Deterministic Algorithms for 3-D Diameter and some 2-D Lower Envelopes *

Edgar A. Ramos †

Abstract

We present a deterministic algorithm for computing the diameter of a set of n points in $\mathbb{R}^3$; its running time $O(n \log n)$ is worst-case optimal. This improves previous deterministic algorithms by Ramos (1997) and Bespamyatnikh (1998), both with running time $O(n \log^2 n)$, and matches the running time of a randomized algorithm by Clarkson and Shor (1989). We also present a deterministic algorithm for computing the lower envelope of n functions of 2 variables, for a class of functions with certain restrictions; if the functions in the class have lower envelope with worst-case complexity $O(f_2(n))$, the running time is $O(f_2(n) \log n)$, in general, and $O(f_2(n))$ when $f_2(n) = \Omega(n^{1+\epsilon})$ for any small fraction $\epsilon > 0$. The algorithms follow a divide-and-conquer approach based on deterministic sampling with the essential feature that planar graph separators are used to group subproblems in order to limit the growth of the total subproblem size.

1 Introduction

We consider the problem of computing the *diameter* of a given set P of n points in 3-d space, that is, the maximum distance between any pair of points in P, and the problem of computing the *lower envelope* of a given set F of n functions of 2 variables, that is, the function defined as the pointwise minimum of the given functions.

†Max-Planck-Institut für Informatik, Saarbrücken, Germany. E-mail: ramos@mpi-sb.mpg.de

Computational Geometry 2000 Hong Kong China

1.1 Previous Work.

3-D Diameter. Yao [32] first showed that the problem can be solved in subquadratic time; he gave a deterministic algorithm with running time $O((n \log n)^{1.8})$. Reporting on the status of the problem in 1985, Preparata and Shamos [27] said: "In spite of its apparent simplicity, the computation of the diameter of a three-dimensional set has been a source of frustration to a great many workers." Then, Clarkson and Shor [12], in their ground-breaking paper on applications of random sampling in computational geometry, gave a simple randomized algorithm which runs in expected time $O(n \log n)$. This is optimal, see *e.g.* [27]. They solve the diameter problem through a reduction to computing the intersection of congruent (equal radius) balls. Since then, it has been a challenge to match that time complexity with a deterministic algorithm: First, Agarwal *et al.* [1] gave an algorithm with running time $O(n^{4/3})$; then Chazelle *et al.* [11] improved the time to $O(n^{1+\epsilon})$, where $\epsilon > 0$ is arbitrary and with the constant in the O notation depending on ϵ; next, Matoušek and Schwarzkopf [25] further improved it to $O(n \log^c n)$ where c is a constant for which they do not give an explicit bound and is possibly very large; then Amato, Goodrich and Ramos [4] improved the time to $O(n \log^3 n)$; finally, Ramos [30] improved it to $O(n \log^2 n)$. All these algorithms use the parametric search technique and deterministic (derandomized) sampling. On a different track, without using derandomization techniques, Ramos gave an algorithm with running time $O(n \log^5 n)$ [29] (this still uses parametric search), and later Bespamyatnikh [8] gave an elementary algorithm also with running time $O(n \log^2 n)$, thus providing an alternative approach simpler than that of [30].

2-D Lower Envelopes. This is an important problem in computational geometry; in particular a 2-d

Voronoi diagram is the vertical projection on the $\mathbb{R}^2$ plane of the lower envelope of the distance functions to the *sites* determining the diagram. There is a considerable amount of work regarding the computation of Voronoi diagrams. Here, we only mention some directly related work. Most algorithms make use of very specific properties and in many cases the only known optimal deterministic algorithms use sampling (derandomization). Let $f_2(n)$ be an upper bound on the worst case complexity for a specific class of functions or Voronoi diagram. When $f_2(n) = \Omega(n^{1+\epsilon})$, the *vertex accounting* approach, introduced by Chazelle to compute optimal hyperplane cuttings [10], leads to an optimal $O(f_2(n))$ worst-case algorithm [30]. When $f_2(n)$ is linear or close to linear, this approch does not seem to work; but then algorithms can take advantage of specific properties, like the fact that each *site* contributes a single face to the diagram. This is the case of Voronoi diagram of segments, or the intersection of congruent balls [4, 7]. However, a general solution, for the case in which $f_2(n)$ is close to linear and the single face property does not hold, does not seem to be know previously. Aggarwal *et al* [3] give an algorithm with complexity $O(n^{2+\epsilon})$, which is almost or essentially optimal in that $O(n^{2+\epsilon})$ is also the best general bound for the complexity of a lower envelope, but we don't know if it can be adapted so that its running time is dependent on the actual complexity $f_2(n)$ of a particular class of functions.

1.2 Our Results

3-D Diameter. We present a new optimal randomized algorithm that can be easily derandomized using current standard techniques, while preserving the optimal running time. Specifically, the derandomization only makes use of the efficient construction of ϵ-*nets* by Matoušek [23]. The algorithm is closely related to that in [11]. A new essential tool, combined with the use of random sampling, is the clustering of subproblems via planar-graph separators. In this, we follow its use by Dehne *et al.* [13] in the context of algorithms for Voronoi diagrams in coarse-grained multicomputers. This allows for a decomposition into subproblems for which, without additional effort, the total size of the subproblems increases only by a small factor. Furthermore, we observe that parametric search is not necessary. The result is a relatively simple divide-and-conquer algorithm, module the ϵ-net computation.

