Discrete Comput Geom 4:467-489 (1989)

Discrete & Computational Geometry

Quasi-Optimal Range Searching in Spaces of Finite VC-Dimension*

Bernard Chazelle[1] and Emo Welzl[2]

[1] Department of Computer Science, Princeton University, Princeton, NJ 08544, USA

[2] Department of Mathematics, Free University of Berlin, Arnimallee 2-6, D-1000 Berlin 33, Federal Republic of Germany

Abstract. The range-searching problems that allow efficient partition trees are characterized as those defined by range spaces of finite Vapnik-Chervonenkis dimension. More generally, these problems are shown to be the only ones that admit linear-size solutions with sublinear query time in the arithmetic model. The proof rests on a characterization of spanning trees with a low stabbing number. We use probabilistic arguments to treat the general case, but we are able to use geometric techniques to handle the most common range-searching problems, such as simplex and spherical range search. We prove that any set of n points in E^d admits a spanning tree which cannot be cut by any hyperplane (or hypersphere) through more than roughly $n^{1-1/d}$ edges. This result yields quasi-optimal solutions to simplex range searching in the arithmetic model of computation. We also look at polygon, disk, and tetrahedron range searching on a random access machine. Given n points in E^2, we derive a data structure of size $O(n \log n)$ for counting how many points fall inside a query convex k-gon (for arbitrary values of k). The query time is $O(\sqrt{kn} \log n)$. If k is fixed once and for all (as in triangular range searching), then the storage requirement drops to $O(n)$. We also describe an $O(n \log n)$-size data structure for counting how many points fall inside a query circle in $O(\sqrt{n} \log^2 n)$ query time. Finally, we present an $O(n \log n)$-size data structure for counting how many points fall inside a query tetrahedron in 3-space in $O(n^{2/3} \log^2 n)$ query time. All the algorithms are optimal within polylogarithmic factors. In all cases, the

* Portions of this work have appeared in preliminary form in "Partition trees for triangle counting and other range searching problems" (E. Welzl), *Proc. 4th Ann. ACM Symp. Comput. Geom.* (1988), 23-33, and "Tight Bounds on the Stabbing Number of Spanning Trees in Euclidean Space" (B. Chazelle), Comput. Sci. Techn. Rep. No. CS-TR-155-88, Princeton University, 1988. Bernard Chazelle acknowledges the National Science Foundation for supporting this research in part under Grant CCR-8700917. Emo Welzl acknowledges the Deutsche Forschungsgemeinschaft for supporting this research in part under Grant We 1265/1-1.

preprocessing can be done in polynomial time. Furthermore, the algorithms can also handle reporting within the same complexity (adding the size of the output as a linear term to the query time).

1. Introduction

Here is the traditional view of range searching in computational geometry: Given a finite collection P of points in E^d and a region $q \subseteq E^d$, report (or count) the points of $P \cap q$. It is understood that the points are given once and for all and that the region q is a *query* to be answered on-line. There is usually a prescribed set of allowable queries, called the *query domain.* A typical example is to take the set of all hyperrectangles (orthogonal range searching), the set of all simplices (simplex range searching), the set of all halfspaces (halfspace range searching), or the set of all d-balls (spherical range searching). To achieve greater generality, it is customary to assign a weight to each point of P and ask for the cumulative weight of $P \cap q$ (that is, the sum of the weights assigned to the points of $P \cap q$). Weights are usually chosen in some algebraic structure, such as a group or a semigroup.

Following Haussler and Welzl [11] we can shed the problem of its geometry and make it purely combinatorial. The pair (E^d, query domain) is replaced by the abstract notion of a *range space* (X, R), where X is an arbitrary set and R is a subset of its power-set 2^X. For convenience, the elements of X are still called *points*; the members of R are called *ranges.* As usual, points are assigned weights. Given a fixed finite subset P of X, the problem is to compute the cumulative weight of $P \cap q$ on-line, where $q \in R$. We assume that membership in a range can be tested in constant time. Then an obvious solution is to store all the weighted points in a list and scan the entire list for each query. This solution uses linear space and has linear query time. In this paper we restrict ourselves to linear-size (or almost linear-size) solutions. Of course, we are primarily interested in sublinear query times.[1]

A popular approach to range searching is the use of *partition trees.* Willard [20] introduced that concept in the context of triangular range searching; the best partition tree for the problem in question was later given in [11]. Applications of partition trees beyond range searching have been found in [8]. A partition tree $\mathcal{T}$ for the input set P is a rooted tree with $|P|$ leaves. Each node v is associated with a *node-set* $N(v)$: if v is a leaf, $N(v)$ is a distinct point of P; otherwise, $N(v)$ is the union of the node-sets of all the leaves descending from v. To avoid redundancy we prohibit any node from having exactly one child. As a data structure, the partition tree need not store its node-sets explicitly but only their cumulative weights. To answer a query $q \in R$, we set a count variable *answer* to the symbolic value *null.* (This value is not a weight *per se*, but rather a semaphore

[1] Throughout this paper the term "sublinear" refers to a function of the form $f(n) = O(n^\alpha)$, where $\alpha < 1$ is a fixed constant.

playing the role of an identity element.) Beginning at the root v of $\mathcal{T}$, we apply the following recursive procedure:

(i) If $N(v) \subseteq q$, then we add the cumulative weight of $N(v)$ to the current value of *answer* and we return.
(ii) If $N(v) \cap q = \emptyset$, we simply return.
(iii) If neither $N(v) \subseteq q$ nor $N(v) \cap q = \emptyset$ holds, then we recurse in all the children of v.

The correctness of the algorithm follows very simply from the definition of a partition tree. The complexity of answering a query depends on how many nodes are visited and how long it takes to answer questions of the form $N(v) \subseteq q$? or $N(v) \cap q = \emptyset$? For the purposes of the first four sections of this paper we sweep the latter under the rug, and concentrate exclusively on the number of nodes visited when answering a query. This is the *arithmetic* view of range searching, where attention is focused on the number of arithmetic operations needed to answer a query and not on the number of steps taken by the algorithm. As it turns out, this restriction is of minor consequence in the geometric applications we discuss in two and three dimensions.

Given a set $A \subseteq X$ we say that the query range q *stabs* A if there exist $x, y \in A$ such that $x \in q$ and $y \notin q$, or, in other words, if neither $A \subseteq q$ nor $A \cap q = \emptyset$. We also say that q *visits* node v if either v is the root of $\mathcal{T}$ or q stabs the node-set of v's father. The *visiting number* of the partition tree $\mathcal{T}$ is the maximum number of nodes visited by any single query.

Informally, the node-sets of a partition tree are building blocks which we use to rewrite sets of the form $P \cap q$ $(q \in R)$ in a more compact fashion. Given a query q, the query-answering algorithm identifies a collection of nodes $v_1, \ldots, v_t$ such that $N(v_1), \ldots, N(v_t)$ partitions the set $P \cap q$. Obviously, the maneuver is of interest only if the number of blocks, t, is substantially smaller than $|P \cap q|$. But this may not always be possible to ensure. The "ultimate" range space, $(X, 2^X)$, makes the visiting number of any partition tree linear. Why is that so? Label the leaves of the tree $1, 2, \ldots, |P|$ from left to right, and form the union P' of the node-sets of all odd-numbered leaves. There exists a query q such that $P' = P \cap q$. To be answered, the query requires the visit of each odd-numbered leaf, which makes the visiting number of $\mathcal{T}$ proportional to $|P|$. Intuitively, good partition trees should exist as long as the range space does not allow queries to "hit" P in a fairly arbitrary manner.

Remarkably, this existence depends on a single parameter, the *Vapnik-Chervonenkis dimension* of the underlying range space. The following definitions originate in [18], albeit in a different context. Given a range space (X, R) and a set $P \subseteq X$, we define $\Pi_R(P) = \{P \cap q \mid q \in R\}$: this characterizes all the ways in which P can be hit by a query. We say that P is *shattered* by R if $\Pi_R(P)$ is the power-set of P. The Vapnik-Chervonenkis dimension of (X, R), or VC-dimension for short, is defined as the size of the largest set P that is shattered by R. If this size is unbounded, the VC-dimension is infinite, and if R is empty, then the dimension is -1. From our previous discussion, it is clear that a range space of infinite VC-dimension gives rise to arbitrarily large point-sets P which admit no

good partition trees. The main contribution of this paper is a proof that the converse is true. More precisely, if (X, R) has finite VC-dimension, then any $P \subseteq X$ of size n admits a partition tree with visiting number in $O(n^{1-1/d} \log^2 n)$, where $d > 1$ is the VC-dimension of the dual range space of (X, R) (a notion defined below).[2] If d is equal to 1, the visiting number is $O(\log^3 n)$. This should be contrasted with a result of Alon *et al.* [1] which establishes the existence of range spaces of finite VC-dimension for which local properties of partition trees alone are insufficient to prove such a result.

