Discrete Comput Geom 8:315–334 (1992)

Discrete & Computational
Geometry
© 1992 Springer-Verlag New York Inc.

Efficient Partition Trees*

Jiří Matoušek

Department of Applied Mathematics, Charles University,
Malostranské nám. 25, 118 00 Praha 1, Czechoslovakia
matousek@cspguk11.bitnet

Abstract. We prove a theorem on partitioning point sets in E^d (d fixed) and give an efficient construction of partition trees based on it. This yields a simplex range searching structure with linear space, $O(n \log n)$ deterministic preprocessing time, and $O(n^{1-1/d}(\log n)^{O(1)})$ query time. With $O(n^{1+\delta})$ preprocessing time, where δ is an arbitrary positive constant, a more complicated data structure yields query time $O(n^{1-1/d}(\log \log n)^{O(1)})$. This attains the lower bounds due to Chazelle [C1] up to polylogarithmic factors, improving and simplifying previous results of Chazelle *et al.* [CSW].

The partition result implies that, for $r^d \leq n^{1-\delta}$, a $(1/r)$-approximation of size $O(r^d)$ with respect to simplices for an n-point set in E^d can be computed in $O(n \log r)$ deterministic time. A $(1/r)$-cutting of size $O(r^d)$ for a collection of n hyperplanes in E^d can be computed in $O(n \log r)$ deterministic time, provided that $r \leq n^{1/(2d-1)}$.

1. Introduction

In this paper we consider a problem known as *simplex range searching*: preprocess a set P of n points in E^d so that, given any query simplex σ, the points in $P \cap \sigma$ can be counted or reported efficiently. Usually this problem is investigated in a more general setting, where a weight function on the points is assumed and a cumulative weight of the points in $P \cap \sigma$ is asked for. The weights are assumed to belong to a semigroup, i.e., we should not use subtractions when computing the answer. We assume that the semigroup operations can be executed in constant time.

---

* Part of this research was performed while the author was visiting at Freie Universität Berlin. The author was partially supported by the ESPRIT II Basic Research Actions Program of the EC under Contract No. 3075 (project Alcom). A preliminary version of this paper appeared in the *Proceedings of the Seventh ACM Symposium on Computational Geometry*, pages 1–9, 1991.

As is typical in computational geometry, we consider the space dimension d as a fixed number (and imagine that it is a small number), thus something depending on the dimension only will be a constant for us. All the algorithms considered are deterministic unless stated otherwise.

The simplex range searching problem has quite a rich history, which reflects its basic importance in computational geometry. We do not try to give an account of this history; we mention only some works directly related to our results.

The possible efficiency of the query answering depends on the amount of storage (and preprocessing time) used. Chazelle [C1] established the following lower bounds: in dimension d and given m units of storage, the worst-case query time is $\Omega((n/\log n)/m^{1/d})$ (in the plane, the bound sharpens to $\Omega(n/\sqrt{m})$; this indicates that, for higher dimensions, the logarithmic factor might be only a product of the proof technique). These lower bounds are valid only under some restrictions on the kind of algorithm used, but so far all known algorithms satisfy these restrictions and there does not seem to be much hope for circumventing the lower bound. Most of the works on upper bounds (i.e., designing efficient simplex range searching algorithms) concern the "small-space" end of the spectrum, i.e., a linear or slightly superlinear storage.

The earlier solutions, starting with Willard [Wi], use the idea of *partition trees.* We briefly explain the idea of a partition tree and its use for query answering. For simplicity, we start with the planar case and half-planar ranges (instead of simplices). What is needed to construct a partition tree is a partitioning scheme of the following kind: Given an n point set, divide the plane into several regions, such that each region contains at least a constant fraction of the points, and that any line misses one (or several) of the regions. Given a query half-plane, we can then treat the points in the regions missed by its boundary line very efficiently: either they all lie inside the query half-plane or all lie outside of it. Thus it remains to handle the points in the regions intersected by the boundary line of the query half-plane. To this end, the partition scheme is applied recursively for subsets in each of the regions, until we reach trivially small subsets (of a constant size). A data structure recording this recursive partition of the point set in a suitable way is called a partition tree. In the query answering process we handle the regions missed by the boundary line of the query half-plane directly, and we proceed recursively down the tree for the regions intersected by the boundary line.

The efficiency of a partition tree is determined by the parameters of the partitioning scheme used. The earlier partition schemes proposed in the literature (e.g., [Wi], [EW], and [YY]) divide the space into a small number of parts, and the ham-sandwich cut theorem or similar theorems are used to establish the existence of such schemes. More efficient partition trees of this kind were devised by Haussler and Welzl [HW]. Their partition scheme has a large but constant number of parts and a probabilistic method is employed for its construction and analysis.

The query time achieved in the partition trees of [HW] is still quite far from the above-quoted lower bound. For the planar case, the first algorithm attaining the lower bound up to a polylogarithmic factor was given by Welzl [We]. The method is refined in [CW]. It works in an arbitrary fixed dimension, even in a more general setting than the geometric one, but in the so-called arithmetic model only. In the arithmetic model we only count the semigroup operations needed to compute the answer to a query, and other operations needed in the algorithm are ignored (or rather left unspecified in the algorithm). Hence the result does not yield an algorithm in the usual sense automatically, and an efficient algorithm is given for dimensions two and three only in [CW].

While the usual partition trees are built "locally," by a recursive application of a partitioning scheme, Welzl's method is a "global" one. However, it can also be crammed under the above-described pattern of a partition tree (actually a single-level one): essentially, it gives a partition scheme, which has $n/2$ regions, each of the regions contains just two points and no line crosses more than $O(\sqrt{n})$ of the regions. This view is a departing point of this paper. With a modification of Welzl's proof, this global partition scheme can be turned into a local one, and we get almost optimal partition trees and an almost optimal solution to the simplex range search problem in higher dimensions.

The most efficient previous solution of the general simplex range searching problem is due to Chazelle *et al.* [CSW]. They give an efficient randomized algorithm almost attaining the lower bounds in all dimensions (up to $O(n^\delta)$ factors[1]). Their approach is again basically similar to "local" partition trees, but instead of a single partition scheme they use a *family* of partition schemes, such that for each hyperplane there is a "good" partition scheme in this family.

In our new solution, the query answering structure is simpler, and there is a reasonably simple randomized algorithm for the preprocessing. The query time and preprocessing time approach the lower bounds more tightly. Also, the partition scheme and partition trees may have applications for which other constructions do not suffice; in this paper we give an application to a surprisingly fast deterministic computation of ε-approximations and ε-cuttings.

Many applications of the partition scheme of Chazelle *et al.* [CSW] are given by Agarwal and Sharir [AS]. In these applications, replacing that scheme by our partition trees results in simpler and sometimes slightly more efficient algorithms. In general, efficient simplex range searching algorithms often give conceptually simple and efficient algorithms for problems where previously problem-specific and often complicated methods have been used.

