Discrete Comput Geom 10:157–182 (1993)

Discrete & Computational
Geometry

Range Searching with Efficient Hierarchical Cuttings*

Jiří Matoušek

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

Abstract. We present an improved space/query-time tradeoff for the general simplex range searching problem, matching known lower bounds up to small polylogarithmic factors. In particular, we construct a linear-space simplex range searching data structure with $O(n^{1-1/d})$ query time, which is optimal for $d = 2$ and probably also for $d > 2$. Further, we show that multilevel range searching data structures can be built with only a polylogarithmic overhead in space and query time per level (the previous solutions require at least a small fixed power of n). We show that Hopcroft's problem (detecting an incidence among n lines and n points) can be solved in time $n^{4/3}2^{O(\log^* n)}$. In all these algorithms we apply Chazelle's results on computing optimal cuttings.

1. Introduction

Building on the works of Agarwal [A] and of the author (e.g., [M2] and [M1]), Chazelle recently found a new approach to the computation of so-called cuttings [C2], and also gave some applications of this result. In this paper we present several results which use his construction in a substantial way.

Mainly we consider geometric range searching problems, whose prototype is the following *simplex range searching problem*: 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, when a weight function on the points is assumed and a cumulative weight of the points in $P \cap \sigma$ is requested. The weights are assumed to belong to a semigroup, i.e., subtractions should not be used when computing the answer.

* Part of this research was done during the First Utrecht Computational Geometry Workshop, supported by the Dutch Organization for Scientific Research (NWO).

This problem has a rich literature, which we do not try to survey here; we only recall the current best bounds. Chazelle [C1] derived lower bounds for this problem: under certain reasonable assumptions on the model of computation, if a simplex range searching algorithm is allowed to use storage at most m, the worst-case query time is no better than $\Omega(n/(m^{1/d}\log n))$ (resp. $\Omega(n/\sqrt{m})$ for $d = 2$; it seems that the logarithmic factor in the denominator of the lower bound for higher dimensions might be only a product of the proof technique). Chazelle *et al.* [CSW] gave an algorithm whose performance matches this lower bound quite closely, and another (simpler and slightly more efficient) solution for the case of roughly linear space was given in [M4].

In this paper we improve the current upper bounds in the full range of the space/query-time tradeoff (Theorem 6.2), further approaching the lower bounds. In particular, we present a linear-space data structure for the simplex range searching problem with query time $O(n^{1-1/d})$ (Theorem 4.1); in view of the above-mentioned lower bounds, this is optimal for $d = 2$, and quite likely also for $d > 2$.

The previously known and new bounds are summarized in Table 1. In each section of the table we first give the general space/query-time tradeoff (the best query time achievable with given space m), and then various special cases and/or improvements for particular values of m. In the table, as well as in the whole paper, δ denotes an arbitrarily small positive constant; the constants hidden in the big-Oh notation may depend on δ. The $O(\ )$ notation is omitted in the table.

We also obtain results concerning the construction of so-called multilevel range searching structures. This idea has been widely used, e.g., in orthogonal range searching problems. In the context of partition trees and similar range searching structures, it appears in the paper by Dobkin and Edelsbrunner [DE], and it has been elaborated in various directions in [CSW], [OSS], [M4], [AS], and other papers. The currently most efficient known method for constructing such structures is based on the results of [CSW]. Let us remark that while the results of [M4] improved the simplex range searching with a linear space and simplified the multilevel structures to some extent, they did not increase the asymptotic efficiency for the multilevel structures. Without going into technical details here, let us indicate the main result about multilevel range searching structures: we show that for the full range of space/query-time tradeoffs, such structures can be built with only a polylogarithmic overhead in space and query time per level of the structure. This improves previous solutions, where an extra factor n^{δ} was necessary.

We also consider Hopcroft's problem: given n lines and n points in the plane, decide whether some point lies on some of the lines. Understanding this problem seems to be one of the major challenges in computational geometry. Many algorithms were published, the most efficient one in Chazelle's paper [C4], with time complexity $O(n^{4/3}\log^{1/3} n)$. It is suspected that $n^{4/3}$ might be the true computational complexity of this problem, although nothing even approaching a proof is known. We show that the running time can be improved to $n^{4/3}2^{O(\log^* n)}$ (here $\log^* x$ is the function defined by $\log^* x = 1$ for $x < 1$,

$$\log^* x = \log^*(\log_2 x) + 1 \text{ for } x \geq 1).$$

Table 1. Results on the simplex range searching problem with semigroup weights.

Space	Query time	Preprocessing	Source
	Lower bounds		
$m\ (n \le m \le n^d)$	$\Omega\left(\frac{n}{\sqrt{m}}\right)$ $(d = 2)$		[C1]
	$\Omega\left(\frac{n}{m^{1/d}\log n}\right)$ $(d > 2)$		
n	$\Omega(\sqrt{n})$ $(d = 2)$		[C1]
	$\Omega\left(\frac{n^{1-1/d}}{\log n}\right)$ $(d > 2)$		
	Previous upper bounds		
$m\ (n \le m \le n^d)$	$\frac{n^{1+\delta}}{m^{1/d}}$	$m^{1+\delta}$	[CSW]
$n^{d+\delta}$	$\log^{d+1} n$	$n^{d+\delta}$	[CSW]
n	$n^{1-1/d}(\log n)^{O(1)}$	$n \log n$	[M4]
n	$n^{1-1/d}(\log \log n)^{O(1)}$	$n^{1+\delta}$	[M4]
	New upper bounds		
$m\ (n \le m \le n^d)$	$\frac{n}{m^{1/d}} \log^{d+1} \frac{m}{n}$	$n^{1+\delta} + m(\log n)^{\delta}$	
n^d	$\log^{d+1} n$	$n^d(\log n)^{\delta}$	
n	$n^{1-1/d}$	$n^{1+\delta}$	

The paper is structured as follows: In Section 2 we recall the definition of cuttings and explain Chazelle's result. In Section 3 we discuss Hopcroft's problem as the most direct application. Section 4 describes a simplex range searching structure with linear space. In Section 5 we introduce a tool for building multilevel data structures, so-called half-space decomposition schemes, and we construct two types of these. In Section 6 we describe a simplex range searching structure with polylogarithmic query time and a combination of data structures which yields a space/query-time tradeoff. We mention various open problems in Section 7.

2. Efficient Hierarchical Cuttings

In many randomized computational geometry algorithms of divide-and-conquer type, as well as in their deterministic counterparts, the following notion of a $(1/r)$-*cutting* turned out to be crucial. A *cutting* is a collection Ξ of closed d-dimensional simplices[1] with disjoint interiors, which cover all E^d. Let H be a

[1] By a simplex we mean an intersection of at most $d + 1$ closed half-spaces, so we also admit unbounded simplices.

collection of n hyperplanes in E^d. For a simplex s, let H_s denote the collection of hyperplanes of H intersecting its interior. A $(1/r)$-*cutting* for H is a cutting Ξ satisfying $|H_s| \le n/r$ for every $s \in \Xi$ (here r is a parameter, $1 \le r \le n$). The *size* of a $(1/r)$-cutting is the number of its simplices.

Chazelle and Friedman [CF] proved that, for every H and $r \le n$, there exists a $(1/r)$-cutting of size $O(r^d)$ for H (which is asymptotically the best possible size), and that such a $(1/r)$-cutting can be computed by a randomized algorithm in expected $O(nr^{d-1})$ time. This running time is optimal if, together with the $(1/r)$-cutting, the collections H_s should also be output, since the sum of the sizes of these collections is $\Omega(nr^{d-1})$ (at least if the hyperplanes are in a general position). After several previous works on deterministic computation of cuttings (see, e.g., [M1] and references therein), Chazelle [C2] was able to match this performance by a deterministic algorithm for the full range of the values of r. The main novel feature in Chazelle's solution is that, together with the required $(1/r)$-cutting, he also computes a certain hierarchy of coarser cuttings. This is usually important in applications of his results; so we introduce a name for it.

We say that a cutting Ξ' *refines* a cutting Ξ if every simplex of Ξ' is completely contained in a single simplex of Ξ. We say that Ξ' *C-refines* Ξ if it refines Ξ and every simplex of Ξ contains at most C simplices of Ξ'. Let H, r be as in the definition of the $(1/r)$-cutting, and let C, $\rho > 1$ be constants. Let $\Xi_0, \Xi_1, \ldots, \Xi_k$ be cuttings such that Ξ_0 is the cutting consisting of the single "simplex" E^d, and every Ξ_i $(1 \le i \le k)$ is a $(1/\rho^i)$ cutting of size $O(\rho^{id})$ which C-refines Ξ_{i-1}, and $\rho^{k-1} < r \le \rho^k$ (thus $k = \Theta(\log r)$). Then we call the sequence $\Xi_0, \ldots, \Xi_k$ an *efficient hierarchical* $(1/r)$*-cutting (for* H*).*[2]

If s is a simplex of Ξ_i and s' is a simplex of Ξ_{i-1} containing it, we call s' the *father* of s and s a *son* of s'.

