Discrete Comput Geom 5:427-448 (1990)

Discrete & Computational
Geometry

Construction of ε-Nets

Jiří Matoušek

Department of Computer Science, Charles University,
Malostranské nám. 25, 118 00 Praha 1, Czechoslovakia

Abstract. Let S be a set of n points in the plane and let ε be a real number, $0 < \varepsilon < 1$. We give a deterministic algorithm, which in time $O(n\varepsilon^{-2} \log(1/\varepsilon) + \varepsilon^{-8})$ (resp. $O(n\varepsilon^{-2} \log(1/\varepsilon) + \varepsilon^{-10})$ constructs an ε-net $N \subset S$ of size $O((1/\varepsilon)(\log(1/\varepsilon))^2)$ for intersections of S with double wedges (resp. triangles); this means that any double wedge (resp. triangle) containing more that εn points of S contains a point of N. This gives $O(n \log n)$ deterministic preprocessing for the simplex range-counting algorithm of Haussler and Welzl [HW] (in the plane).

We also prove that given a set L of n lines in the plane, we can cut the plane into $O(\varepsilon^{-2})$ triangles in such a way that no triangle is intersected by more than εn lines of L. We give a deterministic algorithm for this with running time $O(n\varepsilon^{-2} \log(1/\varepsilon))$. This has numerous applications in various computational geometry problems.

1. Introduction and Statement of Results

The study of combinatorial properties of finite point sets in E^d and their dual counterpart—arrangements of hyperplanes—is one of the fundamental issues in algorithmic and combinatorial geometry. It is closely related to the construction of algorithms for rapid processing of half-space and simplex range queries. The problem here is to preprocess a finite set of points in E^d (d a small fixed integer) so that, given a query range (half-space or simplex), we can quickly count the number of points inside the range (or, more generally, perform some semigroup operation on values associated with the points).

A significant progress in this area was the data structure of Haussler and Welzl [HW]. It uses $O(n)$ storage (for an n-point set in E^d) and yields $O(n^\alpha)$ query time, for every fixed $\alpha > d(d-1)/(d(d-1)+1)$. Let us remark that several newer algorithms are superior to this algorithm in most respects—see [W1], [CW], [A2], and [M3].

The preprocessing algorithm of Haussler and Welzl [HW] is very efficient, but is randomized one. The original motivation of this paper was to find a deterministic

preprocessing for that algorithm: this goal has been completely attained for the planar case. However, it turned out that the methods used to achieve this have many more consequences, and they allow us to remove randomization from many other algorithms in computational geometry. We state the results first and then we briefly indicate further developments in this area.

Let us start with some definitions.

A *range space* Σ is a pair (X, R), where X is a set and R is a set of subsets of X. Members of R are called *ranges* of Σ. A range space Σ is *finite* if X is finite. The notion of range space allows us to treat range-searching problems in an abstract setting.

Let $\Sigma = (X, R)$ be a finite range space, let ε be a nonnegative real number, and let N be a subset of X. We say that N is an *ε-net for* Σ if N intersects each range $r \in R$ with $|r| > \varepsilon|X|$. This notion is perhaps the most important one introduced in [HW].

Let k be a positive integer and let S be a finite set of points in the plane. An *H^k-range* is defined as an intersection of at most k closed half-planes. We denote by $H^k(S)$ the range space $(S, \{S \cap r;\ r \text{ is an } H^k\text{-range}\})$.

For $k \geq 4$, an H^k-range can be partitioned into at most $k - 2$ H^3-ranges, and thus an ε-net for $H^3(S)$ is a $(k-2)\varepsilon$-net for $H^k(S)$. Hence in constructing ε-nets, we can restrict ourselves to the cases $k = 2$ and $k = 3$ (the case $k = 1$ being easy). This observation is due to Welzl.

For the special case of range spaces $H^k(S)$ (k fixed), the theory of ε-nets developed in [HW] says that if a random sample of size $(C/\varepsilon)\log(1/\varepsilon)$ (C a sufficiently large constant) is drawn from S, it will be an ε-net for $H^k(S)$ with high probability. However, no polynomial deterministic algorithm for constructing such a small ε-net was known. We have the following result:

Theorem 1.1. *Let S be a set of n points in the plane and let ε be a real number with $0 < \varepsilon < 1$. We can construct an ε-net for $H^k(S)$ of size $O((1/\varepsilon)(\log(1/\varepsilon))^2)$ in time $O(\varepsilon^{-2}\log(1/\varepsilon)\, n + \varepsilon^{-(2k+4)})$ (for $k = 2$ and $k = 3$).*

The method in [HW] then gives

Corollary 1.2. *For any fixed $\delta > 0$, we can deterministically preprocess a set S of n points in the plane in time $O(n \log n)$, store the results in space $O(n)$, and then answer triangle range-counting queries for the set S in time $O(n^{2/3+\delta})$.*

Let us remark that the range-counting algorithm from the above corollary remains probably the most efficient one known with almost linear-time deterministic preprocessing. It can also be used for general semigroup queries on triangular ranges, and in this respect it seems to be the most efficient one with almost linear-time preprocessing even among randomized algorithms.

As we have already mentioned, in the light of new developments the auxiliary results in the proof of Theorem 1.1 turned out to be perhaps more significant than the theorem itself, in particular the following:

Theorem 1.3. *Given a set L of n lines, a number $\varepsilon > c/n$ (c a sufficiently large constant), and a rectangle R, which contains all the intersections of lines of L in its*

interior, it is possible to subdivide R into $O(\varepsilon^{-2})$ triangles in such a way that the interior of no triangle is intersected by more than εn lines of L. Such triangles can be deterministically constructed in time $O(\varepsilon^{-2}\log(1/\varepsilon)n)$.

In an actual implementation, the rectangle R need not be actually computed; we use it mainly to avoid handling unbounded regions.

Analogs of Theorem 1.3 can be easily proved (once it is known how to do it) using random sampling; in this case a randomized algorithm is obtained, and the number of triangles in the subdivision is slightly larger (by a factor $\log(1/\varepsilon)$). It is the context of random-sampling methods where most of potential applications of Theorem 1.3 originated (let us quote, e.g., the papers [EG*], [CS], and [EGH*]).

The number of triangles in Theorem 1.3 is asymptotically the best possible (as is easily seen by comparing the total number of intersections of the lines with the maximum possible number of intersections inside a single triangle). The existence of a subdivision with the optimal number of triangles was independently established by Chazelle and Friedman [CF] (they even solve the problem for an arbitrary dimension; they use the term *simplicial packing*). They also a give a deterministic algorithm for such a subdivision, but with much worse time complexity ($O(n^6/\varepsilon)$ for the planar case).

The algorithms from Theorems 1.3 and 1.1 are asymptotically optimal if ε is a fixed parameter. For smaller values of ε (e.g., comparable to a power of $1/n$), Agarwal [A1] improved the time complexity in Theorem 1.3 to $O((n/\varepsilon)\log n(\log(1/\varepsilon))^{\omega})$ ($\omega < 3.3$ is a constant). A further improvement in dimension 2 and results for a general dimension were obtained recently [M2].

In applications, Theorem 1.3 and its analogs are usually used as a divide-and-conquer principle in geometric algorithms, allowing us to divide a problem involving lines and/or points into smaller subproblems (defined by the triangles). We do not describe the applications here. The reader may find an extensive survey in [A1] and [A2] (resp. in their journal versions).

The plan of this paper is as follows: Section 2 contains some definitions and facts that we use. In Section 3 we discuss an algorithm for the selection of the kth leftmost intersection of a given set of lines. Building on the methods in [CSSS], we modify the algorithm for the case when only an approximate selection is desired. In Section 4 we prove Theorem 1.3, and we obtain another by-product of our proof—an algorithm for the construction of so-called approximate leveling for a set of lines. Section 5 describes the construction of ε-nets and proves Theorem 1.1. Section 6 gives some more results on the computational complexity of finding ε-nets; in particular, it shows that finding a smallest ε-net for $H^3(S)$ is an NP-complete problem.

2. Preliminaries

The symbol $a \ll b$ means that the ratio b/a is sufficiently large, i.e., it exceeds some constant which could be computed from the proof where the symbol is used. If A and B are sets, $A \div B$ denotes their symmetric difference.

We denote the set $\{p, p+1, \ldots, q\}$ ($p \le q$ integers) by $[p, q]$, and $[p]$ means $[1, p]$.

Let us add something more about range spaces. If $\Sigma = (X, R)$ is a range space and $Y \subset X$, we define the *subspace of* Σ *induced by* Y as the range space $(Y, \{Y \cap r; r \in R\})$. Range spaces we deal with are finite, but they will be defined as subspaces of infinite range spaces (e.g., with $X = E^2$), where the ranges have natural geometric definitions.