2-D Lower Envelope. We consider a class of functions such that the the lower envelope allows a decomposition into cells such that: (i) the cells, which we call *trapezoids*, are of bounded complexity; (ii) the size of the decomposition is of the same order as the complexity of the lower envelope; (iii) the adjacency graph for the decomposition is of bounded degree;[1] and (iv) whether the set below a trapezoid, which we call a *brick*, intersects a function can be described by a first order predicate of bounded size. Let $f_2(n)$ be an upper bound on the complexity of the lower envelope for the class of functions under consideration. We describe an algorithm that for n functions that runs in time $O(f_2(n)\log n)$, in general, and $O(f_2(n))$ when $f_2(n) = \Omega(n^{1+\epsilon})$, for any small $\epsilon > 0$. The algorithm uses a divide-and-conquer approach based on deterministic sampling. It combines the use of graph separators to limit the growth of the subproblem sizes, as in the diameter algorithm, with the vertex accounting approach of Chazelle [10]. This is a refinement over the randomized algorithms in [13, 14, 20] for Voronoi diagrams with the single face property.

This paper consists of two main sections dedicated to each of the two problems under consideration. In the next subsection, we review the use of planar graph separators for clustering subproblems.

1.3 Clustering via Graph Separators

The goal of the clustering is to minimize the size of the boundary between subproblems and, hence, to minimize the number of elements shared by subproblems. In this, we follow the work of Dehne *et. al.* [13]. First, we recall the planar graph separator result [21] in the particular form we need.

Lemma 1 *Let $\mathcal{G}(V, E)$ be a planar graph with $|V| = \eta$ and vertex degree bounded by a constant. Then V can be partitioned into sets V_1 and V_2 such that $|V_1|, |V_2| \leq \lceil \eta/2 \rceil$ and the number of edges connecting V_1 and V_2 is $O(\sqrt{\eta})$. Furthermore, such sets can be determined in $O(\eta)$ time.*

The original separator theorem guarantees the existence of a vertex separator of size $O(\sqrt{\eta})$. Because of the bounded vertex degree, this implies an edge separator of asymptotically the same size. Also the original theorem guarantees only a 2/3 to 1/3 balance, and by iterating it the 1/2 to 1/2 balance is

[1] This is unfortunately very restrictive. It is needed in order to apply the clustering via planar graph separators (see 1.3). However, it is likely that it still works even if the degree is not bounded, using a weighted graph separator. This would allow an arbitrary trapezoidal decomposition obtained with vertical rays (that is, constant x-coordinate), which would be more satisfactory. We are still investigating this.

obtained. Though this adds some further unnecessary complication to the algorithm, it facilitates the argument and computation below.

As in [13], the clustering is performed by iterating the separator construction, until *clusters* (groups) consisting of at most t vertices are obtained. Thus, with l so that $2^i = \eta/t$, the total separator size is big-O of

$$\sum_{i=0}^{l} 2^i \left(\frac{\eta}{2^i}\right)^{1/2} = O\left(\frac{\eta}{t^{1/2}}\right). \quad (1)$$

Thus, there is a gain $\Omega(t^{1/2})$ in the number of edges separating groups respect to the $O(\eta)$ edges separating the vertices of the original graph.

2 3-D Diameter

2.1 Intersection of Congruent Balls

Let P be a set of n points in $\mathbb{R}^3$. For $p \in P$, let $b(p,r)$ be the ball of radius r centered at p and let $s(p,r)$ be the bounding sphere of $b(p,r)$ (p and r may be omitted when they are understood or not relevant). Let $B = B(P,r) = \{b(p,r) : p \in S\}$ and $\mathcal{B} = \mathcal{B}(B) = \bigcap_{b\in B} b$. $\mathcal{B}$ is a convex body which we call a *spherical polytope*. The boundary $\mathrm{bd}(\mathcal{B})$ of $\mathcal{B}$ consists of facets, edges and vertices corresponding to the intersection of one, two and three bounding spheres respectively (we assume that the point set P is in general position so that more than three bounding spheres have empty intersection; this can be achieved using symbolic perturbation [16, 34]). A facet of $\mathcal{B}$ is the intersection of *spherical caps* on the sphere that supports the facet (each cap resulting from the intersection with another ball), which we call a *spherical polygon*. We point out that a spherical polygon can be bounded by only two edges, and that the intersection circle of two bounding spheres can contribute more than one edge to $\mathrm{bd}(\mathcal{B})$. The *size* of $\mathcal{B}$, denoted $|\mathcal{B}|$, is the total number of facets, edges and vertices. In general, if the radii of the balls are not equal, $|\mathcal{B}|$ can be $\Omega(|B|^2)$. However, for congruent balls, $|\mathcal{B}|$ is $O(|B|)$ as a result of a "convexity" property of the faces [19],[2] which implies that for the bounding sphere s of each ball in B, $\mathrm{bd}(\mathcal{B}) \cap s$ has at most one connected component. Since the degree of a vertex in $\mathcal{B}$ is exactly three, it follows that $|\mathcal{B}|$ is $O(|B|)$.

[2]For completeness: Let p, q be two points on $\mathrm{bd}(\mathcal{B})$ both on the same bounding sphere s (assuming $|B| \geq 2$, they cannot be antipodal). Then the *geodesic* segment $\widehat{pq}$ connecting p and q on s (a portion of circle with radius equal to the radius of s) is also on $\mathrm{bd}(\mathcal{B})$, otherwise another ball b' contains p, q but not some other point in $\widehat{pq}$, and so b' would have a radius greater than that of b.