In one of our subsequent constructions, we will need the following stronger (and natural) property:

> For every i, j, $0 \le j < i \le k$, and $s \in \Xi_j$, s contains at most $O(\rho^{(i-j)d})$ simplices of Ξ_i. (2.1)

Chazelle's method implies the following:

Theorem 2.1 [C2] (Hierarchical Cutting Lemma). *For every fixed dimension d, there exist constants C, $\rho > 1$, such that, for any H and r, an efficient hierarchical $(1/r)$-cutting for H satisfying the additional condition* (2.1) *exists*[3] *and can be computed deterministically in time $O(nr^{d-1})$ (together with the collections H_s for every simplex $s \in \Xi_k$).*

Let us remark that we can always choose ρ larger than any prescribed constant.

---

[2] This notion of course depends on the values of C, ρ, so, in order to be quite rigorous, we should say something like a "(C, ρ)-efficient hierarchical $(1/r)$-cutting," but since the values of C, ρ are not essential as long as they do not depend on n, we do not use such a cumbersome term.

[3] Condition (2.1) is not explicitly established in [C2], but it follows easily from the construction.

Using the method of [M4] in a straightforward way, we obtain the following corollary (we omit the proof, since the method is explained in [M4] on a very similar case of the computation of a $(1/r)$-cutting):

Corollary 2.2. *There exists a constant $\alpha > 0$ (dependent on the dimension d), such that the following holds: for $r \leq n^{\alpha}$, an efficient hierarchical $(1/r)$-cutting can be computed in time $O(n \log r)$ (without computing the H_s's).*

Usually the main point of interest in an efficient hierarchical $(1/r)$-cutting is the finest cutting Ξ_k, but the hierarchy supplies a useful tree-like "access structure" for Ξ_k. For example, given a query point p, a simplex of Ξ_k containing it can be located by a straightforward descent through the hierarchy, in $O(\log r)$ time.

In one of our applications, we also need to consider a weighted collection of hyperplanes, which is a pair H, w, where H is a set of hyperplanes and $w: H \to R^+$ is a (nonnegative) weight function. For $X \subseteq H$, we write $w(X)$ for $\sum_{h \in X} w(h)$. The notions of a (1/r)-cutting and of an efficient hierarchical $(1/r)$-cutting can be naturally generalized for the weighted case (each simplex in a $(1/r)$-cutting for H, w should be intersected by hyperplanes of total weight at most $w(H)/r$). By a method described in [M2] (using a simple trick and the simulation of simplicity), an algorithm computing an efficient hierarchical $(1/r)$-cutting extends also to the weighted case (with only a constant factor overhead).

For our simplex range searching construction, we need one more concept (originating in [M3] and [CSW]). Let P be a set of n points in E^d, and let $r \leq n$ be a parameter. Let G be a set of $d + 1$ hyperplanes. We call G a *guarding set* for a hyperplane h (with respect to P and r) if the zone of h in the arrangement of G contains less than $n/r^{1/d}$ points of P.

The following lemma immediately follows from [M4]:

Lemma 2.3 (Guarding Set Lemma). *Given P and r, one can compute (in polynomial time) a set $\mathcal{G}$ of $O(r)$ $(d + 1)$-tuples of hyperplanes plus an additional data structure (requiring $O(r)$ space), such that $\mathcal{G}$ contains a guarding set for every hyperplane h, and such a set can be found for given h in $O(\log r)$ time. For $r < n^{\alpha}$ for a suitable constant $\alpha > 0$, $\mathcal{G}$ can be constructed in $O(n \log r)$ time.*

3. Hopcroft's Problem

In this section we prove the following improved upper bound for Hopcroft's problem:

Theorem 3.1. *Given n points and n lines in the plane, the existence of a line/point incidence can be decided in time $n^{4/3}2^{O(\log^* n)}$.*

The algorithm can be easily modified for counting or reporting all the incidences within the same time bound, only the details are slightly more complicated. Also,

there is an obvious generalization of the problem into higher dimensions (detecting or counting hyperplane/point incidences), and our solution can be generalized as well. These generalizations are not presented here.

Our algorithm for Hopcroft's problem is similar in spirit to previous ones (e.g., in [C2]), we only use a more complicated recursion. By an (n, m)-problem, we mean the problem of detecting a line/point incidence for at most n points and at most m lines. By the well-known line/point duality, an (n, m)-problem can be solved in the same time as an (m, n)-problem.

Lemma 3.2. *An (n, m)-problem can be solved in time $O(mr + n \log r)$ plus the time needed for solving $O(r^2)$ $(n/r^2, m/r)$-problems (r is a parameter which can be chosen in range from 1 to $\min(m, \sqrt{n})$).*

Proof. The idea goes back to Edelsbrunner *et al.* [EGH$^+$], and a procedure similar to ours is sketched in [C2]. However, since there are rather delicate issues of degenerate cases to be treated, we give a proof here.

Let H be the set of m lines and let P be the set of n points. Using Theorem 2.1, we compute an efficient hierarchical $(1/r)$-cutting for the set of lines, together with the subcollection H_s of lines intersecting the interior of every simplex s (a triangle in this case) of the finest cutting Ξ_k. Then we locate every point in the cutting Ξ_k. This takes time $O(mr + n \log r)$. Now, for every simplex $s \in \Xi_k$, we know the set P_s of points contained in it (including its sides and vertices). Each point of P which is not a vertex of Ξ_k contributes to at most two P_s's. On the other hand, the points coinciding with vertices contribute at most three members of each P_s, and hence $\sum_{s \in \Xi_k} |P_s| \leq 2n + O(r^2) = O(n)$. For every s, we further partition (if necessary) P_s into subsets of size at most n/r^2, and, for every such subset, we solve the subproblem involving that subset of points and the set H_s of lines. By the previous considerations, we get $O(r^2)$ $(n/r^2, m/r)$-problems. The only incidences which might avoid detection in these subproblems are those formed by a line containing a side of a simplex and a point on that side. However, for every side, we can tell whether it is contained in a line,[4] and take care of this in the point-location phase. □

Proof of Theorem 3.1. Let $T(n, m)$ denote the time needed for solution of an (n, m)-problem. Let $\varphi(x)$ be the function given by

$$\varphi(x) = 1 \qquad \text{for} \quad x < 2,$$

$$\varphi(x) = \log_2 x \, \varphi(\log_2 x) \qquad \text{for} \quad x \geq 2.$$

We begin by considering a problem involving $m\varphi^3(m)$ lines and $m^2\varphi^3(m)$ points (for

[4] One possible way of how to do this: compute the list of all lines defined by the sides, sort it lexicographically by the coefficients in their equation, and search for every line of H in this list; the total running time is $O(r^2 \log r + m \log r) = O(n \log r + mr)$.

these rather mysteriously looking parameters we get a nice recursion). We choose $r = m \log m$, and from Lemma 3.3 we get

$$T(m^2\varphi^3(m), m\varphi^3(m)) \leq O(m^2 \log m\, \varphi^3(m)) + O(m^2 \log^2 m) T(\log m\, \varphi^3(\log m), \log^2 m\, \varphi^3(\log m)).$$

We see that in each subproblem we have a similar form of parameters as in the original problem, only that the role of points and lines is flipped and m is replaced by $\log m$. Denoting $f(m) = T(m^2\varphi^3(m), m\varphi^3(m))$, we thus have

$$f(m) \leq O(m^2 \log m\, \varphi^3(m)) + O(m^2 \log^2 m) f(\log m),$$

and it is easily verified that a solution to this recurrence satisfies

$$f(m) \leq m^2\varphi^4(m) 2^{C \log^* m}$$

for a large enough constant C.

It remains to reduce the (n, n) problem to the case solved above. We can do it in two steps: In the first step we choose $r = n^{1/3}/\varphi(n)$. This yields $O(n^{2/3}/\varphi^2(n))$ problems with at most $n^{2/3}\varphi(n)$ lines and at most $n^{1/3}\varphi^2(n)$ points each. In the second step we flip the role of lines and points, and choose $r = n^{1/3}/\varphi(\log n)$. In this way we get $O(n^{4/3}/(\varphi^2(n)\varphi^2(\log n)))$ subproblems in total, each being a $(\log^2 n\, \varphi^3(\log n), \log n\, \varphi^3(\log n))$-problem. The running time in these two steps is easily checked to be $o(n^{4/3})$. By the previous considerations, each subproblem can be solved in time $\log^2 n\, \varphi^4(\log n) 2^{O(\log^* n)}$, and multiplying this by the number of subproblems, we get the total running time $n^{4/3} 2^{O(\log^* n)}$. □

Theorem 3.1 has an amusing consequence, which was pointed out by Eppstein [Ep]. Namely, it implies the following:

Corollary 3.3. *An algorithm A can be constructed for solving the Hopcroft problem, such that if there exists any algorithm (in the algebraic decision tree model) for the Hopcroft problem with $O(n^{4/3})$ (resp. $o(n^{4/3})$) running time, then algorithm A has $O(n^{4/3})$ (resp. $o(n^{4/3})$) running time.*

Indeed, we may use few levels of the recursion to obtain $O((n/\log\log\log n)^{4/3})$ subproblems of size $O(\log\log\log n)$. Then we exhaustively search (using the exponential-time decision theory of real numbers) for the algebraic decision tree algorithm on problems of that size having minimum complexity. We then run that algebraic decision tree on all the subproblems. This idea has been used in [L], and several applications (e.g., to minimum spanning trees) were mentioned by R. E. Tarjan in an invited lecture at the Second Scandinavian Workshop on Algorithm Theory, 1990. A useful consequence of this result might be that in

searching for an improved algorithm, we may restrict ourselves to the algebraic decision tree model.