A basic combinatorial characteristic of a range space is its dimension, a concept introduced in [VC]. The *Vapnik–Chervonenkis dimension* (or simply *dimension* [VC]) of Σ is the largest integer d for which there exists a d-element set $A \subset X$ such that the set $\{A \cap r; r \in R\}$ consists of all subsets of A. If no such maximal d exists, we say that the dimension of Σ is infinite.

An existential result for ε-nets (obtained by a counting argument) is the following:

Theorem 2.1 [HW]. *Let S be a range space of finite dimension d and let $\varepsilon > 0$. There exists an ε-net for S of size at most $\lceil (8d/\varepsilon) \log(8d/\varepsilon) \rceil$.*

Throughout this paper we use the line-point *duality transform*. This is a transform D, which maps points to lines and nonvertical lines to points, and its main property is that it reverses the relation "lying above" for pairs point–line or line–point. In particular, the half-plane below a line p contains at least k points of a set S iff the point $D(p)$ lies below at least k lines of $D(S)$ (see, e.g., [E] for more information).

For the sake of simplicity we consider only line and point configurations in general position. The results can be easily transferred to the general case (perhaps the easiest way to show this is a perturbation argument—so-called simulation of simplicity, see [E]). An explicit treatment of degeneracies in a similar algorithm can be found in [A1] (journal version). We say that a set of lines L is *in general position* if no three lines of L meet at a common point, no two are parallel, and no two of their intersections have the same x-coordinate. A point set is in general position if no three points are collinear and no two points lie on a common vertical line.

3. Approximate Partitioning of Intersections

In the following N denotes the expression $\binom{n}{2}$. We address the following problem:

Problem 3.1. Given n lines in the plane in general position, an integer $k \in [N]$ and a real number ε $(0 < \varepsilon < 1)$, find a number a such that the number s of intersections of given lines lying to the left of the vertical line $x = a$ satisfies $|k - s| \leq \varepsilon N$.

Cole *et al.* [CSSS] gave an $O(n \log n)$ algorithm solving the exact version of the problem (i.e., with $\varepsilon = 0$) in time $O(n \log n)$ (which is optimal for the exact version). We modify their methods to get a more efficient algorithm when $\log(1/\varepsilon) = o(\log n)$:

Theorem 3.2. *Problem* 3.1 *can be solved by a deterministic algorithm in time* $O(n \log(1/\varepsilon))$.

In the proof it suffices to consider the case when $\log(1/\varepsilon) \ll \log n$; otherwise the result in [CSSS] applies.

The proof of Theorem 3.2 is divided as follows: In Section 3.1 we prove technical lemmas about permutations, and in Section 3.2 we present some preliminaries on sorting networks. Section 3.3 describes an algorithm for solving Problem 3.1, which is relatively simple but slower than we need, and in section 3.4 we sketch the modifications needed to obtain the desired bound.

3.1. Permutations

Let S_n denote the set of permutations of $[n]$, i.e., of bijective mappings of $[n]$ onto itself. For $\pi, \sigma \in S_n$, $\pi \circ \sigma$ denotes the composition of π and σ (as mappings) and π^{-1} is the inverse permutation (mapping) to π. Special permutations denoted by *id* and *rev* are defined by $\mathrm{id}(i) = i$, $\mathrm{rev}(i) = n + 1 - i$ for all $i \in [n]$. *For* $\pi \in S_n$ let $\mathrm{I}(\pi) = \{\{i, j\}; 1 \le i < j \le n, \pi(i) > \pi(j)\}$ be the set of *inversions* of π.

The following lemma gives some elementary properties of the sets of inversions.

Lemma 3.3. *Let* $\pi, \sigma, \pi_1, \pi_2 \in S_n$. *Then*

(i) $|\mathrm{I}(\pi)| = |\mathrm{I}(\pi^{-1})|$,
(ii) $|\mathrm{I}(\sigma) \div \mathrm{I}(\pi)| = |\mathrm{I}(\sigma \circ \pi^{-1})|$,
(iii) $|\mathrm{I}(\pi_1) \div \mathrm{I}(\pi_2)| \le |\mathrm{I}(\sigma \circ \pi_1^{-1})| + |\mathrm{I}(\sigma \circ \pi_2^{-1})|$.

Proof. (i) It is by definition

$$\mathrm{I}(\pi^{-1}) = \{\{\pi(i), \pi(j)\}; \{i, j\} \in \mathrm{I}(\pi)\},$$

and this gives a bijection between $\mathrm{I}(\pi)$ and $\mathrm{I}(\pi^{-1})$.

(ii) We have

$$\begin{aligned}
&|\mathrm{I}(\sigma \circ \pi^{-1})| \\
&\quad = |\{\{i, j\}; i < j \text{ and } \sigma(\pi^{-1}(i)) > \sigma(\pi^{-1}(j))\}| \\
&\quad = |\{\{\pi^{-1}(i), \pi^{-1}(j)\}; \pi^{-1}(i) < \pi^{-1}(j) \text{ and} \\
&\qquad ((i < j \text{ and } \sigma(\pi^{-1}(i)) > \sigma(\pi^{-1}(j))) \\
&\qquad \text{or } (i > j \text{ and } \sigma(\pi^{-1}(i)) < \sigma(\pi^{-1}(j))))\}| \\
&\quad = |\{\{p, q\}; p < q \text{ and } ((\pi(p) > \pi(q) \text{ and } \sigma(p) < \sigma(q)) \\
&\qquad\qquad \text{or } (\sigma(p) < \sigma(q) \text{ and } \pi(p) > \pi(q)))\}| \\
&\quad = |\mathrm{I}(\sigma) \div \mathrm{I}(\pi)|.
\end{aligned}$$

(iii) By a basic property of the symmetric difference of sets we get

$$|\mathrm{I}(\pi_1) \div \mathrm{I}(\pi_2)| \le |\mathrm{I}(\sigma) \div \mathrm{I}(\pi_1)| + |\mathrm{I}(\sigma) \div \mathrm{I}(\pi_2)|,$$

and by applying (ii) we get that the last expression equals

$$|\mathrm{I}(\sigma \circ \pi_1^{-1})| + |\mathrm{I}(\sigma \circ \pi_2^{-1})|. \qquad \square$$

We say that permutations $\pi_0, \pi_1, \ldots, \pi_k$ form a *decreasing sequence of permutations* if, for every $i = 1, 2, \ldots, k$, there is an index $j_0 = j_0(i)$ such that if we denote $p = \pi_{i-1}(j_0 + 1)$, $q = \pi_{i-1}(j_0)$, then $p < q$, $\pi_i(j_0) = p$, $\pi_i(j_0 + 1) = q$, and $\pi_i(j) = \pi_{i-1}(j)$ for all $j \notin \{j_0, j_0 + 1\}$. A deceasing sequence of permutations arises when we lay the elements of $[n]$ in a row in the order π_0 and then sort them by exchanges of neighboring elements which are not in correct order. We see that $\mathrm{I}(\pi_i^{-1}) = \mathrm{I}(\pi_{i-1}^{-1}) \setminus \{\{p, q\}\}$ and $|\mathrm{I}(\pi_i)| = |\mathrm{I}(\pi_0)| - i$.

A decreasing sequence of permutations $Q = (\pi_0 = \mathrm{rev}, \pi_1, \ldots, \pi_N = \mathrm{id})$ is called *complete*. In the context of line arrangements, a complete decreasing sequence of permutations naturally arises when we sweep a (nondegenerate) line arrangement by a vertical line from left to right.

The following lemma says, roughly speaking, that if we leave out a sufficiently large gap in a complete decreasing sequence of permutations, then we cannot reconstruct the middle permutation in the unknown part too exactly. This will be used in the proof of correctness of our algorithm.

Lemma 3.4. *Let $\tau, \tau' \in S_n$ be permutations such that there exists a decreasing sequence of permutations of length $2r + 1$ beginning with τ and ending with τ' (thus $|\mathrm{I}(\tau)| = |\mathrm{I}(\tau')| + 2r$) and suppose that $n \gg 1$ and $r \gg n^{4/3}$. Then there exist decreasing sequences of permutations $\{\rho_0 = \tau, \rho_1, \ldots, \rho_{2r} = \tau'\}$ and $\{\sigma_0 = \tau, \sigma_1, \ldots, \sigma_{2r} = \tau'\}$ such that $|\mathrm{I}(\rho_r) \div \mathrm{I}(\sigma_r)| > r^{3/2}/3n$.*

Proof. It is sufficient to prove the lemma for $\tau' = \mathrm{id}$ (then $|\mathrm{I}(\tau)| = 2r$); the general case is obtained by composing all the permutations under consideration with τ'^{-1}.