The structure of $\mathcal{B}$ is very similar to that of a 2-d Euclidean Voronoi diagram. In fact, Clarkson and Shor [12], used their random incremental approach to construct $\mathcal{B}$ in optimal expected time $O(n \log n)$ much in the way the standard Euclidean Voronoi diagram is constructed. An optimal deterministic algorithm was given by Amato *et al.* [4], by a derandomization of a divide-and-conquer approach. The previous best algorithm for the diameter problem in [30] was based on such an optimal algorithm, in combination with parametric search. In contrast, the new algorithm requires the computation of an intersection of congruent balls only for small collections, and so a simpler algorithm suffices.

2.2 Spherical-simplices and ϵ-Nets

Spherical-simplices. A spherical polytope $\mathcal{B}$ is decomposed into *spherical-simplices* as follows. First, each of the facets is decomposed into *spherical-triangles* by taking an arbitrary vertex of the facet and joining it to each of the other non-adjacent vertices of the facet with a geodesic segment (note that such geodesic segments do not intersect inside the facet). Then, we choose an arbitrary vertex of $\mathcal{B}$ and join it to the spherical triangles, thus obtaining a collection of $O(|\mathcal{B}|) = O(|B|)$ *spherical simplices.*[3] For $\mathcal{B} = \mathcal{B}(B)$, we denote this decomposition by $\mathcal{T}(B)$. The size of $\mathcal{T}(B)$ is $O(|B|)$. For a set of spheres S and a spherical simplex Δ, the *conflict list* $S_{|\Delta}$ is the set of those $s \in S$ such that $s \cap \Delta \neq \emptyset$. We denote the set of all the lists $S_{|\Delta}$, $\Delta \in T$, by $T[S]$ and, abusing notation, denote the total size $\sum_{\Delta\in T} |S_{|\Delta}|$ by $|T[S]|$. Analogously, for a set of points P, $P_{|\Delta}$ denotes those contained in Δ.

Spherical-simplices define a collection $\mathcal{S}_P$ of *ranges* (subsets of) on the set of points P as follows: $R \subseteq P$ is in $\mathcal{S}_P$ if for some radius $r > 0$ and some spherical-simplex Δ determined by spheres of radius r, $R = S_{|\Delta}$, where $S = S(P,r)$. The range space $(P, \mathcal{S}_P)$ has a constant VC-exponent, that is, for some constant c, $|\mathcal{S}_P| = O(|P|^c)$. This is most easily verified through *linearization* [33, 2, 25].[4] For this, we only need to

[3]We point out that we have no canonical way to choose the vertex used in the triangulation of each facet polygon, so that the resulting decomposition into spherical simplices satisfies the properties of *configuration spaces* [12, 26, 24] needed for stronger sampling bounds to hold (choosing the bottom vertex does not work). Fortunately, we do not need such stronger bounds; it suffices to have a triangulation.

[4]Let $\mathbb{L}^{d,k}$ be the collection of *linear cells* in $\mathbb{R}^d$ of complexity at most k, where a linear cell in $\mathbb{R}^d$ is the union of convex polyhedra, and its complexity is the total number of its faces. The range space induced by a class of range-objects $\mathbb{T}$ on a class of point-objects $\mathbb{X}$ is *linearizable* if there are integers d, k

verify that for a spherical-simplex Δ and a sphere s, $s \cap \Delta \neq \emptyset$ can be described by a first-order predicate of size $O(1)$ in the theory of closed fields (one formed from polynomials inequalities using Boolean connectives and quantifiers). See [25] for a justification of this.

Nets. An ϵ-net N for a range space $(X, \mathcal{R})$ is a subset of X such that for each $R \in \mathcal{R}$, if $|R| > \epsilon|X|$, then $N \cap R \neq \emptyset$ [18]. In particular, using the contrapositive, if $N \cap R = \emptyset$, then $|R| \leq \epsilon|X|$, this leads to an effective way to decompose a problem into subproblems with smaller size. Matoušek [23] has shown that, for a *linearizable* range space $(X, \mathcal{R})$ of constant VC-exponent d, there is a $\delta = \delta(d) > 0$ such that for $r \leq n^{\delta}$, a $(1/r)$-net of size $O(r \log r)$ can be computed in time $O(n \log r)$, where $n = |X|$. Linearization translates the computation of an ϵ-net for the range space $(P, \mathcal{S}_P)$ into the computation of an ϵ-net for ranges induced by the union of $O(1)$ simplices in a higher dimensional space $\mathbb{R}^d$, to which the algorithm in [23] directly applies.

2.3 Conflict List Computation

Let B be a set of balls and let S be a set of spheres, all of equal radius, and let $T = \mathcal{T}(B)$, the decomposition of $\mathcal{B} = \mathcal{B}(B)$ into spherical simplices. We want to compute efficiently all the conflict lists $S_{|\Delta}$, $\Delta \in T$. This is done by computing the conflicts of each $s \in S$ as follows: First determine a point v in $s \cap \mathrm{bd}(\mathcal{B})$, if it exists, using a *Dobkin-Kirkpatrick hierarchy*, and then determine all the conflicts by walking on the triangulation of $\mathrm{bd}(\mathcal{B})$, following the boundary of the facet supported by s in $\mathcal{B}(B \cup \{b\})$, where b is the ball bounded by s. The correctness follows from the fact that s contributes a connected facet to $\mathcal{B}(B \cup \{b\})$. Next, we elaborate on the hierarchy construction and the walk.