As we mentioned earlier the visiting number of a partition tree focuses on the arithmetic (or rather algebraic) component of the query-answering process. We can go even further in that direction and define the arithmetic complexity of a range-searching problem [10], [22]. In that model, a data structure is merely a collection of precomputed cumulative weights. The query time counts only the minimum number of stored operands needed to form the answer to a given query; it says nothing about the time to find the operands in the data structure. In truth, each weight stored is associated with a certain subset of P, called a *generator*: the time to answer a query q is equal to the minimum number of generators whose union gives $P \cap q$. A range space of infinite VC-dimension gives rise to arbitrarily large point-sets P which admit no linear-size solutions with sublinear query time. Conversely, we will prove that if (X, R) has finite VC-dimension, then any $P \subseteq X$ of size n admits a linear-size solution with an $O(n^{1-1/d} \alpha(n) \log n)$ query time, where $d > 1$ is the VC-dimension of the dual of (X, R) and $\alpha(n)$ is a functional inverse of Ackermann's function. If $d = 1$, the query time is $O(\alpha(n) \log^2 n)$.

Section 2 begins with some background material on shatter functions and ε-nets [11]. In Section 3 we define the notion of stabbing numbers and establish its link with the visiting numbers of partition trees. We also prove a lower bound on how good partition trees can be. The two main results of this paper are established in Section 4. We turn to specific geometric problems in Section 5. We prove that any set of n points in E^d admits a spanning tree which cannot be cut by any hyperplane (or hypersphere) through more than roughly $n^{1-1/d}$ edges. This result yields quasi-optimal solutions to simplex and spherical range searching in the arithmetic model of computation. Section 6 is concerned with polygon, disk, and tetrahedron range searching on a random access machine. Given n points in E^2 we derive a data structure of size $O(n \log n)$ for counting how many points fall inside a query convex k-gon (for arbitrary values of k). The query time is $O(\sqrt{kn} \log n)$. If k is fixed once and for all (as in triangular range searching), then the storage requirement drops to $O(n)$. We also describe an $O(n \log n)$-size data structure for counting how many points fall inside a query circle in $O(\sqrt{n} \log^2 n)$ query time. Finally, we present an $O(n \log n)$-size data structure for counting how many points fall inside a query tetrahedron in 3-space in $O(n^{2/3} \log^2 n)$ query time. All the algorithms are optimal within polylogarithmic factors. In all cases, preprocessing can be done in polynomial

[2] All logarithms are taken to the base 2.

time. Furthermore, the algorithms can also handle reporting within the same complexity (adding the size of the output as a linear term to the query time).

2. Preliminaries on Range Spaces

Let (X, R) be a range space. For any integer $n \geq 0$, let $\pi(n)$ be the maximum size of the collection $\Pi_R(P)$, over all subsets P of X of size at most n: $\pi(n)$ is called the (*primal*) *shatter function* of (X, R). Roughly speaking, $\pi(n)$ indicates the maximum number of ways a set of n points can be stabbed. There is a strong relationship between the VC-dimension of a range space and its shatter function. For example, if the dimension is infinite we have $\pi(n) = 2^n$ (pick any P such that $\Pi_R(P) = 2^P$ and $|P| \geq n$). The converse is obviously true. Furthermore, if the dimension is $d < +\infty$, we have $\pi(n) = O(n^d)$ [18], [16]. We give a proof for completeness. Let $\Phi_d(n)$ be the maximum size of $\{P \cap q \mid q \in R\}$, over all $P \subseteq X$ of size n. We have $\Phi_0(n) = \Phi_d(0) = 1$, so let us assume that $d, n > 0$ and let p be a point of P. The number of sets $P \cap q$ can be written as $A + B$, where A is the number of sets of the form $(P \setminus \{p\}) \cap q$ and B is the number of sets $P \cap q$ which can be expressed as the disjoint union of $P \cap q'$ $(q' \in R)$ and $\{p\}$. In the latter case, the sets $P \cap q'$ cannot shatter any subset of $P \setminus \{p\}$ of size d, therefore $B \leq \Phi_{d-1}(n-1)$. Since, obviously $A \leq \Phi_d(n-1)$, we have the recurrence $\Phi_d(n) \leq \Phi_d(n-1) + \Phi_{d-1}(n-1)$, which proves the claim that $\Phi_d(n) = O(n^d)$. It must be noted that the shatter function of a range space of finite VC-dimension may not always be of the form $\Theta(n^d)$; indeed, Welzl and Wöginger [19] can construct a range space with a shatter function in $\Theta(n \log n)$. One last property worth mentioning is that if $\pi(n) = o(n)$, then $\pi(n) = O(1)$ and therefore R is finite. This follows from Proposition 2.19 of [2]. We summarize these facts below.

Lemma 2.1. *Let π denote the shatter function of a range space (X, R).*

(i) *(X, R) has infinite VC-dimension if and only if $\pi(n) = 2^n$, for all $n \geq 0$.*
(ii) *If (X, R) has VC-dimension d, then $\pi(n) = O(n^d)$.*
(iii) *If $\pi(n) = o(n)$, then $\pi(n) = O(1)$ and R is finite.*

Dudley [6] has shown that the set of range spaces of finite VC-dimension is closed under union, intersection, and complementation. We use a special case of this result in the following. For completeness we include a proof.

Lemma 2.2. *Let (X, R) be a range space of finite VC-dimension d and let $\hat{R} = \{(q \cup q') \setminus (q \cap q') \mid q, q' \in R\}$ be the set of symmetric differences between sets of R. Then the range space $(X, \hat{R})$ has finite VC-dimension.*

Proof. Since a range of $(X, \hat{R})$ is defined by two ranges of (X, R), its shatter function cannot exceed the square of the shatter function of (X, R). The proof follows from Lemma 2.1(i) and (ii). □

In some sense it can be argued that X and R play symmetrical roles in the range space (X, R). In the same way as a range is associated with all the points in it, we can associate a point with all the ranges that contain it. This suggests introducing the set $X^* = \{R_x \mid x \in X\}$, where $R_x = \{q \in R \mid x \in q\}$. The pair (R, X^*) is a range space, called the *dual* of (X, R) [2]. If every pair of points in X is stabbed by at least one range of R, then we easily check that, up to isomorphism, duality is involutory; in other words, the dual of (R, X^*) is isomorphic to (X, R). The shatter function of (R, X^*), denoted $\pi^*(n)$, is also called the *dual shatter function* of (X, R). Let Q be a set of ranges in R. Any maximal subset of X which is stabbed by no range $q \in Q$ is called a *cell* of Q. Equivalently, we can consider the relation which puts in the same equivalence class the points of X with the same membership relationship with respect to the ranges of Q. The equivalence classes are the cells of Q: their set is denoted by $\Pi_X^*(Q)$. It is clear that $\pi^*(n)$ is equal to the maximum size of $\Pi_X^*(Q)$, over all subsets Q of R of size at most $n \geq 0$. The following lemma is proven in [2]. Again we include a proof for the sake of completeness.

Lemma 2.3. *A range space has finite VC-dimension if and only if its dual also has finite VC-dimension.*

Proof. Because duality is an involution (under the stabbing conditions described earlier), it suffices to prove that the dual of a range space (X, R) of infinite VC-dimension is also of infinite VC-dimension. Let $P = \{p_0, p_1, \ldots, p_{n-1}\}$ be a subset of X of size $n = 2^k$ which is shattered by R. Define k subsets $P_1, \ldots, P_k \subseteq P$ as follows: P_i contains p_j if and only if the binary representation of j over k bits has a 1 as its ith most significant bit. Because P is shattered, each P_i can be matched to a range $q_i \in Q$ such that $P_i = P \cap q_i$. By construction the only q_j's that contain p_i are those whose indices correspond to a 1 in the binary representation of i. Therefore, all combinations are achieved and $\Pi_X^*(\{q_1, \ldots, q_k\}) = 2^k$. Since k can be made arbitrarily large we conclude from Lemma 2.1(i) (and the fact that "all n" and "infinitely many n" are in this case equivalent) that the dual range space of (X, R) has infinite VC-dimension. □

We now turn to the crucial concept of ε-nets introduced in [11]. Let (X, R) be a range space and let $\varepsilon < 1$ be a positive real. Given a nonempty finite subset P of X, a subset N of P is called an *ε-net* of P for R if, for any $q \in R$, the inequality $|P \cap q| > \varepsilon |P|$ implies that $N \cap q \neq \emptyset$. The notion can be extended to the case of multisets P without difficulty. In that case, the cardinality is to be understood with multiplicity counted in, as in $|\{1, 1\}| = 2$.

