Use in Geometry

BERNARD CHAZELLE AND JOEL FRIEDMAN
Department of Computer Science
Princeton University
Princeton, NJ 08544

Abstract: The combination of divide-and-conquer and random sampling has proven very effective in the design of fast geometric algorithms. A flurry of efficient probabilistic algorithms have been recently discovered, based on this happy marriage. We show that all those algorithms can be derandomized with only polynomial overhead. In the process we establish results of independent interest concerning the covering of hypergraphs and we improve on various probabilistic bounds in geometric complexity. For example, given n hyperplanes in d-space and any integer r large enough, we show how to compute, in polynomial time, a simplicial packing of size $O(r^d)$ which covers d-space, each of whose simplices intersects $O(n/r)$ hyperplanes. Also, we show how to locate a point among n hyperplanes in d-space in $O(\log n)$ query time, using $O(n^d)$ storage and polynomial preprocessing.

Bernard Chazelle wishes to acknowledge the National Science Foundation for supporting this research in part under Grant CCR–8700917. Joel Friedman wishes to acknowledge the National Science Foundation for supporting this research in part under Grant CCR–8858788, and the Office of Naval Research under Grant N00014–87–K–0467.

1. Introduction

Recent efforts have demonstrated the considerable power of randomization in speeding up geometric algorithms, especially at the low end of the complexity spectrum. Efficient Las Vegas algorithms have been discovered for range searching, triangulation, nearest neighbors, convex hulls, hidden-surface removal, motion-planning, etc. [1,2,3,4,5,6,8,9,12,19,25]. But for all their appeal, probabilistic algorithms raise a disturbing question: how much distortion in the deterministic complexity of a problem do we let in by allowing randomization? Put more bluntly, what can we hope to learn from probabilistic models? The issue has been under intense scrutiny lately, especially in the context of the *NC* vs. *RNC* question. A popular technique for removing randomization is to check if mutual independence is necessary, and if not, reduce the probability space to allow exhaustive searching (Luby [15]). Another approach is the method of ***conditional probabilities*** (Raghavan [18], Spencer [21], Pach and Spencer [17]). The basic idea is to estimate the probabilities of failure associated with the nodes of the computation tree. The difficulty is that estimating these probabilities may not always be doable in polynomial time. Finally, if all else has failed, there always remains the option of looking for a deterministic solution without a probabilistic model in mind...

The main contribution of this work is to show that most of the Las Vegas algorithms used in computational geometry can be made deterministic. In doing so, we also improve on some of the probabilistic complexity bounds obtained earlier. Our starting point is the observation that randomization is often confined to repeated random sampling over a fixed domain. Invariably, the desired feature of a random sample is to behave somewhat like a *point cover* (Lovász [14]) or likewise a *puncture set* (Spencer [20]). We say "somewhat" because we must add a twist to the definition of these notions. Briefly, the setting is this: let H be a hypergraph of n vertices and $m = n^{O(1)}$ edges. Given an integer r, pick an r-sample (i.e., a subset of r vertices) such that every edge of size past a certain threshold contains at least one sample vertex. This model is nice to work with, but it is a tad too general to give good results. Looking at the computational geometry literature where random sampling has been used, two distinct types of scenarios appear. In one case the covering requirement is confined to a particular subset of the edges specified by the sample itself (e.g., Clarkson [2,3,4,5], Reif and Sen [19]). Furthermore, the hypergraph is assumed to obey a certain ***bounded vertex dependency*** condition. In the other scenario (Haussler and Welzl [12]) we restrict ourselves to hypergraphs induced by range spaces of finite *Vapnik-Chervonenkis dimension* [24]. In this framework, a certain parameter of the hypergraph akin to its clique number is bounded above by a constant. Numerous other applications fitting either of these scenarios have been found recently (e.g., [1,6,8,9,12,25]).

Our main result is a framework which unifies both random sampling scenarios and yield deterministic solutions for the underlying covering problems. In the process we establish results of independent interest concerning the covering of hypergraphs. For this we modify Lovász's *greedy cover algorithm* [13]. The result is this: Given a hypergraph with n vertices and m edges, each of size $\geq \alpha n$, the algorithm computes an r-sample that intersects every edge e of the hypergraph in $\Omega(|e|r/n)$ vertices, where $r = O\big((\log n + \log m)/\alpha\big)$. This means that for each edge the number of covered vertices is at least a fixed fraction of the "expected" value in the probabilistic model. By comparison, Lovász's algorithm guarantees only one covered vertex per edge.

The tools for computing covers which we present in this paper are powerful enough to derandomize just about every probabilistic algorithm proposed in computational geometry. It must be noted, however, that our deterministic constructions can be worthless if the underlying problem is of very low complexity. For example, the fastest known deterministic algorithm for triangulating a simple n-gon runs in time $O(n \log\log n)$ (Tarjan and Van Wyk [22])†: its probabilistic counterpart is slightly better, running in time $O(n \log^* n)$ (Clarkson et al. [6]). Our methods incur substantial, albeit polynomial, overhead and are too heavy-handed for such a delicate line of business. Still, there is plenty of room left for interesting applications, in particular, in the area of geometric data structures. Here are a few examples (in the following, ε denotes any positive real).

(1) *Post-Office.* Given n points in E^d, there exists a data structure of size $O\left(n^{\lceil d/2 \rceil + \varepsilon}\right)$ for answering nearest-neighbor queries in $O(\log n)$ time (Clarkson [2]).

(2) *Halfplane Range Search.* Given n points in E^d, there exists a data structure of size $O\left(n^{d+\varepsilon}\right)$ for counting how many points lie on one side of a query hyperplane in time $O(\log n)$ (Clarkson [3]). There is also a data structure of size $O\left(n^{\lfloor d/2 \rfloor + \varepsilon}\right)$ for reporting all k points on one side of a query hyperplane in time $O(k + \log n)$ (Clarkson [4]).

(3) *Simplex Range Search.* Given n points in E^d, there exists a data structure of size $O(n)$ for counting how many points lie inside a query simplex in time $O\left(n^{\frac{d(d-1)}{d(d-1)+1}+\varepsilon}\right)$ (Haussler and Welzl [12]).

(4) *Probabilistic Divide-and-Conquer.* Given n hyperplanes in d-space and a parameter $r > 0$, there exists a simplicial cell decomposition of E^d of size $O\left(r^d\right)$ such that each simplex intersects $O\left(n(\log r)/r\right)$ hyperplanes (Clarkson [3], Edelsbrunner et al. [9]). The decomposition can be computed probabilistically in polynomial time.

Using our techniques, all the probabilistic algorithms listed above can be made deterministic. The conversions entail little else than re-implementing the random calls made by the algorithms. Also, we are able to improve upon some of the complexity bounds stated above. For example, given n hyperplanes in d-space and any integer r large enough, we show how to compute, in polynomial time, a simplicial packing of size $O\left(r^d\right)$ which covers d-space, each of whose simplices intersects $O(n/r)$ hyperplanes. Also, we show how to locate a point among n hyperplanes in d-space in $O(\log n)$ query time, using $O\left(n^d\right)$ storage and polynomial preprocessing.

2. Frames and Covers

A hypergraph H is a pair (V, E), where V is a finite set (the vertices) and E is a set of nonempty subsets of V (the edges). We say that H is a multi-hypergraph if we allow empty and multiple edges. In that case, E is to be understood as a multiset. A subset of V of size r is called an *r-sample*. We define a *frame* $\mathcal{F}$ as a pair $(H; \varphi)$, where H is a hypergraph and φ is a homomorphism from the semilattice $\left(2^V, \cap\right)$ to $\left(2^E, \cap\right)$ (which is to say that $W' \subseteq W \subseteq V$ implies $\varphi(W') \subseteq \varphi(W)$); we

† Throughout this paper, unless specified otherwise, logarithms are taken to the base 2.

also require that $\varphi(V) = E$. The frame is said to be of dimension d if d is a positive real constant and, for each $W \subseteq V$, the size of $\{ W \cap e \mid e \in \varphi(W) \}$ is at most $c|W|^d$, for some constant c. Note that the dimension is not defined uniquely and that bounded dimension does not imply that $|\varphi(W)|$ is polynomial in W. The ratio $\min\{ |e|/|V| : e \in E \}$ is called the *threshold* of the frame. Finally, we say that an r-sample R is an *r-cover* (or *cover*, for short) for the frame if it has a nonempty intersection with every edge of $\varphi(R)$. Here is our main result about frames.

Theorem 2.1. *Let $\mathcal{F}$ be a frame of dimension d with n vertices and let $r \leq n$ be any integer larger than some constant. If the threshold of $\mathcal{F}$ is at least $c(\log r)/r$, for some appropriate constant c, then it is possible to find an r-cover for the frame in $O(rn^{d+1})$ time.*

Before giving the proof (section 3), let us relate this result to the way random sampling has been used in the computational geometry literature. There are two basic "schools" to which the majority of randomized geometric algorithms can be attached one way or the other. One can be traced to (Clarkson [2,3,4,5], Reif and Sen [19]) and is modeled by hypergraphs of *bounded vertex dependency* (in our terminology). The other class of randomization (Haussler and Welzl [12]) concerns hypergraphs of finite *Vapnik-Chervonenkis dimension*. We give formal definitions of these notions below and describe the type of random sampling attached to them. Then we show that these are special cases of a frame. In section 4 we give a construction that in some sense removes the $\log r$ factor for a certain type of hypergraph of bounded vertex dependency, which we call *robust*.