In a related paper [M4], the method of this paper is adapted to the half-space range reporting problem, where it significantly improves known results for dimensions $d \geq 4$. These results in turn can be applied, e.g., for simple and efficient ray-shooting algorithms in higher dimensions and numerous other problems, see [AM2] and [AM1]. The author recently gave a further improvement of the simplex range searching results (based on a work by Chazelle [C2]), in particular attaining $O(n^{1-1/d})$ query time with linear space [M3].

[1] Throughout this paper, δ stands for an arbitrary but fixed positive constant, and the multiplicative factors implicit in the big-Oh notation may depend on it.

2. Preliminaries on Cuttings

In this section we give some definitions and results on cutting arrangements of hyperplanes which we need in the following.

A *cutting* is a collection of d-dimensional closed simplices[2] with disjoint interiors, whose union is the whole space E^d. The *size* of a cutting is the number of its simplices. Let H be a collection of n hyperplanes and Ξ be a cutting. For a simplex $\Delta \in \Xi$, let H_Δ denote the collection of hyperplanes of H intersecting the interior of Δ. A cutting Ξ is an *ε-cutting* for H provided that $|H_\Delta| \leq \varepsilon n$ for every simplex $\Delta \in \Xi$.

It is sometimes convenient to work with weighted collections of hyperplanes. A weighted collection of hyperplanes is a pair (H, w), where H is a collection of hyperplanes and $w: H \to \mathbb{R}^+$ is a weight function on H. If $X \subseteq H$ we write just $w(X)$ for $\sum_{h \in X} w(h)$. The notions introduced for unweighted collections of hyperplanes can usually be generalized for weighted collections in an obvious way. For example, a cutting Ξ is an *ε-cutting for* (H, w), provided that, for every simplex Δ of Ξ, the collection H_Δ has total weight $w(H_\Delta) \leq \varepsilon w(H)$.

The main known results about cuttings and their computability are summarized in the following theorem:

Theorem 2.1.

(i) [CF] *For any collection of n hyperplanes and a parameter $r \leq n$, there exists a $(1/r)$-cutting of size $O(r^d)$ (which is asymptotically the best possible size).*

(ii) *A cutting as in (i) can be computed in $O(nr^{d-1})$ time by a randomized algorithm* [CF] *or by a deterministic algorithm* [C2].

(iii) [M2] *Any algorithm computing $(1/r)$-cuttings for unweighted collections of hyperplanes can be converted into one computing $(1/r)$-cuttings for weighted collections, with the same asymptotic bounds on the running time and on the size of the resulting cutting.*

Let us remark that in (ii) we can also compute the collections H_Δ for all simplices Δ within the same time bound.

The preprocessing for our algorithm given in the following uses the computation of cuttings as in (ii). In order to give the reader a more complete picture of how complicated the preprocessing algorithm really is, we describe a randomized algorithm for computing a $(1/r)$-cutting of asymptotically optimal size:

1. Pick a random sample of r hyperplanes of H, construct its arrangement, and triangulate it in a specific way (so-called canonical or bottom-vertex triangulation, see, e.g., [CF]).
2. For every simplex Δ of this triangulation, compute the collection H_Δ of hyperplanes intersecting it, and set $t_\Delta = |H_\Delta| r/n$.
3. Pick a sample R_Δ of $Ct_\Delta \log t_\Delta$ hyperplanes of the collection H_Δ (C a suitable

[2] By a simplex, we mean an intersection of $d + 1$ half-spaces, thus we admit also unbounded simplices.

constant), and let R'_Δ be R_Δ plus the $d+1$ hyperplanes bounding the simplex Δ.

4. Triangulate the portion of the arrangement of R'_Δ inside Δ. Form a cutting Ξ as the set of all simplices thus obtained for all Δ.
5. Check if Ξ is a $(1/r)$-cutting for H and if the size of Ξ is at most $C'r^d$, for a suitable constant C'. These conditions can be verified in $O(nr^{d-1})$ time, and if they are not satisfied, the algorithm is restarted.

This algorithm (properly implemented) has expected running time $O(nr^{d-1})$. The proof is given in [CF] (and rephrased in [M2]).

3. Partitioning Point Sets

In this section we describe an efficient partition scheme for point sets.

Let P be an n-point set in E^d. A *simplicial partition* for P is a collection

$$\Pi = \{(P_1, \Delta_1), \ldots, (P_m, \Delta_m)\},$$

where the P_i's are pairwise disjoint subsets (called the *classes* of Π) forming a partition of P, and each Δ_i is a relatively open simplex containing the set P_i.

Let us remark at this point that the simplex Δ_i may also contain other points of P than those of P_i. Also, we admit not only full-dimensional simplices but also simplices of any dimension $k \leq d$. This is sometimes necessary for point sets P in degenerate positions; for a general position, we may take only full-dimensional simplices in the simplicial partitions in the forthcoming Partition Theorem.

The simplicial partitions we work with satisfy the following condition:

$$\max\{|P_i|; (P_i, s_i) \in \Pi\} < 2 \min\{|P_i|; (P_i, s_i) \in \Pi\}, \tag{1}$$

so all the classes have roughly the same size.[3]

If h is a hyperplane and Δ is a simplex, we say that h *crosses* Δ if $h \cap \Delta \neq \emptyset$ and $\Delta \not\subseteq h$ (thus a hyperplane does not cross a lower-dimensional simplex contained in it). Let h be a hyperplane; we define the *crossing number* of h (relative to Π) as the number of simplices among the Δ_i's crossed by h. The crossing number of Π is defined as the maximum of crossing numbers over all hyperplanes h. For the construction of efficient partition trees, we are interested in simplicial partitions with as small crossing number as possible.

Our main result is the following:

Theorem 3.1 (Partition Theorem). *Let P be an n-point set in E^d $(d \geq 2)$, let s be an integer parameter, $2 \leq s < n$, and set $r = n/s$. There exists a simplicial partition* Π

[3] This is a strengthening of the notion of *fine* simplicial partition defined in the preliminary version of this paper, where only the upper bound on the cardinality of the classes was imposed. The lower bound comes for free from the proof, and it can be useful in some applications of our results.

for P, whose classes P_i satisfy $s \le |P_i| < 2s$, and whose crossing number is $O(r^{1-1/d})$. This bound is asymptotically tight in the worst case.

A special case of this result, with $s = 2$, was proved by Chazelle and Welzl [CW], by improving a method developed by Welzl [We]. Our proof uses Welzl's method; the main new ingredient is the application of cuttings, which enables us to handle arbitrary values of s.

The proof is based on the following lemma:

Lemma 3.2. *Let P, n, s be as above, and let Q be a set of hyperplanes. Then there exists a simplicial partition Π for P, whose classes P_i satisfy $s \le |P_i| < 2s$ for every i, and such that the crossing number of every hyperplane of Q relative to Π is $O(r^{1-1/d} + \log|Q|)$.*

Proof. We inductively construct the disjoint sets $P_1, P_2, \ldots \subseteq P$, and simplices $\Delta_1, \Delta_2, \ldots, P_i \subseteq \Delta_i$. Suppose that $P_1, \ldots, P_i$ have already been constructed, and set $P'_i = P \setminus (P_1 \cup \cdots \cup P_i)$, $n_i = |P'_i|$. If $|n_i| < 2s$, we set $P_{i+1} = P'_i$, $\Delta_{i+1} = E^d$, $m = i + 1$, and $\Pi = \{(P_1, \Delta_1), \ldots, (P_m, \Delta_m)\}$, which finishes the construction.

