A Deterministic Algorithm for the Three-Dimensional Diameter Problem*

Jiří Matoušek
Katedra aplikované matematiky, Universita Karlova,
Malostranské nám. 25, 118 00 Praha 1, Czechoslovakia and
Institut für Informatik, Freie Universität Berlin
Arnimallee 2-6, W 1000 Berlin 33, Germany

Otfried Schwarzkopf
Vakgroep Informatica, Universiteit Utrecht,
Postbus 80.089, 3508 TB Utrecht, the Netherlands

Abstract

We give a deterministic algorithm for computing the diameter of an n point set in three dimensions with $O(n \log^c n)$ running time, c a constant.

1 Introduction

We consider the following problem: Given an n-point set P in three-dimensional space, compute its diameter $\mathrm{diam}(P)$, defined as the maximum distance between a pair of points in P. This problem was solved by Clarkson and Shor [CS89] by a randomized algorithm with optimal expected running time $O(n \log n)$. However, it seems quite difficult to construct a deterministic algorithm with a comparable asymptotic efficiency.

Chazelle et al. [CEGS92] applied Megiddo's *parametric search* [Meg83] in connection with the Clarkson-Shor algorithm and further techniques and obtained a deterministic $O(n^{1+\varepsilon})$ algorithm, where $\varepsilon > 0$ is an arbitrarily small constant. In this paper, we improve this running time to $O(n \operatorname{polylog} n)$ (we use $\operatorname{polylog} n$ as a generic notation for a function of the form $(\log n)^c$, $c > 0$ a constant).

Our basic approach is the same as that of [CEGS92]. Also the basic scheme for achieving the improvement has appeared in previous works: While Chazelle et al. divide the problem into a constant number of subproblems, we refine the "divide" step and use a larger number of subproblems (n^α for a small but positive constant α). With this change, some computations which are trivially done for a constant number of subprob ems become more demanding.

In order to apply parametric search in this situation, it is helpful to change the usual approach to parametric search. In many presentations and applications of the parametric search technique, the need for designing a *parallel* algorithm is emphasized. What is really required, however, is a sequential algorithm, only with comparisons involving the value of the unknown parameter collected into "batches" of independent comparisons. This has already been pointed out by various authors, and we consider it an important part of the parametric search method, which is very easy to overlook. It is this slight shift of attention which enables us to solve problems which we could not overcome before.

Theorem 1 *Given a set P of n points in three-dimensional space, the diameter of P can be determined by a deterministic algorithm in time $O(n \operatorname{polylog} n)$.*

The algorithm is too complicated and has too large constants hidden in the asymptotic notation to be of any practical value; it is only a theoretical contribution to the problem of deterministic asymptotic complexity of the diameter problem. And, one must ask, are all these heavy tools really necessary, or are we just blinded by the various techniques available to overlook a simple elementary solution?

2 Diameter and Parametric Search

Let P be a given set of n points in 3-space. A quadratic algorithm for computing $\mathrm{diam}(P)$ is trivial. Using randomization, Clarkson and Shor [CS89] have obtained an algorithm whose expected running time (over the

*This research was supported by the Netherlands' Organization for Scientific Research (NWO) and partially by the ESPRIT Basic Research Action No. 7141 (project ALCOM II). J. M. acknowledges support by Humboldt Research Fellowship. Part of this research was done while he visited Utrecht University.

25th ACM STOC '93-5/93/CA,USA

internal randomizations that it performs) is $O(n \log n)$. They first transform the problem to the following *ball problem*:

> Given n unit-balls and n points in 3-space, determine whether any point lies outside the common intersection of the balls.

This problem is then solved using a randomized incremental algorithm. In their algorithm, both the transformation to the ball problem, and the solution of the latter involve randomization.

Chazelle et al. [CEGS92] considered the problem of making this algorithm deterministic. Fortunately, the transformation to the ball problem can be made deterministic by applying Megiddo's parametric search [Meg83]. Unfortunately, this transformation adds some additional requirement on the solution of the ball problem, and Chazelle et al. [CEGS92] could only give a deterministic solution to the latter with running time $O(n^{1+\varepsilon})$, for an arbitrarily small $\varepsilon > 0$. As stated in the introduction, we will be able to get closer to the complexity of the randomized algorithm. Matching the randomized complexity remains as a challenge for future research.