Dobkin-Kirkpatrick (DK) Hierarchy Let $B = B(P, r)$. A hierarchy for $\mathcal{B}(B)$ similar to that for a 3-d convex polytope by Dobkin and Kirkpatrick [15] can be defined for $\mathcal{B}(B)$. The *DK hierarchy* is a sequence $\mathcal{B}(B_i)$, $i = 0, \ldots, k$, where $B_0 = B$, $B_{i+1} \subseteq B_i$, $B_k = O(1)$, $|B_{i+1}| \leq \alpha|B_i|$ for some constant $0 < \alpha < 1$, and $D_i = B_i - B_{i+1}$ consists of a set of balls such that: (i) the facets of $b, b' \in D_i$ in $\mathcal{B}_i$ are not adjacent, (ii) the facet of $b \in D_i$ in $\mathcal{B}_i$ has $O(1)$ adjacent facets in $\mathcal{B}_i$. The existence of such set D_i follows from the planarity of the adjacency graph of $\mathrm{bd}(\mathcal{B}_i)$ [15]. Note that $k = O(\log |B|)$. The DK hierarchy can be computed easily in time $O(|B|)$.

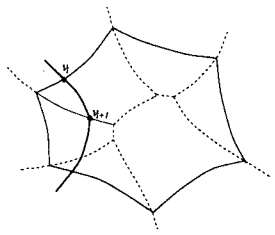

Figure 1: Search in the DK hierarchy.

Let $s = s(p, r)$ and $b = b(p, r)$. The DK hierarchy is used to determine a point v in the intersection of s and $\mathcal{B}(B)$, if it exists (this is also similar to the use of the DK hierarchy for convex polytopes): Start by locating a point v_k in $s \cap \mathcal{B}$ in time $O(1)$ if it exists, else halt with failure; from v_{i+1} in $\mathcal{B}_{i+1}$ one can obtain v_i in $\mathcal{B}_i$ in time $O(1)$ if it exists by the properties of the DK decomposition: either $v_i = v_{i+1}$ because v_{i+1} is already a point in $\mathcal{B}_i$, or only $O(1)$ edges need to be checked. If v_{i+1} does not exist, halt with failure. See Fig. 1. Thus, in time $O(\log |B|)$, a point $v = v_0$ in $s \cap \mathcal{B}$ is obtained, if it exists.

Walk. The walk for s starts at the point v, and follows the boundary of the facet of s in $\mathcal{B}(B \cup \{b\})$ through the triangulation for which an appropriate adjacency data structure is available. This walk takes time linear in the number of conflicts found: This is the case because the walk for s only visits triangles that actually conflict with s and, second, a given triangle t can only be visited at most $O(1)$ times (because the boundary of the facet supported by s in $\mathcal{B}(B \cup \{b\})$ intersects the boundary of t $O(1)$ times). Therefore, the total time for computing the conflict lists is $O(|S| \log |B| + |T[S]|)$.

2.4 Clustering

We now apply the clustering to our problem: Let $\mathcal{B} = \mathcal{B}(B)$ for some set of balls B, and let $T = \mathcal{T}(B)$ be its decomposition into spherical-simplices. Let $\mathcal{G}$ be the dual graph of T (or more properly, of its restriction to $\mathrm{bd}(\mathcal{B})$). Let G be the set of clusters resulting from the procedure just described and let S

and maps $\varphi : \mathbb{X} \to \mathbb{R}^d$ and $\psi : \mathbb{T} \to \mathbb{L}^{d,k}$ such that for $x \in \mathbb{X}$ and $\Delta \in \mathbb{T}$, $x \cap \Delta \neq \emptyset$ iff $\varphi(x) \in \psi(\Delta)$. A general procedure to obtain φ and ψ is as follows (see [25]): Start with a first-order predicate Π in the theory of closed fields (one formed from polynomials inequalities using Boolean connectives and quantifiers) that describes when $x \cap \Delta \neq \emptyset$; then, using a quantifier elimination method, rewrite Π as a disjunction of several conjunctions of polynomial inequalities; finally, by introducing a variable for each monomial that appears in the polynomial inequalities, obtain linear inequalities that correspond to a linear cell.

be a set of n spheres (all balls in B and spheres in S are congruent). For $\square \in G$, let $S_{|\square}$ denote the conflict list of S in $\square$, let $G[S]$ be the set of all conflict lists $S_{|\square}$, $\square \in G$, and let $|G[S]|$ denote the total conflict list size $\sum_{\square \in G} |S_{|\square}|$. We want to bound $|G[S]|$, under the assumption that for any $\Delta \in T$, $|S_{|\Delta}| \leq \kappa$. We observe that the *conflict region* of any $s \in S$, that is, the union of the spherical triangles bounding the spherical simplices with which it conflicts, is connected (because a sphere contributes a single facet). Thus, if the conflict region of a sphere does not intersect the boundaries between clusters, then the sphere conflicts with only one cluster. This is illustrated in the figure where the boundary of a cluster is shown with a continuous line, and the boundaries of the conflict regions for some spheres are shown with broken lines. Thus, using clustering as already discussed, we

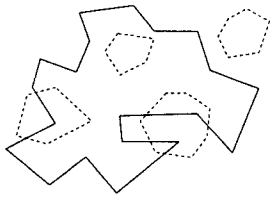

Figure 2: Connectedness of conflict regions.

obtain the total size of subproblems $|G[S]|$ is bounded by

$$n + C\frac{\eta}{t^{1/2}} \cdot \kappa.$$

In our application, we have that B is a $(1/r)$-net. So $\eta = C' r \log r$ and $\kappa = n/r$. Then, choosing $t = r^{1/2}$, we have that $|G[S]|$ is bounded by

$$n \cdot \left(1 + C_0 \frac{\log r}{r^{1/4}}\right).$$

Choosing $r = f(n)$ sufficiently large, *e.g.*, $r = \Theta(n^{\epsilon})$, the total subproblem size remains $O(n)$ through all recursive levels of the computation. Actually, $r = \Theta(\log^c n)$ is sufficient. It would be interesting if $r = \Theta(1)$ could be used, as in this case the construction of a $(1/r)$-net is considerably simpler. Unfortunately, the analysis is not sufficient to support this.