4. Simplex Range Searching: Linear Space

As mentioned in the introduction, the best query time for simplex range searching with linear space was achieved in [M4]—$O(n^{1-1/d}(\log\log n)^{O(1)})$. In some special cases the $(\log\log n)^{O(1)}$ factor can be replaced by a much smaller one, $2^{O(\log^* n)}$. In this section we show

Theorem 4.1. *There exists a simplex range searching structure with $O(n)$ space, $O(n^{1-1/d})$ query time, and $O(n^{1+\delta})$ preprocessing time.*

Proof. The proof combines and somewhat refines ideas of [CW], [CSW], and some previous works of the author on this and related problems.

For technical reasons, we describe a way to construct a subset $P' \subseteq P$ of at least half of the points of P, and a simplex range searching structure for this P' with performance as in the theorem. In order to get a simplex range searching structure for the whole P, we perform the above-mentioned construction for P, then for $P\backslash P'$, etc., obtaining a logarithmic number of range searching structures with geometrically decreasing sizes. A query for P can be answered by querying each of these structures; the performance of this combined data structure is as required.

We also assume that the points of P are in general position; degenerate cases can be handled using simulation of simplicity (see, e.g., [Ed]).

Let us describe the components and properties of the simplex range searching structure. One part is a collection $\Psi_0 = \{s_1, \ldots, s_t\}$ of $t = n^{1/d}\log n$ full-dimensional simplices (not necessarily disjoint). For every $i = 1, 2, \ldots, t$ we also have a set $P_i \subset P$ of cardinality at most $n/2t$, with $P_i \subset s_i$. The P_i's are disjoint and together they form the above-mentioned set P'.

For every s_i, there is a rooted tree whose nodes are simplices, with s_i as the root. Every nonleaf node of this tree has $O(1)$ sons. These sons are simplices with disjoint interiors which together cover their father. The trees have depth at most $q = O(\log n)$. We note that every point of P_i is contained in exactly one leaf simplex. As will be apparent from the construction, leaf nodes may occur at all levels of the trees, not only at the deepest one.

We let Ψ_j be the collection of all simplices which lie at distance j from the root in some of the trees.

Given the general position of P, we may assume that none of the points of P is contained in the boundary of a simplex of some Ψ_j. For a simplex $s \in \Psi_j$, we let $P(s) = P_i \cap s$, where s_i is the root of the tree containing s.

For a hyperplane h, let $K_j(h)$ be the set of simpices of Ψ_j intersected by h, and let $L_j(h)$ be the leaf simplices from $K_j(h)$. Let $K(h) = K_1(h) \cup \cdots \cup K_q(h)$, and similarly for $L(h)$.

In what follows we will show how to construct (in polynomial time) the

above-described objects, satisfying the following conditions (for every hyperplane h):

$$\sum_{j=0}^{q} |\Psi_j| = O(n), \tag{4.1}$$

$$|K(h)| = O(n^{1-1/d}), \tag{4.2}$$

$$\sum_{s \in L(h)} |P(s)| = O(n^{1-1/d}). \tag{4.3}$$

The simplex range searching structure will then be complemented as follows: with every nonleaf simplex s of the Ψ_j's, we store the total weight of the points of $P(s)$, and for the leaf simplices we store the list of points of $P(s)$. Storing the simplices, the tree structures and this additional information requires $O(n)$ space, by (4.1).

Given a query simplex σ, the weight of points in $P' \cap \sigma$ is computed as follows:

1. We set a variable w to the total weight of the point sets P_i such that the simplices $s_i \in \Psi_0$ are completely contained in σ. We also set a variable $\mathscr{C}$ to the set of all simplices of Ψ_0 intersected by the boundary of σ.
2. We repeat the following step until $\mathscr{C} = \emptyset$, then we output w as the answer.
3. We remove one simplex s from $\mathscr{C}$. If it is a leaf simplex, we directly test the membership of every point of $P(s)$ in σ, and increase w by the weight of $P(s) \cap \sigma$. If s is a nonleaf simplex, we determine the position of each of its sons with respect to σ. We add those intersecting the boundary of σ to $\mathscr{C}$, and we increase w by the total weight of the point sets corresponding to the sons completely contained in σ.

Conditions (4.2) and (4.3) guarantee that the time spent in the execution of steps 2–3 will be $O(n^{1-1/d})$. If step 1 is executed trivially, by inspecting all the simplices of Ψ_0, it needs time $O(t) = O(n^{1/d} \log n)$. For $d > 2$, this is $o(n^{1-1/d})$; for $d = 2$, we have to use one more auxiliary data structure in order to execute this step in time $O(n^{1/2})$. The requirements on the performance of the auxiliary structure are quite mild, and, e.g., a data structure described in [M4] is more than sufficient.

Our simplex range searching structure has a linear storage, $O(n^{1-1/d})$ query time, and a polynomial-time preprocessing, as is shown below. Later (in Section 5.5), we show how to reduce the preprocessing time to $O(n^{1+\delta})$.

It remains to show that the data structure with the above-described properties can indeed be constructed.

In order to gain some intuition about the construction, the reader is advised first to look into [M4], where a similar procedure is used to solve a simpler task. Here we only give a sketchy description of the basic ideas and then a formal proof, which might look somewhat scary at first sight.

The problem considered in [M4] is to partition a point set into classes of a prescribed size, each class being contained in some simplex, and in such a way that no hyperplane crosses too many of these simplices. The following are the

basic ingredients in this simpler construction. First, it is shown that it is sufficient to guarantee a small crossing number only for the hyperplanes of a carefully chosen small collection H, in our case the ones from the guarding sets as in Lemma 2.3. If a $(1/r)$-cutting for H is constructed, then an average hyperplane of H crosses only a $(1/r)$ fraction of all simplices of the cutting. In order to get a worst-case bound on the number of crossed simplices for all hyperplanes of H, the simplices are selected one by one. Each new simplex is selected as one of the simplices of a suitable $(1/r)$-cutting. This $(1/r)$-cutting is computed anew in each step, so that the hyperplanes which happen to intersect many of the already selected simplices are favored over the other hyperplanes. To this end, a weight function w on the hyperplanes is defined: the current weight of a hyperplane h during the construction is equal to $2^{\kappa(h)}$, where $\kappa(h)$ is the number of simplices constructed so far which are crossed by h. Each $(1/r)$-cutting is computed with respect to the current weights. A relatively easy calculation with the weights then gives the worst-case crossing number bound.

In our forthcoming construction, each step constructs not only a single simplex, but a whole tree-like structure of simplices. To this end we compute an efficient hierarchical cutting and we select one simplex from a suitable intermediate cutting in the hierarchy, and together with it we take all simplices from the finer levels of the hierarchy contained in it. The efficient hierarchical cutting is again computed with respect to certain current weighting of the hyperplanes of H. The weight function is more complicated than in the above-mentioned case. Essentially this is because it has to "watch" the crossing numbers in all levels of the tree-like structures. For the range searching data structure, the deepest (finest) level is the most crucial one, thus it is given the largest importance in the weight function. The requirements on the crossing numbers of the higher-level simplices are less strict, but they still have to be reflected in the weight function. An additional technical complication arises because the points of P need not be uniformly distributed among the simplices of the hierarchical cuttings used in the construction. For the finest-level simplices, it is not enough to guarantee a small number of simplices crossed by a hyperplane, but it is also necessary that the number of points appearing in the simplices crossed by any hyperplane is not too large. Another term in the weight function accounts for this. On the other hand, it could happen that many of the selected simplices on the deeper levels will be almost empty. This does not matter for the crossing-number argument concerning the hyperplanes of H, but it would cause problems when arguing about the crossing number of a general hyperplane. For this reason, we omit such almost empty simplices from our data structure. A more formal description of the construction follows.

We begin by applying Lemma 2.3 for the set P and with $r = n$. This yields a collection $\mathscr{G}$ of guarding sets, and we let H be the union of all sets from $\mathscr{G}$.