The required sequences are obtained as follows: the first one corresponds to sorting of τ by repeated exchanges of the first two neighboring elements which are currently not in the correct order. For the second sequence, we do a similar procedure, but we always take the last two neighboring elements in the wrong order. Formally, we define the sequences of permutations inductively, specifying the indices $j_0(i)$ in the definition of a decreasing sequence: for $\{\rho_i\}$, let $j_0(i)$ be the least index such that $\rho_{i-1}(j_0(i)) > \rho_{i-1}(j_0(i) + 1)$, and for $\{\sigma_i\}$, let $j_0(i)$ be the largest index such that $\sigma_{i-1}(j_0(i)) > \sigma_{i-1}(j_0(i) + 1)$.

Let us put $s_i = \min\{s;\ \rho_s(1) < \rho_s(2) < \cdots < \rho_s(i)\}$ and $t_i = \min\{t;\ \sigma_t(i) < \sigma_t(i + 1) < \cdots < \sigma_t(n)\}$. By the definition of $\{\rho_i\}$ and $\{\sigma_i\}$, we see that $s_{i+1} - s_i \le i$, $t_i - t_{i+1} \le i$. Let p be the maximal index with $s_p \le r$ and let q be minimal with $t_q \le r$.

Let us denote $I_1 = \mathrm{I}(\rho_{s_p})$, $I_2 = \mathrm{I}(\sigma_{t_q})$, and $I = \mathrm{I}(\tau)$. The sets I_1 and $\mathrm{I}(\rho_r)$ differ by at most n inversions, and similarly for I_2 and $\mathrm{I}(\sigma_r)$, so it will be sufficient to show that the symmetric difference of I_1 and I_2 is big, namely that $|I_1 \div I_2| \ge r^{3/2}/2n$ (by our assumption on the values of n and r, we have $n \ll r^{3/2}/n$). The reason for using I_1 and I_2 instead of $\mathrm{I}(\rho_r)$ and $\mathrm{I}(\sigma_r)$ is that I_1 and I_2 have a nicer structure, namely

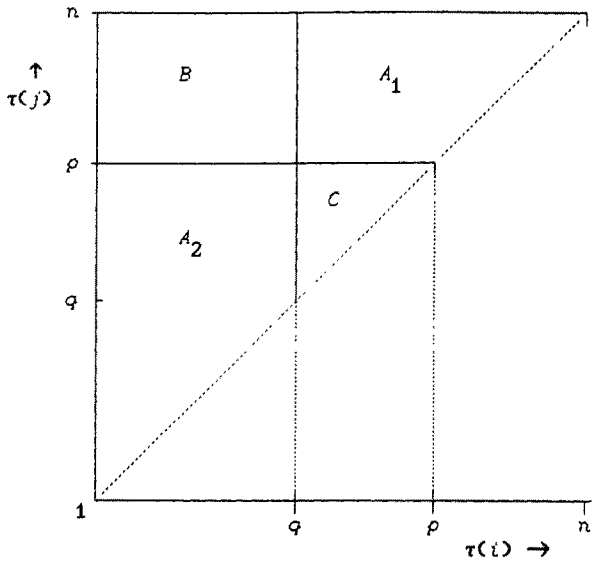

Fig. 1

it is

$$I_1 = \{\{i,j\} \in I;\ \tau(i) > p \text{ or } \tau(j) > p\}$$

and

$$I_2 = \{\{i,j\} \in I;\ \tau(i) < q \text{ or } \tau(j) < q\}.$$

The subsets I_1, I_2 partition I into four parts as follows (see the scheme shown in Fig. 1):

$$A_1 = I_1 - I_2,$$
$$A_2 = I_2 - I_1,$$
$$B = I_1 \cap I_2,$$
$$C = I \setminus (I_1 \cup I_2).$$

We know that $|I_1| \approx |I_2| \approx r$ (here $x \approx y$ means that $|x - y| = O(n) \ll r$) and $|I| = 2r$. Since I_1 is the disjoint union of A_1 and B and I_2 is the disjoint union of A_2 and B, it is $|A_1| \approx |A_2|$, and further we get $|C| = |I| - |I_1 \cup I_2| = |I| - |I_1| - |I_2| + |I_1 \cap I_2| \approx 2r - r - r + |B| = |B|$.

We want to show that $|I_1 \div I_2| = |A_1| + |A_2| \geq r^{3/2}/2n$. Let us assume the contrary; then in particular $|A_1| + |A_2| \ll r$ and since $|B| + |C| = |I| - |A_1| - |A_2|$ and $|B| \approx |C|$, we get $|B| > r/2$, $|C| > r/2$.

We have $C = I \setminus (I_1 \cup I_2) = \{\{i, j\} \in I;\ q \leq \tau(i),\ \tau(j) \leq p\}$. If we denote $d = p - q + 1$, we get that C is a set of pairs of elements of a d-element set, so $d(d-1)/2 \geq |C| \geq r/2$, and $d > \sqrt{r}$.

For any pair $\{k, i\} \in B$ $(k < i)$ we have $\tau(i) \in [1, q)$, $\tau(k) \in (p, n]$, thus for every j with $\tau(j) \in [p,q]$ it is $\tau(i) < \tau(j) < \tau(k)$ and hence we get

$$\{j, i\} \in A_2 \qquad (\text{if } j < i) \tag{*}$$

or

$$\{k, j\} \in A_1 \qquad (\text{if } j > k) \tag{**}$$

(at least one of the above cases must occur). These are at least d inversions in $A_1 \cup A_2$ for every pair $\{k, i\} \in B$. Now every pair of the form (*) can be obtained for at most $q \leq n$ pairs $\{k, i\}$, and each pair of the form (**) can be obtained for at most $n - p \leq n$ pairs $\{k, i\}$. From this we get

$$|I_1 \div I_2| = |A_1| + |A_2| \geq |B| \cdot d/n \geq r^{3/2}/2n$$

as required. □

Remark. We can construct a permutation τ showing that our bound in this lemma is the best possible (up to a multiplicative constant) for our choice of the sequences $\{\rho_i\}$, $\{\sigma_i\}$. However, perhaps we could get a better bound for other pairs of decreasing sequences.

3.2. Sorting Networks

In the algorithm solving Problem 3.1, we use sorting networks. We assume that the reader has already encountered this notion (a usual formalization of this notion can be found, e.g., in [AKS]); we give a formalization (one of possible formalizations) in order to explain further notions we need.

A *parallel step* P of width n is a set $P = \{(p_1, q_1), \ldots, (p_s, q_s)\}$ of ordered pairs, such that $p_1, q_1, p_2, q_2, \ldots, p_s, q_s$ are distinct members of $[n]$ (thus $s \leq n/2$). The ordered pair (p_i, q_i) is called a *comparator*. A *network* H of width n and depth D will be an ordered D-tuple $H = (P_1, P_2, \ldots, P_D)$, where the P_i are parallel steps of width n.

The computation of a network $H = (P_1, P_2, \ldots, P_D)$ on an input vector $(i_1, \ldots, i_n)$ proceeds as follows: In the first step, i_p is compared with i_q for every $(p, q) \in P_1$, and whenever $i_p > i_q$, these entries in the input vector are exchanged. The input vector modified in this way is then processed by the second parallel step, etc.

The computation of the network obviously depends only on the permutation π, which orders the input vector (i.e., such that $i_{\pi(1)} < i_{\pi(2)} < \cdots < i_{\pi(n)}$). Let $H(\pi)$ denote the permutation performed on the input vector by the network when the input vector is ordered by π. Thus, H is a sorting network in the usual sense iff $H(\pi) = \pi$ for every π.

Let us call the pairs (p, q) of indices such that the elements i_p and i_q of the input vector are ever compared during the computation of H (we mean the original

indices of the elements, although they might have been moved to other places before the comparison) the *questions of H for π* (where π is as before). We also define the questions of ith step of H for π in an obvious way. Note that the permutation $H(\pi)$ depends solely on the answers to the questions for π, and does not use any other information about π.

We call a network H an *ε-sorting network*, if, for every $n \in S_n$, $|\mathrm{I}(H(\pi) \circ \pi^{-1})| \leq \varepsilon N$ (thus for $\varepsilon = 0$ we get that $H(\pi) = \pi$). We call a network H a *t-tolerant ε-sorting network* if any network arising from H by deleting at most t comparators is an ε-sorting network.