Let us start by describing the application of parametric search (see e.g., Megiddo's original work [Meg83] or the paper of Chazelle et al. [CEGS92] for a more detailed explanation of the parametric search paradigm): We assume that we are given an algorithm $\mathcal{A}$ that solves the following "fixed-size" problem: Given n points P, and a parameter $\delta > 0$, determine whether $\mathrm{diam}(P)$ is less than, equal to, or greater than δ. (Observe that this is a restricted version of the ball problem mentioned above, since we have to determine whether any point in P lies outside the intersection of the balls of radius δ around the points in P).

The algorithm $\mathcal{A}$ plays a dual role in the overall algorithm for diameter computation. First, it serves as a *oracle* which, given a specific value δ^* of δ, decides whether $\delta^* < \mathrm{diam}(P)$, $\delta^* = \mathrm{diam}(P)$ or $\delta^* > \mathrm{diam}(P)$. Second, it serves as so-called *generic algorithm* (let us remark that in general, these roles can be played by different algorithms in parametric search applications, but for us it is suitable to use $\mathcal{A}$ in both roles). For this second role, we require that $\mathcal{A}$ uses the information about the parameter δ in a restricted manner. That is, we assume that δ is not explicitly available to $\mathcal{A}$, but rather that $\mathcal{A}$ has access to an oracle that allows it to test whether δ is less than or equal to some value $\delta^* > 0$. Note that this allows the algorithm $\mathcal{A}$ to test the sign of arbitrary polynomials of constant-bounded degree in δ (it finds the roots of the polynomial, which is considered as a constant-time operation in the model of computation we use, and then it locates the position of δ among them). When describing the algorithm $\mathcal{A}$ in the sequel, we will usually first explain its oracle function, with a specific value of δ, and then we will comment on the way the use of the actual value of δ is restricted to a small number of oracle calls, in the generic algorithm role of $\mathcal{A}$.

Let us denote the number of calls to the oracle made by $\mathcal{A}$ for an input of size n in the worst case by $Q(n)$, and the running time of $\mathcal{A}$ (with each oracle call charged by one time unit only) by $T(n)$. This makes sense for the case of explicitly given δ, when an oracle call is reduced to a single comparison of real numbers.

We are now ready to apply parametric search to the diameter problem. We run our algorithm $\mathcal{A}$ in the generic role on the set P and with parameter $\delta = \mathrm{diam}(P)$. This definition of δ may seem strange, since $\mathrm{diam}(P)$ is exactly what we intend to compute in the end, but recall that $\mathcal{A}$ cannot access δ directly. The only thing it can do is to ask the oracle whether $\delta \leqslant \delta^*$ for some specific values $\delta^* > 0$; these values are computed from P and the previous answers of the oracle. Since we can implement the oracle by $\mathcal{A}$ in the oracle role, it follows that the total running time of the *generic* $\mathcal{A}$ including the oracle calls is

$$T(n) + Q(n)T(n) = O(T(n)Q(n)).$$

Finally, the generic algorithm $\mathcal{A}$ will stop and answer "EQUAL". This is not surprising, since this is the appropriate answer for $\delta = \mathrm{diam}(P)$. However, it is not difficult to verify that in order to be able to answer "EQUAL", one of the calls to the oracle must also have returned "EQUAL". By keeping track of the values δ^* used in the calls to the oracle, we find $\mathrm{diam}(P)$. (Actually, we can stop the whole computation as soon as a call to the oracle returns with answer "EQUAL").

We summarize our exposition of parametric search for the diameter problem in a lemma.

Lemma 2 *Let $\mathcal{A}$ be a deterministic algorithm for the ball problem on n points which uses at most $Q(n)$ calls to the oracle describing the parameter δ and with running time bounded by $T(n)$. Then the diameter of a set of n points in three dimensions can be computed by a deterministic algorithm in time $O(T(n)Q(n))$.*

3 A deterministic algorithm for the ball problem

We want to give an algorithm for the following (slightly generalized) problem: Given a set P of n points, a set Q of m points, and a parameter $\delta > 0$ (in the form of an oracle), determine whether $Q \subset K := \bigcap_{p \in P} B(p, \delta)$, where $B(p, \rho)$ denotes the ball around p with radius ρ.