We say that a multi-hypergraph $H = (V, E)$ has *bounded vertex dependency* if each edge e is associated with a nonempty subset $\sigma(e)$ of V, called its *signature*. Furthermore, for each edge e and any $W \subseteq V$, we have $|\sigma(e)| \leq d$ and $|\sigma^{-1}(W)| \leq c$ (counted as a multiset), for some constants c and d. Given an r-sample R we define the *domain* of the sample, denoted $H(R)$, as the multiset of edges whose signatures are subsets of R. An edge e is called *t-deficient* if (i) $e \in H(R)$, (ii) $R \cap e = \emptyset$, and (iii) $|e| \geq tn/r$. Informally, an edge is t-deficient if (i) it is relevant, (ii) the sample fails to hit it and (iii) that miss represents a deviation from the expected number of hits by a factor of at least t. (Note that an empty edge in $H(R)$ is always 0-deficient, no matter what.) It is not difficult to prove that a random r-sample (from the uniform distribution over r-subsets) will leave no edge $(c \log r)$-deficient, for some constant c. We turn to frames for an effective construction.

Theorem 2.2. *Given an n-vertex multi-hypergraph H of vertex dependency d and any integer $r \leq n$ larger than some constant, it is possible to compute an r-sample that leaves no edge of H $(c \log r)$-deficient, for some constant c. The running time is $O(rn^{d+1})$.*

Proof: We can assume that $H = (V, E)$ has no edges of size less than $cn(\log r)/r$, for we can always remove them. We define $\varphi(W)$ as the set (not multiset) $\{ e \in E \mid \sigma(e) \subseteq W \}$. We easily check that φ carries $(2^V, \cap)$ homomorphically and that $|\{ W \cap e \mid e \in \varphi(W) \}| = O(|W|^d)$. The hypergraph $H^* = (V, \varphi(V))$ creates a frame $(H^*; \varphi)$ of dimension d, to which we apply Theorem 2.1. ∎

Let $H = (V, E)$ be a hypergraph of n vertices. We define the *VC-dimension* of H as the maximum size of any $W \subseteq V$ such that $\{ W \cap e \mid e \in E \} = 2^W$. Given an r-sample $R \subseteq V$ we now say that an edge e is *t-deficient* if (i) $R \cap e = \emptyset$ and (ii) $|e| \geq tn/r$. We know that for any $r \leq n$ large enough there exists an r-sample which leaves no edge $(c \log r)$-deficient, for some constant c (Haussler and Welzl [12]). This is related to the central notion of *ε-nets*. Again, Theorem 2.1 gives an effective construction.

Theorem 2.3. *Given an n-vertex hypergraph of VC-dimension d and any integer $r \leq n$ larger than some constant, it is possible to compute an r-sample that leaves no edge of the hypergraph $(c \log r)$-deficient, for some constant c. The running time is $O(rn^{d+1})$.*

Proof: Again, we assume that H has no edges of size less than $cn(\log r)/r$. We define φ as the trivial homomorphism which maps every subset of V to E. We know from (Sauer [23], Vapnik and Chervonenkis [24]) that the hypergraph H has $O(n^d)$ edges. The hypergraph $(W, \{ W \cap e \mid e \in E \})$ obtained by restricting H to $W \subseteq V$ has *VC*-dimension at most d, so we also have $|\{ W \cap e \mid e \in \varphi(W) \}| = O(|W|^d)$. The pair (H, φ) forms a frame of dimension d, so we can call on Theorem 2.1 to conclude. ∎

3. Covering a Frame

This entire section is devoted to proving Theorem 2.1. Let $\mathcal{F} : (H = (V, E); \varphi)$ be a frame of dimension d with n vertices and let $r \leq n$ be a positive integer. We shall assume that r is larger than some appropriate constant and that the threshold of the frame is at least $7d(\log r)/r$. As we recall, this means that each edge of E contains at least $7dn(\log r)/r$ vertices. We will successively prove the existence of an r-cover of size r and then we will show how to compute it in $O(rn^{d+1})$ time.

To begin with, let us see how far we get with the obvious approach, which is to pick a random r-sample R. The probability that R fails to intersect a given edge of H is at most

$$\binom{n - \lceil 7dn(\log r)/r \rceil}{r} \Big/ \binom{n}{r} \leq \left(1 - \frac{7d \log r}{r}\right)^r \leq \exp(-7d \log r) \leq 1/r^{7d}.$$

Since $E = \varphi(V)$, there are $O(n^d)$ edges, so the probability that R is not a cover for $\mathcal{F}$ is $O(n^d/r^{7d})$. This probability can be made less than $1/2$ for, say, any $r \geq \sqrt{n}$.†

Assume now that $r < \sqrt{n}$. To prove the existence of an r-cover, we follow the approach of (Vapnik and Chervonenkis [24], Haussler and Welzl [12]) and compare the behavior of r-samples and $(2r)$-samples. Given a $(2r)$-sample S we say that an edge e is a *witness* for S if $2d \log r \leq |S \cap e| \leq r$. Let $\pi(r)$ be the probability that a random r-sample R fails to be a cover for $\mathcal{F}$. Failure means that $R \cap e = \emptyset$ for some edge $e \in \varphi(R)$. Pick one such edge, call it e^R, and choose a random r-sample R'

† All the results of this paper are trivial if the vertex set has a constant number of vertices. Therefore we shall implicitly assume that n always exceeds any appropriate constant.

in $V \setminus R$. With very high probability R' will "do things right", that is, make the edge e a witness for $R \cup R'$. Let $p(r)$ be the probability that at least one edge $e \in \varphi(R)$ both fails to intersect R and is a witness for $R \cup R'$ (where both R and R' are picked randomly, with the condition $R \cap R' = \emptyset$). We have

$$p(r) \geq \pi(r) \times \text{Prob}\left[\, |R' \cap e^R| \geq 2d \log r \,:\, \exists e \in \varphi(R);\, R \cap e = \emptyset \right],$$

and therefore

$$p(r) \geq \left(1 - \sum_{0 \leq k \leq \lfloor 2d \log r \rfloor} h(u,k)\right) \pi(r), \tag{3.1}$$

where $h(u,k) = \binom{u}{k}\binom{m-u}{r-k} / \binom{m}{r}$, $m = n - r$, and $u = \lceil 7dn(\log r)/r \rceil$. Using the trivial approximations

$$\frac{(a-b)^b}{b!} \leq \binom{a}{b} \leq \frac{a^b}{b!},$$

we find

$$h(u,k) \leq \frac{u^k (m-u)^{r-k}}{(m-r)^r} \binom{r}{k}.$$

From Robbins' approximation of the factorial [11],

$$t^t e^{-t} \sqrt{2\pi t} e^{1/(12t+1)} < t! < t^t e^{-t} \sqrt{2\pi t} e^{1/(12t)},$$

we derive

$$\binom{r}{k} \leq \frac{(r-k)^k}{k^k (1-k/r)^r} \leq \left(\frac{r}{k}\right)^k \left(1 + \frac{2k}{r}\right)^r,$$

for $1 \leq k \leq r/2$, and using the fact that $r < m/2$,

$$h(u,k) \leq \left(\frac{ru}{mk}\right)^k \left(1 - \frac{u}{m}\right)^{r-k} \left(1 + \frac{2k}{r}\right)^r \left(1 + \frac{2r}{m}\right)^r.$$

The function $f(x) = (M/x)^x$ achieves its maximum at $x = M/e$, therefore

$$h(u,k) \leq \exp\left(-\frac{u}{m}\left(r - k - \frac{r}{e}\right) + 4d \log r + \frac{2r^2}{m}\right) \leq r^{-d/3}$$

for r large enough. From (3.1) it follows that

$$p(r) \geq \left(1 - (d \log r + 1) r^{-d/3}\right) \pi(r) \geq \pi(r)/2. \tag{3.2}$$

Let S be a $(2r)$-sample of V and let $p(S,r)$ be the probability that, given a random r-sample R in S, there exists at least one edge $e \in \varphi(S)$ that is a witness for S and fails to intersect R. Let us now interpret $p(r)$ slightly differently. It is the probability that, given a random $(2r)$-sample S and a random r-sample $R \subseteq S$, there is at least one edge $e \in \varphi(R)$ that is a witness for S and fails to intersect R. Because φ is a homomorphism, we have

$$p(r) \leq \max\{\, p(S,r) \mid S \subseteq V,\ |S| = 2r \,\}. \tag{3.3}$$

Now, consider a $(2r)$-sample S for which edge e is a witness. A random r-sample in S will miss e entirely with probability at most

$$\binom{2r - 2d\log r}{r} \Big/ \binom{2r}{r} \leq (1 - d(\log r)/r)^r = o(1/r^{9d/8}).$$

But for the frame $\mathcal{F}$ to be of dimension d implies that

$$\left|\{\, S \cap e \mid e \in \varphi(S) \,\}\right| = O(r^d),$$

therefore $p(S, r) = o(1)$. The existence of an r-cover for $\mathcal{F}$ follows from (3.2) and (3.3). We summarize our results below.

Lemma 3.1 *Let $\mathcal{F}$ be a frame of dimension d with n vertices and let $r \leq n$ be any integer larger than some constant. If the threshold of $\mathcal{F}$ is at least $7d(\log r)/r$, then a random r-sample provides a cover for $\mathcal{F}$ with probability at least a half.*

Unfortunately, the existence proof falls short of even remotely suggesting a constructive method. If we were always allowed large samples, however, Lovász's *greedy cover algorithm* [13] would work just fine. The method is simple and it is reviewed below. Our strategy will be to modify the greedy algorithm in order to strenghten the covering quality of the computed sample. We will then iterate the modified algorithm a few times to find a small sequence of samples $R_1 \supseteq \cdots \supseteq R_k$ out of which an r-cover can be easily derived.