Lemma 2.4 [11]. *Let (X, R) be a range space of VC-dimension $d \geq 1$ and let $\varepsilon < 1$ be a positive real. For every multiset P of points in X there exists an ε-net of P for R, with*

$$|N| \leq \left\lceil \frac{8d}{\varepsilon} \log \frac{8d}{\varepsilon} \right\rceil.$$

The lemma will be applied to a dual range space to argue that, given a range space of finite VC-dimension (X, R) and two finite sets $P \subseteq X$ and $Q \subseteq R$, there are two points $p, p' \in P$ such that the pair $\{p, p'\}$ is not stabbed by more than roughly $|Q|/|P|^b$ ranges of Q, for some constant $b > 0$.

3. Stabbing Numbers of Spanning Paths

Let P be the input set of a range-searching problem whose underlying range space (X, R) has finite VC-dimension. We prove in Section 4 that a certain permutation of P, call it, $p_1, \ldots, p_n$, is such that no range of R can stab more than "a few" pairs of the form $\{p_i, p_{i+1}\}$. Consequently, given a query q, the set $P \cap q$ can be expressed as the union of a few intervals of the form $p_i, p_{i+1}, \ldots, p_{j-1}, p_j$. Computing the cumulative weights of these intervals and adding them will give the desired answer. As it turns out, computing a *partial sum*, which is the name for the cumulative weight of a query interval, is a well-studied problem which has a very efficient solution [21]. A permutation of the input points can be regarded as a one-path spanning tree, $\{p_1, p_2\}, \ldots, \{p_{n-1}, p_n\}$. As we shall see, good spanning paths of the type above lead to good partition trees as well. We discuss the relationship between spanning trees, spanning paths, and partition trees below. But we need an additional piece of terminology. Given a spanning tree T of P and a range $q \in R$, let $\sigma(q)$ denote the number of edges of T stabbed by q (an edge is a set of two points). The maximum value of $\sigma(q)$ over all ranges q in R is called the *stabbing number* of T and is denoted $\sigma(T)$.

Lemma 3.1. *Let (X, R) be a range space and let P be a set of n points in X:*

(i) *If T is a spanning tree of P, then there exists a spanning path with a stabbing number at most twice that of T.*

(ii) *If $\mathcal{P}$ is a spanning path of P, then there exists a balanced binary partition tree for P with a visiting number at most $2\sigma(\mathcal{P})\lceil \log n \rceil + 1$.*

(iii) *If $\mathcal{T}$ is a partition tree for P, then there exists a spanning path whose stabbing number does not exceed the visiting number of $\mathcal{T}$.*

Proof. (i) Connect together the vertices of T in the order given by a depth-first traversal of the tree. This gives us a spanning path whose stabbing number is at most twice that of T. Indeed, let e be an edge of the spanning path that is stabbed by a range q. If e is not an edge of T, then it creates a cycle in T, at least two of whose edges are stabbed by q. Because of the depth-first labeling, no edge of T thus needs to be charged more than twice, which proves our claim.

To prove (ii), build a complete binary tree $\mathcal{T}$ on n leaves and associate the points of the spanning path, in sequence, with the leaves of $\mathcal{T}$ from left to right. If the node-set of an internal node of $\mathcal{T}$ is stabbed by a query range q, then the subtree rooted at that node must have two consecutive leaves l and l' such that the edge $\{x, x'\}$ of the spanning path is stabbed by q, where $N(l) = \{x\}$ and

$N(l') = \{x'\}$. By definition there are no more than $\sigma(\mathcal{P})$ stabbed edges, hence no more than $\sigma(\mathcal{P})\lceil \log n \rceil$ stabbed node-sets. Each stabbed node-set gives rise to two visited nodes, which accounts for all of them, save the root.

As regards (iii), assign an arbitrary left-to-right order among the children of every internal node of $\mathcal{T}$, and let $l_1, \ldots, l_n$ denote the leaves of $\mathcal{T}$ from left to right. Next, form the spanning path $(x_1, x_2, \ldots, x_n)$ of P, where $\{x_i\}$ is the node-set of l_i. If the edge $\{x_i, x_{i+1}\}$ is stabbed by a range q, we charge this event to the unique child of the nearest common ancestor v of l_i and l_{i+1} that is also an ancestor of l_i (or l_i itself). Since $N(v)$ is necessarily stabbed by q, the child which takes the charge is visited. Furthermore, such a node cannot be charged twice. □

The problem is now to compute spanning trees of a low stabbing number. But before doing so, we establish a simple lower bound on the minimum stabbing number of a spanning tree. This tells us where to set our sights.

Lemma 3.2. *Let (X, R) be a range space with a dual shatter function $\pi^*(m)$ in $\Omega(m^d)$, for $d \geq 1$. Then, for any n_0, there exists a set P of $n > n_0$ points in X such that every spanning tree of P has a stabbing number at least $cn^{1-1/d}$, for some constant $c > 0$.*

Proof. Let $a > 0$ be a constant such that $\pi^*(m) \geq am^d$ for infinitely many integers $m > 0$. For such a value of m, there exists a set Q of m ranges in R with at least am^d cells. Now let P be a set of $n = \lceil am^d \rceil$ points in X, no two of which lie in the same cell of Q. Each edge of any spanning tree T of the points is stabbed by at least one of the m ranges of R, therefore one range must stab at least $(n-1)/m = \Omega(n^{1-1/d})$ edges of T. □

From Lemma 3.1(iii) we conclude that under the conditions of Lemma 3.2 no partition tree can have a stabbing number in $o(n^{1-1/d})$.

4. Computing Spanning Trees of a Low Stabbing Number

Let (X, R) be a range space with dual shatter function $\pi^*(m) = O(m^d)$, for some constant $d \geq 1$. Let P be a set of n points in X and let Q be a multiset of m ranges in R. There exist two points $p, p' \in P$ such that the set $\{p, p'\}$ is not stabbed by more than roughly $m(\log n)/n^{1/d}$ ranges of Q (counting multiplicities). This fact, which is proven below, is the main building block for constructing a spanning tree of a low stabbing number. The idea is to connect p and p' by an edge and discard one of the points from further consideration. If we iterated in this fashion, we would obtain a tree T whose edges are guaranteed not to be stabbed by too many ranges of Q. In turn, this would ensure that most ranges in Q stab only few edges of T. This technique is similar to the greedy algorithms for computing minimum spanning trees. To strengthen the result and ensure that *all* and not just *most* ranges stab few edges, we use a weighting mechanism. Once the first

edge of T is chosen, we identify every range stabbing it and duplicate it. Any edge chosen subsequently has a similar effect on all the ranges stabbing it, including those already duplicated. This is a way of influencing the choice of future tree edges: if a range q stabs a new edge of the tree, each pair of points stabbed by q is, in effect, moved further apart (in the pseudodistance defined by the number of ranges stabbing a pair of points) and made less likely to be subsequently picked as a tree edge. The duplicating process increases the size of the multiset Q geometrically, but the progression rate is kept fairly small thanks to the edge-selection process. On the other hand, every time a given range stabs a new tree edge it is duplicated. This also gives us a geometric progression, but one of much higher rate. Consequently, no range can be duplicated too much and the stabbing number is thus kept low.

Lemma 4.1. *Let (X, R) be a range space with dual shatter function $\pi^*(m) = O(m^d)$, for some constant $d \geq 1$. Let P be a set of n points in X and let Q be a multiset of m ranges in R. There exists a pair of points in P which is not stabbed by more than $cm(\log n)/n^{1/d}$ ranges of Q, where c is a constant.*

Proof. For $x \in X$, let R_x be the set of ranges in R that contain x, and, for $x, y \in X$ $(x \neq y)$, let R_{xy} be the set of ranges in R that stab $\{x, y\}$. The range space $(R, \{R_x \mid x \in X\})$ is the dual of (X, R) and is therefore of finite VC-dimension (Lemma 2.1). Because of Lemma 2.2 and the fact that $R_{xy} = (R_x \cup R_y) \setminus (R_x \cap R_y)$ we find that the range space $\mathcal{R} = (R, \{R_{xy} \mid x, y \in X, x \neq y\})$ also has finite VC-dimension. This implies (Lemma 2.4) that for every ε $(0 < \varepsilon < 1)$ there exists an ε-net N of Q for $\mathcal{R}$, with $|N| \leq b(1/\varepsilon)\log(1/\varepsilon)$, for some constant b. Since $\pi^*(m) = O(m^d)$, the setting $\varepsilon = c(\log n)/n^{1/d}$ gives us $\pi^*(|N|) < n$, for some appropriate choice of a constant c. By the pigeonhole principle, two points of P must fall in the same cell of N, therefore the pair which they form cannot be stabbed by more than $\varepsilon m = cm(\log n)/n^{1/d}$ ranges of Q. □

The next result shows how to get the construction of the spanning tree started. The technique is used in several different contexts, so we have expressed the lemma in terms of a parameter function β.