Let now $n_i \ge 2s$. For a hyperplane $h \in Q$, let $\kappa_i(h)$ denote the number of simplices among $\Delta_1, \ldots, \Delta_i$ crossed by h. We define a weighted collection (Q, w_i) by setting $w_i(h) = 2^{\kappa_i(h)}$ for every $h \in Q$.

Let us choose a parameter t_i as large as possible, but such that there exists a $(1/t_i)$-cutting Ξ_i for (Q, w_i), whose simplices have at most n_i/s faces of all dimensions in total. By Theorem 2.1, t_i can be chosen such that $t_i \ge c(n_i/s)^{1/d}$ for some positive constant c. By the pigeonhole principle, some of the relatively open faces of the simplices of the cutting Ξ_i contains at least s points of P'_i. Let Δ_{i+1} be some such relatively open face. Since $s \ge 2$, the dimension of Δ_i is at least 1. Among the at least s points of P'_i contained in Δ_{i+1}, let us choose an s-point subset (arbitrarily) and call it P_{i+1}. This finishes the description of the construction.

Let us establish the bound on the crossing numbers of the hyperplanes of Q relative to the simplicial partition Π. The bound is obtained by estimating the final total weight $w_m(Q)$ of the hyperplanes of Q in two different ways.

The weight $w_m(h)$ of a hyperplane $h \in Q$ with crossing number κ is equal to 2^κ. Therefore $\kappa \le \log_2 w_m(Q)$.

Let us consider how $w_{i+1}(Q)$ increases compared with $w_i(Q)$. Let Q_{i+1} denote the collection of the hyperplanes of Q crossing Δ_{i+1}. For the hyperplanes of Q_{i+1}, the weight increases twice, and for the others it remains unchanged. From this we get

$$w_{i+1}(Q) \le w_i(Q) - w_i(Q_{i+1}) + 2w_i(Q_{i+1}) = w_i(Q)\left(1 + \frac{w_i(Q_{i+1})}{w_i(Q)}\right).$$

We claim that $w_i(Q_{i+1}) \le w_i(Q)/t_i$. Indeed, this is clear from the definition of a $(1/t_i)$-cutting if Δ_i has dimension d. For a simplex of lower dimension we note that if a hyperplane crosses a face of a full-dimensional simplex, then it also intersects the interior of that simplex; this is easily seen by induction on dimension.

Hence

$$w_{i+1}(Q) \leq w_i(Q)\left(1 + \frac{1}{t_i}\right) \leq w_i(Q)\left(1 + \frac{s^{1/d}}{cn_i^{1/d}}\right)$$

and using $w_0(Q) = |Q|$, $n_i = n - is$, $r = n/s$, $m = \lfloor r \rfloor$, we get

$$w_m(Q) \leq |Q| \prod_{i=0}^{m-1} \left(1 + \frac{1}{c(r-i)^{1/d}}\right).$$

Taking logarithms and using the inequality $\ln(1 + x) \leq x$ we get

$$\begin{aligned}\log w_m(Q) &\leq \log|Q| + \frac{1}{c}\sum_{i=0}^{m-1} \frac{1}{(r-i)^{1/d}} \leq \log|Q| + \frac{1}{c}\sum_{i=0}^{m-1} \frac{1}{(m-i)^{1/d}} \\ &= \log|Q| + \frac{1}{c}\sum_{j=1}^{m} \frac{1}{j^{1/d}}.\end{aligned}$$

Bounding the last sum by integral, we finally obtain

$$\kappa \leq \log_2 w_m(Q) = O(\log|Q| + r^{1-1/d}).$$

This concludes the proof of Lemma 3.2. □

The second ingredient in the proof of the Partition Theorem is the following lemma, saying that when estimating the crossing number of our simplicial partitions, we may concentrate on a fairly small number of "test" hyperplanes instead of considering all possible hyperplanes:

Lemma 3.3 (Test Set Lemma). *For an n-point set $P \subseteq E^d$ and a parameter r, there exists a set Q of at most r hyperplanes, such that, for any simplicial partition $\Pi = \{(P_1, \Delta_1), \ldots, (P_m, \Delta_m)\}$ for P satisfying $|P_i| \geq s$ for every i, the following holds: if κ_0 is the maximum crossing numbers of hyperplanes of Q relative to Π, then the crossing number of Π is bounded by*

$$(d + 1)\kappa_0 + O\left(\frac{n}{sr^{1/d}}\right).$$

Proof. Let $H = \mathscr{D}(P)$ be the collection of hyperplanes dual to the points of P (see, e.g., [E] for the definition and relevant properties of the duality transform). According to Theorem 2.1(i), we can choose a $(1/t)$-cutting Ξ for H whose simplices have at most r vertices in total, where $t = \Omega(r^{1/d})$. Let V be the set of all vertices of the simplices of Ξ, and put $Q = \mathscr{D}(V)$. We claim that Q has the desired property.

Let h be any hyperplane and let G be the set of vertices of a simplex Δ of Ξ containing the dual point $\mathscr{D}(h)$. By the assumption, each of the $d + 1$ hyperplanes dual to the points of G crosses at most κ_0 simplices of the simplicial partition Π, and it remains to bound the number of simplices of Π which are crossed by the hyperplane h but by no hyperplane of $\mathscr{D}(G)$. Such simplices Δ_i must be completely contained in the zone of h in the arrangement of $\mathscr{D}(G)$, and hence this zone must also contain the points of their corresponding classes P_i in its interior. It is elementary to verify, using the properties of the duality transform, that any point of P lying in the interior of the zone of h in the arrangement of $\mathscr{D}(G)$ dualizes to a hyperplane of H intersecting the interior of the simplex Δ (see also [CSW]), and there are at most $O(n/r^{1/d})$ such hyperplanes in H. Hence the zone of h may contain at most this many points of P, and this shows that there are at most $O(n/sr^{1/d})$ simplices of Π completely contained in the zone of h in the arrangement of G. □

Proof of Theorem 3.1. In order to obtain the desired simplicial partition, we first use Lemma 3.3 with P, s, and r as in the statement of the Partition Theorem. We get a set Q of at most r hyperplanes and we use Lemma 3.2, obtaining a simplicial partition Π, whose classes have size between s and $2s$, and such that the crossing number of any hyperplane of Q is at most $O(\log|Q| + r^{1-1/d}) = O(r^{1-1/d})$. Then, by the property of Q guaranteed by the Test Set Lemma, the crossing number of Π is at most $(d+1)O(r^{1-1/d}) + O(n/sr^{1/d}) = O(r^{1-1/d})$.