Ajtai *et al.* constructed a famous sorting network of depth $O(\log n)$ [AKS]. We need the following property of their network:

Lemma 3.5. *Three exists a constant C such that, for each n and $0 < \varepsilon < 1$, the first $C \log(1/\varepsilon)$ steps of the sorting network of* [AKS] *of width n form an ε-sorting network.*

Proof. This can be calculated from the inductive hypothesis in [AKS] (stated in Section 8 of [AKS] as a theorem). The calculation is straightforward but not quite short and we omit the details. □

Lemma 3.6. *Let H be an ε-sorting network of depth D and let $t \leq \varepsilon n/2^{D+1}$. Then H is t-tolerant 2ε-sorting.*

Proof. Removing a single comparator from a parallel step of the network may result in exchanging the position of two elements entering the step $j + 1$, which in turn may affect four elements entering step $j + 2$, etc., so the position of at most 2^D elements of $H(\pi) \circ \pi^{-1}$ is affected. Each element with changed position may contribute by less than n inversions, hence removing t comparators increases the number of inversions only $t \cdot 2^D \cdot (n - 1) \leq \varepsilon N$. □

Corollary 3.7. *There exists a constant C such that, for $n \gg 1$ and $0 < \nu < 1$, we can construct a t-tolerant ν-sorting network $H = H(n, \nu)$ of width n and of depth $O(|\log \nu|)$, where $t = \Omega(\nu^C n)$.* □

Remark. By a more careful analysis of the sorting network in [AKS] (using the properties of expander graphs), we can show that Corollary 3.7 holds for a much better value of t.

3.3. The Algorithm

In this section we give a simpler version of the algorithm for Problem 3.1. Our exposition is based on that in [CSSS], which in turn uses the principles outlined by Megiddo [Me]. We start by describing the basic algorithm for the exact solution of Problem 3.1.

Let $\lambda_1, \ldots, \lambda_n$ be the given lines, ordered by increasing slopes. Let u_{ij} denote the x-coordinate of the intersection of λ_i and λ_j and let $t_1 < t_2 < \cdots < t_N$ denote the elements of the set $\{u_{ij};\ 1 \leq i < j \leq n\}$ in sorted order. Given k, we seek t_k.

For b, a real number not belonging to the set $\{t_1, \dots, t_N\}$, let $\pi(b)$ be the permutation which orders the intercepts of $\lambda_1, \dots, \lambda_n$ with the vertical line $x = b$, thus, under obvious notation, $y_{\pi(1)}(b) > \cdots > y_{\pi(n)}(b)$ (so, for $b < t_1$, $\pi(b) = \mathrm{rev}$ and, for $b > t_N$, $\pi(b) = \mathrm{id}$; our numbering of lines is opposite to the numbering in [CSSS]). We extend the definition also to $b = t_i$, putting $\pi_i = \pi(t_i) = \pi(t_i + \delta)$, where $0 < \delta < \min_j\{t_{j+1} - t_j\}$, and we denote $\pi_0 = \mathrm{rev}$. Then $\pi_0, \pi_1, \dots, \pi_N$ form a complete decreasing sequence of permutations and $|\mathrm{I}(\pi_i)| = N - i$.

We do not know the value of t_k, but for any i, j we can decide whether $y_i(t_k) \le y_j(t_k)$ (which is the same as $\pi_k^{-1}(i) > \pi_k^{-1}(j)$; $i < j \in [n]$) by the following method: we find u_{ij}, the intersection of λ_i and λ_j, we compute $\pi(u_{ij})$ (by sorting the intercepts at u_{ij}), and evaluate $|\mathrm{I}(\pi(u_{ij}))|$. From this we obtain s such that $u_{ij} = t_s$. Then we can deduce whether u_{ij} lies before or after t_k; in the first case $\pi_k^{-1}(i) < \pi_k^{-1}(j)$, in the second case the opposite inequality holds.

Now the basic algorithm for the exact selection of the kth intersection works as follows: We consider some sorting network H. We proceed as if we tried to sort the intercepts of the lines with the (unknown) vertical lines $x = t_k$ by the network H, i.e., to compute $H(\pi_k)$. Each parallel step of the network H gives us a set of questions $\{(i_1, j_1), \dots, (i_s, j_s)\}$. We find the median z of $\{u_{i_1 j_1}, u_{i_2 j_2}, \dots, u_{i_s j_s}\}$, we compute the value of $K_z = N - |\mathrm{I}(\pi(k))|$ and compare it with k. Three cases may occur:

(i) $K_z > k$. In this case certainly $t_k < z$ and all the questions (i, j) with $u_{ij} \ge z$ can be answered (which is half of the questions since z was the median).

(ii) $K_z < k$. In this case certainly $t_k > z$ and all the questions (i, j) with $u_{ij} \le z$ can be answered.

(iii) $K_z = k$. This is actually the desired case, since $t_k = z$ is the answer, so we may finish.

If cases (i) or (ii) have occurred, we consider the yet unresolved questions instead of the original set and repeat the same procedure for the median of the corresponding intersections, etc.

After roughly $\log n$ repetitions of this procedure, all questions of the first parallel step of the network H have been resolved, and we may proceed with the next step in the same way.

In the exact case, it is not difficult to argue that this algorithm must finish by an occurrence of case (iii). Now we want to modify this algorithm for an approximate selection of the kth intersection with a prescribed accuracy. We must do everything approximately: the underlying network will be only ν-sorting (for a certain ν depending on ε), the inversion counting will be approximate and in each step we only resolve most of the questions, dropping some of them unresolved. In this case it seems much more difficult to prove that the algorithm actually works as desired.

We state an auxilliary result, immediately following from [CSSS]:

Lemma 3.8 [CSSS]. *Let $\alpha_1, \dots, \alpha_n$ be a sequence of numbers, let n denote the (unknown) permutation transforming this sequence into sorted order, and let $\varepsilon \in (0, 1)$ be a real number. Then the number $|\mathrm{I}(\pi)|$ can be approximately computed with accuracy εN in time $O(n \log(1/\varepsilon))$.*

Let us now describe the algorithm for the approximate selection. Instead of t_k, we seek only some $a \in [t_{k-\lfloor \varepsilon N \rfloor}, t_{k+\lfloor \varepsilon N \rfloor}]$.

We put $v = \varepsilon^{3/2}/K$, where K is a sufficiently large constant (hence $\log(1/v) = O(\log(1/\varepsilon))$), and let $H = H(n, v)$, $t = v^C n$ be as in Corollary 3.7. Let $D = O(\log(1/\varepsilon))$ be the depth of H. Let $r = \lfloor \varepsilon N/3 \rfloor$.

We process one parallel step of H by one, analogously to the exact algorithm. We find the median z of the set $\{u_{i_1 j_1}, u_{i_2 j_2}, \ldots, u_{i_s j_s}\}$ of questions of the processed parallel step, and we compute a number K_z, which approximates $N - |\mathrm{I}(\pi(z))|$ with accuracy $\leq r$ (in time $O(n \log(1/\varepsilon))$ by Lemma 3.8). Depending on the value of K_z, we again consider three cases:

(i) $K_z > k + 2r$. In this case certainly $t_k < z$ and all the questions (i, j) with $u_{ij} \geq z$ can be answered.

(ii) $K_z < k - 2r$. In this case certainly $t_k > z$ and all the questions (i, j) with $u_{ij} \leq z$ can be answered.

(iii) $|K_z - k| \leq 2r$. This is the desired case, since $a = z$ is a solution to the original problem, so we may finish.

If case (iii) does not occur, we drop the resolved half of the questions and repeat the above procedure with the remaining half of the questions, etc., until the number of unanswered questions is $\leq t/D$. This happens after repeating the above step $O(\log(1/\varepsilon))$ times. Then we delete the $\leq t/D$ comparators corresponding to the unanswered questions from the network H and proceed to the next parallel step of the network.

Each parallel step of the network requires time $O(n(\log(1/\varepsilon))^2)$ and there are $O(\log(1/\varepsilon))$ parallel steps, hence the total time of computation is $O(n(\log(1/\varepsilon))^3)$.

Lemma 3.9. *The algorithm described above always finishes by the occurrence of case (iii), and thus it solves Problem* 3.1 *in time* $O(n(\log(1/\varepsilon))^3)$.