Lemma 4.2. *Let (X, R) be a range space of finite VC-dimension and let $\beta(n)$ be a decreasing function (for $n > n_0$) which tends to 0 as n goes to infinity. Assume that, for any finite set $P \subseteq X$ and any finite multiset $Q \subseteq R$, there exists a pair of points in P which is not stabbed by more than $|Q|\beta(|P|)$ ranges of Q. Given a set P of n points in X, it is then possible to form a forest of trees with at least $n/2$ edges, such that every range of R stabs $O(n\beta(n/2) + \log n)$ edges.*

Proof. Let P be a set of n points in X and let Q_0 be a minimum set of ranges in R such that $\{P \cap q \mid q \in R\} = \{P \cap q \mid q \in Q_0\}$. If (X, R) has VC-dimension d, we know from Lemma 2.1 that $|Q_0| = O(n^d)$. Let $\{p, p'\}$ be a pair of points in P that is not stabbed by more than $|Q_0|\beta(n)$ ranges of Q_0. We make the pair $\{p, p'\}$ the first edge of our forest. The ranges of Q_0 that stab this edge are not nearly as

fresh and young as the others, so we duplicate each of them, thus producing a multiset Q_1. Next, we iterate the whole procedure with respect to the set of points $P\backslash\{p\}$ and the new multiset of ranges. Note that to duplicate a range with multiplicity μ means to give it multiplicity 2μ. All in all, we iterate through the procedure $\rho = \lceil n/2 \rceil$ times. The size of the final multiset Q_ρ is at most

$$(1+\beta(n-\rho+1))^\rho |Q_0| \leq |Q_0|\, e^{\rho\beta(n/2)} \leq |Q_0|\, e^{n\beta(n/2)}.$$

Because of the duplication policy, no range can stab more than $\log |Q_\rho|$ edges. The fact that $|Q_0| = O(n^d)$ completes the proof. □

We conclude with the main result of this section: the existence of spanning paths of a low stabbing number for range spaces of finite VC-dimension. This existence is characteristic of finite VC-dimensionality.

Theorem 4.3. *Let (X, R) be a range space with a dual shatter function $\pi^*(m)$ in $O(m^d)$, for some constant $d \geq 1$. Any nonempty set of n points in X admits a spanning path with the stabbing number $O(n^{1-1/d} \log n)$, if $d > 1$, and $O(\log^2 n)$, if $d = 1$.*

Proof. Let P be a set of n points in X. From Lemma 4.1 we easily check that the function $\beta(n) = c(\log n)/n^{1/d}$ satisfies all the conditions of Lemma 4.2. In particular, we derive the finite VC-dimensionality of (X, R) from the fact that its dual shatter function is bounded by a polynomial (Lemmas 2.1 and 2.3). Therefore, there exists a forest spanning at least half the points of P, with a stabbing number in $O(n^{1-1/d} \log n)$. Keep one point per tree of the forest and apply the same construction to the remaining points. Iterate on this process until the number of points left falls below $n^{1-1/d} \log n$. Finally, connect the remaining points via an arbitrary spanning path. This procedure produces a spanning tree of P with a stabbing number at most proportional to

$$\sum_{k \geq 0} ((n/2^k)^{1-1/d} \log(n/2^k)).$$

This gives $O(n^{1-1/d} \log n)$, if $d > 1$, and $O(\log^2 n)$, if $d = 1$. Lemma 3.1(i) completes the proof. □

Theorem 4.3 not only characterizes the existence of good spanning trees, it also shows how to construct them. First, compute a set Q_0 of representative ranges (hopefully, in time polynomial in $\pi(n)$, where $\pi(n)$ is the primal shatter function of (X, R)). If testing membership in a range is tractable, then we easily construct a good spanning tree (along the lines of Theorem 4.3) in polynomial time. Note that the duplication mechanism might generate weights of exponential size. We can cope with that difficulty by using linear-size arrays to emulate long computer words. From Lemma 3.1 we conclude to the existence of a balanced binary partition tree with sublinear visiting number, if and only if the range-searching problem is defined over a range space of finite VC-dimension. Recall

that this result is mostly of theoretical interest. Indeed, the visiting number is a realistic complexity measure only if testing intersection between a node-set and a query range can be performed in constant time (or at least reasonably fast). Sometimes that problem alone might be almost as difficult as the original range-searching problem (cf. our discussion of simplex range searching in the next section).

What about the complexity of range searching in the arithmetic model [10], [22]? Recall that in that model, a data structure is a collection of precomputed weights, each associated with a certain subset of P, called a generator. The query time is measured as the maximum, over all queries $q \in R$, of the minimum number of generators whose union is $P \cap q$. In this way, note that a data structure essentially works for all weight assignments: if we change the weights of the input points, all we have to do is re-evaluate the cumulative weight of each generator and the new data structure will work just the same. A spanning path of P provides the terrain for an efficient data structure. Preprocess the sequence of weights along the path, following the method for the partial sum problem described in [21]. With this preprocessing, the cumulative weight of any interval along the spanning path can be computed in time $O(\alpha(n))$, where $\alpha(n)$ is a functional inverse of Ackermann's function (see the Appendix). The storage requirement is $O(n)$. Returning to our range-searching problem, we conclude that any query can be answered in time $O(\sigma\alpha(n))$, where σ is the stabbing number of the spanning path. Once again, bear in mind that this solution is incomplete because it brushes aside the problem of computing the interval decomposition. In the arithmetic model, however, none of that work would be charged anyway. Therefore our solution, though unrealistic as it may be, can be meaningfully compared against the best lower bounds obtained in the arithmetic model. This is what we do next, right after summarizing our results below.

Theorem 4.4. *Given a range space (X, R) of finite VC-dimension and a set P of n points in X, there exists a partition tree for P with a visiting number $O(n^b \log^2 n)$, for some constant $b < 1$. In the arithmetic model, there also exists a linear-size data structure with query time in $O(n^b \alpha(n) \log n)$. The constant b can be set to $1 - 1/d$, where d is the least integer ≥ 1 such that the dual shatter function $\pi^*(m)$ is in $O(m^d)$. If $d = 1$, then the visiting number (resp. query time in the arithmetic model) is $O(\log^3 n)$ (resp. $O(\alpha(n) \log^2 n)$). All these results are optimal within polylogarithmic factors.*

How do we justify our claim of optimality? By calling on Lemmas 3.1 and 3.2 in the case of partition trees. But what about the complexity of range searching in the arithmetic model? It suffices to prove that for any $d > 1$ there exists a range-searching problem whose dual shatter function $\pi^*(m)$ is in $O(m^d)$, and for which any linear-size solution has a worst-case query time of $\Omega(n^{1-1/d}/\log n)$ in the arithmetic model. (Notice that we can ignore the case $d = 1$.) But this lower bound is a particular case of a space-time tradeoff proven in [3] for simplex range searching in d-space over a semigroup: Given a collection P of n weighted points in E^d and a query simplex q, compute the cumulative weight of $P \cap q$. It

is easily verified that the dual shatter function of the primal range space is in $O(m^d)$.

Observe that the optimality result is stronger for partition trees than for the arithmetic model. The reason is that in the former case the lower bound holds for *any* range space of finite VC-dimension, whereas in the latter we must exhibit one specific range space. We have already seen that no good partition trees exist if the dimension is infinite. We can strengthen this result and prove that no linear-size, sublinear query-time solution can exist in the arithmetic model. This is actually an immediate consequence of Theorem 3.2 of [3]. Adapted to our purposes, the theorem implies that given any range space of infinite VC-dimension and any integer function $p(n)$ $(n < p(n) \leq 2^n)$, to ensure a query time of less than $cn/\log(p(n)/n)$ requires the use of $\Omega(p(n))$ storage, where c is some appropriate constant. This establishes the following characterization.