Note that the intersection set K is a convex body, bounded by spherical facets (supported by a ball $B(p,\delta)$ for some $p \in P$). The facets are bounded by circular edges (circular arcs in three dimensions, arising as the intersection of the boundaries of two balls of radius δ). Since all balls have equal radius, the combinatorial complexity of K is only linear in n (see [CS89]). Furthermore, the facets of K are "convex" in the following sense [CS89]: Let f be a facet of K, x and y be points in the facet f. Then the shorter one of the two great circle arcs connecting x and y lies completely in f (note that x and y cannot be antipodes of each other, unless there is only one ball).

Clarkson and Shor's randomized algorithm [CS89] works as follows: It computes the body K by a randomized incremental construction, picks some point o in the interior of K, and centrally projects the edges of K from o on a unit-sphere around o. This results in a planar map for which a planar point location structure is constructed. It then locates every point of Q using this structure. On the top level of the algorithm, one has $P = Q$; the above generalized version of the ball problem will be a typical subproblem appearing in a recursion in our algorithm.

The part which appears difficult to do deterministically is the construction of K. Instead of an incremental construction, we will use a divide-and-conquer approach. It will also be convenient to combine the recursive construction of K with the location of the points of Q in K.

In the first step, we will identify a subset R of P. We will compute $K_R := \bigcap_{p\in R} B(p,\delta)$. The surface of K_R consists of vertices, edges (circular arcs) and facets (portions of unit spheres bounded by circular arcs). The total number of these features is $O(|R|)$. We form a decomposition $\mathcal{T}(R)$ of K_R into $O(|R|)$ cells, each cell $\sigma \in \mathcal{T}(R)$ being described by a constant number of real parameters. We call these cells (for a lack of a better name) *bricks*. A natural way to produce such a decomposition would be as follows: for every facet f of K_R, choose a point $v_f \in f$, and subdivide f into "spherical triangles" by connecting v_f to the vertices of f by great circle arcs. Then choose a point o in the interior of K_R and for every spherical triangle τ on the surface of K_R, connect each its point with o by a segment, forming a brick ("spherical tetrahedron") corresponding to τ. This is essentially the way which is used (and works) in Chazelle et al's method [CEGS92] (although their definition contains some formal imprecisions). It turns out, however, that this decomposition method is not good enough for our purposes, and we have to choose another one, analogous to vertical decomposition in the plane. The precise definition will be given in Section 4, for time being we suppose that we have the subdivision $\mathcal{T}(R)$ of K_R into $O(|R|)$ bricks of some kind.

For every brick $\sigma \in \mathcal{T}(R)$ we compute the set Q_σ of points in Q lying in σ, and also the set P_σ of $p \in P$ such that σ is not completely contained in $B(p,\delta)$. If any point of Q turns out to lie outside K_R, we are done. Otherwise, we proceed by solving the subproblems defined by the sets P_σ, Q_σ and by δ for every $\sigma \in \mathcal{T}(R)$. When the cardinality of either P_σ or Q_σ drops below some constant, we solve the problem naively.

Thus, we obtain the following recursion for the time $T(n,m)$ necessary to solve the ball problem.

$$T(n,m) = S(n) + S'(n,m) + \sum_{\sigma\in\mathcal{T}(R)} T(n_\sigma, m_\sigma).$$

Here, $S(n)$ is the time necessary to identify a proper subset R, triangulate it and identify the subsets P_σ for all $\sigma \in \mathcal{T}(R)$. $S'(n,m)$ is the time necessary to identify the subsets Q_σ; and n_σ and m_σ are the cardinalities of P_σ and Q_σ, resp. Note that $\sum_{\sigma\in\mathcal{T}(R)} m_\sigma = m$.

Let $r > 1$ be a parameter bounded by a constant. Using ε-net theory and results from [Mat91a], it is possible to find, in time $S(n) = O(n \log r)$, a subset $R \subset P$ of size $O(r \log r)$ with the property that $n_\sigma \leqslant n/r$ for every $\sigma \in \mathcal{T}(R)$. With these values and $S'(n,m) = O(m)$, the above recurrence solves to $O(m \log n + n^{1+\varepsilon})$, for a constant $\varepsilon > 0$ which can be made arbitrarily small by increasing r. This is essentially the solution by Chazelle et al. [CEGS92].

To get rid of the polynomial factor $O(n^\varepsilon)$, however, we want to find a suitable subset R of larger size, as in the following lemma.