Proof. Assume the contrary; then we can compute the permutation $\sigma = F(\pi_k)$ for some network F arising from H by deleting at most $D \cdot t/D = t$ comparators, and, since H is t-tolerant v-sorting, it must be $|\mathrm{I}(\sigma \circ \pi_k^{-1})| \leq vN$. At the same time, case (iii) has never occurred and so the information about order of intercepts was computed only for vertical lines $x = z$ such that $\pi(z)$ and $\pi(t_k)$ differ by at least r inversions. Therefore all the information from which σ was computed can be inferred solely from the knowledge of permutations $\pi_0, \pi_1, \ldots, \pi_{k-r-1}, \pi_{k+r+1}, \pi_{k+r+2}, \ldots, \pi_N$ and it does not depend on $\pi_{k-r}, \ldots, \pi_{k+r}$. Now let us apply Lemma 3.4 with $\tau = \pi_{k-r}, \tau = \pi_{k+r+1}$; it allows us to define two complete decreasing sequences of permutations $P = \{\pi'_0, \ldots, \pi'_N\}$ and $P'' = \{\pi''_0, \ldots, \pi''_N\}$ such that

$$\pi_j = \pi''_j = \pi'_j \qquad \text{for} \quad i \notin [k - r, k + r]$$

and

$$|\mathrm{I}(\pi'_k) \div \mathrm{I}(\pi''_k)| > 2vN. \tag{*}$$

We know that $F(\pi'_k) = F(\pi''_k) = \sigma$ (since the information given to the network was the same) and, because F is v-sorting, it should be $|\mathrm{I}(\sigma \cdot \pi'^{-1}_k)| \leq vN$,

$|\mathrm{I}(\sigma \circ \pi_k''^{-1})| \le \nu N$. But applying Lemma 3.3 with $\pi_1 = \pi_k'$, $\pi_2 = \pi_k''$, we get a contradiction to (*). This proves Lemma 3.8. □

3.4. Improvements

The previous result can be improved in the same way as in [CSSS]; we give only the main ideas.

The first improvement is achieved by the method of Cole [Co]. All the questions of the first step of the network are known; a question of step $j > 1$ depends on the results of two questions of step $j - 1$. A question is "active" if it is known but not yet answered. The idea is to resolve the questions as they became active. A weight 4^{-j+1} is assigned to active questions of step j and at each stage of the algorithm z is taken as the weighted median of currently active questions. As before, this resolves either all the questions (i, j) with $u_{ij} \le z$ or with $u_{ij} \ge z$ (or answers Problem 3.1). It is shown in [Co] that each such stage reduces the total weight of active questions by one-quarter. Therefore after $O(\log(1/\varepsilon))$ stages (each taking time $O(n \log(1/\varepsilon))$) there remain only questions with sufficiently small total weight, and the corresponding comparators are then deleted from the network.

The second improvement saves the factor $\log(1/\varepsilon)$ in the computations of K_z, the approximation to $N - |\mathrm{I}(\pi(z))|$ (here we cannot explain it completely, since the procedure is quite involved; we refer to [CSSS] for details). The algorithm in [CSSS] approximates $|\mathrm{I}(\pi(z))|$ with the minimal necessary accuracy, which allows us to distinguish among cases (i)–(iii). When we know the approximation of $|\mathrm{I}(\pi(z))|$ with a certain accuracy at some value of z, a procedure described in [CSSS] allows us to compute an analogous information at a new value of z (i.e., the value of $|\mathrm{I}(\pi(z))|$ with the same accuracy) in linear time. Sometimes it may happen that this accuracy is already insufficient for the new value of z (it does not allow us to distinguish between cases (i)–(iii)). Then there is another procedure, which in linear time improves the accuracy of the approximation (at a given value of z) twice. The point is that neither of these procedures can be called than $O(\log n)$ times.

The only modification of the algorithm from [CSSS] needed for our purpose is to insert a test to see if the accuracy of approximation reached $\varepsilon N/2$ and in this case we may finish, since then the value of z already lies sufficiently near to t_k. The argument that this must happen during the computation is the same as we used in Lemma 3.9. The total complexity of the algorithm is then $O(n \log(1/\varepsilon))$ as claimed. □

4. Cutting Line Arrangements

In this section we prove Theorem 1.4 on cutting the plane, which is used in the next section. As a by-product, we also obtain a theorem on the construction of so-called approximate levelings.

4.1. *Approximate Levelings*

Let L be a set of n nonvertical lines. The *level* k ($1 \le k \le n$) of L is the set of points $X = (x_0, y_0)$, where y_0 is the maximum number such that the semiline $\{(x_0, y); y \ge y_0\}$ meets at least k lines of L. It is easy to see that each level is an x-monotone polygonal line, whose edges are segments of lines of L.

We say that an x-monotone polygonal line P is a *d-approximate level k* if it lies between the levels $k - d$ and $k + d$ of the set L (we take the closed region between the levels. For $k - d < 1$ we only require that P lies above level $k + d$; similarly for the other end). A *d-approximate leveling* for a set L of n lines is a sequence P_1, $P_2, \ldots, P_{\lfloor n/2d \rfloor}$, where P_i is a d-approximate level $2di$.

We give a method for an efficient construction of approximate levelings; let us briefly review the history of this problem.

Edelsbrunner and Welzl [EW] were perhaps the first who considered approximate levels. Their paper gives a method for constructing approximate levels (the authors use the notion *k-belt* for the region lying between levels k and $n - k$). They first construct the exact level and then its approximation. The construction of level k requires time $O(n \log n + e(\log n)^2)$, where e denotes the number of edges of the level, and from this we obtain the time bound for the construction of a d-approximate leveling of order $O(b(n, n/d) \cdot \log^2 n)$, where $b(n, r)$ denotes the maximum possible number of edges of r distinct levels for a set of n lines.

The best-known upper bound for $b(n, r)$ was given by Welzl [W2]: $b(n, r) = O(n^{3/2} r^{1/2})$, and from this we get the bound $O(n^{3/2}(\log n)^2 \varepsilon^{-1/2})$ for the construction of an εn-approximate leveling by the algorithm in [EW]. A famous conjecture about so-called k-sets essentially says that $b(n, 1) = O(n^{1+\delta})$ for every $\sigma > 0$ (see [E] for references and discussion; further progress on this conjecture was made by Pach *et al.* [PSS]). The validity of this conjecture would imply $b(n, r) = O(n^{1+\sigma} r)$ and a further improvement in the above time bound.

An algorithm for construction of an εn-approximate leveling with time complexity almost linear in n, namely

$$O(\varepsilon^{-2} n (\log n)^2 (\log \log n)^2 (\log \log n + \log(1/\varepsilon))^{1/2}),$$

was given in [M1]. We prove the following improvement:

Theorem 4.1. *Given a set L of n lines and a real number ε, we can find an εn-approximate leveling for L with $O(\varepsilon^{-2})$ edges in total, in time $O(\varepsilon^{-2} \log(1/\varepsilon) n)$.*

An approximate leveling can be applied for an approximate half-planar range counting (and using some well-known reduction techniques, also for other range-counting problems, such as, e.g., triangle range counting); we do not go into details here.

4.2. *Construction of an Approximate Leveling*

We divide the proof of Theorem 1.3 into several steps. First we construct an εn-approximate leveling with $O(\varepsilon^{-3})$ edges in total, then we simplify it to have only

$O(\varepsilon^{-2})$ edges, and finally we show how such a leveling can be used for the desired subdivision.

Let us recall that Theorem 1.3 actually speaks about a subdivision of a rectangle R, containing all the intersections of the lines in its interior. Obviously it suffices to restrict our considerations on such a rectangle; if a subdivision of the whole plane were desired, we may cut the complement of R into unbounded "triangles." Also it suffices to construct an approximate leveling inside the rectangle, the unbounded edges can be easily added.

Let us begin with the construction of an approximate leveling. First we cut the plane into h vertical strips, where h is approximately $4\varepsilon^{-2}$ (rounded to an integer; for a simpler exposition, we assume in the following that the expressions with ε are integers where needed). We require that no strip contains more than $\varepsilon^2 N/2$ intersections of the lines of L This is achieved by applying the algorithm from Theorem 3.2 $(h-1)$ times, for $k = j\varepsilon^2 N/4, j = 1, 2, \ldots, h-1$, with the accuracy of selection at least $\varepsilon^2 N/8$ (i.e., we take $\varepsilon^2/8$ instead of ε in the algorithm from Theorem 3.2). The time needed to obtain the vertical strips is $O(hn \log(1/\varepsilon) = O(n\varepsilon^{-2} \log(1/\varepsilon))$.

Let $v_1, v_2, \ldots, v_{h-1}$ denote the vertical lines separating the strips, numbered from left to right. Let us add two more vertical lines: v_0 on the left of all intersections and v_h on the right of all intersections.

Put $b = 1/2\varepsilon$. Let z_{ij} be the intersection of v_i with the level $2j\varepsilon n, j = 1, 2, \ldots, b$. For each i, the points z_{ij} can be found by computing the intersections of v_i with all lines of L and partitioning them into consecutive groups of $2\varepsilon n$ elements. By a recursive application of a linear-time algorithm for selection of the tth element out of n elements, this partitioning is achieved in time $O(\log(1/\varepsilon)n)$ (first we select the element approximately in the middle and apply the recursion on both parts). Having found the points z_{ij}, we form polygonal lines $P_j = z_{0j}z_{1j}\cdots z_{hj}$ (for each j). In total the construction of these polygonal lines takes time $O(\varepsilon^{-2} \log(1/\varepsilon)n)$.