Theorem 4.5. *In the arithmetic model a range-searching problem admits a linear-size, sublinear query-time solution if and only if the underlying range space has finite VC-dimension. Furthermore, if the dimension is finite, then there exists a partition tree with a sublinear visiting number.*

5. Simplex and Spherical Range Searching

We could use our previous results to establish new upper bounds on the complexity of several geometric searching problems. Unfortunately, the stabbing numbers of the resulting spanning trees exceed the lower bound of Lemma 3.2 by a factor of $\log n$. Lemma 4.2 is the keystone of the construction. Plug in a value for $\beta(n)$ and a partition tree will automatically result. Finding an appropriate function β is what Lemma 4.1 is all about. We will see, however, that in the particular cases of simplex and spherical range searching the geometry of the problems allows us to finetune the bound of Lemma 4.1 by removing the factor of $\log n$. This will result in similar improvements for partition trees and arithmetic-model solutions. The main idea is to study the properties of the pseudodistance defined by the number of ranges stabbing a pair of points. (We cannot quite call this function a distance because two distinct points may be at a distance 0 of each other.) We will show that this pseudodistance satisfies packing properties similar to the Euclidean metric.

Let $p_1, \ldots, p_n$ be n points of E^d, called *sites*, and let $\pi_1, \ldots, \pi_m$ be a finite collection of closed halfspaces (choosing them open would work just the same). To avoid dealing with multisets, we assign a positive real number w_i (a weight) to each halfspace π_i. The sum of all the weights is denoted Δ. Given any two points p and p', we define the pseudodistance $\delta(p, p')$ as the sum $\sum_i w_i$, taken over all halfspaces π_i that stab the pair $\{p, p'\}$. Note that Δ is the (finite) diameter of the entire space. One trivial, yet crucial, property of δ is that it satisfies the triangular inequality.

Let H be the arrangement formed by the hyperplanes bounding the m halfspaces. We assume that the m hyperplanes are in general position and that each

weight w_i is equal to 1. Because of general position, any vertex of H is the intersection of exactly d hyperplanes. Given a point p and a real r, we define the ball $B(p, r)$ as the set of vertices v of H such that $\delta(p, v) \leq r$. The *volume* of $B(p, r)$ denotes its cardinality. The pseudodistance δ shares some fundamental properties with the Euclidean metric. For example, Lemma 5.1 says that, for r not too large, a ball of radius r has volume $\Omega(r^d)$. More important, Lemma 5.2 asserts that in the pseudometric δ the two nearest sites are only $O(\Delta/n^{1/d})$ apart. A similar, well-known fact in E^d is that if a set of n points has Euclidean diameter Δ, then the Euclidean distance between the two nearest points is $O(\Delta/n^{1/d})$.

Lemma 5.1. *Given m halfspaces in general position in E^d, let p be a point of E^d and let r be a real such that $0 \leq r \leq m$. If the halfspaces are assigned unit weight, then the volume of $B(p, r)$ is at least* $\binom{\lfloor r \rfloor}{d} / d!$.

Proof. If p lies on the boundary of some halfspaces, we can always perturb p toward the intersection of those halfspaces (which, because of general position, is nonempty) without changing the pseudodistance between p and any point of E^d.

We now proceed by induction on d. Let $g_d(m, r)$ be the minimum volume of any ball $B(p, r)$ in E^d with respect to any arrangement of m closed halfspaces in general position. If $d = 1$, then we have $g_1(m, r) = \lfloor r \rfloor$. Assume now that $d > 1$. Because of general position there exists a line L passing through p which does not intersect any two bounding hyperplanes at the same point but still intersects each of them. Let $q_1, \ldots, q_m$ be the sequence of intersections between L and the hyperplanes. The sequence is chosen so that the Euclidean distance between p and q_i is nondecreasing; thus, $\delta(p, q_i) \leq i$. Let π_k^* be the bounding hyperplane associated with q_k and let H_k denote the arrangement formed by the $(d-2)$-flats $\pi_k^* \cap \pi_l^*$ $(l \neq k)$. Each H_k is an arrangement of $m-1$ unit-weight hyperplanes in general position in E^{d-1}, and the restriction of δ to π_k^* is itself a pseudodistance of the same type in $(d-1)$-space. Therefore, using the monotonicity of $g_{d-1}(m, r)$ in r and the facts that δ satisfies the triangular inequality and that every vertex lies on exactly d hyperplanes, we have

$$dg_d(m, r) \geq \sum_{1 \leq k \leq \lfloor r \rfloor} g_{d-1}(m-1, r-k) \geq \sum_{1 \leq k \leq \lfloor r \rfloor} \binom{\lfloor r-k \rfloor}{d-1} \Big/ (d-1)!,$$

from which it follows that

$$g_d(m, r) \geq \frac{1}{d!} \sum_{0 \leq k \leq \lfloor r \rfloor - 1} \binom{k}{d-1} = \frac{1}{d!} \binom{\lfloor r \rfloor}{d}. \qquad \square$$

Lemma 5.2. *Given m weighted halfspaces and n points $p_1, \ldots, p_n$ in E^d, if n is large enough, there exist two points p_i and p_j $(i < j)$ such that $\delta(p_i, p_j) \leq 2\lceil d\Delta/n^{1/d} \rceil$, where Δ is the sum of all the weights.*

Proof. We begin by assuming that each weight w_i is equal to 1 and that the set of halfspaces is in general position. If

$$n\binom{\lfloor r \rfloor}{d} \Big/ d! > \binom{m}{d}, \tag{5.1}$$

then there are two points p_i and p_j $(i<j)$ such that both $\delta(p_i, q) \le r$ and $\delta(p_j, q) \le r$, for some point $q \in E^d$. The reason is that, otherwise, the n balls $B(p_i, r)$ would be disjoint and, by Lemma 5.1, their combined volume would exceed the total number of vertices. From the triangular inequality it follows that $\delta(p_i, p_j) \le 2r$. We easily check that (5.1) holds for the assignment

$$r = \lceil dm/n^{1/d} \rceil,$$

provided that $d \ge 2$ and $m \ge n^{1/d}$. If $d = 1$, the lemma is trivial. If $d > 1$ and $m < n^{1/d}$, then there are more points than there are cells (i.e., d-faces) in the arrangement defined by the m halfspaces: indeed, for m not too small, the number of cells is at most $\sum_{0 \le k \le d} \binom{m}{k} \le m^d$ [7]. Because the halfspaces are in general position we can always perturb the points away from the bounding hyperplanes without altering the pseudodistances between points. Having done that, we now find that at least two points belong to the same cell and therefore lie at a pseudodistance 0 of each other. The proof is now complete under our restrictive assumptions.

Let us now turn to the case of positive integral weights $w_1, \ldots, w_m$. The idea is to make w_i copies of each π_i and perturb them a little. Given the nature of this operation we might as well assume that the original halfspaces are not necessarily in general position (this will kill two birds with one stone). We perturb the halfspaces in two steps. First, we move each bounding hyperplane by a small random translation directed toward the outside of the halfspace. This will remove all possible contact between sites and bounding hyperplanes. It will also guarantee that no more than d hyperplanes can intersect in one point. To complete the general positioning, we perturb each halfspace with a (very small) random rotation. This rotation should be small enough so that no site leaves any halfspace in the process. In this way, we achieve general position without changing the pseudodistance between any pair of sites.

For the sake of generality we now consider the case of arbitrary positive real weights. Pick some large integer k and replace each w_i by the integral weight $w_i' = \lfloor kw_i \rfloor$. Our previous generalization shows that there exist two sites p_i and p_j $(i<j)$ such that $\delta'(p_i, p_j) \le 2\lceil d\Delta'/n^{1/d} \rceil$, where $\Delta' = \sum_{1 \le i \le m} \lfloor kw_i \rfloor$ and δ' is the pseudometric δ modified in the obvious way. Let J be the set of indices l such that π_l stabs the pair $\{p_i, p_j\}$. We have

$$\sum_{l \in J} \lfloor kw_l \rfloor \le 2\lceil d\Delta'/n^{1/d} \rceil,$$

therefore

$$\delta(p_i, p_j) = \sum_{l \in J} w_l \le 2\lceil d\Delta/n^{1/d} \rceil + \varepsilon(k),$$

where $\varepsilon(k)$ goes to 0 with $1/k$. Since the inequality holds for arbitrarily large k, the proof is now complete. □

In light of the previous section (especially Lemma 4.2), Lemma 5.2 allows us to conclude to the existence of a spanning tree of P with stabbing number in $O(n^{1-1/d})$. Lemma 3.2 shows that this result is optimal. Since a simplex is the intersection of $d+1$ halfspaces, the spanning tree will have the same stabbing number, up to within a constant factor (dependent on d), with respect to simplices. This is a general principle which holds for any range space defined by a constant number of set-theoretic operations over a given range space. Applying the ideas leading to Theorem 4.4, we have the following result.