Until specified otherwise let us now consider $H = (V, E)$ to be an arbitrary hypergraph with n vertices and $O(n^d)$ edges, without any reference to a frame. Let α be a real (not necessarily a constant) such that $0 < \alpha < 1$, and assume that each edge of H has at least αn vertices. Lovász's greedy cover algorithm starts with the empty sample R. Then, it iterates on the following process, each time adding one vertex to the current sample R, until every edge of H intersects R.

1. Pick the vertex v of maximum degree and add it to the sample R.
2. Delete the vertex v from the current hypergraph as well as all the edges that contain it.

How big is the sample R? Let k be the number of edges in H and let k' be the number of edges in the resulting hypergraph after the first iteration. The number of removed edges, $k - k'$ is at least αk, therefore $k' \leq (1 - \alpha)k$. After $|R|$ iterations the hypergraph will be left with at most $(1 - \alpha)^{|R|} n^d$ edges, up to within a constant factor. This ensures termination within $|R| = O((\log n)/\alpha)$ steps. To implement the algorithm efficiently, we can compute all the vertex-degrees in preprocessing and then find the largest one. After removing the vertex in question, we delete its incident edges: for each of them we update the vector of vertex-degrees. The time required to do this can be charged to the removed edges, so the total running time is $O(n^{d+1})$.

If H were the hypergraph of a frame of dimension d then we could use the greedy algorithm to compute an r-cover, provided that the threshold of the frame exceeded $c(\log n)/r$, for some appropriate constant c. Since our goal is to provide r-covers for frames with thresholds proportional to $(\log r)/r$ and above, we must improve on the greedy algorithm (unless, of course, r is $\geq n^\varepsilon$, for

some fixed $\varepsilon > 0$). Once again, let us forget about frames temporarily and return to a general hypergraph H. Note that random samples of size $c(\log n)/\alpha$, for c large enough, work just as well as the greedy cover. The difference, however, is that the random sample works more uniformly. Indeed, it is easy to prove that for a proper setting of c a random r-sample intersects every edge of H in not just one but roughly $|e|r/n$ places.

To add this nice feature to the greedy cover algorithm we must modify the hypergraph a little and weight its edges. First of all, replace each edge e of size greater than $2\alpha n$ by edges of size between αn and $2\alpha n + 1$, which together partition e. With respect to the new hypergraph $H^+ = (V, E^+)$, we define the *weighted degree* of a vertex v as the sum of the weights assigned to the edges that contain v. The trick is now to refrain from throwing out edges which have already been hit, but instead, lower their weights. From the edges' point of view, this has the effect of sending this message to the algorithm:

> *It's all right to treat me as a kth-class citizen if I've been hit k times already (and k is less than a fraction of my size), but don't go thinking that I am dead (as the greedy cover algorithm would). For if you do, my disappearance will automatically impose a lower bound on the total amount of weight the hypergraph can ever hope to lose, which in turn will produce a contradiction.*

In some sense this approach is dual to the weighting strategy used in (Welzl [25]) for computing spanning trees of low stabbing number.

The Modified Greedy Cover Algorithm: Initialize each edge weight to 1 and R to the empty set. Iterate through steps 1 and 2 exactly $r = \min\{n, \lfloor 4(d+1)(\log n)/\alpha \rfloor\}$ times.

1. Pick the vertex v of maximum weighted degree (among vertices not yet picked) and add it to the sample R.
2. Divide by two the weight of every edge containing v. If the weight of any edge e falls below $1/\sqrt{2^{|e|}}$, then set it to 0.

Lemma 3.2. *Let d be an arbitrary positive real constant and let H be a hypergraph with n vertices and $O(n^d)$ edges. Let α be a real such that $0 < \alpha < 1$ and let $r = \min\{n, \lfloor 4(d+1)(\log n)/\alpha \rfloor\}$. Assume that each edge of H has at least αn vertices and that n is larger than some appropriate constant. Then the modified greedy cover algorithm will compute an r-sample that intersects every edge e of H in at least $c|e|r/n$ vertices, for $c = \frac{1}{25(1+1/d)}$; this represents a fixed fraction of the "expected" value. The running time is $O(rn^{d+1})$.*

Proof: We can obviously assume that $r < n$, and hence, $\alpha \geq (\log n)/n$. Let $w_i(e)$ be the weight of edge e after the ith step, with $w_0(e) = 1$, and let $W_i = \sum_{e \in E_i^+} |e| w_i(e)$, where $E_i^+ = \{ e \in E^+ \mid w_i(e) > 0 \}$. The weighted degree δ of the $(i+1)$st vertex chosen is at least $\sum_{e \in E_i^+} (|e| - h_i(e)) w_i(e)/(n-i)$, where $h_i(e)$ is the number of times the weight of edge e has changed. But $w_i(e) > 0$ implies that $w_i(e) \geq 1/\sqrt{2^{|e|}}$, therefore $|e| \geq 2h_i(e)$. It follows that

$$\delta \geq \sum_{e \in E_i^+} (|e| - h_i(e)) w_i(e)/n \geq \sum_{e \in E_i^+} \frac{|e| w_i(e)}{2n} = \frac{W_i}{2n}.$$

From step 2 of the algorithm we have

$$W_{i+1} \leq W_i - \frac{\delta}{2} \min_{e \in E^+} |e| \leq \left(1 - \frac{\alpha}{4}\right) W_i.$$

When the algorithm terminates we have $W_r \leq (1-\alpha/4)^r W_0$. Since W_0 is $\sum_{e \in E} |e| = O(n^{d+1})$ and $r < n$, we have

$$W_r \leq \exp\left(-\alpha r/4 + (d+1)\ln n + O(1)\right) \leq \frac{1}{n^{d/3}}. \tag{3.4}$$

If R intersects an edge $e \in E^+$ no more than h times then either $1/2^h < \sqrt{2^{|e|}}$, and hence $h > |e|/2$, or we have $w_r(e) \geq 1/2^h$ and

$$W_r \geq |e|/2^h \geq \alpha n/2^h,$$

so that

$$h \geq \log n + \log \alpha - \log W_r.$$

Since each edge $e \in E$ was divided into at least $\lceil |e|/(2\alpha n + 1) \rceil$ edges in E^+, the size of $R \cap e$ is at least either

$$\frac{\alpha n |e|}{4\alpha n + 2} \geq \frac{|e|}{6}$$

or, from (3.4),

$$\frac{(\log n + \log \alpha - \log W_r)|e|}{2\alpha n + 1} \geq \frac{d|e| \log n}{6\alpha n + 3} \geq \frac{|e| r}{25(1 + 1/d)n}.$$

It remains to analyze the complexity of the algorithm. The running time is clearly in $O(rn^{d+1})$, if we assume infinite precision. But, for practical reasons, we prefer to assume that the word-size is only, say, $\lceil \log n \rceil$. To achieve $O(rn^{d+1})$ time, it suffices to show how to compute the weighted degree of a given vertex in $O(n^d)$ time. This involves the addition of $O(n^d)$ numbers, each of the form $1/2^k$ $(0 \leq k \leq n)$. Each weight $1/2^k$ can be stored as k in a single word. In $O(n^d)$ time we can write the degree as a sum $\sum_{0 \leq k \leq n} a_k/2^k$, where a_k is a frequency count which satisfies $0 \leq a_k = O(n^d)$. To evaluate this sum, we add the numbers $a_k/2^k$ in the order $k = n, n-1, \ldots, 1, 0$. For that purpose, we use a virtual computer word made up of n real words (this is generous) and store the sum in binary by chunks of $\lceil \log n \rceil$ bits. To add $a_k/2^k$ to $\sum_{k+1 \leq j \leq n} a_j/2^j$, we find the position of the sole one-bit of 2^{-k} in the virtual word (in constant time) and start the addition right there. The integer a_k is stored over $O(1)$ real words, so we can complete the addition in constant time. The only source of worry is the possibility of long carries. But we have $\sum_{k+1 \leq j \leq n} a_j/2^j = O(n^d a_k/2^k)$, therefore no carry can extend over more than a constant number of real words. This completes the proof. ∎

We have restricted Lemma 3.2 to the case of hypergraphs with a polynomial number of edges only for the sake of the use which we will make of it below. Obviously, except perhaps for the complexity analysis, the result generalizes to any hypergraph of any size in a straightforward manner.