The construction of the data structure proceeds in t phases; in the ith phase we produce P_i, s_i, and the whole tree rooted at s_i.

Let us consider the situation at the beginning of the $(i+1)$st phase, when $P_1, \ldots, P_i$, $s_1, \ldots, s_i$, and their trees have already been constructed. We let

$\Psi_0^{(i)}, \ldots, \Psi_q^{(i)}$ stand for the already constructed parts of $\Psi_0, \ldots, \Psi_q$, and $K_j^{(i)}(h)$, $L_j^{(i)}(h)$ have a meaning analogous to $K_j(h)$, $L_j(h)$.

The construction of P_{i+1}, s_{i+1}, and its associated tree structure in phase $i+1$ begins by defining a weight function w_i on the set H of hyperplanes. For $h \in H$, we set

$$w_i(h) = \exp\left(\frac{\log n}{n^{1-1/d}}\left[\sum_{j=0}^{q} 4^{q-j}|K_j^{(i)}(h)| + \sum_{s \in K_q^{(i)}(h)} |P(s)|\right]\right). \tag{4.4}$$

In this way we obtain a weighted collection[5] of hyperplanes H, w_i. According to Theorem 2.1 and the remarks following it, we construct $\Xi_0, \ldots, \Xi_k$, an efficient hierarchical $(1/n^{1/d})$-cutting for H, w_i, which also satisfies the property (2.1).

The construction of an efficient hierarchical $(1/n^{1/d})$-cutting guarantees that the size of Ξ_i does not exceed $C_1\rho^{id}$, where $\rho > 4$ is as in the definition of an efficient hierarchical cutting and the constant C_1 can be deduced from the proof. We let p be the maximal index with $C_1\rho^{pd} \le t$, thus $\rho^{pd} = \Theta(t)$, and Ξ_p is a $(1/r_p)$-cutting for H, w_i with $r_p = \rho^p = \Theta(t^{1/d})$, and it consists of at most t simplices. We define q (the depth of our tree structures) by $q = k - p$.

We let $\tilde{P}_i = P \setminus (P_1 \cup \cdots \cup P_i)$; thus $|\tilde{P}_i| \ge n/2$. Hence there exists a simplex (which we denote by s_{i+1}) of Ξ_p containing at least $n/2t$ points of $\tilde{P}_i$. We add this s_{i+1} to $\Psi_0^{(i)}$, obtaining $\Psi_0^{(i+1)}$. We select some $n/2t$ points of $\tilde{P}_i$ contained in s_{i+1} as P_{i+1}.

We consider the set S of all simplices in $\Xi_p, \ldots, \Xi_k$ contained in s_{i+1}; these form the tree structure rooted at s_{i+1}, only we need to eliminate some of them. We say that a simplex of S belonging to Ξ_{p+j} $(j = 0, 1, \ldots, q)$ is *fat* if it contains at least 2^{q-j} points of P_{i+1} (at least s_{i+1} must be fat). We put a simplex $s \in S$ belonging to Ξ_{p+j} to the tree rooted at s_{i+1} iff all its predecessors (its father, father of its father, etc., until s_{i+1}) are fat. The simplices of depth j in this tree are added to $\Psi_j^{(i)}$, forming $\Psi_j^{(i+1)}$. This finishes the description of phase $i+1$ of the construction. It is straightforward to check that the construction can be executed in time polynomial in n and r.

By property (2.1) of the efficient hierarchical cutting used for the construction we have

$$|\Psi_j^{(i+1)} \setminus \Psi_j^{(i)}| = O(\rho^{jd}), \tag{4.5}$$

and the total weight w_i of hyperplanes of H intersecting each simplex of $\Psi_j^{(i+1)} \setminus \Psi_j^{(i)}$ is

$$O\left(\frac{w_i(H)}{\rho^{j+p}}\right) = O\left(\frac{w_i(H)\rho^{q-j}}{n^{1/d}}\right). \tag{4.6}$$

[5] Let us remark that in an actual implementation, the computation of the weight can be performed with quite limited precision; see [M3] for a more detailed analysis in a similar problem.

Since (4.5) implies that we add at most $O(\rho^{qd}) = O(n/t)$ new simplices in each of the t phases, property (4.1) follows. It remains to establish (4.2), (4.3), for which we use the following lemma:

Lemma 4.2. *For every hyperplane $h \in H$,*

(i) $|K(h)| = O(n^{1-1/d})$,
(ii) $|K_j(h)| = O(n^{1-1/d}4^{-(q-j)})$, $j = 0, 1, \ldots, q$,
(iii) $\sum_{s \in K_q(h)} |P(s)| = O(n^{1-1/d})$.

Before proving this lemma we derive the required properties (4.2), (4.3) from it. Let h be a general hyperplane and let $G \subset H$ be its guarding set. Any simplex intersecting h but none of its guarding hyperplanes lies completely in the zone of h in the arrangement of G, and thus all such simplices together contain at most $O(n^{1-1/d})$ points of P.

In order to establish (4.2) for h, let us consider the set F of fathers of the simplices of $K(h)$; it is enough to show $|F| = O(n^{1-1/d})$. Let F_0 be the set of simplices of F intersecting some hyperplane of G; by Lemma 4.2(i) we have $|F_0| = O(n^{1-1/d})$. Each of the simplices of $F_1 = F \setminus F_0$ is completely contained in the zone of h in the arrangement of G. Let us consider the simplices of F_1 belonging to Ψ_j. Their corresponding subsets of P are disjoint, each has at least 2^{q-j} points (since the nonleaf simplices are fat), and together they contain at most $n^{1-1/d}$ points; hence their number is $O(n^{1-1/d}2^{-(q-j)})$. Summing up, we also get $|F_1| = O(n^{1-1/d})$.

As for property (4.3), first we consider the leaf simplices intersecting h but not its guarding set; since the point sets of the leaf simplices are disjoint, these contain $O(n^{1-1/d})$ points in total. Hence it suffices to show property (4.3) for every hyperplane $h \in H$. For the leaf simplices from Ψ_q intersected by h, we already have the bound $O(n^{1-1/d})$ directly from Lemma 4.2(iii). For $j < q$, using Lemma 4.2(ii) and the fact that the leaf simplices from Ψ_j contain less than 2^{q-j} points each, we get that there are at most $O(n^{1-1/d}2^{-(q-j)})$ points in the leaf simplices of Ψ_j intersecting h. The bound (4.3) follows by summation on j. □

Proof of Lemma 4.2. Clearly (ii) implies (i). Below we prove the estimate $\log w_i(H) = O(\log n)$. By our choice of the weight function, this implies that, for every h,

$$\sum_{j=0}^{q} 4^{q-j}|K_j(h)| + \sum_{s \in K_q(h)} |P(s)| = \frac{n^{1-1/d}}{\log n} \log w_i(h) \le \frac{n^{1-1/d}}{\log n} \log w_i(H) = O(n^{1-1/d}),$$

from which (ii) and (iii) follow.

Let us investigate the increase in the total weight of the hyperplanes of H when passing from phase i to phase $i+1$, i.e., the relation between $w_i(H)$ and $w_{i+1}(H)$.

First we show that the weight of every hyperplane increases by at most a constant factor in every phase. Indeed, in view of (4.5), the weight of any hyperplane

increases by a factor not exceeding

$$\exp\left(\frac{\log n}{n^{1-1/d}}\left[\sum_{j=0}^{q} 4^{q-j}O(\rho^{jd}) + \frac{n}{2t}\right]\right) = \exp(O(1))$$

(we have used $\rho^{qd} = \Theta(n/t) = \Theta(n^{1-1/d}/\log n)$ and $\rho^d > 4$).

Let us now imagine that we are adding the new simplices one by one. We can naturally define intermediate weights (among w_i and w_{i+1}) of hyperplanes of H after adding a part of the new simplices, by replacing $K_j^{(i)}(h)$ in definition (4.4) by the set of simplices intersected by h among those of $\Psi_j^{(i)}$ and the new ones added so far. By adding a new simplex $s \in \Psi_j$, the intermediate weight increases only for the hyperplanes intersecting that simplex. Using (4.6), we get that, for $j < q$, by adding a simplex of Ψ_j the intermediate weight grows by a factor at most

$$\begin{aligned} f_j &= 1 + O\left(\frac{(\exp(4^{q-j}\log n/n^{1-1/d}) - 1)\rho^{q-j}}{n^{1/d}}\right) \\ &\le 1 + O\left(\frac{4^{q-j}\rho^{q-j}\log n}{n}\right) \\ &\le \exp\left(O\left(\frac{4^{q-j}\rho^{q-j}\log n}{n}\right)\right) \end{aligned} \tag{4.7}$$

(we have used the inequalities $1 + x \le e^x \le 1 + 2x$, the second one valid for $x \le 1$). For $j = q$, the weight grows by a factor of f_q and then also by a factor of

$$f(s) = 1 + O\left(\frac{\exp(|P(s)|\log n/n^{1-1/d}) - 1}{n^{1/d}}\right) \le \exp\left(O\left(\frac{|P(s)|\log n}{n}\right)\right). \tag{4.8}$$

We thus obtain the estimate

$$w_t(H) \le w_0(H)\left(\prod_{s\in\Psi_q} f(s)\right)\prod_{j=0}^{q} f_j^{|\Psi_j|}.$$

Using $w_0(H) = O(n)$, $|\Psi_j| = O(t\rho^{jd})$, and substituting from the above estimates for f_j and $f(s)$, we get the desired result $w_t(H) = \exp(O(\log n))$. □

5. Half-Space Decomposition Schemes

In this section we continue considering various improvements in solutions to geometric range searching problems, implied by Chazelle's results on hierarchical cuttings. We now set up an abstract framework.