Theorem 5.3. *Simplex range searching on n points in E^d can be performed in $O(n^{1-1/d}\alpha(n))$ query time and $O(n)$ storage in the arithmetic model. It can also be solved with a partition tree with a visiting number $O(n^{1-1/d}\log n)$. These bounds are optimal within logarithmic factors.*

This improves upon previous work on this problem [20], [9], [23], [24], [11]. It must be mentioned that the solutions just quoted also hold on a random access machine. See also [8] for probabilistic variants. The best previous partition tree [11] has a visiting number of $O(n^{d(d-1)/(d(d-1)+1)+\varepsilon})$, for any $\varepsilon > 0$.

Let us now generalize our technique to spherical range searching. The range space (X, R) now consists of $X = E^d$ and the set R of all closed d-balls. As shown in [24] spherical range searching in E^d is a special case of halfspace range searching in E^{d+1}. Theorem 5.3 is thus ready for action. Also, it is easy to see that the range space is of finite VC-dimension, which makes the problem amenable to Theorem 4.4. We will obtain better results, however, if we can treat d-balls as we did halfspaces and stay in d-space. As halfspaces are bounded by hyperplanes, d-balls are bounded by $(d-1)$-spheres, which we call *hyperspheres*. The difficulty with hyperspheres is that any d of them should not be expected to intersect always, as was the case with hyperplanes in general position. This will necessitate a revision of our volume-based argument.

We now have n sites $p_1, \ldots, p_n$ and m d-balls $\pi_1, \ldots, \pi_m$ in E^d. For consistency, we assume that the sites themselves lie on a d-sphere S^d in E^{d+1}. This does not really matter since S^d can always be chosen very big. On the other hand, it allows us to redefine each π_i as the intersection of S^d with some halfspace. As usual, each "d-ball" π_i is assigned a real weight $w_i > 0$. The reason for switching to S^d is to salvage the induction used in the proof of Lemma 5.1. The pseudodistance δ is still defined exactly the same way, that is, $\delta(p, p')$ is the cumulative weight of all the d-balls stabbing the pair of points $\{p, p'\}$. Note that the sum of the weights is no longer the diameter of the space (at least not always).

Let H be the arrangement formed by the m bounding hyperspheres, denoted $\pi_1^*, \ldots, \pi_m^*$. For the time being we assume that the set of defining halfspaces is in general position and that each of the m d-balls is assigned unit weight ($w_i = 1$). The ball $B(p, r)$ is defined just as before, but its volume is not. The reason is that the volume of S^d might no longer be on the order of m^d. If things went

nicely, then every set of d hyperspheres would intersect in exactly two points and the volume of S^d would be precisely $2\binom{m}{d}$. Unfortunately, we might have much fewer intersections. To cope with this problem, we put H in normal form by adding dummy vertices to it. A *normalized* arrangement consists of real and dummy vertices. The volume of the ball $B(p, r)$ is now defined as the number of *dummy* vertices that it contains. But what are those dummy vertices, anyway?

We define the normalization procedure by induction on the dimension d. If $d = 1$, an arrangement of m 0-spheres in general position on S^1 has precisely $2m$ real vertices: we place a dummy vertex at each real vertex. Assume now that $d > 1$. For each hypersphere π_i^* in turn, consider the arrangement of $(d-2)$-spheres formed by intersecting π_i^* with each π_j^* $(j \neq i)$. If the intersection is empty, then we replace π_j^* by a new hypersphere that intersects π_i^* and leaves the current set of intersections $\{\pi_i^* \cap \pi_l^* \mid 1 \leq l \leq j\ (l \neq i)\}$ in general position. The replacement of π_j^* is called its *i-substitute*. We can now carry out the normalization procedure recursively with respect to the new set of $m-1$ $(d-2)$-spheres of the form $\pi_i^* \cap \pi_j^*$ $(j \neq i)$. Note that this process may produce several dummy vertices with the same location in S^d. We easily verify that a normalized arrangement of m hyperspheres in S^d has exactly $2\binom{m}{d}d!$ dummy vertices. (A simple interpretation of this number is that each sequence of d hyperspheres produces exactly two dummy vertices.) Recall that dummy vertices do not affect the definition of δ and that the volume of $B(p, r)$ counts *only* dummy vertices. We are now ready to revisit Lemma 5.1.

Lemma 5.4. *Given a normalized arrangement of m d-balls in S^d in general position, let p be a point of S^d and let r be a real such that $0 \leq r \leq m$. If the d-balls are assigned unit weight, then the volume of $B(p, r)$ is at least $2\binom{\lfloor r \rfloor}{d}$.*

Proof. By using a perturbation argument similar to the one used in the proof of Lemma 5.1 we can assume that p does not lie on any of the m hyperspheres. As usual, we now proceed by induction on d. Let $g_d(m, r)$ be the minimum volume of any ball $B(p, r)$ in S^d. If $d = 1$, then we have $2m \geq 2r$ vertices on S^1. This implies that we can walk clockwise from p until we cross $\lfloor r \rfloor$ vertices, and then do the same counterclockwise without interference. It follows that $g_1(m, r) \geq 2\lfloor r \rfloor$ (which is tight). Assume now that $d > 1$ and let p' be a point of S^d which maximizes the distance $\rho = \delta(p, p')$. Because the hyperspheres are in general position we can always assume that p' does not lie on any of them. If $\rho \leq r$, then the volume of $B(p, r)$ is that of S^d, that is, $2\binom{m}{d}d! \geq 2\binom{\lfloor r \rfloor}{d}$. Assuming now that $\rho > r$, consider a circle (that is, the intersection of S^d with a two-dimensional flat of E^{d+1}) which contains both p and p' and avoids any $\pi_i^* \cap \pi_j^*$ $(i < j)$. Think now of a point q moving continuously along the circle from p to p' in some given direction. The distance $\delta(p, q)$ goes from 0 to ρ by steps of $+1$ or -1. Therefore,

q crosses at least $\lfloor r \rfloor$ *distinct* hyperspheres at points $q_1, \ldots, q_l$, such that $\delta(p, q_k) \leq k$ $(1 \leq k \leq l)$. The hypersphere $\pi^*_{i_k}$ passing through each q_k is now regarded as the underlying space S^{d-1} of an arrangement H_k of unit-weight $(d-2)$-spheres. To ensure that the arrangement consists of $m-1$ spheres, we include all the k-substitutes defined in the normalization of H. Note that, by inheriting the dummy vertices of H, the arrangement H_k is itself normalized. Let δ_k be the usual pseudodistance in S^{d-1} defined with respect to H_k. It is interesting to observe that because of the k-substitutes δ_k is not, in general, the restriction of δ to $\pi^*_{i_k}$. However, the δ-distance between any two points of $\pi^*_{i_k}$ cannot exceed their δ_k-distance. Using the monotonicity of $g_{d-1}(m, r)$ in r and the fact that δ satisfies the triangular inequality, we have

$$g_d(m, r) \geq \sum_{1 \leq k \leq \lfloor r \rfloor} g_{d-1}(m-1, r-k) \geq 2 \sum_{1 \leq k \leq \lfloor r \rfloor} \binom{\lfloor r-k \rfloor}{d-1}$$

$$\geq 2 \sum_{0 \leq k \leq \lfloor r \rfloor - 1} \binom{k}{d-1} = 2\binom{\lfloor r \rfloor}{d}. \qquad \square$$

Lemma 5.5. *Given m weighted d-balls and n points $p_1, \ldots, p_n$ in E^d, if n is large enough, there exist two points p_i and p_j $(i<j)$ such that $\delta(p_i, p_j) \leq 2\lceil d\Delta / n^{1/d} \rceil$, where Δ is the sum of all the weights.*

Proof. It is almost identical to that of Lemma 5.2. We briefly sketch it. As usual, we first assume that each weight w_i is equal to 1 and that the set of d-balls is in general position. Our first task is to normalize the arrangement of the hyperspheres. Since the total number of dummy vertices is $2\binom{m}{d}d!$, satisfying the inequality (5.1) again ensures the existence of two points p_i and p_j $(i<j)$ such that both $\delta(p_i, q) \leq r$ and $\delta(p_j, q) \leq r$, for some point $q \in E^d$ (Lemma 5.4). From the triangular inequality it follows that $\delta(p_i, p_j) \leq 2r$. As we saw earlier, (5.1) is satisfied for

$$r = \lceil dm / n^{1/d} \rceil,$$

provided that $d \geq 2$ and $m \geq n^{1/d}$. If $d = 1$, then, for accounting purposes, let us embed the supporting line in S^1. That way we have n fundamental intervals, one of which, $p_i p_j$, contains at most $2m/n$ endpoints of 1-balls. Obviously, $\delta(p_i, p_j) \leq 2m/n$, which proves the lemma. Assume now that $d > 1$ and $m < n^{1/d}$. We need to evaluate the maximum number $f(d, m)$ of cells in an arrangement of m hyperspheres in S^d, We have the recurrence $f(1, m) = 2m$ and $f(d, m) = f(d, m-1) + f(d-1, m-1)$. Using a path-counting method (e.g., [14]), we easily find that, for m large enough,

$$f(d, m) = 2 \sum_{0 \leq k \leq d} \binom{m-1}{k} \leq m^d < n.$$

But if $d > 1$, $f(d, m)$ is no less than the maximum number of cells in an arrangement of m hyperspheres in E^d. The remainder of the proof is identical to that of Lemma 5.2. □

From Lemmas 5.4 and 5.5 we draw the same conclusions as expressed in Theorem 5.3. But to begin with we state a nice geometric result of independent interest.