Lemma 3 *There exists a positive constant α, such that for an n point set P in three dimensions, and a parameter $\delta > 0$ given as an oracle, we can identify in time $O(n \log n)$ a subset R of P, satisfying the following conditions:*

(i) $|R| \leqslant r := n^\alpha$,

(ii) *for all $\sigma \in \mathcal{T}(R)$ we have $n_\sigma \leqslant c_1(n/r)\log r$, and*

(iii)
$$\sum_{\sigma\in\mathcal{T}(R)} n_\sigma \leqslant c_2 n,$$

for constants $c_1, c_2 > 0$. Furthermore we can compute the intersection K_R of the balls $B(p,\delta)$ and identify the sets P_σ for all $\sigma \in \mathcal{T}(R)$ within the same time bound. The number of calls to the oracle describing δ is in $O(\log^2 n)$.

Given another m point set Q, we can compute Q_σ for every $\sigma \in \mathcal{T}(R)$ in $O(m \log n)$ time and with $O(\log m \log n)$ additional oracle calls.

This lemma will be proved in the next section. Using this result as the divide step of our algorithm for the ball problem, our recurrence becomes

$$T(n,m) = O((n+m)\log n) + \sum_{\sigma\in\mathcal{T}(R)} T(n_\sigma, m_\sigma)$$

with $\sum_{\sigma\in\mathcal{T}(R)} n_\sigma \leqslant c_2 n$ and $\sum_{\sigma\in\mathcal{T}(R)} m_\sigma = m$. It is straightforward to check that the solution of this recurrence satisfies $T(n,m) \in O((n+m)\,\mathrm{polylog}\, n)$ (the power of the logarithm is determined by the constants α and c_2).

It remains to analyze the total number of oracle calls. We note that the computations in the subproblems are independent of each other and they can thus be executed in a pseudo-parallel fashion. Specifically, the depth of the recursion is $O(\log\log n)$, and every subproblem appearing in the recursion needs at most $O(\log^2 n)$ oracle calls. Thus, the oracle calls in this computation can be grouped into $O(\log^2 n \log\log n)$ rounds of *independent* calls (that is, the values of δ^* for the calls in one round do not depend on the outcome of the other calls in that round). Using binary search, each round can be answered using only $O(\log n)$ actual oracle calls (this is the standard trick to reduce the number of evaluations of the oracle, when the generic algorithm is fully parallel, [Meg83]), resulting in $O(\log^3 n \log\log n)$ calls altogether (this is a crude bound, a more refined analysis is possible). In view of Lemma 2, this proves Theorem 1. It remains to describe the "divide" step in detail.

4 The divide step

In this section we give a proof of Lemma 3. First we establish several auxiliary results.

Canonical decomposition. We will need that the decomposition $\mathcal{T}(R)$ of K_R defined in Section 3 is a *canonical decomposition* in the sense of [CS89, CF90]. Let f be a facet of K_R and $s = B(p_1,\delta)$ its supporting sphere. We define the poles of s as the points of s with the largest and with the smallest z-coordinate, and through every vertex v of the facet f, we pass the great circle through v and the poles[1]. Each such circle intersects f in a connected arc, and these arcs subdivide f into "spherical trapezoids". If f does not contain a pole, then these spherical trapezoids look analogously to trapezoids in the plane; if it does, then a spherical trapezoid has a bowtie shape — its two sides formed by the great circle arcs cross at the pole.

[1] If v happens to be a pole itself, we introduce no great circle for it.

We then pick a point o in the interior of K_R. A clever choice for o is the center of the smallest enclosing ball for the point set P, which can be precomputed in linear time using a deterministic algorithm by Dyer [Dye92]. Note that K_R must contain this point, unless K is empty. In the latter case we are done[2].

We extend all spherical trapezoids on the surface of K_R to bricks by taking the union over all line segments connecting o with a point in the spherical trapezoid. This finishes the definition of the decomposition $\mathcal{T}(R)$ of K_R. It is straightforward to check that this decomposition has $O(|R|)$ bricks.

Let $\mathcal{F}$ denote the set of all bricks that appear in a decomposition $\mathcal{T}(R)$ for some subset $R \subseteq P$.