5.1. *Definitions*

Let Γ be a family of subsets of E^d (the reader may imagine the set of all half-spaces). A Γ-*decomposition scheme* $\mathscr{D}$ is a function assigning to every finite subset $P \subset \mathrm{E}^d$ a data structure $\mathscr{D}(P)$, with the following components:

- A collection of designated subsets of P, called the *canonical sets*. These subsets are not usually represented by lists of their points; rather the data structure stores only certain information associated with every canonical set (e.g., the number of points). We call this information the *secondary structure* of $\mathscr{D}$; formally this is a function $S_{\mathscr{D},P}$ defined on the collection of canonical sets of $\mathscr{D}(P)$.
- An algorithm which, given a set $\gamma \in \Gamma$, outputs the names of certain canonical sets, whose disjoint union is equal to $P \cap \gamma$ (the *decomposition of* γ). Here "names" practically means pointers to the secondary structures of these sets.[6]

A basic example which can be regarded as a decomposition scheme in the above sense are the *range trees* (see, e.g., [PS]). Here the family Γ is the family of all rectangular axis-parallel boxes. Numerous other examples can be obtained from known geometric range searching structures.

The most direct application of a Γ-decomposition scheme $\mathscr{D}$ is for range searching with ranges belonging to Γ. To this end we define the secondary structure $S_{\mathscr{D},P}(C)$ for a canonical set C in the data structure $\mathscr{D}(P)$ as the total weight of points in C (e.g., if we need a Γ-range counting structure, the weight of C will be its cardinality). The total weight of points of $P \cap \gamma$ is the sum of the weights of the sets in the decomposition of $P \cap \gamma$, and it can be computed in an additional time proportional to the number of sets in the decomposition.

Sometimes it is useful to consider a Γ-decomposition scheme without specifying the secondary structures; when we want to emphasize this point of view, we speak about an *abstract* Γ-decomposition scheme.

The idea of multilevel range searching structures is captured by the notion of *composition* of decomposition schemes. Let $\mathscr{D}_1$ be an abstract Γ_1-decomposition scheme and let $\mathscr{D}_2$ be an abstract Γ_2-decomposition scheme. Let us define the secondary structure of $\mathscr{D}_1$ by setting, for a set P and its canonical subset C, $S_{\mathscr{D}_1,P}(C) = \mathscr{D}_2(C)$. In this way we naturally obtain an (abstract) Γ-decomposition scheme (where $\Gamma = \{\gamma_1 \cap \gamma_2; \gamma_1 \in \Gamma_1, \gamma_2 \in \Gamma_2\}$), which we denote by $\mathscr{D}_1 \circ \mathscr{D}_2$ and call the *composition* of $\mathscr{D}_1$ and $\mathscr{D}_2$. Indeed, given a set $\gamma = \gamma_1 \cap \gamma_2$, $\gamma_i \in \Gamma_i$, we consider the canonical sets forming the decomposition of $P \cap \gamma_1$ given by $\mathscr{D}_1(P)$, and, for every such set C, we take the collection of canonical sets forming the decomposition of $C \cap \gamma_2$ given by $\mathscr{D}_2(C)$. Taking these collections together, we obtain a decomposition of $P \cap \gamma_1 \cap \gamma_2$. For instance, if $\mathscr{D}$ is a half-space decomposition scheme, then $\mathscr{D} \circ \mathscr{D} \circ \cdots \circ \mathscr{D}$ ($(d+1)$-fold composition) is a simplex decomposition scheme.

---

[6] We could give the definition in a still more abstract setting by defining a decomposition scheme for a set system $(X, \mathscr{R})$ (or *range space* in the terminology used in some computational geometry papers). In this paper we remain in the geometric setting.

Similarly a composition of an abstract Γ_1-decomposition scheme with a Γ_2-range searching structure, obtaining a range searching structure for ranges of the form $\gamma_1 \cap \gamma_2$, $\gamma_1 \in \Gamma_1$, $\gamma_2 \in \Gamma_2$, can be defined.

The effectivity of a decomposition scheme is determined by various parameters: How many canonical sets are there? What is the distribution of their sizes? How many canonical sets are used in the decomposition of a range from Γ? How are their sizes distributed, and what is the preprocessing time for building the structure and the running time of the decomposition algorithm? The importance of various parameters may depend on a specific application. In particular, if the secondary structures use only a constant amount of space each (as, e.g., in the above-mentioned example of a Γ-range searching structure), we are interested in the number of canonical sets only (in total and in the decomposition). If, on the other hand, the decomposition scheme is used as a "top-level" structure in composed (multilevel) structures, then the sizes of the canonical subsets play an important role.

In order to express another common type of nesting in range searching structures, we also define a *partial Γ-decomposition scheme* (Fig. 1). The definition is similar to the Γ-decomposition scheme, but all canonical sets are divided into two subcollections: the *inner sets* and the *remainder sets*. The decomposition for a set $P \cap \gamma$ is a collection of disjoint canonical sets, containing both inner and remainder sets. Each inner set of the decomposition is completely contained in $P \cap \gamma$, while the remainder sets cover the remaining points of $P \cap \gamma$ (but they may also contain some other points of P). The general goal is that a possibly small number of canonical sets is used and all the remainder sets are small. For an application in a data structure, it is usually necessary to use some other decomposition schemes for the remainder sets.

For partial decomposition schemes, it makes sense to define two types of composition: *via inner sets*, where we define the secondary structures for the inner sets of the first decomposition scheme (in this case an abstract partial Γ_1-

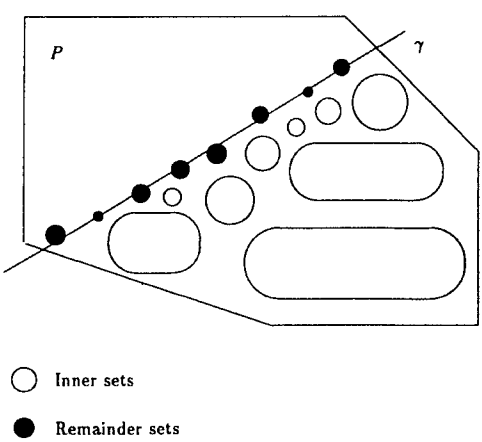

Fig. 1. A schematic illustration of a partial half-space decomposition scheme.

decomposition scheme can be composed with an abstract partial Γ_2-decomposition scheme, yielding an abstract partial Γ-decomposition scheme with

$$\Gamma = \{\gamma_1 \cap \gamma_2; \gamma_1 \in \Gamma_1, \gamma_2 \in \Gamma_2\},$$

or *via remainder sets*, where we define the secondary structures for the remainder sets of the first decomposition scheme (here it usually only makes sense to compose abstract partial Γ-decomposition schemes with the same Γ; in particular, by composing an abstract partial Γ-decomposition scheme with an abstract Γ-decomposition scheme, we obtain another Γ-decomposition scheme).

In the remainder of this section we concentrate on constructions of efficient partial half-space decomposition schemes, applicable in the construction of multi-level range searching structures.

5.2. *The Large Space Case*

In this section we consider a half-space decomposition scheme with $O(n^d)$ canonical sets, which provides decompositions consisting of $O(\log n)$ canonical sets. The following theorem is a rather direct consequence of Chazelle's cutting result and a construction of [CSW]. We formulate the result in terms of a partial decomposition scheme, which allows us to apply it in "composed" data structures whose storage requirements can be adjusted between almost linear and roughly $O(n^d)$, while the query time varies accordingly.