2.5 Algorithm

Let $d_F(p, Q)$ denote the furthest distance between p and the point set Q, and $d_F(P, Q)$ denote the furthest distance between the point sets P and Q. The box below has the outline of procedure `Furthest`(P, Q), which determines $d_F(P, Q)$. In Step 9, Centers(X) denotes the set of centers of the spheres in X, and $P_{|\square}$ denotes the set of points $P \cap \square$. The diameter problem for a point set P is solved by a call to `Furthest`(P, P).

`Furthest`(P, Q)

1. If $|Q| = O(1)$ or $|P| = O(1)$ then compute $d_F(P, Q)$ by examining all pairs and return this value
2. Compute a $(1/|Q|^{\delta})$-net $N \subseteq Q$ with respect to spherical-simplex ranges
3. Construct a furthest point data structure $\mathcal{D}_N$ for N
4. For each $p \in P$, compute $d_p = d_F(p, N)$ using $\mathcal{D}_N$
5. Let $D = \max_{p \in P} d_p$, $B_N = B(N, D)$ and $S_Q = S(Q, D)$
6. Construct $\mathcal{B}_N = \mathcal{B}(B_N)$ and $T_N = \mathcal{T}(B_N)$
7. Cluster the simplices in T_N into a set of clusters G_N, each of size $O(|Q|^{\delta/2})$
8. For each $\square \in G_N$, determine $S_{Q|\square}$ and $P_{|\square}$
9. For each $\square \in G_N$, $D_{\square} \leftarrow$ `Furthest`$(P_{|\square}, \text{Centers}(S_{Q|\square}))$
10. Return $\max(D, \max_{\square \in G_N} D_{\square})$

Figure 3: Furthest pair procedure.

Correctness. Step 1 takes care of the basis case in which one of the sets has size $O(1)$. Step 2 computes a $(1/r)$-net with $r = |Q|^{\delta}$ to be used as the sample for partitioning the problem. The value $D = d_F(P, N)$ obtained in Step 5 is a lower bound for $d_F(P, Q)$. Then the spherical polytope $\mathcal{B}_N$ is computed for balls of radius D and the spherical simplices in its decomposition are clustered in Step 7 as described in the previous section. Consider a cluster $\square \in G_N$, and its two sets $S_{Q|\square}$ and $P_{|\square}$. Clearly, for $q \in Q - S_{Q|\square}$ and $p \in P_{|\square}$, the distance between p and q is smaller than D, so this pair does not need to be considered any further. This justifies recursing, in Step 9, on the pairs $(P_{|\square}, \text{Centers}(S_{Q|\square}))$. We point out that, as presented, there is no guarantee of size reduction for the set P; to correct this, the roles of the sets P and Q in the recursive calls could be switched.

Step Details and Time Analysis. Step 1 computes the farthest distance by looking at each pair; this takes time $O(|P|)$ if $|Q| = O(1)$, or $O(|Q|)$ if $|P| = O(1)$. Step 2 is performed using an algorithm by Matoušek [23], which computes an *ϵ-approximation*, with $\epsilon = O(1/|Q|^{\delta})$, as an intermediate step; this takes time $O(|Q| \log r) = O(|Q| \log |Q|)$ and N has size $O(r \log r) = O(|Q|^{\delta} \log |Q|)$. Step 3 computes the furthest point Voronoi diagram of N and a corresponding point location data structure so that a query can be answered in time $O(\log |N|)$; it can be completed using time and space $O(|Q|^2)$ using a divide-and conquer approach based on sampling [5] (this reference deals with the closest point Voronoi diagram, but the approach is the same; this algorithm

is considerably simpler than the one by Chazelle *et al.* [9]). Then, given this data structure, Step 4 takes time $O(|P|\log|Q|)$. Step 6 can be performed in time $O(|N|\log|N|)$ [4], but here it suffices a simpler $O(|N|^{1+\epsilon})$ time algorithm that uses the divide-and-conquer approach based on sampling (without worrying about the usual "blow-up" of the problem sizes). Step 7 can also be performed very efficiently, indeed in time linear in $|T|$ [17], but again a simpler $O(|T|\log|T|)$ time algorithm, which simply uses the linear time graph separator algorithm [21] independently in each level of the iteration in the clustering procedure. In Step 8, the conflict lists $S_{Q|\square}$ are computed by first computing the conflict lists $S_{Q|\Delta}$ as indicated in Subsection 2.3, and then grouping them (eliminating duplicates in the process). The lists $P_{|\square}$ are computed by performing a point location in T using a data structure with logarithmic query time; for this, we can use the data structure resulting from the construction of $\mathcal{B}_N$ in Step 6 (if a point is on a boundary, then it is assigned to one of the adjacent simplices, so that each point is assigned to a single subproblem). Thus, the computation of the conflict lists takes time $O(|Q|\log|T| + |T[S_Q]|)$ and $O(|P|\log|T|)$ respectively. Note that $|T[S_Q]| = (|Q|/r)O(r\log r) = O(|Q|\log|Q|)$.