Lemma 4.2. *The polygonal lines $P_1, P_2, \ldots, P_b$ form an εn-approximate leveling for the set L.*

Proof. Let $X = z_{(i-1)j}$ be the intersection of level $2j\varepsilon n$ with v_{i-1} and let $Y = z_{ij}$ be

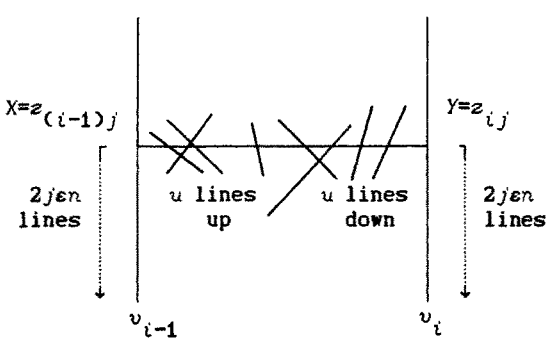

Fig. 2

the intersection of that level with v_i (Fig. 2). In order to prove that P_j is an εn-approximate level $2j\varepsilon n$, it suffices to show that every such segment XY is crossed by at most εn lines of L. The number u of lines crossing XY upward (when going from left to right) must be the same as the number of lines crossing XY downward, since the number of lines below X and below Y is the same. Every line crossing XY upward intersects every line crossing XY downward somewhere inside the strip between v_{i-1} and v_i, so we have $u^2 \leq \varepsilon^2 N/2 < \varepsilon^2 n^2/4$. From this we get $u \leq \varepsilon n/2$, and thus there are at most εn lines crossing XY in total (a more precise reasoning here gives a better constant, allowing us to take $h = \varepsilon^{-2}$). □

4.3. *Simplifying an Approximate Leveling*

Now we show how the above approximate leveling can be simplified to have $O(\varepsilon^{-2})$ edges in total, paying the price of lowered accuracy. Before we prove this, we need several auxiliary results.

Let $V_1, V_2, \ldots, V_{m-1}$ be the vertices of a level k ordered by their abscissa. We define the *p-simplification* of level k as the polygonal line with vertices $V_p, V_{2p}, \ldots, V_{\lfloor (m-1)/p \rfloor p}$ and the left (right) unbounded edge parallel to the left (right) unbounded edge of level k. Thus the p-simplification of a level with m edges has $\lceil m/p \rceil$ edges.

Lemma 4.3 [EW]. *The p-simplification of a level k is a $\lfloor p/2 \rfloor$-approximate level k.*

Lemma 4.4. *Let $K_1, K_2, \ldots, K_m$ be disjoint subsets of $[n]$ such that $|K_i| \geq d$ for each i. Then there exist indices $k_1 \in K_1, \ldots, k_m \in K_m$ such that the levels $k_1, k_2, \ldots, k_m$ have at most n^2/d edges in total.*

Proof. Let b_i denote the number of edges of level i. The levels k, $k \in K_i$ for some i, have at most n^2 edges in total. For each i we choose k_i such that

$$b_{k_i} \leq \left(\sum_{j \in K_i} b_j \right) \Big/ |K_i| \leq \left(\sum_{j \in K_i} b_j \right) \Big/ d,$$

then the sum of b_{k_i} for $i \in [m]$ is bounded by n^2/d. □

Lemma 4.5. *Let AB and CD be two vertical segments and let $a = AC$ and $b = BC$ be two x-monotone polygonal lines having $O(n)$ edges. Suppose that there exists a polygonal line of m edges joining segments AB and CD and lying between a and b. Then a polygonal line with these properties can be found in time $O(n)$.*

Proof. The monotone polygon bounded by the polygonal lines a and b and by the segments AB and CD can be easily triangulated in linear time. Then we can apply the linear-time algorithm of Suri [S] for computing the link distance of the segment AB from the segment CD, i.e., the minimal possible number of edges of a polygonal path joining AB to CD and lying inside the polygon (the result in [S] is stated for computing the link distance of points, but the technique is applicable also for

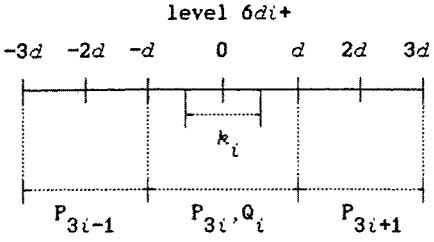

Fig. 3

segments). An optimal path can also be constructed by the algorithm and it is easy to see that it can be made x-monotone in linear time. □

Lemma 4.6. *Let polygonal lines $P_1, \dots, P_{\lfloor n/2d \rfloor}$ with m edges in total form a d-approximate leveling for a set L of n lines. Then we can construct a $3d$-approximate leveling for L* with *$O(n^2/d^2)$ edges in total, in time $O(m)$.*

Proof. By the definition of d-approximate leveling, the region between P_{3i-1} and P_{3i+1} must contain all levels $6di - d$, $6di - d + 1, \dots, 6di + d$ (see the scheme shown in Fig. 3). Applying Lemma 4.4 with $K_i = [6di - d/2, 6di + d/2]$, we see that there are levels $k_1, k_2, \dots, k_{\lfloor n/6d \rfloor}$ with at most n^2/d edges in total, such that $k_i \in K_i$ for every i.

Let Q_i be the d-simplification of level k_i. The polygonal lines Q_i have $O(n^2/d^2)$ edges in total, and, by Lemma 4.3, Q_i lies between levels $6di - d$ and $6di + d$, thus also between P_{3i-1} and P_{3i+1}. By Lemma 4.5 we can find polygonal lines R_i with $O(n^2/d^2)$ edges in total, each R_i lying between P_{3i-1} and P_{3i+1}. Since the polygonal lines P_i form a d-approximate leveling, R_i also lies between levels $6di - 3d$ and $6di + 3d$, thus the polygonal lines R_i form a $3d$-approximate leveling. The claimed time bound follows from Lemma 4.5. □

Coming the construction of an $(\varepsilon/3)n$-approximate leveling with $O(\varepsilon^{-3})$ edges with the previous lemma, we obtain Theorem 4.1.

4.4. *Cutting the Plane*

Now we finish the proof of Theorem 1.3. First we show that the approximate leveling constructed above has only $O(n/\varepsilon)$ intersections with the lines of L.

Lemma 4.7. *Let the polygonal lines $P_1, \dots, P_{\lfloor n/2d \rfloor}$, with m edges in total, form a d-approximate leveling for a set L of n lines. Then the number of intersections of the lines of L with all the edges of the polygonal lines P_i is $O(md + n^2/d)$.*

Proof. We bound the number of intersections of the lines of L with the polygonal lines with even indices (the case of odd indices being analogous). Applying Lemma

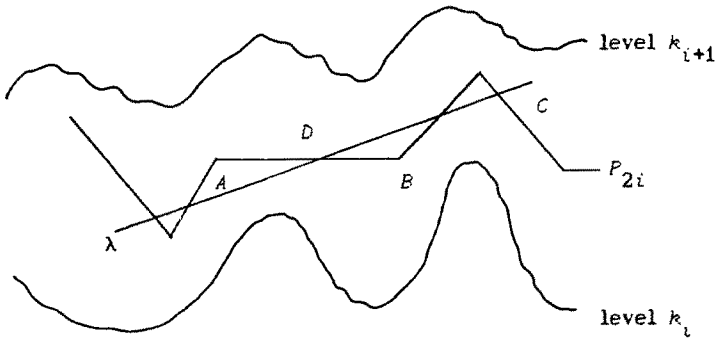

Fig. 4

4.4 with $K_i = [2(2i-1)d - d, 2(2i-1)d + d]$, we see that we may choose levels $k_1, k_2, \ldots, k_{\lfloor n/4d \rfloor}$ with at most $n^2/2d$ edges in total such that P_{2i} lies between levels k_i and k_{i+1}.

Let us consider the lines intersecting P_{2i}. We say that an intersection of a line $\lambda \in L$ with P_{2i} is *good* if it is immediately preceded or immediately followed on λ by an intersection of λ with level k_i or k_{i+1} (e.g., point C in Fig. 4 is a good intersection). The total number of good intersections (for all i) is at most twice the number of edges of levels $k_1, k_2, \ldots, k_{\lfloor n/4d \rfloor}$ (since the number of edges of a level is also an upper bound for the number of intersections of that level with the lines of L), thus at most n^2/d. If an intersection of λ with an edge AB of P_{2i} is not good (point D in Fig. 4), then it means that any vertical line through the segment AB meets λ inside the area between levels k_{i+1} and k_i. Therefore, for a fixed edge AB there are at most $k_{i+1} - k_i + 1 \leq 6d$ such lines. Summing up the intersections which are not good over all edges of P_{2i} and adding the good intersections, we get the claimed bound. □

By Lemma 4.6 we may assume that we already have an εn-approximate leveling with $O(\varepsilon^{-2})$ edges in total. We convert it into the desired subdivision. We divide the regions between the successive approximate levels by vertical segments into convex quadrilaterals (and each quadrilateral is then divided into two triangles).