Theorem 5.1. *There exists a partial half-space decomposition scheme $\mathscr{L}$, such that, for an n-point set P and a prescribed parameter $r \le n$, the data structure $\mathscr{L}(P)$ has the following parameters: the inner sets can be partitioned into $O(\log r)$ collections $\mathscr{C}_0, \mathscr{C}_1, \ldots, \mathscr{C}_{k-1}$, such that $\mathscr{C}_i$ contains $O(\rho^i)$ sets of size at most n/ρ^i (where ρ is a constant, $\rho^{k-1} < r \le \rho^k$). For any half-space γ, each $\mathscr{C}_i$ $(i < k)$ contributes one inner set to the decomposition of $P \cap \gamma$. There are $O(r^d)$ remainder sets of size at most n/r each, and the decomposition of $P \cap \gamma$ contains only one remainder set. The decomposition can be found in $O(\log r)$ time, and the data structure can be built in $O(nr^{d-1})$ time.*

Proof. We let H be the collection of hyperplanes dual to the points of P, and we compute an efficient hierarchical $(1/r)$-cutting $\Xi_0, \ldots, \Xi_k$ for H as in Theorem 2.1. In order to simplify our considerations, we assume that the hyperplanes of H are in general position and also that the simplices of each Ξ_i are in general position with respect to H (this can be achieved by simulation of simplicity, see [Ed]). We now regard the canonical sets as sets of hyperplanes of H. The collection of remainder sets consists of the sets of hyperplanes intersecting a simplex s for all $s \in \Xi_k$. For $i < k$, the collection $\mathscr{C}_i$ contains the following sets: we consider a simplex $s \in \Xi_i$ and its son $s' \in \Xi_{i+1}$, and we include in $\mathscr{C}_i$ the set of hyperplanes intersecting s and lying completely below s', and also a similar set with "below" replaced by

"above." The claims about the distribution of sizes of canonical sets and the preprocessing time are immediate from Theorem 2.1.

In our dual setting, a decomposition of $P \cap \gamma$ corresponds to a decomposition of the set of hyperplanes of H lying below (or above) the point q dual to the boundary of γ; we speak about the "below" case only. The decomposition is formed as follows: let s_i be the simplex of Ξ_i containing q, then the decomposition contains the set of hyperplanes intersecting s_i and lying below s_{i+1}, $0 \le i < k$, plus the set of hyperplanes intersecting s_k. This decomposition can be computed in $O(\log r)$ time, and it is easily verified that it has the required parameters. □

5.3. *The Small Space Case*

Here we consider decomposition schemes whose canonical subsets have a roughly linear total size. In the following result we again combine an idea from [CSW] with the hierarchical cutting result, this time in a more complicated way.

Theorem 5.2. *There exists a partial half-space decomposition scheme $\mathscr{S}$, such that, for an n-point set P and a prescribed parameter $r \le n$, the data structure $\mathscr{S}(P)$ has the following parameters:*

There are $O(r \log r)$ inner sets with sizes summing up to $O(n \log^2 r)$, and $O(r \log r)$ remainder sets of size at most n/r each. For every half-space γ, the decomposition of $P \cap \gamma$ consists of $O(r^{1-1/d})$ sets, and if $\mathscr{C}$ stands for the collection of inner sets used in this decomposition, then

$$\sum_{C \in \mathscr{C}} |C|^{1-1/d} = O(n^{1-1/d} \log r).$$

The decomposition can be found in $O(r^{1-1/d})$ time. The structure can be built in time polynomial in n and r, and, for $r < n^{\alpha}$ for a certain constant $\alpha > 0$, the preprocessing time can be made $O(n \log^2 r)$.

Proof. We assume that the set P is in general position. This is no loss of generality, since the degenerate cases can be handled using simulation of simplicity.

Let us now apply Lemma 2.3 for P and r, obtaining a set $\mathscr{G}$ of $O(r)$ guarding sets. For a set P in general position, we may also assume that the hyperplanes forming $\mathscr{G}$ are in general positions as well (in particular, they are all distinct) and that none of them contains a point of P; this can again be guaranteed by the simulation of simplicity.

We now describe the construction of the partial half-space decomposition scheme. It consists of the data structure mentioned in the Guarding Set Lemma 2.3 (for finding a guarding set in $\mathscr{G}$ for every hyperplane) and of $p = O(\log r)$ components. Each component is associated with a subset $\bar{\mathscr{G}}_i$ of the guarding sets, and it is used for decomposition of half-spaces whose bounding hyperplanes have the guarding set in $\bar{\mathscr{G}}_i$.

The construction proceeds inductively, producing one component at a time. At the beginning of ith stage we have a current set $\mathscr{G}_i$ of guarding sets which remain to be treated ($\mathscr{G}_1 = \mathscr{G}$). We define a set H_i of hyperplanes as the union of the guarding sets of $\mathscr{G}_i$. We compute an efficient hierarchical $(1/r^{1/d})$-cutting $\Xi_0^{(i)}, \ldots, \Xi_k^{(i)}$ for H_i according to Theorem 2.1. In what follows we drop the superscript (i) for the cuttings. Again, we assume that all the simplices of the Ξ_j's are in general positions with respect to the hyperplanes of H_i as well as to the points[7] of P.

We locate, in total time $O(n \log r)$, the points of P within the simplices of every Ξ_j. For a simplex $s \in \Xi_j$, we set $P(s) = P \cap s$.

We say that a simplex $s \in \Xi_j$ is *fat* if $|P(s)| \geq 2^{k-j} n/r$. We say that s is *active* if all his predecessors (i.e., his father, father of his father, etc.) are fat, and s is a *leaf* simplex if it is active and either belongs to Ξ_k or is not fat.

With these definitions we can describe the canonical sets in the ith component. The inner sets are the subsets of P contained in the active simplices. Next, we consider the subsets of P contained in the leaf simplices, and we partition each of these sets whose size exceeds n/r into subsets with sizes between $n/2r$ and n/r. The resulting sets form the collection of the remainder sets in our decomposition scheme.

Since the cuttings $\Xi_1, \ldots, \Xi_k$ have $O(r)$ simplices in total, the ith component of the decomposition scheme has $O(r)$ inner sets. The inner sets coming from each Ξ_j are disjoint and have total size at most n, hence the total size of the inner sets of the ith component is $O(n \log r)$. The remainder sets are pairwise disjoint, and each has size at most n/r. Since they were defined by starting with $O(r)$ subsets and then by splitting subsets of size larger than n/r into parts larger than $n/2r$, the total number is $O(r)$.

We are now going to define $\mathscr{G}_i \subseteq \mathscr{G}_i$, the subset for which the current hierarchical cutting is "good." To this end we define a weight function on the guarding sets. For a hyperplane h, we let $K_j(h)$ be the set of simplices of Ξ_j intersected by h, $K(h) = K_1(h) \cup \cdots \cup K_k(h)$ and we define the weight of h by

$$w(h) = \sum_{j=1}^{k} 4^{k-j} |K_j(h)| + \frac{1}{k} \sum_{s \in K(h)} \left(\frac{r}{n} |P(s)| \right)^{1-1/d} + \sum_{s \in K_k(h)} \frac{r}{n} |P(s)|. \qquad (5.1)$$

Finally, the weight of a guarding set $G \in \mathscr{G}_i$ is the sum of weights of its hyperplanes.

Lemma 5.3. *The average weight of a guarding set $G \in \mathscr{G}_i$ is $O(r^{1-1/d})$.*

Proof. Since Ξ_j is a $(1/\rho^j)$-cutting for H_i (where $\rho^{k-1} < r^{1/d} \leq \rho^k$), each $s \in \Xi_j$ is

[7] This triple application of the simulation of simplicity looks fearful, at least from an algorithmic point of view. We could completely avoid it, but the presentation would become much more complicated.

intersected by no more than $|H_i|/\rho^j$ hyperplanes of H_i. Summing over all $O(\rho^{jd})$ simplices in Ξ_j, we get $\sum_{h\in H_i}|K_j(h)| = O(|H_i|\rho^{j(d-1)})$, and so the average value of $|K_j(h)|$ is $O(\rho^{j(d-1)})$. Similarly,

$$\sum_{h\in H_i}\sum_{s\in K_j(h)}|P(s)|^{1-1/d} = \sum_{s\in\Xi_j}|P(s)|^{1-1/d}|\{h\in H_i;\ h\cap s\neq\emptyset\}|$$
$$\leq \frac{|H_i|}{\rho^j}\sum_{s\in\Xi_j}|P(s)|^{1-1/d},$$

and, by Hölder's inequality,

$$\sum_{s\in\Xi_j}|P(s)|^{1-1/d} \leq n^{1-1/d}|\Xi_j|^{1/d} = O(n^{1-1/d}\rho^j),$$

so the average value of $\sum_{s\in K_j(h)}|P(s)|^{1-1/d}$ is $O(n^{1-1/d})$. A similar but simpler calculation shows that the average value of $\sum_{s\in K_k(h)}|P(s)|$ is $O(n/r^{1/d})$. These bounds imply the lemma. □

We define $\mathcal{G}_i$ (the "good" guarding sets for the ith component) as the ones whose weight exceeds the average at most twice; thus $\mathcal{G}_i$ contains at least half of the elements of $\mathcal{G}_i$. We set $\mathcal{G}_{i+1} = \mathcal{G}_i\backslash\mathcal{G}_i$. The whole construction finishes when $\mathcal{G}_{i+1} = \emptyset$. For every hyperplane h from the union of the guarding sets of $\mathcal{G}_i$, the bound on the weight function implies

$$|K_j(h)| = 4^{-(k-j)}O(r^{1-1/d}), \tag{5.2}$$

$$\sum_{s\in K_k(h)}|P(s)| = O(n/r^{1/d}), \tag{5.3}$$

$$\sum_{s\in K(h)}|P(s)|^{1-1/d} = O(n^{1-1/d}\log r). \tag{5.4}$$

Now let γ be a half-space with a bounding hyperplane h, whose guarding set G belongs to $\mathcal{G}_i$. We define the decomposition of $P\cap\gamma$. The inner sets are those corresponding to all active simplices completely contained in γ, whose father is not contained in γ. The remainder sets are the ones corresponding to all leaf simplices intersected by h. It is straightforward to check that all sets in this decomposition are disjoint and that the whole $P\cap\gamma$ is covered.