We determine the total running time by considering all recursive calls of the procedure at the same level. Let M_i and N_i be the sum of the sizes of the sets P and the sets Q for all the calls in the i-th level, and let n_i be an upper bound on the size of the sets Q in the i-th level. From the previous analysis, we obtain that the running time in the i-th level is $O((M_i+N_i)\log n_i)$. Clearly $M_i = n$ and, because of the analysis of the clustering,

$$N_i \le N_{i-1}\cdot\left(1+\frac{C_0\log n_{i-1}}{n_{i-1}^{\delta/4}}\right) \le N_{i-1}\cdot\left(1+\frac{1}{n_{i-1}^{\delta'}}\right)$$

for $\delta' = \delta/4+\epsilon$, with $\epsilon > 0$ arbitrarily small and n_{i-1} sufficiently large, and so

$$N_i \le n\cdot\prod_{j=0}^{i-1}\left(1+\frac{1}{n_j^{\delta'}}\right).$$

On the other hand, $n_i \le n_{i-1}^{1-\delta/2}$ and so $n_i \le n^{(1-\delta/2)^i}$. The worst case for N_i occurs when $n_i = n^{(1-\delta/2)^i}$, and thus we obtain

$$N_i \le n\cdot\prod_{j=0}^{i-1}\left(1+\frac{1}{n^{(1-\delta/2)^i\delta'}}\right) = O(n).$$

Finally, since $\sum_{i\ge 0}\log n_i = O(\log n)$, then the total running time is $O(n\log n)$.

Theorem 2 *The diameter of a 3-d point set of size n can be computed deterministically in optimal time $O(n\log n)$.*

As a final remark, in a randomized version of the algorithm, Step 2 finds a net by simply taking a random sample, and substituting other procedures for possibly simpler randomized algorithms.

3 2-D Lower Envelopes

3.1 Problem

We consider the problem of computing the *lower envelope* of a set F of functions $f : \mathbb{R}^2 \to \mathbb{R}$. The lower envelope $\mathcal{L}(F)$ is the function that is the pointwise minimum over all the functions in F, and one is interested in a description into *cells*, in each of which $\mathcal{L}(F)$ is achieved by the the same subset of F. We assume a *nondegeneracy* by which in these 0-, 1- and 2-dimensional cells, only 1, 2 and 3 functions achieve the minimum, respectively.

Recall our assumptions (i-iv) described in the Introduction, regarding the existence of a decomposition of $\mathcal{L}(F)$ into trapezoids. For Voronoi diagrams, such a decomposition can usually be obtained by tracing a *geodesic* from each vertex in a face to the *site* determining the cell. For $R \subseteq F$, we let $\mathcal{T}(R)$ denote the set of bricks below $\mathcal{L}(F)$. The conflict list $F_{|\Delta}$ of a brick $\Delta \in \mathcal{T}(R)$ is the set of functions in F that intersect it.

In computing a 2-d lower envelope, the corresponding 1-d version appears naturally as the restriction of the 2-d lower envelope to a *curtain*, the set of points below an edge. Given a curtain γ, the restriction is denoted $\mathcal{L}_{|\gamma}(F)$. For a brick Δ, $\mathcal{T}_{|\Delta}(R)$ denotes the decomposition into bricks of $\mathcal{L}(R)$ restricted to Δ (and similarly for clusters).

We assume that there are functions $f_1(n)$ and $f_2(n)$ that give upper bounds on the complexity (number of vertices and edges) in a 1-d and 2-d lower enevelope. The only assumptions we make about them are: $f_1(n) = O(n^{1+\delta})$ for a very small δ, and $f_2 = \Omega(n)$. The condition on $f_1(n)$ is appropriate in view of the upper bounds for Davenport-Schinzel sequences [31].

3.2 Algorithm

The construction proceeds in rounds and is determined by a sequence of probabilities p_i. Let $n_i = 1/p_i$ and $r_i = p_i/p_{i-1} = n_{i-1}/n_i$. The result of the i-th round is a set T_i of bricks that cover the lower envelope. These bricks are grouped into clusters G_i:

A region $\square \in G_i$ is the union of a set of bricks $T_{i,\square}$ (so it is not of constant description complexity) and has associated a set $R_{i,\square} \subseteq F_{|\square}$ which is the set of functions bounding the bricks $T_{i,\square}$ from above. Thus, $T_i = \bigcup_{\square \in G_i} T_{i,\square}$. Unlike the strict divide-and-conquer, in the i-th round, the algorithm performs a *clean-up* within each cluster in G_{i-1}, that is, the courtains inside the clusters of the $(i-1)$-st round are not final, and are modified according to the new sample obtained for the cluster. The construction in the i-th round proceeds as follows. For each $\square \in G_{i-1}$, take a p_i-sample $R_\square \subseteq F_{|\square} - R_{i-1,\square}$, and let $R_{i,\square} = R_{i-1,\square} \cup R_\square$. As usual in derandomization, the sample is actually computed via an ϵ-approximation. Then compute $T_{i,\square} = \mathcal{T}_{|\square}(R_{i,\square})$ – this is the partial clean-up – and use it to decompose $\square$ into a set of smaller clusters $G_\square$, by *clustering* the bricks in $T_{i,\square}$, as described in Sec. 1.3. The set of all these newer subclusters is G_i. A parameter t_i determines how big the clusters are: each cluster in G_i consists of $O(t_i)$ cells. Thus, for $\Delta \in T_{i,\square} = \mathcal{T}(R_{i,\square})$ and $\square' \in G_\square$:

$$n_\Delta = O(n_i \log r_i) \quad \text{and} \quad n_{\square'} = O(n_i t_i \log r_i).$$

The parameters are determined by the following choice: $r_i = n_{i-1}^{\alpha}$ and $t_i = r_i^{1/2}$. The appropriate choice of the constant α follows from the analysis below. This choice implies $n_i = n^{(1-\alpha)^i}$, $r_i = n^{(1-\alpha)^i \alpha}$ and $p_i = 1/n_i = n^{-(1-\alpha)^i}$.