Lemma 4 *The decomposition $\mathcal{T}(R)$ defined above is a canonical decomposition of K_R in the following sense:*

(i) Every brick σ is defined by at most five points, i.e. σ appears in $\mathcal{T}(U)$ for some subset $U \subseteq P$ with $|U| \leqslant 5$.

(ii) A brick $\sigma \in \mathcal{F}$ appears in $\mathcal{T}(R)$ exactly if the defining points are in R, and no point of P_σ is in R. (Recall that P_σ is the set of points $p \in P$ such that $B(p,\delta)$ does not contain σ.)

Proof: Consider a brick σ appearing in some decomposition $\mathcal{T}(R)$ for $R \subseteq P$. It is completely defined by its spherical trapezoid, which lies in some facet f on the boundary of a sphere $B(p_1,\delta)$. The spherical trapezoid is defined by at most two arcs γ bounding f and by at most two vertices on them. The bounding arcs γ_1, γ_2 are defined by intersections of $B(p_1,\delta)$ with some balls $B(p_2,\delta)$ and $B(p_3,\delta)$ (if there is only one arc we may formally put $p_3 = p_2$). The vertices of the arcs are defined by intersections of γ_1 or γ_2 with some other two balls $B(p_4,\delta)$ and $B(p_5,\delta)$ (again, we may put $p_5 = p_4$ if there is only one defining vertex). We set $U = \{p_1, p_2, \ldots, p_5\}$. Consider now the decomposition $\mathcal{T}(U)$. It is easy to check that σ will appear in this decomposition, so we have proven (i) and can now define σ as $\sigma(\delta, p_1, p_2, \ldots, p_5)$ (note that although $\mathcal{T}(U)$ consists of several bricks, the ordered 5-tuple of points determines a unique σ). To prove (ii), imagine that we start from the decomposition $\mathcal{T}(U)$ and then add the points of $R \setminus U$ one by one. It is easy to check that the brick σ does not disappear from the current decomposition by adding a new point p unless the complement of the ball $B(p,\delta)$ intersects σ. □

[2] Dyer's algorithm is one of sophisticated subroutines in our algorithm, but we can avoid this one rather easily. We formulate the algorithm with its use mainly for simplicity of exposition.

Linearization lemma. On several places of the proof of Lemma 3, we will deal with rather complicated predicates involving nonlinear polynomial inequalities. The main example will be the predicate "is σ contained in $B(p,\delta)$?", where σ is a brick, $p \in P$ is a point and $\delta > 0$ is a real parameter. Using an observation of Yao and Yao [YY85], we can nevertheless deal with such predicates using techniques for "linear" objects (points and hyperplanes), by "lifting" the problem into a suitable space of higher dimensions. A precise formulation is given in the following lemma. Let us call a subset of $\mathbb{R}^d$ a *linear cell*, if it can be expressed as a union of $O(1)$ sets, each of which is an intersection of $O(1)$ halfspaces in $\mathbb{R}^d$.

Lemma 5 *Let $\Pi(x_1, x_2, \ldots, x_k, a_1, a_2, \ldots, a_\ell)$ be a first-order predicate in the theory of real closed fields (one formed from polynomial inequalities using Boolean connectives and quantifiers). Then there exists a constant d (depending on Π) and mappings φ, ψ as follows:*

- *The mapping φ assigns to every k-tuple $x = (x_1, \ldots, x_k)$ a point $\varphi(x) \in \mathbb{R}^d$. The mapping is given by bounded-degree polynomials in $x_1, \ldots, x_k$.*
- *The mapping ψ assigns to every ℓ-tuple $a = (a_1, \ldots, a_\ell)$ a linear cell $\psi(a) \subseteq \mathbb{R}^d$. The functions describing the coefficients in equations of the halfspaces defining $\psi(a)$ are given by bounded-degree polynomials in $a_1, a_2, \ldots, a_\ell$.*
- *For any a, x, $\Pi(x, a)$ holds iff the point $\varphi(x)$ belongs to the linear cell $\psi(a)$.*

Proof: The procedure for obtaining such mappings φ, ψ is based on Yao and Yao's observation and it is discussed in [Mat91]. First, we convert the formula for Π into an equivalent quantifier-free formula (one composed of polynomial inequalities using Boolean connectives) using a quantifier elimination method, see e.g., [Ren92]. This formula can be rewritten as a disjunction of several conjunctions of polynomial inequalities.