Finally we show that the bound on the crossing number is asymptotically tight in the worst case, extending an argument due to Welzl. Indeed, suppose that we want a simplicial partition for an n-point set, with classes of size between s and $2s$. Choose a set Q of hyperplanes in general position, such that its arrangement has at least $\lceil n/(s-1)\rceil$ distinct cells; it suffices to take $O(r^{1/d})$ hyperplanes, $r = n/s$. Divide the n points into clusters by at most $s-1$ points each, and place the clusters into distinct cells of the arrangement of Q. In any simplicial partition with classes of size at least s, every simplex crosses at least one hyperplane of Q. Hence, on the average, a hyperplane of Q crosses $\Omega(r^{1-1/d})$ simplices of the simplicial partition. □

We now start considering an algorithmic form of the Partition Theorem.

Lemma 3.4. *For a value of r bounded by a constant, a simplicial partition as in Theorem 3.1 can be constructed in time $O(n)$.*

Proof. It suffices to go through the steps of the preceding proof of Theorem 3.1 and verify that they can be executed in the claimed time bound. For the Test Set Lemma we need to compute a $(1/t)$-cutting for a collection of n hyperplanes, with $t = O(1)$, which can be done in $O(n)$ time by Theorem 2.1(ii). In the proof of Lemma 3.2, the collection Q has a constant number r of hyperplanes, we make $r = O(1)$ steps and we always deal with a constant number of simplices only, so the only thing which might cause problems would be the arithmetic with the potentially large powers of two standing for the weights. In the algorithm the weights are only used for calculation of the cuttings. The calculation of a cutting for a weighted collection (H, w) in Theorem 2.1(iii) starts by replacing (H, w) by a multiset containing $m(h) = \lfloor |H| w(h)/w(H) \rfloor + 1$ copies of each hyperplane $h \in H$, but the procedure still works (with worse constants) if we use any number between $m(h)$ and $2m(h)$ of copies of h. Hence it suffices to know the value of $w(h)/w(H)$ with relative accuracy $\frac{1}{2}$, so we can use ordinary integer arithmetic for these calculations.

For later reference, let us also note that if r is not bounded by a constant, the above proof gives an algorithm polynomial in n and r for computing the simplicial partition. □

This result has the following consequence:

Corollary 3.5. *Let $\delta > 0$ be a positive constant. Given an n-point set $P \subseteq E^d$ and an integer parameter s, $2 \leq s < n$, a simplicial partition for P satisfying $s \leq |P_i| < 2s$ for every class P_i and with crossing number $O(r^{1-1/d+\delta})$ can be constructed in time $O(n \log r)$.*

Proof. A simple observation is that we can "compose" the constructions of a simplicial partition. Namely, we may first construct a simplicial partition Π with class sizes between s_1 and $2s_1$ and with crossing number at most $Cr_1^{1-1/d}$, where $r_1 = n/s_1$ and C is some absolute constant which can be estimated by inspection of the proof of the Partition Theorem. Then for every class P_i of this simplicial partition we construct a simplicial partition Π_i with class sizes between s_2 and $2s_2$ and with crossing number at most $Cr_2^{1-1/d}$, $r_2 = 2s_1/s_2$. Apparently the union of all these secondary simplicial partitions is a simplicial partition for P with class sizes between s_2 and $2s_2$ and with crossing number not exceeding $2C^2r^{1-1/d}$, where $r = n/s_2$. This refinement can be iterated more times, and we lose a constant factor at every iteration compared with a direct construction.

In our case we choose a sufficiently large constant r_0 and we iterate the construction with parameter $s_1 = \lfloor n/r_0 \rfloor$, $s_2 = \lfloor s_1/r_0 \rfloor$, etc. The number of iterations needed to achieve class size at most n/r is about $D = \log r/\log r_0$, thus the multiplicative factor we lose in the crossing number is of order $(2C)^D = r^{\log 2C/\log r_0}$. By choosing the constant r_0 large enough, we can make this factor smaller than r^δ for arbitrary fixed $\delta > 0$. □

Our further goal is to prove a stronger algorithmic version of the Partition Theorem, with a better bound on the crossing number than in the preceding corollary. To this end we consider a fast computation of $(1/r)$-cuttings, and we obtain results of independent interest.

4. Approximations, Cuttings, and an Efficient Partitioning Algorithm

The starting observation is a relation between simplicial partitions and ε-approximations with respect to simplices.

Let P be a finite point set in E^d, and let $A \subseteq P$. We say that A is an *ε-approximation* for P (with respect to simplices), provided that for every simplex

σ it is

$$\left| \frac{|A \cap \sigma|}{|A|} - \frac{|P \cap \sigma|}{|P|} \right| < \varepsilon.$$

The notion of ε-approximation originated in a paper by Vapnik and Chervonenkis [VC] in a more general setting; the name ε-approximation comes from the paper by Haussler and Welzl [HW]. From a theorem in [VC], we get that, for each P and r, there exists a $(1/r)$-approximation for P of size at most $O(r^2 \log r)$. This size can be further improved to $O(r^{2-2/(d+1)} \log^2 r)$, see [MWW]. As shown in [M1], a $(1/r)$-approximation of size $O(r^2 \log r)$ can be computed in time $O(n(r^2 \log r)^{d(d+1)})$.

The notion of ε-approximation can be immediately generalized to the case when A is equipped by a weight function w: a weighted collection (A, w) is an ε-approximation for P, provided that $|w(A \cap \sigma)/w(A) - |P \cap \sigma|/|P|| < \varepsilon$ for every simplex σ.

We have the following observation:

Observation 4.1. *Let P be an n-point set in E^d and let $\Pi = \{(P_1, \Delta_1), \ldots, (P_m, \Delta_m)\}$ be a simplicial partition for P, with class sizes not exceeding s and with crossing number κ. For each i, let a_i be one point of P_i and put $w(a_i) = |P_i|$. Then $(A = \{a_1, \ldots, a_m\}, w)$ is an ε-approximation for P (with respect to simplices), where $\varepsilon = (d+1)\kappa s/n$.*

Proof. For a simplex σ, the difference in the number of points of P inside σ and the total weight of points of A inside σ is caused only by the classes of the simplicial partition whose corresponding simplices are crossed by the boundary of σ. The number of points in such simplices is at most $(d+1)\kappa s$, and the observation follows. □

We may now apply Corollary 3.5 together with the preceding observation, obtaining the following:

Theorem 4.2. *Given an n-point set $P \subseteq E^d$ and a parameter t, a $(1/t)$-approximation of size $O(t^{d+\delta})$ for P (with respect to simplices) can be computed in time $O(n \log t)$.*

The approximation size attained in this theorem is far from optimal, and no substantial improvement is possible with the present method (via partition trees); the forthcoming strengthening of Corollary 3.5 can only remove the t^δ factor in the size of the $(1/t)$-approximation (see below). If the required value of t (measuring the accuracy of the approximation) is not too large, we can use this as a first reduction in the problem size, and then use another algorithm (with a larger running time) to improve further the size of the $(1/t)$-approximation.