First, we add vertical subdividing segments in both directions from every vertex of the approximate levels. There are only $O(\varepsilon^{-2})$ such segments (Fig. 5).

Each vertical side of the resulting quadrilaterals intersected by at most $4\varepsilon n$ lines of L, as the polygonal lines form an εn-approximate leveling. We vertically subdivide each quadrilateral with the nonvertical sides intersected by more than $4\varepsilon n$ lines of L, and this subdivision is done so that the nonvertical sides of each resulting quadrilateral are intersected by at least $2\varepsilon n$ lines. Applying Lemma 4.7 with $d = \varepsilon n$, $m = O(\varepsilon^{-2})$, we see that the number of intersections of lines of L with the approximate levels is $O(n/\varepsilon)$ and thus the number of vertical subdivisions added in the second step is again $O(\varepsilon^{-2})$. Therefore we have subdivided the covering rectangle of L into $O(\varepsilon^{-2})$ convex quadrilaterals, each being intersected by at most $8\varepsilon n$ lines of L.

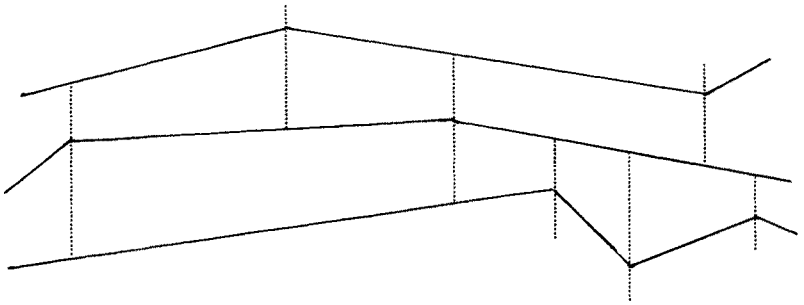

Fig. 5

It remains to show that the second step can be accomplished sufficiently fast. We may proceed as follows: we compute all the intersections of lines of L with the approximate levels in time $O(\varepsilon^{-2}n)$ (testing each line against each edge) and then group the intersections on each edge into consecutive groups of size at most εn. This is done in time $O((1/\varepsilon)\log(1/\varepsilon)n)$ by the method we have indicated in the construction before Lemma 4.2. Now finding the subdividing vertical segments is straightforward. Let us remark that the complexity of the above procedure can be lowered using an approximate version of a suitable range-search algorithm, but we do not give the detals since this does not affect the overall complexity.

5. Selecting an ε-Net

In this section we prove the result about ε-net construction for range spaces $\mathrm{H}^k(S)$. First we define a dual counterpart of the range space $\mathrm{H}^k(S)$; we need a slightly more general definition:

Let C be a collection of subsets of E^2 and let k be a natural number (again the interesting cases are $k = 2$ and $k = 3$). A *k-combination* of C is an ordered pair $K = (\{X_1, X_2, \ldots, X_m\}, (X_{m+1}, X_{m+2}, \ldots, X_k\})$, where $0 \le m \le k$ and the X_i are elements of C. We say that a nonvertical line λ *realizes* the k-combination K if $X_1, \ldots, X_m$ lie above λ and $X_{m+1}, \ldots, X_k$ lie below λ. Let L be a set of lines; we define a range space $\mathrm{DH}^k(L) = (L, \{\{\lambda \in L;\ \lambda \text{ realizes } K\};\ K \text{ a } k\text{-combination of points of } \mathrm{E}^2\})$.

Let S be a point set in general position. Then all ranges of $\mathrm{H}^k(S)$ arise as intersections of S with H^k-ranges determined only by planes with nonvertical bounding lines, and by the properties of duality transform, the range spaces $\mathrm{H}^k(S)$ and $\mathrm{DH}^k(D(S))$ are isomorphic, in particular ε-nets are preserved by the transform.

We need the following fact about ε-nets in $\mathrm{H}^k(S)$:

Lemma 5.1. *For every finite point set S, $0 < \varepsilon < 1$, and for every fixed k, the range space $\mathrm{H}^k(S)$ has an ε-net of size $O((1/\varepsilon)\log(1/\varepsilon))$.*

The proof follows from Theorem 2.1 and the method of Lemma 4.5 of [HW].

The following lemma links the results of the preceding section to the construction of ε-nets:

Lemma 5.2. *Let L be a set of n lines, let R be a rectangle containing all intersections of lines of L, and let R be covered by arcwise connected sets $Q_1, \dots, Q_C$, such that every Q_i meets at most δn lines of L. Let us say that a subset $N \subset L$ has property R_k iff every k-combination of $\{Q_1, \dots, Q_C\}$ realized by more than εn lines of L is realized by some line of N. Then*

(i) *Any $N \subset L$ with property R_k is an $(\varepsilon + k\delta)$-net for $\mathrm{DH}^k(L)$.*
(ii) *Let us assume that, moreover, each Q_i is a triangle. If a set $N \subset L$ is an ε-net for $\mathrm{DH}^{3k}(L)$, then it has property R_k.*

Proof. (i) Let $K = (\{X_1, X_2, \dots, X_m\}, \{X_{m+1}, X_{m+2}, \dots, X_k\})$ be a k-combination of points, realized by more than $(\varepsilon + k\delta)n$ lines of L. For each X_i, choose $j(i)$ such that $X_i \in Q_{j(i)}$. At most $k\delta n$ lines of L can meet some $Q_{j(i)}$, the remaining ones among those realizing K must realize the k-combination $K_1 = (\{Q_{j(1)}, \dots, Q_{j(m)}\}, \{Q_{j(m+1)}, \dots, Q_{j(k)}\})$ (by the arcwise connectedness of each Q_j), and by property R_k there must be a line λ in N realizing K_1. This λ also realizes K.

(ii) The set of lines realizing a k-combination of triangles can be expressed as a set of lines realizing a certain $3k$-combination of points. □

We want to select a subset $N \subset L$ with property R_k and as small as possible. This problem can be formulated as an instance of a set-covering problem. This is the following computational problem: Given a system $\Sigma = \{S_1, \dots, S_k\}$ of subsets of a set X, find a subsystem of Σ of minimum cardinality whose sets contain all points of X. In our application, the points of a set system Σ will be the k-combinations of $\{Q_1, \dots, Q_C\}$ realized by more than εn lines of L, and the sets of Σ will be the sets of k-combinations realized by the lines of L. In this terminology, a subset $N \subset L$ has the property R_k iff it corresponds to a subsystem covering all points of Σ.

Let us call two lines *equivalent* (relative to the given triangles $Q_1, \dots, Q_C$) if one of the double wedges determined by the lines contains no vertex of the triangles. It is easily seen that there are only $O(C^2)$ nonequivalent positions a line can have, hence our system Σ has only $O(C^2)$ distinct sets and $O(C^k)$ points. Its incidence matrix can be easily computed in time $O(nC + C^{k+2})$.

The set-covering problem whose instance we want to solve is NP-complete in general (see [GJ]), but we may use a simple algorithm, which runs in time proportional to the number of entries of the incidence matrix. This algorithm (called a "greedy algorithm") selects the sets into the covering subsystem one by one and always takes a set covering the maximum number of points not covered by the previously selected sets. The size of the resulting covering subsystem is bounded as follows:

Lemma 5.3 [L], [Ch]. *Let $\Sigma = (S_1, \dots, S_n)$ be a set system with $|S_i| \le k$ for all i, and suppose that S has a covering subsystem consisting of at most m sets. Then the greedy algorithm selects a covering subsystem of size at most $m(\ln k + 1)$.*

By Lemmas 5.3 and 5.2(ii), the greedy algorithm selects a covering subsystem of Σ whose size is at most $O(k \log C)$ times bigger than the size of a smallest ε-net for $\mathrm{DH}^{3k}(L)$. This covering subsystem gives an $(\varepsilon + k\delta)$-net for $\mathrm{DH}^k(L)$ by 5.2(i). Putting this together with Theorem 1.3 (with $\delta = \varepsilon/2k$ instead of ε, thus $C = O(\varepsilon^{-2})$) and using the bound from Lemma 5.1 on the size of the $(\varepsilon/2)$-net for $\mathrm{DH}^{3k}(L)$, we get Theorem 1.1.