We view the polynomials occurring in these inequalities as polynomials in the x_i variables with coefficients being polynomials in the a_i variables. We let M be the set of all monomials $\mu = \mu(x)$ in $x_1, \ldots, x_k$ occurring in these polynomials. Then we set $d = |M|$, we imagine that the coordinates in $\mathbb{R}^d$ are indexed by the monomials of M, and we define the mapping φ as follows: given a k-tuple $x = (x_1, \ldots, x_k)$ of real numbers, the point $\varphi(x) = (y_\mu)_{\mu \in M} \in \mathbb{R}^d$ is defined by $y_\mu = \mu(x_1, \ldots, x_k)$ (the value of the monomial μ evaluated at the given k-tuple x). If we consider one polynomial inequality occurring in our formula, of the form $\sum_{\mu \in M} g_\mu(a)\mu(x) \geqslant 0$ (each g_μ a polynomial in $a_1, \ldots, a_\ell$), it is satisfied for given x, a iff the point $\varphi(x)$ lies in the halfspace $\{(y_\mu)_{\mu \in M} \in \mathbb{R}^d;\ \sum_{\mu \in M} g_\mu(a) y_\mu \geqslant 0\}$. A conjunction of several inequalities then corresponds to a membership of $\varphi(x)$ in an intersection of several halfspaces, and a disjunction of several conjunctions corresponds to a union of several such intersections. The equations of the halfspaces are given by the g_μ polynomials in the a_i variables and they can also be read off from the formula. □

Point location method. As a first application of the above linearization lemma, we will discuss a way the subsets P_σ as in Lemma 3 can be computed efficiently, when we already have the decomposition $\mathcal{T}(R)$ of the body K_R into bricks.

A brick σ can be described by specifying δ and a vector $z_1, \ldots, z_{15}$ of real parameters, expressing the five points defining σ. We write $\sigma = \sigma(\delta, z)$. A point $p \in \mathbb{R}^3$ is given by its three coordinates (u_1, u_2, u_3). Then the predicate "$\sigma \not\subseteq B(p,\delta)$" can be written as a first-order predicate in the variables δ, u_i and z_i (this is somewhat laborious but possible). Using Lemma 5 for this predicate, we get mappings φ, ψ as follows: $\varphi = \varphi(\delta, u_1, u_2, u_3)$ assigns to a value of δ and a point $p = (u_1, u_2, u_3)$ a point in $\mathbb{R}^d$ (for some perhaps large but constant-bounded d), $\psi = \psi(z)$ assigns to a parameter vector z a linear cell in $\mathbb{R}^d$, and $\sigma(\delta, z) \not\subseteq B(p,\delta)$ is equivalent to $\varphi(\delta, u_1, u_2, u_3) \in \psi(z)$.

We use these mappings for computing the sets P_σ (given the set P and the decomposition $\mathcal{T}(R)$) as follows. For every brick $\sigma \in \mathcal{T}(R)$, $\sigma = \sigma(\delta, z)$, we compute the set $\psi(z)$ (note that this is independent of δ). Then we preprocess the arrangement of these sets in $\mathbb{R}^d$ for all $\sigma \in \mathcal{T}(R)$ for fast point location. This can be done by preprocessing the full arrangement of all hyperplanes defining our sets for point location (several methods for point location in hyperplane arrangements are known, see e.g., [Cha92]). This yields a data structure with space and preprocessing time polynomial in $|\mathcal{T}(R)|$, thus, for α small enough, no more than linear in n. The query time will be $O(\log n)$. Then, for every point $p \in P$, we locate the point $\varphi(\delta, p)$ using this data structure, which tells us to which sets P_σ does p belong. The total time will be $O(n \log n)$. As for the oracle calls, each step of a point location query with some point p involves a comparison of the point $\varphi(\delta, p)$ against some (explicitly known) hyperplane, and such a comparison can be decided by a bounded number of oracle calls. Since the n point location queries are independent and can be executed in a pseudo-parallel fashion, the total number of oracle calls is $O(\log^2 n)$.

A quite similar approach can be used to compute the subset Q_σ. Here we need a data structure for location of points in the decomposition $\mathcal{T}(R)$. We apply Lemma 5

on the predicate "$q \in \sigma$" (for a point q and a brick $\sigma = \sigma(\delta, z)$), and we build a point location structure in the same manner as above. The set Q_σ can thus be detected in $O(m \log n)$ time with $O(\log m \log n)$ oracle calls. Let us remark that an alternative — and perhaps more natural — way is to use a planar point location structure for point location in $\mathcal{T}(R)$, and then discuss how such a data structure is "parameterized" with δ, so that the preprocessing and point location queries only need a small number of oracle calls.