Theorem 3.3. *Let H be a hypergraph with n vertices and m edges, and let α be an arbitrary real between 0 and 1 such that each edge of H has at least αn vertices. There exist two constants b and $c > 0$ such that running the modified greedy cover algorithm r steps, for any $r \geq b(\log n + \log m)/\alpha$, will produce an r-sample that intersects every edge e of H in at least $c|e|r/n$ vertices, which represents a fixed fraction of the "expected" value.*

Let us now return to our frame $\mathcal{F}$ of dimension d and threshold α. Lemma 3.2 gives us a bootstrapping method for computing an r-cover. We iterate the modified greedy algorithm three times to get samples $R_1 \supseteq R_2 \supseteq R_3$, from which we derive R. The details follow. As we have observed, the hypergraph H has $O(n^d)$ edges, therefore we can apply the theorem to it. We obtain a sample R_1 of size

$$r_1 = \min\Big\{n, \lfloor 4(d+1)(\log n)/\alpha \rfloor\Big\}.$$

We know that R_1 intersects every edge e of the frame in at least $c|e|r_1/n$ vertices. If $r_1 \leq r$ then we stop (we'll justify this later). Otherwise, we extract the sub-hypergraph $H_1 = (R_1, E_1)$, where $E_1 = \{\, R_1 \cap e \,|\, e \in \varphi(R_1) \,\}$. Let φ_1 be the H_1-induced restriction of φ: for each $W \subseteq R_1$, $\varphi_1(W) = \{\, R_1 \cap e \,|\, e \in \varphi(W) \,\}$. We easily verify that $\mathcal{F}_1 = (H_1; \varphi_1)$ is also a frame of dimension d, so we can apply Lemma 3.2 to it (replacing α by $c\alpha$). This provides us a sample $R_2 \subseteq R_1$ of size

$$r_2 = \min\Big\{r_1, \lfloor 4(d+1)(\log r_1)/(c\alpha) \rfloor\Big\}.$$

If $r_2 \leq r$, again we stop. Otherwise, we iterate through the same process one more time, now with respect to the hypergraph $H_2 = (R_2, E_2)$. In general, for all $i > 1$, we have $\mathcal{F}_i = (H_i; \varphi_i)$, where $H_i = (R_i, E_i)$, $E_i = \{\, R_i \cap e \,|\, e \in \varphi_{i-1}(R_i) \,\}$, and for each $W \subseteq R_i$, $\varphi_i(W) = \{\, R_i \cap e \,|\, e \in \varphi_{i-1}(W) \,\}$. Returning to our iteration, this will produce a sample $R_3 \subseteq R_2$ of size

$$r_3 = \min\Big\{r_2, \lfloor 4(d+1)(\log r_2)/(c^2\alpha) \rfloor\Big\}.$$

If $r_3 \leq r$, as usual, we stop. Else, we compute an r-cover for $\mathcal{F}_3$ by trying out all possible r-samples of R_3 until we are successful. (We shall assign α below so that success is *always* guaranted and the cost of an exhaustive search is very small.)

To summarize, either we get an r-cover for $\mathcal{F}_3$ or we obtain one sample R_i $(1 \leq i \leq 3)$ which intersects every edge of H_{i-1}, where $|R_i| \leq r < |R_{i-1}|$ (setting $H_0 = H$ and $R_0 = V$). An important remark is that in the last three cases we can augment the sample which we have obtained into an r-cover for $\mathcal{F}$. Indeed, precisely because R_i intersects every edge of H_{i-1}, we can add any vertex of R_{i-1} into R_i without changing the fact that R_i is a cover for $\mathcal{F}_{i-1}$. But a cover for any $\mathcal{F}_j$ is also a cover for $\mathcal{F}$, which proves our claim. Now, how do we assign α to guarantee the success of the exhaustive search for an r-cover? We easily check that $r_3 \leq b\left(\log\log\log n + \log\frac{1}{\alpha}\right)/\alpha$, for some constant b. Setting

$$\alpha = \frac{7d\log r}{c^3 r} + \frac{2b}{r}\log\frac{r}{2b},$$

we ensure that $c^3\alpha > 7d(\log r)/r$. Since the threshold of $\mathcal{F}_3$ is at least $c^3\alpha$, Lemma 3.1 guarantees the existence of the desired r-cover for $\mathcal{F}_3$. What is the complexity of the algorithm? If we stop with R_1, R_2, or R_3, the running time $O(rn^{d+1})$, as indicated by Lemma 3.2. Assume that we end up computing an r-cover for $\mathcal{F}_3$. We can verify that

$$r_3 \leq \left(\frac{1}{2} + \frac{bc^3\log\log\log n}{7d\log r}\right) r.$$

Since $r < r_3$ it follows that $r = (\log\log n)^{O(1)}$ and therefore $r_3 = (\log\log n)^{O(1)}$. Computing the r-cover of $\mathcal{F}_3$ takes no more than $O(r_3^{r+d+1}) = O(n)$ time. This completes the proof of Theorem 2.1.

4. Multi-Hypergraphs with Bounded Vertex Dependency

So far, we have been willing to accept deficiencies as large as $\log r$. But are those covers optimal? The answer is yes, if we limit ourselves to hypergraphs of large, yet finite, *VC*-dimension (Pach [16]). In this section we look at multi-hypergraphs of bounded vertex dependency, where the situation is a bit different. We cannot quite eliminate the factor $\log r$, but we present a scheme which will turn out to have the same effect in the geometric applications discussed later.

Let $H = (V, E)$ be a multi-hypergraph of n vertices and m edges. We shall assume that H has bounded vertex dependency. This, we recall, means that each edge e has a signature $\emptyset \subset \sigma(e) \subseteq V$. For each edge $e \in E$ and any $W \subseteq V$, we have $|\sigma(e)| \leq d$ and $|\sigma^{-1}(W)| \leq c$, for some constants c and d. The domain, $H(R)$, of an r-sample R is the collection of edges whose signatures are subsets of R. An edge e is called *t-deficient* if (i) $e \in H(R)$, (ii) $R \cap e = \emptyset$, and (iii) $|e| \geq tn/r$. Given a subset $W \subseteq V$ and a random r-sample R (from the uniform distribution over the set of r-samples) we are interested in the conditional expectation

$$\mu(W; r, t) = E\big[\,\#\,t\text{-deficient edges in } H \,\big|\, R \supseteq W\,\big],$$

defined if $r \geq |W|$. Let $E(W; t)$ be the multiset of edges $e \in E$ of size $\geq tn/r$ such that $e \cap (\sigma(e) \cup W) = \emptyset$. The following expression will be useful in the next two lemmas:

$$\mu(W; r, t) = \sum_{e \in E(W;t)} \frac{\binom{n-\tilde{r}-\tilde{\sigma}}{r-\tilde{r}-\tilde{\sigma}}}{\binom{n-\tilde{r}}{r-\tilde{r}}} \left(1 - \frac{|e|}{n-\tilde{r}-\tilde{\sigma}}\right)\left(1 - \frac{|e|}{n-\tilde{r}-\tilde{\sigma}-1}\right)\cdots\left(1 - \frac{|e|}{n-r+1}\right), \quad (4.1)$$

where $\tilde{r} = |W|$ and $\tilde{\sigma} = |\sigma(e) \setminus W|$. Each summand is to be understood as 0 if $r < \tilde{r} + \tilde{\sigma}$ or $r > n - |e|$.

Lemma 4.1. *The inequality $\mu(\emptyset; r, t) < 2cr^d e^{-t/2}$ holds for any t, r such that $2d < r \leq n$.*

Proof: From (4.1) and the fact that $\binom{a-c}{b-c}/\binom{a}{b} \leq (b/a)^c$, we derive

$$\mu(\emptyset; r, t) \leq \sum_{f \in E(\emptyset;t)} \left(\frac{r}{n}\right)^{|\sigma(f)|} \left(1 - \frac{|f|}{n}\right)^{r-|\sigma(f)|} \leq \sum_{f \in E} \left(\frac{r}{n}\right)^{|\sigma(f)|} e^{-t(r-|\sigma(f)|)/r}.$$

Rewriting the sum $\sum_{f \in E} (r/n)^{|\sigma(f)|}$ as $\sum_{1 \leq j \leq d} (r/n)^j \times (\#$ edges $f \in E$ such that $|\sigma(f)| = j)$, we find that

$$\mu(\emptyset; r, t) \leq e^{-t/2} \sum_{1 \leq j \leq d} (r/n)^j c n^j,$$

which completes the proof. ∎

Lemma 4.2. *The function $\mu(W; r, t)$ can be evaluated for any r and t within an absolute error of $1/n^2$ in $O(n^{d+1})$ time. With $O(n^{d+1})$ preprocessing it is possible to derive $\mu(W \cup \{v\}; r, t)$ from $\mu(W; r, t)$ in $O(n^d)$ time.*

Proof: We can rewrite (4.1) as

$$\mu(W;r,t) = \sum_{e \in E(W;t)} \binom{n-\tilde{r}-\tilde{\sigma}-|e|}{r-\tilde{r}-\tilde{\sigma}} \Big/ \binom{n-\tilde{r}}{r-\tilde{r}},$$

or in other words,

$$\mu(W;r,t) = \sum_{e \in E} \delta(W;t,e) \sum_{1 \le i \le 6} f_i(n,r,|e|,|W|,|\sigma(e) \setminus W|),$$

where $\delta(W;t,e)$ is the characteristic function of the multiset $E(W;t)$ and each f_i is a factorial (or its reciprocal) involving a simple arithmetic expression on its parameters. All the parameters and the values of δ can be computed in $O(n^{d+1})$ time. If we precompute $j!$ and its reciprocal for $j = 1,2,\ldots,n$, then we can complete the computation of $\mu(W;r,t)$ in $O(n^d)$ time. If the word-size is around $\log n$, we can compute each factorial and reciprocal in constant time with a relative error of at most $1/n^{2d+4}$. This gives us a relative error on the computed value of $\mu(W;r,t)$ of less than $1/(n^{d+3})$. Since the exact value does not exceed cn^d, the absolute error is less than $1/n^2$. To compute $\mu(W \cup \{v\};r,t)$ incrementally from $\mu(W;r,t)$ in $O(n^d)$ time it suffices to observe that the parameters and the values of δ in each summand can be updated in constant time. ∎

From Lemma 4.1 we derive $\mu(\emptyset;r,2\lceil d \log r\rceil) < 1/2$, for r larger than some constant. Therefore, with probability greater than a half, a random r-sample R hits every edge $e \in H(R)$ of size $|e| > 3dn(\log r)/r$ vertices. Using the technique of *conditional probabilities* (Raghavan [18], Spencer [21]), we remove the randomization by computing the sample R incrementally. Starting from $W = \emptyset$, we observe that as long as $|W| < r$, there exists $v \in V \setminus W$ such that $\mu(W \cup \{v\};r,t) \le \mu(W;r,t)$. This is because

$$\mu(W;r,t) = \frac{1}{n-|W|} \sum_{v \in V \setminus W} \mu(W \cup \{v\};r,t).$$

If we could compute such a vertex v we would add it to our partial sample W, and iterate in this fashion until $|W| = r$. In the end, we would have $\mu(R;r,t) < 1/2$, which is to say that there are no t-deficient edges in $H(R)$. Because of finite precision, however, we must content ourselves with an approximation, $\tilde{\mu}(W;r,t)$, of $\mu(W;r,t)$. From Lemma 4.2 we can choose v such that $\tilde{\mu}(W \cup \{v\};r,t) \le \mu(W;r,t) + 1/n^2$; for this v we know that $\mu(W \cup \{v\};r,t) \le \mu(W;r,t) + 2/n^2$. This method therefore finds an R with $\mu(R;r,t) \le \mu(\emptyset;r,t) + 2r/n^2$. Setting $t = 2\lceil d \log r\rceil$ we find that $\mu(R;r,t) \le 1/2 + 2r/n^2 < 1$, for r large enough (Lemma 4.1). The time complexity follows directly from Lemma 4.2.