The computation for $\square \in G_{i-1}$ in the i-th round is summarized in Fig. 4.

Step Details. Let $n_\square = |F_{|\square}|$. In Step 1, the approximation is with respect to *brick ranges*; the computation uses an algorithm of Matoušek [23] and requires time $O(n_\square \log(1/\epsilon_i))$. Step 2 is performed using the method of conditional probabilities [28] and so requires time polynomial in $|A_\square|$; the constant α is chosen so that this is $O(n_\square)$. Step 3 is performed by reducing it to a point location among hyperplanes via linearization, so that it requires time $O(n_\square \log |R_{i,\square}| + |T_{i,\square}[F]|)$. Finally, Step 4 is performed as described in Section 1.3 and requires time polynomial in $|R_{i,\square}|$, and so $O(n_\square)$. Here, we use need the assumption that the adjacency graph for the decomposition is of bounded degree. The correctness is clear. Though, this procedure does not produce the decomposition $\mathcal{T}(F)$, for most purposes (*e.g.* point location or ray-shooting) the decomposition obtained is satisfactory.

Analysis. We now analize the algorithm to verify that its total running time is $O(n \log n + f_2(n))$. Let

2D-Lower-Envelope (i-th round)

Input: Cluster $\square$, its set of bricks $T_{i-1,\square}$, with conflict lists respect to F.
Output: New decomposition $T_{i,\square}$, with conflict lists respect to F, and its clustering $G_\square$.

1. Compute an ϵ_i-approximation $A_\square$ for $F_{|\square}$, where $1/\epsilon_i = (r_{i-1}t_{i-1})^c$, for a constant c. Let $\tilde{p}_\square = p_i |F_{|\square}|/|A_\square|$, the sampling probability with respect to $A_\square$.
2. Compute a sample $R_\square \subseteq A_\square$ and $T_{i,\square} = \mathcal{T}_{|\square}(R_{i,\square})$ where $R_{i,\square} = R_{i-1,\square} \cup R_\square$ so that :
   (i) $|T_{i,\square}|$ and $|T_{i,\square}[A_\square]|$ have values at most a constant factor greater than the corresponding expected values for a $\tilde{p}_\square$-sample, and
   (ii) for each $\Delta \in T_{i,\square}$, $|F_{|\Delta}| = O(\log(\tilde{p}_\square \tilde{n}_\square)/\tilde{p}_\square)$.
3. Compute the conflict lists $T_{i,\square}[F]$.
4. Perform the clustering using a separator decomposition on the adjacency graph $\mathcal{G}_\square$, and let $G_\square$ be the new set of subregions in $\square$.

Figure 4: Lower envelope procedure

J_i° be the total number of edges bounding the regions $\square$ in G_i.

Lemma 3 *Let $X_i = J_{i-1}^\circ \cdot f_1(r_i \log r_{i-1}) + f_2(p_i n)$. Then the decompositions $T_{i,\square}$ computed in Step 2 satisfy:*

(i) $\sum_{\square \in G_{i-1}} |T_{i,\square}| = O(X_i)$.

(ii) $\sum_{\square \in G_{i-1}} |T_{i,\square}[F]| = O(X_i/p_i)$.

Proof. Let $\tilde{n}_\square = |A_\square|$. Let $r_\square = \tilde{p}_\square \tilde{n}_\square = p_i n_\square$, the expected size of $R_\square$. From the previous round, $n_\square = O(t_{i-1} n_{i-1} \log r_{i-1})$ and so $r_\square = O(t_{i-1} r_i \log r_{i-1})$. The size of $T_{i,\square}$ is big-O of the number of vertices of $\mathcal{L}_{|\square}(R_{i,\square})$, including those on the boundaries. For a bounding courtain γ, from the previous round $|F_{|\gamma}| = O(n_{i-1} \log r_{i-1})$. Since $|A_{\square|\gamma}| = O(|F_{|\gamma}|(\tilde{n}_\square/n_\square) + \epsilon_i \tilde{n}_\square)$, then the expected value of $|R_{\square|\gamma}|$ is $O(\tilde{p}_\square |A_{\square|\gamma}|) = O(r_i \log r_{i-1} + \epsilon_i r_\square)$, and since $r_\square = O(t_{i-1} r_i \log r_{i-1})$, then $|R_{\square|\gamma}| = O(r_i \log r_{i-1})$. Thus, the number of vertices of $\mathcal{L}_{|\square}(R_{i,\square})$ on the bounding courtains is $O(J_{i-1}^\circ \cdot f_1(r_i \log r_{i-1}))$, and this justifies the first term in (i). Now, we bound the number of interior vertices. Let $\mathrm{vert}(X, \square)$ denote the vertices of the arrangement of X in $\square$. For a vertex v, let $t_v = p_i |F_{|v}|$ and $\tilde{t}_v = \tilde{p}_\square |A_{\square|v}|$, the *excess* with respect to F and $A_\square$. The expected number of vertices of $\mathcal{L}_{|\square}(R_{i,\square})$ is big-O of

$$\sum_{v \in \mathrm{vert}(A_\square, \square)} \tilde{p}_\square^3 (1-\tilde{p}_\square)^{|A_{\square|v}|} \leq \sum_{v \in \mathrm{vert}(A_\square, \square)} \tilde{p}_\square^3 \cdot e^{-\tilde{t}_v}.$$