For the result about cuttings we have to pass to the dual setting. If H is a collection of hyperplanes and $A \subseteq H$, we call A an ε-approximation for H (with respect to segments), provided that for every segment e we have

$$\left| \frac{|A_e|}{|A|} - \frac{|H_e|}{|H|} \right| < \varepsilon,$$

where H_e (resp. A_e) denotes the set of hyperplanes of H (resp. of A) crossing the segment e. It is elementary to verify that if P is a point set and A is its ε-approximation with respect to simplices, then $\mathscr{D}(A)$ is an ε-approximation for $\mathscr{D}(P)$ with respect to segments. The use of approximations for an efficient computation of cuttings stems from the following easy lemma (we again use the weighted form of approximations):

Lemma 4.3 [M2]. *Let (A, w) be an ε-approximation for H and let Ξ be an ε'-cutting for (A, w). Then Ξ is a $d(\varepsilon + \varepsilon')$-cutting for H.*

Hence, in a computation of a $(1/r)$-cutting for a collection H of n hyperplanes, we can proceed as follows: We first compute a $(1/2dr)$-approximation of size $O(r^{d+\delta})$ for H, in time $O(n \log r)$ according to Theorem 4.2. Then we apply Theorem 2.1 to compute a $(1/2dr)$-cutting for this approximation, in time $O(r^{2d+\delta-1})$. This yields a $(1/r)$-cutting for the original collection H. We summarize this in a proposition:

Proposition 4.4. *Let $r \leq n^{1/d}$. Then a $(1/r)$-cutting of size $O(r^d)$ for a collection of n hyperplanes in E^d can be computed in time $O(n \log r + r^{2d+\delta-1})$. In particular, for $r \leq n^{1/(2d+\delta-1)}$, the running time is $O(n \log r)$.*

Now we use these improved results on cutting computations to strengthen our results on the computation of simplicial partitions. We begin with a technical lemma on range counting.

Lemma 4.5. *Let C be a prescribed constant, let P be an n-point set in E^d, and let $r \leq n^\alpha$ be a parameter, where $\alpha = \alpha(C) > 0$ is a constant. Then we can build in $O(n \log r)$ time a data structure, which computes the number of points of P in a given query simplex in $O(n/r^C)$ time, and which can be maintained under deletions of points from P, in $O(1)$ time per delete operation (the value of n in the query time remains the original cardinality of P, even after deletions).*

Proof. There are several possible ways to proceed, we use one applying simplicial partitions. Let t be a sufficiently large power of r. In the preprocessing phase we compute in $O(n \log r)$ time a simplicial partition Π for P, with at most t classes and with crossing number $O(t^{1-1/d+\delta})$. For every $(P_i, \Delta_i) \in \Pi$, we store the simplex Δ_i, the list of points of P_i, and their number. Given a query simplex σ, we first find the simplices of Π crossed by some of the hyperplanes bounding σ, and we directly test the points of the corresponding classes for membership in σ. We then add the point counts for all simplices of Π completely contained in σ, obtaining the number of points of $P \cap \sigma$. The query time will be

$$O(t + (n/t)t^{1-1/d+\delta}) = O(t + n/t^{1/d-\delta}).$$

If t is a large enough power of r and $\alpha(C)$ is small enough, this does not exceed $O(n/r^C)$. Finally the deletion of a point is performed by marking the point as deleted in the appropriate list, and updating the appropriate point count. □

Lemma 4.6. *Let P, n, s, r be as in Theorem 3.1 and let $r \leq n^\alpha$, where $\alpha > 0$ is a certain constant. A simplicial partition as in the Partition Theorem can be constructed in $O(n \log r)$ time.*

Proof. We again go through the proof of the Partition Theorem. First we have to compute a $(1/t)$-cutting as in the Test Set Lemma (with $t = \Omega(r^{1/d})$) in time $O(n \log r)$, which we can do by Proposition 4.4.

The remaining steps can mostly be performed in time depending polynomially on r and not depending on n. The only exception is when we select a face of a simplex of the cutting Ξ_i containing enough points of the set P_i'. To this end we need to count the points in these faces (and then report the points inside the selected face). This requires $O(r^2)$ simplex range counting (and fewer reporting) queries on an n-point set which, moreover, dynamically evolves—the points are deleted from it. By Lemma 4.5, these queries can all be performed in $O(n \log r)$ time including the preprocessing, if $\alpha > 0$ is chosen small enough. □

Theorem 4.7. *Let P, n, s, r be as in Theorem 3.1.*

(i) *For every fixed $\delta > 0$, a simplicial partition for P whose classes P_i satisfy $s \leq |P_i| < 2s$ and whose crossing number is $O(r^{1-1/d})$ can be constructed in time $O(n \log r)$, provided that $s \geq n^\delta$ (with the constant in the bound on the crossing number dependent on δ).*

(ii) *For any s, a simplicial partition for P whose classes P_i satisfy $s \leq |P_i| < 2s$ and whose crossing number is $O(r^{1-1/d}(\log r)^{O(1)})$ can be constructed in time $O(n \log r)$.*

(iii) *For any s, a simplicial partition for P whose classes P_i satisfy $s \leq |P_i| < 2s$ and whose crossing number is $O(r^{1-1/d})$ can be constructed in time $O(n^{1+\delta})$ (again with the constant in the bound on the crossing number dependent on δ).*

Proof. We again iterate the partition construction similarly as in the proof of Corollary 3.5, but using Lemma 4.6 this time. When a current point set has size m, we set the parameter s to $\lfloor m^{1-\alpha} \rfloor$ for the next iteration. Thus after the ith iteration, the size of the classes of the current partition are about $n^{(1-\alpha)^i}$. In order to get claim (i), it suffices to iterate this partition construction a constant number of times (until $(1-\alpha)^i$ drops below δ). We thus lose a constant factor in the crossing number only. For claim (ii) we iterate the construction at most $O(\log \log n)$ times, losing a $2^{O(\log \log n)} = \log^{O(1)} n$ factor in the crossing number altogether. For claim (iii) we iterate the construction a constant number of times, until we reach classes of size $m = O(n^{\delta'})$ for a small enough δ'. For each of these small classes we still need to compute a simplicial partition with class sizes between s and $2s$, where s may still be very small compared with m. We now use the polynomial-time procedure for computing a simplicial partition without restrictions on the parameter value (implied by the proof of the Partition Theorem). Since the classes for which this procedure is applied are small enough, the total running time is of order $O(n^{1+\delta})$. □

The preceding theorem allows us to improve the previously mentioned computation of ε-approximations and ε-cuttings a little more. In Theorem 4.2 we can compute a $(1/t)$-approximation of size $O(t^d)$ in $O(n \log t)$ time, provided that $t^d \leq n^{1-\delta}$ for a fixed $\delta > 0$. In Proposition 4.4 we then get an improved running time $O(n \log r + r^{2d-1})$ for $r \leq n^{1-\delta}$. This improves Theorem 2.1 if $r^d \leq n$. However, the result can be combined with Theorem 2.1, and we can improve the running time for cutting computation also for larger values of r, as follows: We choose a suitable value of a parameter r_1, we compute a $(1/r_1)$-cutting for H as in Theorem 2.1(ii) together with the collection H_Δ for every simplex of this cutting. Then we compute a (r_1/r)-cutting for each $H_\Delta \cup B_\Delta$, where B_Δ are the hyperplanes bounding Δ. Similarly as in the outline of a randomized cutting algorithm following Theorem 2.1, this yields a $(1/r)$-cutting for H. In order to balance the running times appropriately, we put

$$r_1 = \left\lceil \frac{r^{1+1/2(d-1)}}{n^{1/2(d-1)}} \right\rceil,$$

and by straightforward analysis we obtain

Theorem 4.8. *Let $r \leq n^{1-\delta}$ for some fixed $\delta > 0$. Then a $(1/r)$-cutting of (asymptotically optimal) size $O(r^d)$ for a collection of n hyperplanes in E^d can be computed in time $O(n \log r + \sqrt{n} r^{d-1/2})$.*