Lemma 4.3. *In $O(rn^{d+1})$ time it is possible to compute an r-sample R that leaves no edge of the hypergraph $(3d \log r)$-deficient, for any r larger than some constant.*

This result is really no different from Theorem 2.2. Only the technique is: it sets the stage for the computation of better covers. Ideally, we would like to say that as long as an edge is of size $> bn/r$, for some constant $b > 0$, a random r-sample which contains its signature will hit it almost certainly. Such a statement requires looking further into the geometry of the specific applications. In the meantime, we will finetune our analysis of t-deficient edges.

Lemma 4.4. *There exists a constant $b > 1$ such that the inequality $\mu(\emptyset; r, t) \leq \mu(\emptyset; \lfloor r/t \rfloor, 0)/2^t$ holds for any t, r, n, where $b \leq dt \leq r \leq n$.*

Proof: We shall assume that $b \leq dt \leq r \leq n$, for an appropriately large constant b. Let e be a fixed edge of H and let R be a random r-sample. We define $p(e; r, t)$ as the conditional probability that e is t-deficient, given that $e \in H(R)$. Assume that e and its signature $\sigma(e)$ are disjoint. We have

$$\binom{n - |\sigma(e)|}{r - |\sigma(e)|}\binom{n - |e| - |\sigma(e)|}{\lfloor r/t \rfloor - |\sigma(e)|}p(e; r, 0) = \binom{n - |e| - |\sigma(e)|}{r - |\sigma(e)|}\binom{n - |\sigma(e)|}{\lfloor r/t \rfloor - |\sigma(e)|}p(e; \lfloor r/t \rfloor, 0),$$

from which we derive

$$\binom{n - \lfloor r/t \rfloor}{r - \lfloor r/t \rfloor}p(e; r, 0) = \binom{n - \lfloor r/t \rfloor - |e|}{r - \lfloor r/t \rfloor}p(e; \lfloor r/t \rfloor, 0).$$

Note that the equality still holds if $e \cap \sigma(e) \neq \emptyset$, since we then have $p(e; r, 0) = p(e; \lfloor r/t \rfloor, 0) = 0$. It follows that

$$p(e; r, 0) \leq \left(1 - \frac{|e|}{n - \lfloor r/t \rfloor}\right)^{r - \lfloor r/t \rfloor} p(e; \lfloor r/t \rfloor, 0) \leq \exp\left(\frac{-|e|(r - \lfloor r/t \rfloor)}{n - \lfloor r/t \rfloor}\right) p(e; \lfloor r/t \rfloor, 0).$$

Assume that $|e| \geq tn/r$. Since $p(e; r, t)$ is non-increasing in t we derive that, for b large enough,

$$p(e; r, t) \leq p(e; \lfloor r/t \rfloor, 0)/(2.5)^t. \tag{4.2}$$

We use 2.5 and not 2 to give us a little breathing room for later. Note that if $|e| < tn/r$ and therefore e is not t-deficient, (4.2) holds trivially. Now we know that (for all $t \geq 0$)

$$\mu(\emptyset; r, t) = \binom{n}{r}^{-1} \sum_{e \in E} \binom{n - |\sigma(e)|}{r - |\sigma(e)|} p(e; r, t). \tag{4.3}$$

We easily verify that for b large enough

$$\binom{n - |\sigma(e)|}{r - |\sigma(e)|} \Big/ \binom{n}{r} \leq (dt)^d \binom{n - |\sigma(e)|}{\lfloor r/t \rfloor - |\sigma(e)|} \Big/ \binom{n}{\lfloor r/t \rfloor}. \tag{4.4}$$

From (4.3) and (4.4) we derive

$$\mu(\emptyset; r, t) \leq (dt)^d \binom{n}{\lfloor r/t \rfloor}^{-1} \sum_{e \in E} \binom{n - |\sigma(e)|}{\lfloor r/t \rfloor - |\sigma(e)|} p(e; r, t),$$

and from (4.2) and (4.3)

$$\mu(\emptyset; r, t) \leq (dt)^d (2.5)^{-t} \mu(\emptyset; \lfloor r/t \rfloor, 0).$$

We complete the proof by choosing b, and hence t, large enough. ∎

Assume now that H is *robust*, meaning that there exists a constant $a > 0$ such that $\mu(\emptyset; \lfloor r/t \rfloor, 0) \leq a\mu(\emptyset; r, 0)$, for any $t > a$. Robustness may seem to be rather unnatural, since it appears to tell us that a small random sample causes no more deficiencies than a big one. This may actually happen, if the set $H(R)$ of relevant edges grows fast with the size of R. Hypergraphs originating from geometric range spaces tend to be robust, as we shall see later in the applications section.

It follows from Lemma 4.4 that, for a constant α large enough and any t, r, n ($a < \alpha \leq dt \leq r \leq n$), we have $\mu(\emptyset; r, t) \leq \alpha\mu(\emptyset; r, 0)/2^t$. Since μ is non-increasing in t, by increasing the value of α if necessary, we can always extend the validity of the previous inequality to any $t \geq 1$. Also, we can extend it to any $t \leq r$ by writing

$$\mu(\emptyset; r, t) \leq \frac{\alpha}{2^{t/d}}\mu(\emptyset; r, 0).$$

Given $W \subseteq V$, we are now interested in

$$\Phi(W; r) = \alpha\mu(W; r, 0) - \sum_{1 \leq t \leq r} 2^{t/(2d)}\mu(W; r, t).$$

From the previous discussion we can re-adjust α one more time, so that $\Phi(\emptyset; r) \geq 0$, for all r, n ($\alpha \leq r \leq n$). Given a random r-sample R, let $\chi(t)$ be the number of t-deficient edges. The quantity $\Phi(W; r)$ is the expected value of $\alpha\chi(0) - \sum_{1 \leq t \leq r} 2^{t/(2d)}\chi(t)$, conditioned on $W \subseteq R$. To say that it is nonnegative implies the existence of an r-sample $R \supseteq W$ such that for any t ($1 \leq t \leq r$) $\chi(t) \leq \alpha\chi(0)/2^{t/(2d)}$. Because of finite-precision problems we weaken this condition a little and say that an r-sample R is *conformal* (where α is understood) if

$$\chi(t) \leq \frac{\alpha\chi(0) + 1}{2^{t/(2d)}},$$

for $t = 1, 2, \ldots, r$. Knowing that $\Phi(\emptyset; r) \geq 0$ tells us that a conformal r-sample exists. Knowing that $\Phi(W; r) \geq 0$, where $|W| = r$, tells us that W is conformal.

Once again, we can apply the method of conditional probabilities to make the construction of a conformal r-sample effective. We begin with $W = \emptyset$. As long as $|W| < r$, we have

$$\Phi(W; r) = \frac{1}{n - |W|} \sum_{v \in V \setminus W} \Phi\big(W \cup \{v\}; r\big),$$

which implies the existence of $v \in V \setminus W$ such that $\Phi\big(W \cup \{v\}; r\big) \geq \Phi(W; r)$. Starting with $\Phi(\emptyset; r) \geq 0$ we thus end up with $\Phi(R; r) \geq 0$. Because of finite precision, however, we use the approximation of μ discussed in Lemma 4.2. The relative error on μ is at most $1/\big(n^{d+3}\big)$, therefore the absolute error on Φ is $O\big(1/n^2\big)$. For n large enough, the resulting sample R will satisfy $\Phi(R; r) \geq -1/n$, and therefore will be conformal.

Computing $\Phi(W; r)$ can be done by evaluating μ at r points, which takes $O\big(rn^{d+1}\big)$ time (Lemma 4.2). From there we compute a conformal r-sample in $O\big(r^2n^{d+2}\big)$ time. We can do better, however. Assume that we have computed $\Phi(W; r)$, and in particular, $\mu(W; r, 0)$ and that we wish to evaluate $\Phi\big(W \cup \{v\}; r\big)$. We can get $\mu\big(W \cup \{v\}; r, 0\big)$ in $O\big(n^d\big)$ time (Lemma 4.2). Going back to the

expression of $\mu(W;r,t)$ used in the proof of Lemma 4.2, we can write $\mu(W;r,t) = \mu(W;r,t-1)+\Delta(t)$, where

$$\Delta(t) = \sum_{e\in E}\big(\delta(W;t,e) - \delta(W;t-1,e)\big) \times \sum_{1\le i\le 6} f_i(n,r,|e|,|W|,|\sigma(e)\setminus W|).$$

However, $\big(\delta(W;t,e) - \delta(W;t-1,e)\big)$ is nonzero for at most one value of t, therefore each $\Delta(t)$ $(1 \le t \le r)$ can be computed in $O(n^d/r)$ amortized time. This allows us to compute $\Phi(W\cup\{v\};r)$ in $O(n^d)$ time, which gives us a running time of $O(rn^{d+1})$ time for computing a conformal r-sample.