Theorem 5.6. *Any set of n points in E^d admits a spanning path, only $O(n^{1-1/d})$ of whose edges can be stabbed by any d-ball (or halfspace). This upper bound is optimal in the worst case.*

Theorem 5.7. *Spherical range searching on n points in E^d can be performed in $O(n^{1-1/d}\alpha(n))$ query time and $O(n)$ storage in the arithmetic model. It can also be solved with a partition tree with a visiting number $O(n^{1-1/d}\log n)$. These bounds are optimal within logarithmic factors.*

6. Disk, Tetrahedron, and Polygon Range Searching

What happens if we have a more realistic model of computation such as a random access machine or a pointer machine? Let us look at spherical range searching in two dimensions, which is often called *disk* or *circular range searching*. Given n points in E^2, count how many points lie inside a query circle. We can use the partition tree of Theorem 5.7. But this requires implementing the operation: given a query disk D and a node-set $N(v)$, check whether $N(v)$ lies (i) entirely inside D, (ii) entirely outside D, or (iii) neither of the above. This can be done by building both the nearest- and furthest-neighbor Voronoi diagrams and preprocessing them for fast point location [7], [13], [15]. Checking the furthest-neighbor diagram will tell us about (i), while the nearest-neighbor diagram will handle (ii), and therefore (iii). Each operation will cost $O(\log n)$ time, which will bring the query time to $\log n$ times the visiting number of the partition tree, that is, $O(\sqrt{n}\log^2 n)$.

Here is an equivalent way of doing things, which illustrates the kinship between Voronoi diagrams and convex hulls (see [7] for details on this relationship). Following Yao [24], we lift the problem to E^3 and reduce it to intersecting a query plane with a polygonal curve $C = (p_1, \ldots, p_n)$: map the site $(x, y, 0)$ to the point (x, y, z), where $z = x^2 + y^2$, and map the query circle $(x-a)^2 + (y-b)^2 = r^2$ to the plane $z = a(2x-a) + b(2y-b) + r^2$. The sites contained in a query circle are precisely those mapping below the plane associated with the circle. The question is now: given a plane π, find all the edges of C split by π. To do so, we set up a complete binary tree of n leaves. The leaves are associated with $p_1, \ldots, p_n$, from left to right. Each internal node v of the tree is associated with the set $S(v)$ consisting of all the points whose corresponding leaves are descendants of v. The idea is to store in v a pointer to the convex hull of $S(v)$. Using a linear-size data structure [5] we can check if a query halfspace stabs the set

$S(v)$ in $O(\log n)$ time. Using straightforward arguments, we conclude to the existence of an $O(n \log n)$-size data structure for computing all k intersections between C and a query plane in time $O((k+1)\log^2 n)$.

Theorem 6.1. *Disk range searching on n points can be performed in $O(\sqrt{n}\log^2 n)$ query time and $O(n \log n)$ storage on a random access machine or a pointer machine.*

Let us now go up one dimension and consider simplex range searching in 3-space (also known as *tetrahedron range searching*). Using the partition tree of Theorem 5.3, we need the following primitive: given a finite set S of points in 3-space and a tetrahedron q, determine whether S lies entirely inside q, or entirely outside q, or neither of the above. As it turns out, this is not so easy to decide. Instead, we simply check whether S is stabbed by any of the halfspaces bounding q, which can be done in logarithmic time and linear space [5]. If the answer is no, then we can immediately answer the previous question. Otherwise, we cannot conclude, but it is safe to proceed down the recursion anyway, since the stabbing numbers of the four planes add up to $O(n^{2/3})$. We easily derive the following result.

Theorem 6.2. *Tetrahedron range searching on n points can be performed in $O(n^{2/3}\log^2 n)$ query time and $O(n \log n)$ storage on a random access machine or a pointer machine.*

Finally, we consider the polygon range-searching problem. Given a set P of n weighted points in E^2 and a query *convex* k-gon K, compute the cumulative weight of $P \cap K$. We still assume that the underlying model of computation is a random access machine or a pointer machine. Let $p_1, \ldots, p_n$ be the input points ordered along the spanning path provided by Theorem 5.6. This gives us an n-gon Π which, unfortunately, might not be simple. It is obvious, however, that the diagonal-switching trick of the traveling salesman problem will make the polygon Π simple without increasing the stabbing number. This idea is developed in [8] to which we refer the reader for details. The gist of the method is to take all intersecting edges (a, b) and (c, d) and replace them by either (a, c) and (b, d), or (a, d) and (b, c), whichever choice keeps the polygonal line connected.

It has been shown in [4] that ray-shooting in Π can be performed in $O(\log n)$ time, using $O(n)$ space. (Ray-shooting means finding the first hit of a ray directed toward Π.) This allows us to find the intersections of Π and K in time proportional to $k\sqrt{n}\log n$, which gives us the desired interval decomposition of the vertices of Π. From there, we complete the computation in an overall time of $O(k\sqrt{n}\log n)$. This is a simple exercise, so let us move on. The factor $k\sqrt{n}$ is an asymptotic upper bound on the maximum number of intersections between the boundaries of Π and K. We show below that a clever choice of Π can limit this number to $O(\sqrt{kn})$.

Let k be a fixed integer between 1 and n. We subdivide the set of n points into k subsets of size $\lceil n/k \rceil$ or less. The first subset includes the $\lceil n/k \rceil$ leftmost points, the second subset includes the points whose x-coordinates have ranks ranging between $\lceil n/k \rceil + 1$ and $2\lceil n/k \rceil$, etc. Apply Theorem 5.6 to each subset and turn the polygonal curves into simple polygons $C_1, \ldots, C_k$.

We claim that the edges of any convex k-gon K can stab only a total of $O(\sqrt{kn})$ edges among $C_1, \ldots, C_k$. Why is that so? Imagine $k-1$ vertical lines separating the C_i's. We can cut up edges of K so as to fit between two consecutive vertical lines without introducing more than $2(k-1)$ new edges. Then we can ray-shoot each edge separately and collect the pieces in the obvious way. Adding partial sums, as described in Section 3, provides the desired answer in time $O(\sqrt{kn} \log n)$. Note that the convexity of K is used only to keep the number of preprocessing cuts small. The same method works if, say, K is monotone in a fixed direction. By building $\lfloor \log n \rfloor$ data structures, one for each value of $k = 2^i \leq n$, we achieve the following result.

Theorem 6.3. *Polygon range searching on n points (with convex k-gons as queries and $k \leq n$) can be performed in $O(\sqrt{kn} \log n)$ query time and $O(n \log n)$ storage on a random access machine or a pointer machine; k is the number of vertices of the convex query polygon. The same result holds if the polygon K is monotone in a fixed direction. If k is fixed once and for all, then the storage requirement is only $O(n)$.*

We have shown that for a fixed value of k, a set of n points in E^2 admits a spanning tree of stabbing number $O(\sqrt{kn})$ with respect to polygon range searching. We can generalize the proof of Lemma 3.2 and prove the optimality of this result in the asymptotic sense. We construct a set of n points by carefully arranging k building blocks. A building block is any set affinely equivalent to the m^2 vertices of an m-×-m square grid, with $m = \lfloor \sqrt{n/k} \rfloor$. Given a direction l and a small angle θ, we define an (l, θ)-*block* as a building block whose diagonal has direction l and whose grid axes form an angle θ (Fig. 1).

Now take a regular $9k$-gon centered at the origin and pick k consecutive edges in the northeast quadrant. On each edge chosen, place a small (l, θ)-block, where l is the direction of the edge and θ is a very small angle, say, $\pi/9^{9^k}$. The block should extend over, say, a ninth of the edge (Fig. 2). We define the *canonical lines* of the grid $\{(i, j) \mid 0 \leq i, j \leq m-1\}$ to be the $m+1$ horizontal lines $y = i - \frac{1}{2}$ $(0 \leq i \leq m)$ and the $m+1$ vertical lines $x = i - \frac{1}{2}$ $(0 \leq i \leq m)$. The two extreme vertical and horizontal lines form a square enclosing the grid, which we call its *box*. By affinity, every block has a box and a set of canonical lines.