Interestingly, this computation is significantly faster than if we only wanted to *verify* that some collection of triangles (in the plane) is a $(1/r)$-cutting for given collection of n lines, by counting the number of intersections of every side of the triangles with the lines, using the fastest-known algorithms (see [A]). For instance, for $r = n^{1/3}$, computing the number of intersections with a given collection of n lines for $O(r^2) = O(n^{2/3})$ segments by Agarwal's algorithm requires about $n^{10/9}$ time, while our results yield a $(1/r)$ cutting in almost linear time in this case.

5. Range Searching

Using Theorem 4.7, it is now straightforward to establish the following:

Theorem 5.1. *Given a set of n points in E^d, we can preprocess it for simplex range searching (with semigroup weights) in time $O(n \log n)$, store the results in space $O(n)$, and then answer queries in time $O(n^{1-1/d}(\log n)^{O(1)})$.*

Proof. We use the simplicial partition from Theorem 4.7 recursively, building a partition tree. The leaves of the tree form a partition of P into constant-sized subsets. Each inner node v of this tree corresponds to a subset P_v of P and to a simplicial partition Π_v of P_v. Of that simplicial partition, we store only the simplices and the cumulative weights of the corresponding subsets in the node v; the classes themselves are only represented implicitly, each corresponding to a child of v. The simplicial partitions Π_v are constructed as in Theorem 4.7(i) with $s = \lfloor |P_v|^{1/d} \rfloor$. Hence the size of the sets corresponding to the nodes at level i of the partition tree are of order n^{1/d^i}, so unlike previous partition trees, ours has depth $O(\log\log n)$.

In the query answering process we proceed as follows: Given a query simplex σ, we start in the root. Being in a node v, we take the simplices of the simplicial partition Π_v one by one, and we handle directly those contained in σ or disjoint from σ, and we proceed into the corresponding children nodes for the other simplices.[4] Each of the latter ones must be crossed by at least one of the hyperplanes bounding the query simplex, so we recurse in $O(r^{1-1/d})$ simplices only ($r = n/s$).

For the query time $T(n)$, we get a recurrence

$$T(n) \le O(r) + O(r^{1-1/d})T\left(\frac{2n}{r}\right), \qquad r = n^{1-1/d},$$

with initial condition $T(n) = O(1)$ for n smaller than some constant. The solution is easily verified to be of the claimed form $T(n) = O(n^{1-1/d}(\log n)^{O(1)})$.

Since an m-point node of the partition tree uses storage $O(m^{1-1/d})$, the total space occupied by the tree is linear (we group the nodes of the partition tree according to their depth, we bound the size for every level and we find that the size of leaves forms a constant fraction of the total size). The preprocessing time for constructing the simplicial partition in a node with m points is $O(m \log m)$ by Theorem 4.7(i), and since the number of points in nodes decreases as a double exponential with their depth in the tree, the total preprocessing time is $O(n \log n)$. □

Let us remark that an analogue of Theorem 5.1 for range reporting can be proved along similar lines (with query time $O(n^{1-1/d}(\log n)^{O(1)} + k)$, where k is the number of reported points).

Chazelle *et al.* [CSW] describe a "fast" simplex range searching data structure with query time $O((\log n)^{d+1})$ and storage and preprocessing time $O(n^{d+\delta})$. This data structure can be combined with the partition trees to get a space/query time tradeoff. We may stop the construction of our partition tree at an appropriate level (on a level where the classes corresponding to nodes have some prescribed size). Instead of the subtrees of the partition tree below that level, we store the fast data structures for the corresponding point sets at these nodes; this is quite similar to a construction in [CSW] and we omit the details. We also note that

[4] Note that we can handle both open and closed query simplices.

the randomized construction of the "fast" structure can immediately be made deterministic using Theorem 2.1.

Corollary 5.2. *Given a set P of n points in E^d and a parameter m, $n \leq m \leq n^d$, we can preprocess P for simplex range searching, deterministically in time $O(m^{1+\delta})$, store the results in space of the same order, and then answer queries in time*

$$O\left(\frac{n}{m^{1/d}} \log^{d+1} n\right).$$

6. Reducing the Query Time

In this section we show that we can further improve the query time for simplex range searching with linear space. However, this improvement complicates the algorithm and so far we also have to sacrifice something in the preprocessing time:

Theorem 6.1. *Given a set of n points in E^d, we can preprocess it for simplex range searching in time $O(n^{1+\delta})$, store the results in space $O(n)$, and then answer queries in time*

$$O(n^{1-1/d}(\log \log n)^{O(1)}).$$

For point weights which can be subtracted (i.e., belonging to a group, as for instance for range counting), query time

$$O(n^{1-1/d}2^{O(\log^* n)})$$

can be achieved. Also for reporting the points in the query simplex we can get query time $O(n^{1-1/d}2^{O(\log^* n)} + k)$, where k is the number of reported points.

Proof. Let us look why the $(\log n)^{O(1)}$ factor in the query time appears in Theorem 5.1: it is because in each level of the partition tree we lose a constant factor in the "efficiency" (crossing number) of the partition. From this point of view the best thing to do would be to have a single-level partition tree, thus returning back to the construction of Welzl [We], [CW]. However, another difficulty arises here: we know that only a small number of simplices in the simplicial partition crosses the boundary of a query simplex, but we have to detect them, more precisely to solve the following two problems:

A. Which simplices of the simplicial partition are crossed by the boundary of the query simplex?
B. What is the cumulative weight of points associated with the simplices of the simplicial partition which are completely contained in the query simplex?

In the proof of Theorem 5.1 the size of the simplicial partition was not bigger than the query time aimed at, so these problems could have been solved trivially—by inspecting all the simplices. This enforces $\Omega(\log \log n)$ levels of the partition tree. For a smaller number of levels, the value of r in each node has to be closer to the number of points in that node, and we need some secondary data structure for Problems A and B.

Let us begin with Problem B. We note that a simplex is contained in another simplex iff all its vertices are. We consider a more general situation, and we show the following:

Lemma 6.2. *Let $k \geq 1$ be a fixed integer, let A be a set of n ordered k-tuples of points in E^d and suppose that every k-tuple is equipped by a semigroup weight. We can compute (in time and space $O(n(\log n)^{C_k})$) a data structure $S_k(A)$ which allows us to compute the total weight of all k-tuples of A completely contained in a query simplex in time $O(n^{1-1/d}\exp(C_k'\sqrt{\log n}))$ (C_k, C_k' constants).*

Proof. We proceed by induction on k. For $k = 1$, the problem is solved by Theorem 5.1, so let $k > 1$. For simplicity of description, let us suppose that all points occurring as the components of the k-tuples of A are distinct. Let F be the set of the first components the k-tuples of A.