We now want to establish the bounds concerning the decomposition. It is sufficient to show the following three estimates for every hyperplane h (recall that

$L(h)$ denotes the set of leaf simplices intersected by h):

$$|K(h)| = O(r^{1-1/d}), \tag{5.5}$$

$$\sum_{s \in L(h)} |P(s)| = O\left(\frac{n}{r^{1/d}}\right), \tag{5.6}$$

$$\sum_{s \in K(h)} |P(s)|^{1-1/d} = O(n^{1-1/d} \log r). \tag{5.7}$$

Indeed, from (5.5) and the fact that each simplex has a bounded number of sons, we infer the bound for the number of inner sets in the decomposition and also a bound for the running time of the algorithm for finding the decomposition. Then (5.5) and (5.6) imply the claim about the number of remainder sets in the decomposition, and (5.7) implies the bound on $\sum_{C \in \mathscr{C}} |C|^{1-1/d}$ in the theorem.

The proof of (5.5) and (5.6) from (5.2) and (5.3) is almost identical to the proof of (4.2), (4.3) from Lemma 4.2, and we leave it to the reader. As for (5.7), this condition holds for every $h \in H_i$ (it is just (5.4)). Let h be a general hyperplane with a guarding set $G \in \mathscr{G}_i$. If a simplex $s \in K(h)$ intersects some hyperplane of G, then its contribution to the estimated sum is already accounted for. Hence it suffices to consider the contribution of the set K_1 of simplices of $K(h)$ contained in the zone of h. Each of the $n/r^{1/d}$ points in that zone is contained in $O(\log r)$ simplices of $K(h)$, so $\sum_{s \in K_1} |P(s)| = O(n \log r / r^{1/d})$. Using $|K_1| \leq |K(h)| = O(r^{1-1/d})$ and Hölder's inequality, we get $\sum_{s \in K_1} |P(s)|^{1-1/d} = O(n^{1-1/d}(\log r)^{1-1/d})$. This gives (5.7).

We have almost finished the proof of Theorem 5.2. The claim about polynomiality of the preprocessing time is obvious from the construction. Finally, all the steps of the construction can be executed in time which only depends on r (polynomially), except for the computation of the guarding sets and the location of the points in the simplices, but these two steps can both be performed in total time $O(n \log^2 r)$ (spending $O(n \log r)$ time for each of the $O(\log r)$ components of the decomposition scheme). For a sufficiently small r, the $n \log r$ term dominates the polynomial in r. This finishes the proof of Theorem 5.2. □

If we did not care about the preprocessing time, we could set $r = n$ in Theorem 5.2, and use it directly as a device for building multilevel range searching structures with almost linear space. We can improve the preprocessing (at the expense of space) by composing partial half-space decomposition schemes with smaller values of r, in a constant number of levels.

Corollary 5.4. *There exists a half-space decomposition scheme $\mathscr{S}_0$ such that, for an n-point set P, the data structure $\mathscr{S}_0(P)$ has the following parameters:*

The sum of sizes of the canonical sets is $O(n \log^{O(1)} n)$. If $\mathscr{C}$ is the collection of canonical sets used in the decomposition of $P \cap \gamma$, then $|\mathscr{C}| = O(n^{1-1/d})$ and

$$\sum_{C \in \mathscr{C}} |C|^{1-1/d} = O(n^{1-1/d}(\log n)^{O(1)}).$$

The decomposition can be found in $O(n^{1-1/d})$ time. The structure can be built in time $O(n^{1+\delta})$.

Proof. First we note that by setting $r = n$, we can indeed obtain a half-space decomposition scheme from Theorem 5.2 (the remainder sets are singletons, so we can add the appropriate ones to the decompositon as inner sets).

Now let $\mathscr{S}$ be the (abstract) partial half-space decomposition scheme from Theorem 5.2, with a particular choice of the parameter r: for an n point set P, we choose $r = n^{\alpha}$, where α is as in Theorem 5.2. We set $\mathscr{D}_1 = \mathscr{S}$, and we define $\mathscr{D}_{i+1}$ as the composition of $\mathscr{S}$ with $\mathscr{D}_i$ via remainder sets. We observe that $\mathscr{D}_k(P)$ is a partial half-space decomposition scheme whose remainder sets have size at most $n^{(1-\alpha)^k}$; hence by choosing a large enough constant for k, the remainder sets of $\mathscr{D}_k$ have size at most $n^{\delta'}$ for a prescribed constant $\delta' > 0$. We finally compose this $\mathscr{D}_k$ via remainder sets with the half-space decomposition scheme mentioned at the beginning of this proof, obtaining a half-space decomposition scheme. All the parameters of this decomposition scheme correspond to the corollary, as may be verified by an elementary calculation. □

5.4. *Example of a Multilevel Data Structure*

For illustration let us use the above result for constructing a simple multilevel data structure. Given a set S of n segments in the plane, we want to preprocess it in such a way that, for a query line λ, we can count the number of segments intersecting λ quickly. Suppose that all endpoints of the segments are distinct, and that the segments are nonvertical.

Let L denote the set of left endpoints of the segments of S, and, for $x \in L$, let $e(x)$ denote the right endpoint of the segment with the left endpoint x. For a suitable half-plane decomposition scheme $\mathscr{D}$, we first construct the data structure $\mathscr{D}(L)$, and, for each of its canonical subset $C \subseteq L$, we define the secondary structure $S_{\mathscr{D},L}(C)$ as a suitable half-plane range counting structure for the set $e(C) = \{e(x); x \in C\}$. In this way we obtain the desired data structure. A query is answered as follows. For a query line λ, we let γ be one of the half-planes defined by λ. Using $\mathscr{D}(L)$ we find a decomposition of $L \cap \gamma$ into canonical sets. For every canonical set C in this decomposition, we use the secondary structure to count the number of points of $e(C)$ contained in the half-plane complementary to γ. The sum of these counts over the canonical decomposition gives the number of segments with the left endpoint inside γ and the right endpoint outside γ. The other possible case (with left and right exchanged) is handled symmetrically.

Let us now consider a specific instance of the above construction. For the half-plane range counting structure, we take one with $O(n)$ space and $O(\sqrt{n})$ query time (Theorem 4.1). For the half-plane decomposition scheme, let us choose the scheme obtained from Theorem 5.2 by setting $n = r$. The space occupied by the resulting data structure is proportional to the sum of sizes of the canonical sets, which is $O(n \log^2 n)$. The query time is proportional to the sum of square roots

of the sizes of the sets in the decomposition, which is $O(\sqrt{n}\log n)$. With these parameters, the data structure may require a rather large preprocessing time; it can be traded for query time using Corollary 5.4.

Note that in the above construction, we use essentially a composition of a half-plane decomposition scheme with a half-plane range counting data structure, only this composition is formally more complicated (it uses a bijection linking the top and bottom level structures together).

The results of this section can also be applied in a similar manner in more complicated examples, as are those in [AS]. Each new level of such structures now brings only a polylogarithmic overhead in space and query time. At present we do not consider it useful to extend our formal framework to enable a sufficiently general precise statement of this result.

5.5. *Reducing the Preprocessing Time for Linear Space Range Searching*

In order to show that the preprocessing time in Theorem 4.1 can be reduced to $O(n^{1+\delta})$ (from the polynomial bound derived in Section 4), we use one more type of a partial simplex decomposition scheme, whose existence directly follows from the results of [M4]:

Lemma 5.5. *There exists a partial simplex decomposition scheme $\mathscr{T}$, such that, for an n-point set P and a parameter $r \le n^{1-\delta}$ for some fixed $\delta > 0$ (which can be chosen), the data structure $\mathscr{T}(P)$ has the following parameters: There are $O(r)$ inner sets, and $O(r)$ remainder sets of size at most n/r each. The decomposition of $P \cap \sigma$ for every simplex σ consists of $O(r^{1-1/d})$ inner and remainder sets, and it can be found in $O(r^{1-1/d})$ time. The structure can be built in time $O(n \log r)$.*

Let us remark that the decomposition scheme in this lemma has better parameters in some respects than that of Theorem 5.2, but it does not say anything about the distribution of sizes of the inner sets in the decomposition, and it can only be built for r not too close to n.