Computing a good sample in polynomial time. It remains to describe how to compute the "good" sample R. First we prove a weaker statement, namely that a suitable R can be found in polynomial time.

Lemma 6 *Let P be a set of n points, $r < n$ be a given parameter and let the parameter $\delta > 0$ be given as an oracle. We can identify in polynomial deterministic time a subset R of P such that*

(i) $|R| \leqslant r$,

(ii) *for all $\sigma \in \mathcal{T}(R)$ we have $n_\sigma \leqslant c_1(n/r)\log r$, and*

(iii)

$$\sum_{\sigma \in \mathcal{T}(R)} n_\sigma \leqslant c_2 n$$

for constants $c_1, c_2 > 0$. The number of calls to the oracle describing δ is in $O(\log n)$. Within the same bounds we can compute the combinatorial structure of K_R.

Proof: We appeal to the usual randomization results ([CS89]) to conclude that when an r-element subset R is chosen randomly from P, the bounds in the lemma will hold with a positive probability. Applying derandomization techniques (i.e. the Raghavan-Spencer method) as in [CF90, Mat91a], we can then find a subset R that fulfills the requirements of the lemma in polynomial time.

To be more specific, let us use the result from [CF90]. To this end, we consider the hypergraph $(P, \mathcal{S})$, where $\mathcal{S}$ is the set of all subsets S of P which can be defined as

$$S = \{p \in P;\ \sigma \not\subseteq B(p, \delta)\},$$

for some brick σ which occurs in the canonical decomposition $\mathcal{T}(R)$ for some subset $R \subseteq P$. Lemma 4 implies that the number of such σ is only polynomial in $|P|$ (since each one is determined by an at most five-element subset of P). Given this hypergraph (explicitly), [CF90] shows that a sample R satisfying (i)–(iii) can be computed in polynomial time. We thus only need to compute the hypergraph $(P, \mathcal{S})$ in polynomial time and with $O(\log n)$ oracle calls. This is not difficult, as we do not care for the exponent of the polynomial bounding the running time. Namely, we list all at most five-element subsets U of P, and for every such U we determine the bricks in the canonical decomposition corresponding to it. For every such brick, we then compute the subset of points of P defined by it. All the comparisons involving δ are independent and can be done in a single round, thus $O(\log n)$ oracle calls suffice to resolve all of them. The rest of the calculations only deals with the hypergraph and requires no more information about δ.

The combinatorial structure of K_R and its decomposition $\mathcal{T}(R)$ can be determined by one more round of polynomially many (in r) independent oracle calls. This proves Lemma 6. □

Computing an approximation. Now that we can compute a suitable sample in polynomial time, we first compute a sufficiently small subset A of P that 'approximates" P so well that it will be sufficient to apply Lemma 6 to A instead of P. This trick has been used in several previous works, see [Mat91a].

We consider a set system $(P, \mathcal{S})$, where $\mathcal{S}$ is defined similarly as in the proof of Lemma 6; a set $S \subseteq P$ belongs to $\mathcal{S}$ if it can be expressed as the subset of points of P whose balls of radius ρ do not completely contain a certain brick σ. The difference to the above proof is that we consider all radii ρ simultaneously, instead of a single value $\rho = \delta$. Hence the set system $(P, \mathcal{S})$ does not depend on δ.

A subset $A \subseteq P$ is called a $(1/r)$-approximation for $(P, \mathcal{S})$, if for any $S \in \mathcal{S}$ we have

$$\left| \frac{|A \cap S|}{|A|} - \frac{|S|}{|P|} \right| \leqslant \frac{1}{r}.$$

Suppose that A is a $(1/r)$-approximation for $(P\ \mathcal{S})$, where $r := n^\alpha$ is as in Lemma 3. Let $R \subseteq A$ be a subset with at most r elements satisfying the conditions (ii),(iii) of Lemma 3 with P replaced by A (and for some value of δ), that is, $|A_\sigma| \leqslant c_1(|A|/r)\log r$ for every $\sigma \in \mathcal{T}(R)$ and $\sum_{\sigma \in \mathcal{T}(R)} |A_\sigma| \leqslant c_2|A|$, c_1, c_2 constants. For any brick σ, we have $P_\sigma \in \mathcal{S}$ (by the definition of the set system $(P, \mathcal{S})$), and $A_\sigma = A \cap P_\sigma$. It is then straightforward to verify, using the definition of a $(1/r)$-approximation, that the subset $R \subseteq A \subseteq P$ also fulfills the conditions (ii),(iii) of Lemma 3 for P, only with somewhat larger constants c_1, c_2.