Lemma 4.5. *If H is robust and r is large enough, it is possible to compute a conformal r-sample in time $O(rn^{d+1})$ time.*

We are getting closer to our goal of wiping out t-deficiency for any $t > 0$. To avoid a major roadblock, however, we must expand the definition of a sample. First, if W is a subset of V we define H^W to be the signature-induced restriction of H to the vertices of W. More precisely, the edges of H^W are of the form $e\cap W$, for all $e \in H(W)$. We can regard H^W as just another multi-hypergraph of bounded vertex dependency; note that the constants c and d used for H apply just the same. We define an *r-blossom* to be a collection of samples $B = (R, R_1, \ldots, R_k)$ such that (i) $|R| = r$, (ii) $k \le |H(R)|$ (counted as a multiset) and (iii) for any edge e that is 1-deficient with respect to R, there is an integer $\ell(e)$ such that the sample $R_{\ell(e)}$ is a subset of e that intersects every edge f of $H^e(R_{\ell(e)})$ of size $|f| \ge \beta n/r$, for some fixed constant $\beta > 1$. The sample R is called the *base* of the blossom. The *weight* of B refers to the total number of 0-deficient edges in H^e (with respect to $R_{\ell(e)}$), over all edges e in E.

Theorem 4.6. *If H is robust and r is large enough, it is possible to compute an r-blossom of weight at most proportional to the number of 0-deficient edges in H (with respect to the base of B). The running time is $O(rn^{d+1})$ time.*

Proof: The first item on the agenda is to compute a conformal r-sample R. To complete it into an r-blossom, we go through each 1-deficient edge e in turn and perform the following operations. First, we compute the restriction H^e explicitly. Let $s = \lceil |e|r/n + r_0\rceil^2$, where r_0 is the constant of Lemma 4.3. If $s \le |e|$ then from the same lemma we can compute a s-sample R' of e that leaves H^e free of $(3d\log s)$-deficient edges (*with respect to H^e and R'*). We can verify that R' fits the role of $R_{\ell(e)}$. Indeed, let f be an edge of $H^e(R')$ of size $|f| \ge \beta n/r$. To see that f intersects R', it suffices to check that $3d(\log s)|e|/s \le \beta n/r$, for β large enough. If $s > |e|$ then we choose e as the sample. The collection of samples obtained in the process constitutes the r-blossom B. What is its weight? It is zero if there are no 0-deficient edges, so we can assume that $\chi(0) > 0$. Recall that $\chi(t)$ denotes the number of t-deficient edges in H (with respect to R). We have

$$\text{weight}\,(B) \le \sum_{1\le t\le r} c(t+r_0)^{2d}\chi(t-1).$$

Because R is conformal and $\chi(0) > 0$,

$$\text{weight}\,(B) \le \sum_{1\le t\le r} c(t+r_0)^{2d}\big(\alpha\chi(0)+1\big)/2^{\frac{t-1}{2d}} = O\big(\chi(0)\big).$$

From Lemmas 4.3 and 4.5 we find that the running time is $O(n^{2d+2})$. But again using the geometrically decaying nature of t-deficient edges, we can finetune the analysis. Computing R takes $O(rn^{d+1})$ time. For each edge e that is 1-deficient, we can compute the hypergraph H^e in $O(|e|^{d+1})$ time with proper preprocessing. Computing the sample $R_{\ell(e)}$ takes $O(s|e|^{d+1})$, where $s = \lceil |e|r/n + r_0 \rceil^2$. Because R is conformal the total cost is at most proportional to

$$rn^{d+1} + \sum_{1 \le t \le r} \left(\frac{tn}{r}\right)^{d+1} (t + r_0)^2 (\alpha\chi(0) + 1)/2^{\frac{t-1}{2d}}.$$

Since the number of 0-deficient edges in H (with respect to R) is at most cr^d, the computation of the blossom takes $O(rn^{d+1})$. ∎

5. Random Sampling and Computational Geometry

We will now illustrate the utility of our techniques by describing an improved method for geometric divide-and-conquer and looking at higher-dimensional point location. The problem is this: Given an arrangement of n hyperplanes in d-space and a query point q, find in which face of the arrangement the point q lies. We will show that with polynomial preprocessing and $O(n^d)$ space, any query can be answered in $O(\log n)$ time. This improves upon Clarkson's solution [3] by a factor of n^ε space, not to mention, of course, the tractability of the preprocessing. Much of our solution is based on an improved method for geometric divide-and-conquer. This is a problem of independent interest, which we thus treat separately.

5.1. Geometric Divide-and-Conquer

Given n hyperplanes in d-space and any integer r larger than some constant, we wish to subdivide d-space into $O(r^d)$ simplices, none of which intersects more than bn/r hyperplanes, for some constant b. A slightly weaker version of the theorem (where bn/r is to be replaced by $bn(\log r)/r$) is used in (Clarkson [2,3]) and (Edelsbrunner at al [8,9]). The basic idea is to pick a random subset R of r hyperplanes and argue that, with high probability, each simplex of *any* triangulation $\mathcal{T}$ of R intersects $O(n(\log r)/r)$ hyperplanes. Theorem 2.2 can be readily applied to make this method deterministic. To remove the factor $\log r$, however, we must use a more sophisticated approach based on conformal samples. One difficulty in trying to use conformality is that statements of the form, "*the number of simplices in $\mathcal{T}$ that intersect more than kn/r hyperplanes decreases geometrically with k*" applies to the entire set of all possible triangulations and not to each individual one. This justifies the introduction of a special type of triangulation.

We begin with a quick review of fundamental geometric concepts. Details can be found in (Clarkson [2], Edelsbrunner [7]). A set is ***polyhedral*** if it is the intersection of a finite number of closed halfspaces. For convenience, we define a ***k-simplex*** as the relative interior of a polyhedral set of dimension k with $k+1$ vertices. An ***arrangement*** $\mathcal{A}(K)$ of a finite collection K of hyperplanes in E^d is the cell complex induced by the hyperplanes. Each element of the complex is the relative

interior of some polyhedral set bounded by hyperplanes in K: it is called a k-face if its dimension is k. A cell complex is *simplicial* if its elements are simplices. (We shall use the word "simplex" when no assumption is made about dimensionality.) A triangulation of d-space is a simplicial cell complex (i.e., a complex composed exclusively of simplices) that covers E^d. A triangulation of $\mathcal{A}(K)$ refers to a simplicial cell complex which subdivides the arrangement $\mathcal{A}(K)$. To handle unbounded faces uniformly we embed the arrangement in projective d-space. Again, for convenience, we might want to avoid wrapping around infinity by adding the ideal hyperplane into the arrangement itself. (This is topologically equivalent to intersecting the arrangement with a large d-ball.) In this way, unbounded faces of $\mathcal{A}(K)$ are no different from bounded ones.

A convenient way to triangulate $\mathcal{A}(K)$ (Clarkson [2]) is to proceed recursively and first triangulate the n arrangements in $(d-1)$ space formed by intersecting each hyperplane with the $n-1$ others. Then for each cell (i.e., the closure of a d-face) F of $\mathcal{A}(K)$, pick a vertex v (i.e., a 0-face), called the *apex*: for each $(d-1)$-face f of F that is not incident upon v, and for each $(d-1)$-simplex s in f computed recursively, add the (relative) interior of the convex hull of $s \cup \{v\}$ to the triangulation. In this way, every d-simplex is formed from the convex hull of its apex and a *base* $(d-1)$-simplex. This works well as long as F is guaranteed to have at least one vertex, which is immediately implied if F does not contain a whole line. If it does contain a line, then we can rewrite F as the sum of $A+G$ of an affine space A and a certain polyhedral set G ($\dim(G) < d$) which does have a vertex. We can therefore triangulate G and use the "simplices" $\{A+t \mid t \in G\}$. We can also avoid these problems by submitting the input to a random projective perturbation. (Although this might come back to haunt us if we are shooting for a deterministic algorithm...) To simplify our discussion, we shall assume that all the hyperplanes considered are in general position. This applies to the hyperplanes in d-space given as input, as well as their various intersections in lower-dimensional space. Relaxing these assumptions is easy but tedious. We also give ourselves a Cartesian system of coordinates $(x_1, \ldots, x_d)$ and assume that the union of the input and the d fundamental hyperplanes, $x_i = 0$, is in general position, in the sense that any intersection of k input hyperplanes is a $(d-k)$-flat (not at infinity) which inherits a local system of coordinates from $(x_1, \ldots, x_{d-k})$. We now close this excursion and refer the reader to (Clarkson [2]) for proofs of some of the geometric facts which we just mentioned.

We are now ready to introduce the special type of triangulation to which we alluded earlier. Our starting point is the observation that the simplices of the triangulation described in the previous paragraph are entirely specified by a finite number $f(d)$ of hyperplanes of K. How many? at most 2 in 1-space, 5 in 2-space, $d+1+f(d-1)$ in d-space, which gives $f(d) = d(d+3)/2$. We define a *universal triangulation scheme* τ to be a mapping of any set S of at most $d(d+3)/2$ hyperplanes in (projective) d-space to a finite (possibly empty) set of simplices. The elements of $\tau(S)$ are k-simplices for values of k ranging from 0 to d. Given *any* finite collection K of hyperplanes in d-space and a subset S of K of size $\leq d(d+3)/2$, let $T_\tau(K,S)$ be the set (not a multiset) of simplices in $\tau(S)$ that *do not* intersect any hyperplane of K, except for the hyperplanes that contain them. By extension, we define $T_\tau(K)$ as the union of the sets $T_\tau(K,S)$, over all subsets S of K of size at most $d(d+3)/2$. We require that $T_\tau(K)$ form a triangulation of $\mathcal{A}(K)$. A universal triangulation scheme can be regarded as two-step process for triangulating an arrangement: (1) lay out candidate

simplices by checking local conditions; (2) keep empty candidates. The nice feature of this approach is that the simplices do not have to be tested for intersection among themselves. It is not even clear at first that a universal triangulation scheme should always exist. But it does.