We use a standard trick to reduce the preprocessing in Theorem 4.1 to $O(n^{1+\delta})$: we consider the partial simplex decomposition scheme $\mathscr{T}$ from the above lemma, with $r = n^{1-\delta'}$ for a suitable $\delta' > 0$. We compose it via remainder sets with the simplex range searching structure constructed in the proof of Theorem 4.1 (in other words, for every remainder set C of the data structure $\mathscr{T}(P)$ we build that simplex range searching structure and store it as the secondary data structure $S_{\mathscr{T},P}(C)$). Finally, we define the secondary structures for the inner sets of $\mathscr{T}(P)$ as their weights, which yields a simplex range searching data structure. Its space and query-time requirements remain asymptotically the same as before, but the preprocessing time is only $O(n^{1+\delta})$.

6. Simplex Range Searching: Tradeoff

We begin by constructing a simplex range searching data structure with a polylogarithmic query time.

Theorem 6.1. *Let P be an n-point set in E^d. The range searching problem for P with the ranges being intersections of p half-spaces, $1 \le p \le d+1$, can be solved with space $O(n^d/\log^{d-p+1} n)$, query time $O(\log^p n)$, and preprocessing time*

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

Proof. We proceed by induction on p. For $p = 1$, we consider the (abstract) partial half-space decomposition scheme $\mathscr{L}$ from Theorem 5.1, and we set the parameter r to $n/\log^{d/(d-1)} n$; hence the size of the remainder sets will be at most $\log^{d/(d-1)} n$. For every inner set, we store its weight as the secondary structure, and, for every remainder set C, we define the secondary structure as the range searching structure from Theorem 4.1 (this is a composition via remainder sets in the terminology of Section 5). We obtain a half-space range searching structure occupying space $O(nr^{d-1})$. When answering a query, we spend $O(\log r) = O(\log n)$ time by computing the decomposition corresponding to the scheme $\mathscr{L}$, and another

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

time by solving the subproblem for the single remainder set in that decomposition. Finally, the preprocessing will be $O(nr^{d-1} + r^d(n/r)^{1+\delta'}) = O((n/\log n)^d \log^\delta n)$.

For $p \ge 1$, we assume that a range searching structure $\mathscr{R}_p$ with parameters as in the theorem has already been constructed for p. In order to constrct $\mathscr{R}_{p+1}$, we again take $\mathscr{L}$ with the same r as above and define the secondary structures for the remainder sets in the same way, but for the inner sets we use $\mathscr{R}_p$ as the secondary structures (composition via inner sets). It remains to estimate the performance of $\mathscr{R}_p$. Using the information from Theorem 5.1 about the distribution of sizes of the inner sets of $\mathscr{L}$, we get that the storage is

$$\sum_{j=0}^{k} \frac{\rho^{dj}(n/\rho^j)^d}{\log^{d-p+2}(n/\rho^j)} = O\left(\frac{n^d}{\log^{d-p+1} n}\right),$$

the preprocessing time is again a factor of $\log^\delta n$ bigger, and the query time is $O(\log^p n)$. □

Now we are ready to establish the general tradeoff:

Theorem 6.2. *Let P be an n-point set in E^d and let m be a parameter, $n \le m \le n^d$. The range searching problem with the ranges being intersections of p half-spaces,*

$1 \le p \le d+1$, *can be solved with space* $O(m)$, *query time*

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

and preprocessing time $O(n^{1+\delta} + m(\log n)^{\delta})$.

Proof. First let us consider the case $m > n^{1+\delta}$. We take the partial simplex decomposition scheme $\mathscr{T}$ as in Lemma 5.5, we define the secondary structures for the inner sets as the weight of these sets, and we let the secondary structures for the remainder sets be the range searching structures from Theorem 6.1. The space occupied by the resulting range searching structure can be written as

$$O(r(n/r)^d/\log^{d-p+1}(n/r)),$$

and since this should be equal to m, we get $r^{d-1} = n^d/(m \log^{d-p+1}(n/r))$. The query time then will be $O(r^{1-1/d} \log^p(n/r))$. Expressing this in terms of n and m, we obtain the claimed bound. The preprocessing time bound is straightforward.

It remains to handle the case when m is close to n, where r in the above construction would get too close to n to apply Lemma 5.5. Here we build the simplex range searching structure in the same manner as in the proof of Theorem 6.1, only we choose a suitably smaller value of r, namely, the one giving space m, i.e., satisfying $nr^{d-1} = m$. We omit the calculations for this case. □

7. Conclusion

In various results of this paper we have approached the known or suspected lower bounds for the considered problems more closely than the previously known solutions did. However, certainly not all of the presented solutions can be regarded as final or completely satisfactory. We point out some open problems.

For Hopcroft's problem, the main challenge is to establish a lower bound, hopefully $\Omega(n^{4/3})$. As for the upper bound, the $2^{O(\log^* n)}$ factor originates in the following manner: we are unable to solve the problem in a constant number of stages with the present method, essentialy because a nonconstant time is spent for location of every point in the cutting. Every stage contributes a constant multiplicative factor to the "excess" in the number of subproblems. This is because we lack some mechanism to control this, similar to Chazelle's method (he can use the number of intersections of the lines as the control device, but we flip the roles of lines and points at every stage, and such a control device is missing in this situation).

In the range searching area, one problem is to improve the preprocessing times, either for the linear-space simplex range searching structure or for the small space decomposition scheme. However, there seems to be a deeper issue here.

In our linear-space solution to the simplex range searching problem, we used tree-like hierarchies of simplices with a constant branching degree, but with a global control over the number of simplices intersected by a hyperplane. This data structure has two shortcomings (which do not matter in the simplex range searching problem): we cannot (so far) make the hierarchy a single tree (it has about $n^{1/d}$ roots which are rather unrelated to each other), and while we can achieve an optimal number of simplices crossed by a hyperplane (crossing number) in the bottommost level and we have some control over the higher levels, we have not enforced an optimal crossing number for these levels. Both these features prevent a successful application of this construction in multilevel data structures, and we had to use another method (basically that of [CSW]) for this purpose. This method gives a nice hierarchy and almost optimal decompositions, but only for half of the guarding hyperplanes. This means that in fact a logarithmic number of data structures are needed, and what is worse, it does not directly give a simplex decomposition scheme. Again, rather similarly to Hopcroft's problem, it seems that in order to obtain a simplex decomposition scheme with optimal parameters, some new global invariant (playing a role similar to the number of intersections in Chazelle's cutting construction) to control the quality of the decomposition scheme at all levels is necessary.

All the above vague statements have a single goal—to point out that although the simplex range searching problem and related questions may look almost completely solved, a really satisfactory solution may still await discovery.

References

[A] P. K. Agarwal. Partitioning arrangements of lines I: An efficient deterministic algorithm. *Discrete & Computational Geometry*, 5:449–483, 1990.

[AS] P. K. Agarwal and M. Sharir. Applications of a new partitioning scheme. In *Proc. 2nd Workshop on Algorithms and Data Structures*. Lecture notes in Computer Science, vol. 519. Springer-Verlag, Berlin, 1991, pp. 379–391.

[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. *Discrete & Computational Geometry*, 9:145–158, 1993.

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

[CSW] B. Chazelle, M. Sharir, and E. Welzl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. *Algorithmica*, 8:407–429, 1992.

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

[DE] D. Dobkin and H. Edelsbrunner. Space searching for intersecting objects. *Journal of Algorithms*, 8:348–361, 1987.

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

[EGH+] H. Edelsbrunner, L. Guibas, J. Herschberger, R. Seidel, M. Sharir, J. Snoeyink, and E. Welzl. Implicitly representing arrangements of lines or segments. *Discrete & Computational Geometry*, 4:433–466, 1989.

[Ep] D. Eppstein. Private communication, 1992.

[L] L. L. Larmore. An optimal algorithm with unknown time complexity for convex matrix searching. *Information Processing Letters*, 36:147–151, 1990.

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

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

[M3] J. Matoušek. Spanning trees with low crossing number. *Informatique Théorique et applications*, 6(25):103–123, 1991.

[M4] J. Matoušek. Efficeint partition trees. *Discrete & Computational Geometry*, 8:315–334, 1992.

[OSS] M. Overmars, H. Schipper, and M. Sharir. Storing line segments in partition trees. *BIT*, 30:385–403, 1990.

[PS] F. Preparata and M. I. Shamos. *Computational Geometry—An Introduction*. Springer-Verlag, New York, 1985.

Received April 1, 1992, and in revised form January 10, 1993.