To prove Lemma 3, we proceed as follows. First, given P, we compute a $(1/r)$-approximation A for the set system $(P, \mathcal{S})$, with $|A|$ much smaller than $n = |P|$. Then we apply Lemma 6 with A in the role of P, and we find a good sample R. By the above considerations, this R will also be good with respect to P. However, since A is small enough, we can afford to spend time bounded

by a polynomial in $|A|$. The proof of Lemma 3 is finished by computing the decomposition $\mathcal{T}(R)$ (all the combinatorial information needed for this can be gained by a single round of independent oracle calls, thus by $O(\log n)$ actual oracle calls) and detecting the subsets P_σ and Q_σ for every $\sigma \in \mathcal{T}(R)$; this has already been discussed above.

It remains to describe how the $(1/r)$-approximation A is computed. Here we again apply the "linearization" method, Lemma 5. Our predicate will be "$\sigma \not\subseteq B(p,\rho)$", $\sigma = \sigma(\rho, z)$ a brick and p a point. This time, however, we will regard only the coordinates of the point p in the role of the x_i variables in Lemma 5, both z and ρ will be regarded as the a_i variables. We thus obtain a mapping φ assigning to a point $p \in \mathbb{R}^3$ a point $\varphi(p) \in \mathbb{R}^d$ (for some constant d) and a mapping ψ assigning to every z and ρ a linear cell in $\mathbb{R}^d$, in such a way that for any p, ρ, z, $\sigma(\rho, z) \not\subseteq B(p,\rho)$ iff $\varphi(p) \in \psi(z,\rho)$. Returning to our set system $(P, \mathcal{S})$, we see that every $S \in \mathcal{S}$ can be expressed as those points of P whose φ-image lies in a certain linear cell in $\mathbb{R}^d$. Since every linear cell can be written as a disjoint union of at most C simplices (C a suitable constant), it suffices to compute a $(1/Cr)$-approximation for the set $\{\varphi(p);\ p \in P\}$ with all its subsets definable by simplices. For this last problem, we can directly use the results of [Mat92], and we get that if $\alpha > 0$ is a small enough constant, our $(1/r)$-approximation A of size $O(r^2 \log r)$ can be computed in $O(n \log n)$ time. This calculation does not involve the parameter δ at all. This finishes the proof of Lemma 3. □

References

[Cha92] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete & Computational Geometry*, 1992. To appear. Preliminary version: *Proc. 32nd IEEE Symposium on Foundations of Computer Science* (1991).

[CEGS92] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. Diameter, width, closest line pair, and parametric searching. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 120–129, 1992.

[CF90] B. Chazelle and J. Friedman. A deterministic view of random sampling and its use in geometry. *Combinatorica*, 10:229–249, 1990.

[CS89] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry II. *Discrete & Computational Geometry*, 4:387–421, 1989.

[Dye92] M. Dyer. A class of convex programs with applications to computational geometry. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 9–15, 1992.

[Mat91] J. Matoušek. Cutting hyperplane arrangements. *Discrete & Computational Geometry*, 6(5):385–406, 1991.

[Mat91a] J. Matoušek. Approximation and geometric divide and conquer. In *Proc. 23rd Annu. ACM Sympos. Theory Comput.*, pages 505–511, 1991.

[Mat92] J. Matoušek. Efficient partition trees. *Discrete & Computational Geometry* 8:315–334 (1992).

[Meg83] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *Journal of the ACM*, 30:852–865, 1983.

[Ren92] J. Renegar. On the computational complexity and geometry of the first order theory of the reals. *Journal of Symbolic Computation*, 1992.

[YY85] F. F. Yao and A. C. Yao. A general approach to geometric queries. In *Proc. 17. ACM Symposium on Theory of Computing*, pages 163–168, 1985.