Proof of Corollary 1.2. The preprocessing for the algorithm in [HW] requires the construction of ε-nets (where ε depends only on the value of δ) of size $O(\varepsilon^{-(1+o(1))})$ for so-called 3-corridors, which are unions of at most three H^2-ranges. An ε-net for H^2-ranges is thus a 3ε-net for 3-corridors. The probabilistic algorithm of [HW] for ε-net construction is then replaced by the algorithm from Theorem 1.1. □

Remark. As noted by Agarwal, we may also replace the use of ε-nets in the algorithm of [HW] by a direct application of Theorem 1.3, which gives a much more direct proof of Corollary 1.2. The ε-nets in [HW] are applied for the following purpose: Given an n-point set S, subdivide the plane so that for any line λ, the number of points of S contained in the regions intersected by λ is at most εn. This can be done directly as follows: subdivide the dual plane into $O(\varepsilon^{-2})$ triangles, each intersected by $\leq \varepsilon n$ lines of $D(S)$. Let P be the set of vertices of these triangles; now take as the desired subdivision of the primal plane the arrangement of $D(P)$. We easily verify that this subdivision has the desired property and $O(\varepsilon^{-4})$ regions, which is even better that the result achieved with ε-nets.

6. More on Computational Complexity of ε-Nets

Further results on the computational complexity of finding ε-nets for range spaces $\mathrm{H}^k(S)$ are the following:

Theorem 6.1.

(i) *There is a polynomial-time algorithm for the following problem: given a set S of n points in the plane, a subset $N \subset S$, a number ε, and an integer k, decide whether N is an ε-net for $\mathrm{H}^k(S)$.*

(ii) *For every $k \geq 3$, the following decision problem is* NP*-complete: given a set S of n points in the plane with integer coordinates, integer m, and number ε, decide whether $\mathrm{H}^k(S)$ admits an ε-net of cardinality $\leq m$ (the cases $k = 1, 2$ are open at present).*

Proof. (i) We search for an H^k-range r containing a maximal number of points of S and avoiding N. We can assume that the half-planes determining r have boundary lines determined by pairs of points of S and that r intersects each vertical line in a (possibly degenerated) segment. Then the problem can be solved in polynomial time by dynamic programming, sweeping the plane by a vertical line from left to right.

A simple (but rather brute-force) approach is to keep (and update with the movement of the sweeping line) the following information: for every pair of lines λ_1, λ_2 determined by the points of S and every $m \leq k$, keep the maximum possible number of points of S which may be contained in a convex m-gon on the left-hand side of the sweeping line, which contains no points of N, has the rightmost vertical side on the sweeping line, and the two sides adjacent to this vertical side lie on λ_1, λ_2. The stopping points of the sweep will be all intersections of pairs of lines determined by points of S. We omit further details.

(ii) For the reduction we use the following NP-complete problem (called VERTEX COVER in [GJ]):

Instance: A simple undirected graph G and an integer k.

Question: Does there exist a set $C \subset V(G)$ of cardinality $\leq k$, such that every edge of G is incident to a vertex belonging to C?

Let the given graph G have m vertices $v_1, \ldots, v_m$. We choose concentric circles c_1, c_2, and c_3, c_1 having the largest and c_3 the smallest radius and the radii differing only by sufficiently small amounts. We choose the point set S_1 as the vertices $A_1, \ldots, A_m$ of a regular m-gon inscribed in c_1. Point set S_2 is defined as follows: for every edge $\{v_i, v_j\}$ of G we choose in S_2 one of the intersections of the segment $A_i A_j$ with c_2. Finally, for every noncollinear triple of points $A, B, C \in S_1 \cup S_2$, we choose in S_3 a point on c_3 that lies in the interior of the triangle ABC and is not collinear with any pair of points of S_1. We put $S = S_1 \cup S_2 \cup S_3$, $n = |S|$, $n_i = |S_i|$; and $\varepsilon = 2/n$. Then we can show

(a) if G has a covering subset of vertices of size s, then $\mathrm{H}^k(S)$ has an ε-net of size $s + n_3$ (the ε-net will be S_3 plus the points of S_1 corresponding to the covering set), and

(b) if $\mathrm{H}^k(S)$ has an ε-net of size t, then G has a covering subset of vertices of size at most $t - n_3 + 2$ (all points of S_3 but at most two must belong to the ε-net, and for every collinear triple of points from $S_1 \cup S_2$ one of them must belong to the ε-net).

Now it is not difficult to verify that S allows a perturbation making the point coordinates rational (without destroying (a) and (b)) and that it can be generated from G in polynomial time. Finally, we observe that it is NP-complete to determine the size of a minimum vertex cover in a graph with accuracy to an additive constant [GJ], so (a) and (b) give the reduction. □

Let us remark that if we take both k and ε as fixed, the decision problem formulated in (ii) becomes trivially solvable in polynomial time: from Theorem 2.1 we obtain a constant upper bound on the size of the optimal ε-net (since $\mathrm{H}^k(S)$ has a fixed dimension), so we may examine all the small subsets of S.

Acknowledgment

I would like to thank Emo Welzl and Pankaj K. Agarwal for many valuable comments and discussions, concerning the final form of the manuscript.

References

[A1] P. K. Agarwal: A deterministic algorithm for partitioning arrangements of lines and its applications, *Proc. 5th Ann. ACM Symposium on Computational Geometry* (1989), pp. 11–21 (full version to apear in *Discrete Comput. Geom.*).

[A2] P. K. Agarwal: Ray shooting and other applications of spanning trees with low stabbing number, *Proc. 5th Ann. ACM Symposium on Computational Geometry* (1989), pp. 315–325 (full version to appear in *Discrete Comput. Geom.*).

[AKS] M. Ajtai, J. Komlós, E. Szemerédi: An $O(n \log n)$ sorting network, *Proc. 15th ACM Symp. on Theory on Computing* (1983), pp. 1–9.

[CF] B. Chazelle, J. Friedman: A deterministic view of random sampling and its use in geometry, Report CS-TR-181-88 (extended abstract in *Proc. 29th Ann. IEEE Symposium on Foundations of Computer Science* (1988), pp. 539–549).

[CW] B. Chazelle, E. Welzl: Quasi-optimal range searching in spaces of finite VC-dimension, *Discrete Comput. Geom.* **4** (1989), 467–490.

[CH] V. Chvátal: A greedy heuristics for the set-covering problem, *Math. Oper. Res.* **4** (1979), 233–235.

[CS] K. Clarkson, P. Shor: Applications of random sampling in computational geometry, II, *Discrete Comput. Geom.* **4** (1989), 387–421.

[CEG*] K. Clarkson, H. Edelsbrunner, L. Guibas, M. Sharir, E. Welzl: Combinatorial complexity for arrangements of curves and surfaces, *Proc. 29th Ann. IEEE Symposium on Foundations of Computer Science* (1988), pp. 568–579.

[Co] R. Cole: Slowing down sorting networks to obtain faster sorting algorithms, *J. Assoc. Comput. Mach.* **31** (1984), 200–208.

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

[E] H. Edelsbrunner: *Algorithms in Combinatorial Geometry*, Springer-Verlag, Heidelberg, 1987.

[EGH*] H. Edelsbrunner, L. Guibas, J. Herschberger, R. Seidel, M. Sharir, J. Snoeyink, E. Welzl: Implicitly representing arrangements of lines or segments, *Discrete Comput. Geom.* **4** (1989), 433–466.

[EW] H. Edelsbrunner, E. Welzl: Constructing belts in 2-dimensional arrangements, *SIAM J. Comput.* **15** (1986), 271–284.

[GJ] M. R. Garey, D. S. Johnson; *Computers and Intractability*, Freeman, San Francisco, 1979.

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

[L] L. Lovász: On the ratio of optimal integral and fractional cover, *Discrete Math.* **13** (1975), 383–390.

[M1] J. Matoušek: *Approximate Halfplanar Range Counting*, KAM Series 59–87, Charles University, Prague, 1987.

[M2] J. Matoušek: Cutting hyperplane arrangements, 6th ACM Symposium on Computational Geometry, 1990.

[M3] J. Matoušek: Spanning trees with low crossing number, to appear in *Inform. Theoret. Applic.*

[Me] N. Megiddo: Applying parallel computation algorithm in the design of serial algorithms, *J. Assoc. Comput. Mach.* **30** (1983), 852–865.

[PSS] J. Pach, W. Steiger, E. Szemerédi: An upper bound for the number of planar k-sets, *Proc. 30th Ann. IEEE Symposium on Foundations of Computer Science* (1989), pp. 72–81.

[S] S. Suri: A linear algorithm for minimum link paths inside a simple polygon, *Comput. Vision Graphics Image Process.* **35** (1986), 99–110.

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

[W1] E. Welzl: Partition trees for triangle counting and other range searching problems, *Proc. 4th ACM Symposium on Computational Geometry* (1988), pp. 23–33.

[W2] E. Welzl: More on k-sets of finite sets in the plane, *Discrete Comput. Geom.* **1** (1986), 95–100.

Received March 15, 1989, and in revised form February 15, 1990.