In order to construct the data structure $S_k(A)$ we build a partition tree as in Theorem 5.1 on the set F, this time setting $r = m/s = \exp(\sqrt{\log m})$ in an m-point node. Let v be a node storing a simplicial partition $\Pi_v = \{(P_1, s_1), \ldots, (P_t, s_t)\}$. For every class $P_i \subseteq F$, let $A_i \subseteq A$ be the set of k-tuples whose first components belong to P_i, and let A_i' be the set of $(k-1)$-tuples arising by removing the first component from the k-tuples of A_i. We compute the data structure $S_{k-1}(A_i')$ and we store it in the node v.

Given a query simplex σ, we proceed as follows: We start in the root of the partition tree. In current (nonleaf) node v we find the simplices of the simplicial partition Π_v completely contained in σ, and, for each such simplex Δ_i, we use the data structure $S_{k-1}(A_i')$ to find the number of k-tuples of A_i contained in σ. Also, we find the simplices of Π_v crossed by the boundary of σ, and we use the appropriate subtrees of v recursively to find the number of k-tuples inside σ belonging to the subsets corresponding to such simplices.

For the query time $T_k(n)$ we get the recurrence

$$T_k(n) \leq O(r) + O(r^{1-1/d})T_k\left(\frac{2n}{r}\right) + O(r)T_{k-1}\left(\frac{2n}{r}\right)$$

with $r = \exp(\sqrt{\log n})$, and we also have $T_{k-1}(n) = O(n^{1-1/d}\exp(C_{k-1}'\sqrt{\log n}))$ by inductive hypothesis. By a straightforward calculation we verify the estimate for $T_k(n)$ from this recurrence. We leave the analysis of space requirements and preprocessing time to the reader. Let us also remark that similar "multilevel" constructions have been used in [CSW] and [AS]. These, however, use a constant value of r in the partition trees, so our approach brings some improvement. □

As for Problem A, we observe that a simplex is crossed by the boundary of another simplex iff some of its edges is. In fact, the simplicial partition guarantees not only a small number of simplices crossing the boundary of the query simplex, but even a small number of simplices crossing any of the hyperplanes bounding the query simplex. Hence it is enough to have a data structure for reporting all segments crossing a query hyperplane h.

We can achieve the following solution:

Lemma 6.3. *For a set of n segments in E^d, there exists a data structure for reporting the segments crossing a query hyperplane, with $O(n(\log n)^{O(1)})$ space and preprocessing time and $O(n^{1-1/d}(\log n)^{O(1)} + k)$ query time, where k is the number of reported segments.*

Proof. Similarly as in Lemma 6.2 we build a partition tree on the set of left endpoints of the segments. The difference to Lemma 6.2 is that, for the secondary data structures on the right endpoints, we require only half-space range reporting. For this problem, very efficient solutions are known in dimensions two and three; the algorithm of Chazelle *et al.* [CGL] in the plane and of Chazelle and Preparata [CP] in 3-space. The query time for these algorithms is $O(\log n + k)$ for an n-point set, where k is the number of points reported. The algorithm in [CGL] uses $O(n)$ space and $O(n \log n)$ preprocessing, the algorithm in [CP] is worse by some logarithmic factors (an improved algorithm for this problem is given by Aggarwal *et al.* [AHT]), which is insignificant for us. An efficient solution has recently been obtained also for higher dimensions (see [M4]): it is possible to solve the half-space range reporting problem with $O(n \log\log n)$ space, $O(n \log n)$ preprocessing time, and $O(n^{1-1/\lfloor d/2 \rfloor}(\log n)^{O(1)} + k)$ query time.

Let us build a partition tree on the set of left endpoints of our segments, setting the parameter s to $\lfloor m^{2/3} \rfloor$ (say) in an m-point node. For each class P_i of the simplicial partition stored in a node v, we let R_i be the set of right endpoints of the segments with the left endpoints in P_i, and we store a half-space range reporting structure for R_i. The total space and preprocessing time for the whole partition tree including the secondary data structures remains close to linear. In an m-point node v we are answering at most $r = m^{1/3}$ half-space range reporting queries on at most $2s$ points each. The additive terms corresponding to the number of reported segments in the complexity of these queries at v sum up to $O(r^{1-1/d})$ (this follows from the bound on the crossing number of the simplicial partition). The sum of the overhead terms for the half-space range reporting queries at v is

$$O(m^{1/3}(m^{2/3})^{1-1/\lfloor d/2 \rfloor}(\log n)^{O(1)}) = O(m^{1-1/d}),$$

and since the depth of the partition tree is $O(\log\log n)$, we get the claimed bound for the total query time. □

In the general case the solution to Problem B is worse than the solution to Problem A. However, if we can subtract the weights of the points, it turns out that Problem B can be solved faster, using the solution of Problem A: at a node v, we pick one representing point in every class P_i of the simplicial partition Π_v, we let its weight be the total weight of the points of P_i and we build the simplex range searching structure as in Theorem 5.1 for these representing points.

Problem B is then solved as follows: We first use the auxiliary data structure to compute the total weight w of the representing points lying inside the query simplex σ. From the solution to Problem A we know the simplices crossing the boundary of σ, and, for each of them, we check whether its representing point lies inside σ. If it does, we have to subtract its weight from w. With this adjustment we obtain the answer to Problem B.

Similarly we can handle Problem B in the case of range reporting.

Now we are ready to prove Theorem 6.1. We equip every node of our basic partition tree (except for the leaves) by auxiliary data structures solving Problems A and B. For an m point node, we choose the size of the classes of the simplicial partition in such a way that the time required for solving Problems A and B is of order $O(m^{1-1/d})$, and the space and preprocessing time for the secondary structures is of order $O(m/\log m)$. Then the total space is easily seen to be linear, and the preprocessing time is $O(n^{1+\delta})$ using Theorem 4.7(iii).

Generalizing the analysis in the proof of Theorem 5.1, we get that if the work on a query in an m-point node is $O(m^{1-1/d})$, then the query time depends on the depth D of the partition tree—it will be $O(n^{1-1/d}2^{O(D)})$. The depth is in turn determined by the smallest value of the parameter s we can afford for an m-point node of the partition tree.

For the general case (subtractions not allowed), our solution to Problems A and B allows us to put $s = 2^{C\sqrt{\log m}}$, where C is a sufficiently large constant. Hence when passing one level down the tree, the logarithm of the node size is reduced to a constant multiple of its square root, which gives the depth $D = O(\log\log\log n)$.