Lemma 5.1. *There exists a universal triangulation scheme in any finite dimension.*

Proof: Let S be a collection of $\leq d(d+3)/2$ hyperplanes in projective d-space and compute its arrangement $\mathcal{A}(S)$ in full, using (Edelsbrunner at al. [10]). Triangulate $\mathcal{A}(S)$, using the following criterion for picking the apex of a face. Choose the vertex of least x_1-coordinate (break ties by using x_2, x_3, etc.) With the use of homogeneous coordinates, this vertex is always well defined (uniquely), even if it lies at infinity. This is called the *canonical triangulation* of S. We define $\tau(S)$ as the collection of simplices in that canonical triangulation. Is τ a valid universal triangulation scheme? We will show that $T_\tau(K)$ itself is the canonical triangulation of K. First, it is clear that any simplex of the canonical triangulation belongs to $\tau(S)$, for some subset S of size $\leq d(d+2)/3$. Moreover, by construction, no simplex of $T_\tau(K)$ can intersect a hyperplane of K, except if it is entirely contained in it.

The last thing to check is that no two distinct simplices $\sigma, \sigma' \in T_\tau(K)$ can intersect. Suppose that they do and let p be a point of the intersection. Our first observation is that σ and σ' have the same dimension k. Suppose that σ is of lesser dimension. Then, p lies in an intersection of $\text{codim}(\sigma)$ hyperplanes, which intersects but does not contain σ'. Therefore one of these hyperplanes must also intersect σ' without containing it, which is a contradiction. It is also easy to see that σ and σ' have the same apex q, for otherwise the simplices would lie in two distinct k-faces of $\mathcal{A}(K)$, because a face can have only one apex. But that is impossible without contradicting the absence of intersection between simplices and non-containing hyperplanes. Now consider the ray from p pointed in the direction $\overrightarrow{qp}$ and let r be the point at which it leaves the intersection of the two simplices. The point r lies on the base simplices of σ and σ'. Furthermore, since p lies in a nonempty open set, what we just said holds true of any p taken in a little k-ball. Consequently, the two base simplices lie on the same $(k-1)$-flat.

We have now set the grounds for a proof by induction on the dimension k of the simplices. If $k = 1$ then we conclude that σ and σ' have no choice left but to be identical, which is a contradiction. If $k > 1$, we observe that the base $(k-1)$-simplices of σ and σ' intersect in r. Since σ and σ' have the same apex, yet are distinct, the base simplices must be distinct. Furthermore, both of them belong to $T_\tau(K)$. The induction hypothesis provides the contradiction. ∎

From now on we use the mapping τ of Lemma 5.1 to supply us with a universal triangulation. Let K be a collection of n hyperplanes in d-space and let $H = (V, E)$ be a multi-hypergraph, with V and E in bijection with, respectively, K and

$$\bigcup\Big(\tau(S) \,|\, S \subseteq K \text{ and } |S| \leq d(d+3)/2\Big).$$

Thus, each edge e is associated with a unique simplex $s(e)$ of $\tau(S)$ (of dimension between 0 and d), where S is a subset of K of size at most $d(d+3)/2$. The subset of vertices in correspondence with

S constitutes the signature $\sigma(e)$ of e. We have $|\sigma(e)| \leq d(d+3)/2$ and $|\sigma^{-1}(W)| = O(d^{2d}) = O(1)$. An edge e is defined as the set of vertices whose corresponding hyperplanes intersect the simplex $s(e)$ but do not contain it.

By construction, H has bounded vertex dependency. Our next step thus is to call upon Theorem 4.6 and compute an r-blossom $B = (R, R_1, \ldots, R_k)$ of weight at most proportional to the number of 0-deficient edges in H. According to the theorem this can be done in $O(rn^{d(d+3)/2+1})$ time. From the main property of a universal triangulation scheme, it appears that an edge e of H is 0-deficient with respect to R if and only if its associated simplex $s(e)$ belongs to the triangulation $T_\tau(R)$. Conversely, every simplex of $T_\tau(R)$ corresponds to at least one 0-deficient edge. If no simplex of $T_\tau(R)$ is 1-deficient, then we have achieved our goal. (By abuse of language, we use R to denote either the r-sample of vertices or its corresponding set of r hyperplanes in K; also, we say that a simplex $s(e)$ is k-deficient to mean that the edge e is so.) If there is at least one 1-deficient edge, then we turn to the samples $R_1, \ldots, R_k$ to take care of undesirable deficiencies.

Let e be the edge of H associated with R_i. Because e is 1-deficient, more than n/r hyperplanes of K intersect the simplex $s(e)$ without containment. It is the purpose of R_i to subdivide $s(e)$. To do so, we compute the intersection of $T_\tau(R_i)$ and $s(e)$, which has the effect of subdividing $s(e)$ into polyhedral sets of bounded size. We can make each set simplicial without multiplying the overall size by more than a constant factor. This gives us a simplicial complex S_i subdividing the closure of $s(e)$. Now remove from S_i all the simplices lying on the boundary of $s(e)$. Doing this for every 1-deficient edge gives us, along with $T_\tau(R)$, a simplicial subdivision P of $\mathcal{A}(R)$. Note that several edges may map to the same simplex $s(e)$, and we might want to keep this in check to avoid unnecessary duplication of work. The subdivision P is a packing (no two simplices intersect) and a covering (the union is d-space). However, it might not be a complex. Computing P can be done within the same time as computing the blossom itself, that is, $O(rn^{d(d+3)/2+1})$ time. Indeed, our computation of the blossom (Theorem 4.6) involves computing the simplices of $T_\tau(R)$ and of each $T_\tau(R_i)$. Clipping the latter, re-triangulating them, and forming the complexes S_i adds only a constant factor to the running time. Note that we can also compute the packing directly from the blossom, if we prefer.

What else can we say about P? The packing consists of simplices, every one of which lies entirely in some simplex of $T_\tau(R)$. We distinguish among the simplices of $T_\tau(R)$ (*first-generation*) and those of the S_i's (*second-generation*). Recall that the dimension of those varies between 0 and d. From Theorem 4.6 we derive that each simplex of P intersects (without containment) $O(n/r)$ hyperplanes of K. The size of P is proportional to the size of $T_\tau(R)$, added to the weight of the blossom. But because R is conformal, this amounts to the number of 0-deficient edges, which is $O(r^d)$. We conclude that, given n hyperplanes in d-space and any integer r large enough, there exists a simplicial packing of size $O(r^d)$, each of whose simplices intersects $O(n/r)$ hyperplanes. The packing can be computed in $O(rn^{d(d+3)/2+1})$ time.

Theorem 5.2. *Given n hyperplanes in d-space and any integer r large enough, there exists a simplicial packing of size $O(r^d)$ which covers d-space, each of whose simplices intersects $O(n/r)$ hyperplanes. The packing can be computed in $O(rn^{d(d+3)/2+1})$ time.*

5.2. Point Location Among Hyperplanes

Let K be a set of n hyperplanes in d-space in general position. Preprocess K so that any query point q can be located in the arrangement $\mathcal{A}(K)$ in $O(\log n)$ time. By locating the point, we mean finding the unique face of the arrangement $\mathcal{A}(K)$ that contains q. The storage allowed is $O(n^d)$. It is clearly sufficient to locate q in the triangulation $T_\tau(K)$. To establish our claim, we need to introduce four separate solutions, A, B, C, and D. Solution A has good query time performance but requires more space than desired. Solution B bootstraps solution A to reduce the space requirement to a better, yet still unacceptable, level. Solution C offers a dual perspective: its space performance is ideal, but the query time is not. Finally, solution D combines the best of solutions B and C. Heavy use is made of conformal samples throughout these stages. In all cases, we compute and store away $T_\tau(K)$: the triangulation will be used to identify the unique simplex that contains the query point q.

Solution A is the derandomized version of a data structure for point location due to Clarkson [3]: its space requirement is $O(n^{d+\varepsilon})$, for any $\varepsilon > 0$, and the query time is $O(\log n)$. Translated in our terminology, the algorithm works as follows. Find a sample R of r hyperplanes, where r is a large constant, which leaves no edge of the τ-induced hypergraph H (the one of the previous section) $(3d \log r)$-deficient (Lemma 4.3). Compute $T_\tau(R)$ and, for each simplex s of the triangulation, identify the set K_s of hyperplanes that intersect (but do not contain) the closure of s. Then take the intersection of all these hyperplanes with the affine closure of s. (Note that the adjective "affine" is used here to distinguish from topological closure.) For convenience, if s is of dimension k, we also include its bounding k-flats. In this way, locating q in $T_\tau(R)$ and $T_\tau(K_s)$, where s is the unique simplex of $T_\tau(R)$ that contains q, will trivially lead to the simplex of $T_\tau(K)$ that contains q. (We will use this pattern time and again.) A simple analysis shows that, for r large enough, this solution has an $O(\log n)$ query time and uses $O(n^{d+\varepsilon})$ space, for any fixed $\varepsilon > 0$. The preprocessing time is $O(n^{d(d+3)/2+1})$.