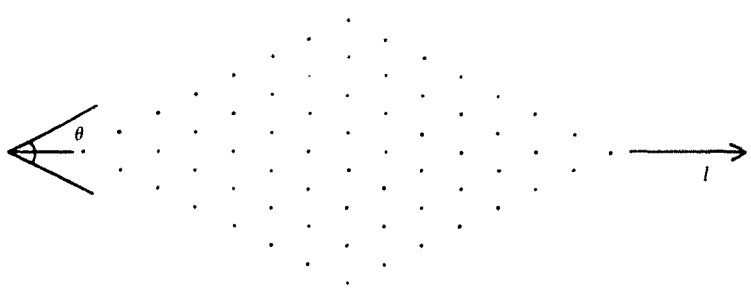

Fig. 1. An (l, θ)-block is affinely equivalent to an m-×-m grid: the angle between its axes is θ and its long diagonal is parallel to l.

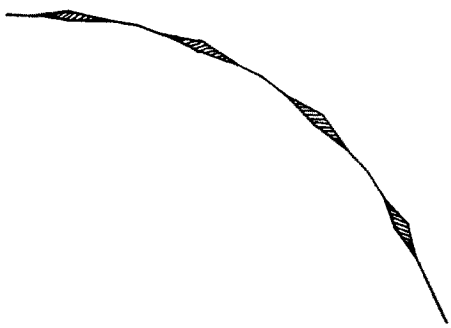

Fig. 2. The set of sites is obtained by arranging k blocks along a regular convex polygon.

Let T be a spanning tree of the set of km^2 sites. Consider the sites of one of the blocks. Each of them is entirely separated from the other sites by canonical lines. On the other hand, each site is connected to at least one other site by an edge of T. This implies that the $2(m+1)$ canonical lines of the block have a total of at least $m^2/2$ intersection points with T. By the pigeonhole principle, one canonical line intersects at least $m/5$ edges of T (for m large enough). Note that the intersections lie in the box of the block. Let us mark this canonical line and do the same for each block. The crucial observation is that these chosen lines form the edges of an unbounded convex k-gon K, each line contributing exactly one edge. Furthermore, all the intersections with T associated with each canonical line happen to lie on its contributed edge. This implies that K stabs at least $mk/5$ edges of T, which is on the order of $\sqrt{kn}$.

Theorem 6.4. *Let k and n be two integers, with $k \leq n$. Any set of n points in E^2 admits a spanning path, only $O(\sqrt{kn})$ of whose edges can be stabbed by any given convex k-gon. This upper bound is optimal in the worst case.*

7. Conclusions

In our entire discussion we have implicitly assumed that each operation on weights could be performed in constant time. This is not the case in the reporting version of the problems, where the cumulative weight of $P \cap q$ is the set itself. It is easy to see, however, that all our upper bounds hold just the same in the reporting case: the only adjustment to be made is adding a term $O(|P \cap q|)$ in the expression of the query times.

We have carefully evaded the issue of preprocessing. It is elementary to check that all the data structures given in this paper can be constructed in polynomial time. This includes—and we add this as a word of caution—the manipulation of large numbers required by the weighting mechanism. The real challenge is to determine how efficiently the construction can be made to be. Recently, Matoušek [12] has given efficient algorithms for the two-dimensional case. Another interesting problem is to determine whether the algorithms of Theorems 5.3 and 5.7 can

be ported to a random access machine. The difficulty here is to determine efficiently the edges of the partition tree that are cut by the query. By using standard techniques we can reduce this problem to that of detecting whether a fixed convex polytope intersects a query hyperplane, using only linear (or quasi-linear) space. To do this in logarithmic or polylogarithmic time in dimension higher than three seems elusive. Finally, the question of providing space-time tradeoffs matching the lower bounds of [3] remains open.

Acknowledgments

We wish to thank the two anonymous referees for their helpful suggestions which helped to improve the presentation of this paper.

Appendix

Below is a definition of the inverse Ackermann function, denoted $\alpha(n)$ [17]. Let $A(i, j)$ be the function defined recursively as follows:

$$A(0, j) = 2j \qquad \text{for any} \quad j \geq 0.$$
$$A(i, 0) = 0 \text{ and } A(i, 1) = 2 \qquad \text{for any} \quad i \geq 1.$$
$$A(i, j) = A(i-1, A(i, j-1)) \qquad \text{for any} \quad i \geq 1 \quad \text{and} \quad j \geq 2.$$

We define $\alpha(n) = \min\{i \mid i \geq 1, A(i, i) > n\}$. For any $m \geq n \geq 1$, we also define the function $\alpha(m, n)$ by $\alpha(m, n) = \min\{i \mid i \geq 1, A(i, 4\lceil m/n \rceil) > \log n\}$. Yao [21] has given a linear-size data structure for partial sum computation: each query is answered in time $O(\alpha(cn, n))$, where c is an appropriate constant. We easily check that $\alpha(cn, n) = O(\alpha(n))$, therefore we are justified to say that the query time is $O(\alpha(n))$.

References

1. Alon, N., Haussler, D., Welzl, E., Wöginger, G. Partitioning and geometric embedding of range spaces of finite Vapnik-Chervonenkis dimension, *Proc. 3rd Ann. ACM Symp. Comput. Geom.* (1987), 331-340.
2. Assouad, P. Densité et dimension, *Ann. Inst. Fourier (Grenoble)* **33** (1983), 233-282.
3. Chazelle, B. Polytope range searching and integral geometry, *Proc. 28th Ann. IEEE Symp. Found. Comput. Sci.* (1987), 1-10. To appear in *J. Amer. Math. Soc.*
4. Chazelle, B., Guibas, L. J. Visibility and intersection problems in plane geometry, *Proc. 1st Ann. ACM Symp. Comput. Geom.* (1985), 135-146. To appear in *Discrete Comput. Geom.*
5. Dobkin, D. P., Kirkpatrick, D. G. Fast detection of polyhedral intersection, *Theoret. Comput. Sci.* **27** (1983), 241-253.
6. Dudley, R. M. Central limit theorems for empirical measures, *Ann. Probab.* **6** (1978), 899-929.
7. Edelsbrunner, H. *Algorithms in Combinatorial Geometry*, Springer-Verlag, Heidelberg, 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 Symp. Comput. Geom.* (1988), 56-69.

9. Edelsbrunner, H., Welzl, E. Halfplanar range search in linear space and $O(n^{0.695})$ query time, *Inform. Process. Lett.* **23** (1986), 289-293.
10. Fredman, M. L. Lower bounds on the complexity of some optimal data structures, *SIAM J. Comput.* **10** (1981), 1-10.
11. Haussler, D., Welzl, E. Epsilon-nets and simplex range queries, *Discrete Comput. Geom.* **2** (1987), 127-151.
12. Matoušek, J. Spanning trees with low stabbing numbers, manuscript, 1988.
13. Mehlhorn, K. *Data Structures and Algorithms 3: Multidimensional Searching and Computational Geometry*, Springer-Verlag, Heidelberg, 1984.
14. Monier, L. Combinatorial solutions of multidimensional divide-and-conquer recurrences, *J. Algorithms* **1** (1980), 60-74.
15. Preparata, F. P., Shamos, M. I. *Computational Geometry*, Springer-Verlag, New York, 1985.
16. Sauer, N. On the density of families of sets, *J. Combin. Theory Ser. A* **13** (1972), 145-147.
17. Tarjan, R. E. Efficiency of a good but not linear set union algorithm, *J. Assoc. Comput. Geom.* **22** (1975), 215-225.
18. 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.
19. Welzl, E., Wöginger, G. On shatter functions of range spaces, manuscript, 1987.
20. Willard, D. E. Polygon retrieval, *SIAM J. Comput.* **11** (1982), 149-165.
21. Yao, A. C. Space-time tradeoff for answering range queries, *Proc. 14th Ann. ACM Symp. Theory Comput.* (1982), 128-136.
22. Yao, A. C. On the complexity of maintaining partial sums, *SIAM J. Comput.* **14** (1985), 277-288.
23. Yao, A. C., Yao, F. F. A general approach to d-dimensional geometric queries, *Proc. 17th Ann. ACM Symp. Theory Comput.* (1985), 163-168.
24. Yao, F. F. A 3-space partition and its applications. *Proc. 15th Ann. ACM Symp. Theory Comput.* (1983), 258-263.

Received August 20, 1988, and in revised form January 15, 1989.