For the case of weights which can be subtracted or for the case of range reporting, the improved solution to Problem B allows us to set $s = (\log m)^{O(1)}$, and thus the passage one level down in the partition tree reduces the node size from m to $(\log m)^{O(1)}$. It is not difficult to calculate that $D = O(\log^* n)$ in this case. This finishes the proof of Theorem 6.1. $\square$

7. Remarks on Dynamization

The data structures described in the previous section can be easily extended to accommodate insertions and deletions of points using a standard approach developed by Bentley [B] and Overmars and van Leeuwen (see [O]). We include this for completeness, since a dynamic version of the simplex range searching algorithm is useful in various geometric applications.

Theorem 7.1. *A simplex range searching data structure with the same asymptotic performance as in Theorem 5.1 can be maintained under insertions and deletions of points, with $O(\log n)$ amortized time per deletion and $O(\log^2 n)$ amortized time per insertion.*

Proof. First we note that it is easy to update the weight of a point in the partition tree from the proof of Theorem 5.1. It is enough to change the sums of weights for the appropriate simplices. For a single point, there are $O(\log \log n)$ simplices whose corresponding subsets contain that point.

If the weights can be subtracted, then such an update is completely straightforward. However, also without subtraction we can recompute the total weight for a node in logarithmic time. To this end we arrange the weights of the children nodes of the considered node as leaves of a balanced binary tree. The inner nodes of this binary tree then contain the appropriate partial sums. If the weight at one leaf of the binary tree is changed, it is enough to recompute weights along the path from this leaf to the root of the binary tree. This amounts to $O(\log r) = O(\log m)$ operations in an m-point node of the partition tree. Summing this update time along a path from a leaf to the root in the partition tree, we get $O(\log n)$ in total (recall that the node size decreases doubly exponentially with depth in the partition tree).

By assigning null weights to points, we can simulate deletions. After $n/2$ deletions we always rebuild the structure from scratch, so that we maintain an appropriate query time when the point set shrinks significantly.

It remains to handle insertions. We know that a static structure for n points can be built in time $O(n \log n)$. If P_1 and P_2 are disjoint point sets, then the answer to a simplex range query on $P_1 \cup P_2$ can be computed from the answers for P_1 and for P_2 in constant time, i.e., the problem is decomposable in the terminology of [O]. Thus, applying the method for dynamizing insertions, we arrive at the following result: $O(\log^2 n)$ amortized insertion time, and asymptotically the same query time as for the static structure. $\square$

Many more questions concerning dynamization could of course be raised. For instance, we may require the time bounds for insertion and deletion to be worst-case rather than amortized (to achieve results similar to the ones given by Schipper and Overmars [SO] for other types of partition trees), or consider more complicated variants of the data structure with some secondary data structures attached to the nodes, etc. It does not seem at present that such questions should present any substantial difficulties.

References

[A] P. K. Agarwal. Partitioning arrangements of lines, II: Applications. *Discrete & Computational Geometry*, 5:533–573, 1990.

[AHT] A. Aggarwal, M. Hansen, and T. Leighton. Solving query-retrieval problems by compacting Voronoi diagrams. In *Proc. 21st ACM Symposium on Theory of Computing*, pages 331–340, 1990.

[AM1] P. K. Agarwal and J. Matoušek. Dynamic half-space range reporting and its applications. Tech. Report CS-91-43, Duke University, 1991. Extended abstract: *Proc. 33rd IEEE Symposium on Foundations of Computer Science*, to appear, 1992.

[AM2] P. K. Agarwal and J. Matoušek. Ray shooting and parametric search. Tech. Report CS-1991-22, Duke University, 1991. Extended abstract: *Proc. 24th ACM Symposium on Theory of Computing*, 1992.

[AS] P. K. Agarwal and M. Sharir. Applications of a new space partitioning scheme. In *Proc. 2nd Workshop on Algorithms and Data Structures*, 1991.

[B] J. L. Bentley. Decomposable searching problems. *Information Processing Letters*, 8:244–251, 1979.

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

[CGL] B. Chazelle, L. Guibas, and D. T. Lee. The power of geometric duality. *BIT*, 25(1), 1985.

[C1] B. Chazelle. Lower bounds on the complexity of polytope range searching. *Journal of the American Mathematical Society*, 2(4):637–666, 1989.

[C2] B. Chazelle. Cutting hyperplanes for divide-and-conquer. Tech. Report CS-TR-335-91, Princeton University, 1991. Preliminary version: *Proc. 32nd IEEE Conference on Foundations of Computer Science*, October 1991. To appear in *Discrete & Computational Geometry*.

[CP] B. Chazelle and F. P. Preparata. Halfspace range searching: An algorithmic application of k-sets. *Discrete & Computational Geometry*, 1:83–93, 1986.

[CSW] B. Chazelle, M. Sharir, and E. Welzl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. In *Proc. 6th ACM Symposium on Computational Geometry*, pages 23–33, 1990.

[CW] B. Chazelle and E. Welzl. Quasi-optimal range searching in spaces of finite VC-dimension. *Discrete & Computational Geometry*, 4:467–490, 1989.

[E] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Sprnger-Verlag, New York, 1987

[EW] H. Edelsbrunner and E. Welzl. Halfplanar range search in linear space and $O(n^{0.695})$ query time. *Information Processing Letters*, 23(6):289–293, 1986.

[HW] D. Haussler and E. Welzl. ε-nets and simplex range queries. *Discrete & Computational Geometry*, 2:127–151, 1987.

[M1] J. Matoušek. Approximations and optimal geometric divide-and-conquer. In *Proc. 23rd ACM Symposium on Theory of Computing*, pages 506–511, 1991.

[M2] J. Matoušek. Cutting hyperplane arrangements. *Discrete & Computational Geometry*, 6(5): 385–406, 1991.

[M3] J. Matoušek. Range searching with efficient hierarchical cuttings. In *Proc. 8th ACM Symposium on Computational Geometry*, pages 276–285, 1992.

[M4] J. Matoušek. Reporting points in halfspace. In *Proc. 32nd IEEE Symposium on Foundations of Computer Science*, pages 207–215, 1991. Also to appear in *Computational Geometry: Theory and Applications*.

[MWW] J. Matoušek, E. Welzl, and L. Wernisch. Discrepancy and ε-approximations for bounded VC-dimension. *Combinatorica*, to appear. Also in *Proc. 32nd IEEE Symposium on Foundations of Computer Science*, pages 424–430, 1991.

[O] M. H. Overmars. *The design of Dynamic Data Structures*. Springer-Verlag, Berlin, 1983.

[SO] H. Schipper and M. H. Overmars. Dynamic partition trees. In *Proc. 2nd Scandavian Workshop on Algorithms Theory*, pages 404–417, 1990. Lecture Notes in Computer Science, Vol. 447. Springer-Verlag, Berlin, 1990. Also to appear in *BIT*.

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

[We] E. Welzl. Partition trees for triangle counting and other range searching problems. In *Proc. 4th ACM Symposium on Computational Geometry*, pages 23–33, 1988.

[Wi] D. E. Willard. Polygon retrieval. *SIAM Journal on Computing*, 11:149–165, 1982.

[YY] F. F. Yao and A. C. Yao. A general approach to geometric queries. In *Proc. 17th ACM Symposium on Theory of Computing*, pages 163–168, 1985.

Received April, 1991, and in revised form January 15, 1992.