Solution B is an improved version of this algorithm: it has the desired $O(\log n)$ query time, but its storage requirement is $O(n^d \log^\varepsilon n)$, for any $\varepsilon > 0$. Let $r(n) = \lceil n/\log n \rceil$, and let R be a conformal sample of $r(n)$ hyperplanes (Lemma 4.5). We compute the data structure recursively with respect to R. The recursion is stopped when the problem has constant size, at which point exhaustive search is used. As usual, for each simplex s of $T_\tau(R)$, we proceed to identify the set K_s of hyperplanes of K that intersect (but do not contain) the closure of s. Then, as usual, we add the bounding flats of s, intersect the hyperplanes with the affine closure of s, etc. The difference is that now we *do not* recurse on K_s, but instead, we set up the previous data structure with respect to it. The algorithm is obvious: first determine recursively the simplex s of $T_\tau(R)$ that contains q; then pursue the search in $T_\tau(K_s)$ using Clarkson's algorithm. Let $S(n)$ and $Q(n)$ be respectively the storage and query time of Solution A. Both $S(n)$ and $Q(n)$ are in $O(1)$, when n is less than a fixed constant. We have the recurrence relation

$$S(n) = S(\lceil n/\log n \rceil) + a \sum_{s \in T_\tau(R)} |K_s|^{d+\varepsilon} + O(n^d),$$

for some fixed a. Because of conformality, the number of sets K_s whose sizes exceed $t \log n$ is at most cn^d/b^t, for some constant $b, c > 1$, therefore

$$S(n) \leq S(\lceil n/\log n \rceil) + a' \left(\frac{n}{\log n}\right)^d \sum_{t \geq 0} (t \log n)^{d+\varepsilon}/b^t,$$

for some constant a'. We can verify by induction that $S(n) = O(n^d \log^{2\varepsilon} n)$. For convenience we can re-adjust ε, so that $S(n) = O(n^d \log^{\varepsilon} n)$. Using similar arguments, we establish that the preprocessing time is $O(n^{d(d+3)/2+2}/\log n)$. Because of conformality we verify that the size of K_s is always in $O(\log^2 n)$. Therefore the query time $Q(n)$ satisfies the recurrence:

$$Q(n) = Q(\lceil n/\log n \rceil) + O(\log \log n),$$

which gives $Q(n) = O(\log n)$. This concludes our discussion of Solution B.

Solution C is quite similar, except that we set $r = \lceil n/2 \rceil$. Each K_s is now of size $O(\log n)$. We locate q in $T_\tau(K_s)$ in $O(\log n)$ time by looking at the triangulation directly, without the help of an extra data structure. To do so, we first locate q in $\mathcal{A}(K_s)$ by finding which hyperplane π is right above q. Then we project q onto π, and we recursively locate the projection in the triangulation of the $(d-2)$-flats on π. This information gives us access to the apex p of a d-simplex whose closure contains q. Next, we consider the ray emanating from q in direction $\overrightarrow{pq}$ and find the first hyperplane π' which it intersects. We project q onto π' (centrally from p) and recurse as before. The time $f(d)$ required for the location satisfies $f(1) = O(\log n)$ and $f(d) = 2f(d-1) + O(\log n)$, which is $O(\log n)$. This gives us a query time of $O(\log^2 n)$ for locating q in $T_\tau(K)$. How about the storage $S(n)$? We have $S(n) = O(1)$, when n is less than a fixed constant. Otherwise, we have the recurrence relation

$$S(n) = S(\lceil n/2 \rceil) + an^d \sum_{t \geq 0} \frac{t}{b^t},$$

for some constant a, which gives $S(n) = O(n^d)$. Finally, the preprocessing time follows trivially from Lemma 4.5. We find that it is in $O(n^{d(d+3)/2+2})$.

We are now ready to describe our last and best solution. Find a conformal sample R of $\lceil n/\log n \rceil$ hyperplanes, and apply Solution B to R. Then, compute $T_\tau(R)$ and, as usual, for each simplex s of the triangulation, identify the set K_s of hyperplanes that intersect (but do not contain) the closure of s and give it the usual treatment. Finally, apply Solution C to each K_s. Location in $T_\tau(R)$ takes $O(\log n)$ time and requires $O((n/\log n)^d \log^{\varepsilon} n) = O(n^d)$ space, for ε small enough. On the other hand, locating q in K_s takes $O((\log \log n)^2)$ time, since each K_s is of size $O(\log^2 n)$. For n larger than a constant, the storage $S(n)$ satisfies

$$S(n) = a(n/\log n)^d \sum_{t \geq 0} \frac{(t \log n)^d}{b^t},$$

for some constant a, which gives $S(n) = O(n^d)$. The preprocessing time is $O(n^{d(d+3)/2+2}/\log n)$. This proves our claim that location among n hyperplanes can be performed in $O(\log n)$ query time, using $O(n^d)$ space.

Theorem 5.3. *Given an arrangement of n hyperplanes in d-space it is possible to locate a query point in $O(\log n)$ time, using $O(n^d)$ storage. The time required for computing the data structure is $O(n^{d(d+3)/2+2}/\log n)$.*

6. Conclusions

Beside improving various complexity bounds, we believe that the main contribution of this paper is to set the dual pursuits of probabilistic and deterministic geometric algorithms on a parallel course. On the algorithmic side, an interesting algorithmic issue is to lower the complexity of our deterministic constructions. Finally, a more exhaustive investigation of applications should be undertaken.

Acknowledgments: We wish to thank K. Clarkson, J. Pach, and J. Spencer for valuable comments about this work.

REFERENCES

1. Agarwal, P., Sharir, M. *Red-blue intersection detection algorithms with applications to motion planning and collision detection*, Proc. 4th Ann. ACM Sympos. Comput. Geom. (1988), 70–80.
2. Clarkson, K.L. *A randomized algorithm for closest-point queries*, SIAM J. Comput. 17 (1988), 830–847.
3. Clarkson, K.L. *New applications of random sampling in computational geometry*, Disc. Comp. Geom. 2 (1987), 195–222.
4. Clarkson, K.L. *Applications of random sampling in computational geometry, II*, Proc. 4th Ann. ACM Sympos. Comput. Geom. (1988), 1–11.
5. Clarkson, K.L., Shor, P.W. *Algorithms for diametral pairs and convex hulls that are optimal, randomized, and incremental*, Proc. 4th Ann. ACM Sympos. Comput. Geom. (1988), 12–17.
6. Clarkson, K.L., Tarjan, R.E., Van Wyk, C.J. *A fast Las Vegas algorithm for triangulating a simple polygon*, Proc. 4th Ann. ACM Sympos. Comput. Geom. (1988), 18–22.
7. Edelsbrunner, H. *Algorithms in Combinatorial Geometry*, Springer-Verlag, Heidelberg, Germany, 1987.
8. Edelsbrunner, H., Guibas, L.J., Hershberger, J., Seidel, R., Sharir, M., Snoeyink, J., Welzl, E. *Implicitly representing arrangements of lines or segments*, Proc. 4th Ann. ACM Sympos. Comput. Geom. (1988), 56–69.
9. Edelsbrunner, H., Guibas, Sharir, M. *The complexity of many faces in arrangements of lines and of segments*, Proc. 4th Ann. ACM Sympos. Comput. Geom. (1988), 44–55.
10. Edelsbrunner, H., O'Rourke, J., Seidel, R. *Constructing arrangements of lines and hyperplanes with applications*, SIAM J. Comput. 15 (1986), 341–363.
11. Erdös, P., Spencer, J. *Probabilistic methods in combinatorics*, Academic Press, New York, 1974.
12. Haussler, D., Welzl, E. *Epsilon-nets and simplex range queries*, Disc. Comp. Geom. 2, (1987), 127–151.
13. Lovász, L. *On the ratio of optimal integral and fractional covers*, Discrete Math. 13 (1975), 383–390.
14. Lovász, L. *Combinatorial problems and exercises*, North-Holland, 1979.
15. Luby, M. *A simple parallel algorithm for the maximal independent set problem*, Proc. 17th Ann. ACM Sympos. on Theory of Comput. (1985), 1–10.
16. Pach, J. Private communication, 1988.
17. Pach, J., Spencer, J. *Explicit codes with low covering radius*, IEEE Trans. Information Theory, to appear.
18. Raghavan, P. *Probabilistic construction of deterministic algorithms: approximating packing integer programs*, Proc. 27th Annu. IEEE Symp. on Foundat. of Comput. Sci. (1986), 10–18.

19. Reif, J.H., Sen, S. *Optimal randomized parallel algorithms for computational geometry*, Proc. 16th Internat. Conf. Parallel Processing, St. Charles, IL, 1987. Full version, Duke Univ. Tech. Rept., CS–88–01, 1988.
20. Spencer, J. *Puncture Sets*, J. Combinat. Theory A, 17 (1974), 329–336.
21. Spencer, J. *Ten lectures on the probabilistic method*, CBMS-NSF, SIAM, 1987.
22. Tarjan, R.E., Van Wyk, C.K. *An $O(n \log\log n)$-time algorithm for triangulating a simple polygon*, SIAM J. Comput. (1988).
23. Sauer, N. *On the density of families of sets*, J. Combinat. Theory A, 13 (1972), 145–147.
24. Vapnik, V.N., Chervonenkis, A.Ya. *On the uniform convergence of relative frequencies of events to their probabilities*, Theory Probab. Appl. 16 (1971), 264–280.
25. Welzl, E. *Partition trees for triangle counting and other range searching problems*, Proc. 4th Ann. ACM Sympos. Comput. Geom. (1988), 23–33.