(We are ignoring vertices on the functions in $R_{i-1,\square}$, which can be handled similarly; we omit the details here.) Now, we claim that because $A_\square$ is an ϵ_i-approximation for $F_{|\square}$, then (proof omitted):

Claim 4 *For $\epsilon_i = O((\beta/r_i)^{c'})$, with c' a constant sufficiently large:*

$$\left| \sum_{v \in \text{vert}(A_\square, \square)} \tilde{p}_\square^3 \cdot e^{-\tilde{t}_v} - \sum_{v \in \text{vert}(F_{|\square}, \square)} p_i^3 \cdot e^{-t_v} \right| \leq \beta \cdot t_{i-1}.$$

For $\beta = O(1)$, the accumulated error over all the clusters in G_{i-1} is $O(t_{i-1} \cdot |G_{i-1}|) = O(J_{i-1}^\circ)$, which is dominated by the first term in (i). Therefore, ignoring this error, the total number of interior vertices is big-O of

$$\sum_{\square \in G_{i-1}} \sum_{v \in \text{vert}(F_{|\square}, \square)} p_i^3 \cdot e^{-t_v} \leq \sum_{v \in \text{vert}(F)} p_i^3 \cdot e^{-t_v}$$
$$\leq \sum_{t=1}^{p_i n} p_i^3 \cdot |\text{vert}(F, (t+1)/p_i)| \cdot e^{-t},$$

where $\text{vert}(F, \ell)$ denotes the set of vertices in level at most ℓ. By results in [12], $|\text{vert}(F, \ell)| = O(\ell^3 \cdot f_2(n/\ell))$, therefore the previous term is big-O of (here is where we use the vertex accounting idea of Chazelle [10], with the added ingredient of taking into account the exponential decay with the excess)

$$\sum_{t=1}^{\infty} p_i^3 \cdot \left(\frac{t+1}{p_i}\right)^3 f_2\left(\frac{p_i n}{t+1}\right) \cdot e^{-t} = O(f_2(p_i n)).$$

This completes part (i). To verify part (ii), we use from [12, 26] that

$$\mathbf{E}[|T_{i,\square}[A_\square]|] = \frac{1}{\tilde{p}_\square} \cdot \mathbf{E}[|T_{i,\square}|],$$

and since $n_\Delta \leq \tilde{n}_\Delta (n_\square / \tilde{n}_\square) + \epsilon_i n_\square$, then

$$\mathbf{E}[|T_{i,\square}[F]|] = \left(\frac{1}{p_i} + \epsilon_i n_\square\right) \cdot \mathbf{E}[|T_{i,\square}|]$$
$$= O\left(\frac{1}{p_i} \cdot \mathbf{E}[|T_{i,\square}|]\right),$$

using $\epsilon_i \leq 1/(t_{i-1} r_i \log r_{i-1})$. From this, part (ii) follows. □

From this claim, and given the analysis for the clustering procedure, we conclude the following recurrence for J_i°, where A and B are some constants:

$$J_i^\circ \leq A J_{i-1}^\circ f_1(r_i \log r_{i-1}) + B \cdot \frac{J_{i-1}^\circ f_1(r_i \log r_{i-1}) + f_2(p_i n)}{t_i^{1/2}}.$$

Here, the first term corresponds to the subcourtains of older courtains due to the refined sampling, and the second term corresponds to the new introduced ones in the interior of the clusters. Form the analysis of the clustering in Sec. 1.3, Eqn. (1) introduces the dividing term $t_i^{1/2}$ which is what limits the growth of the number of bounding courtains counted by J_i°. We solve this recurrence by expanding it, making use of the choice $r_i = n_{i-1}^\alpha$, and the assumptions $f_1(n) = O(n^{1+\delta})$ and $f_2 = \Omega(n)$, with appropriate constants A', C:

$$J_i^\circ \leq A' \cdot J_{i-1}^\circ f_1(r_i \log r_{i-1}) + B \cdot \frac{f_2(p_i n)}{t_i^{1/2}}$$
$$\leq C \cdot \frac{f_2(p_i n)}{t_i^{1/4}}$$

This is the essential feature resulting from the clustering: the number of *spurious* courtains is maintained considerably smaller, by a factor $t_i^{1/4}$, than the expected complexity $O(f_2(p_i n))$ of the decomposition in the i-th round. Therefore, the total total size of the decomposition J_i is bounded as follows

$$\sum_{\square \in G_{i-1}} |T_{i,\square}| \leq A \cdot J_{i-1}^\circ \cdot f_1(r_i \log r_{i-1}) + B \cdot f_2(p_i n)$$
$$\leq A' \cdot \frac{f_2(p_{i-1} n)}{t_{i-1}^{1/4}} \cdot f_1(r_i \log r_{i-1}) + B \cdot f_2(p_i n)$$
$$\leq f_2(p_i n) \cdot \left(\frac{A' r_i^{\delta'}}{t_{i-1}^{1/4}} + B\right)$$
$$\leq C f_2(p_i n).$$

Finally, the total size of the conflict lists is

$$K_i = \sum_{\square \in G_{i-1}} |T_{i,\square}[F]| = O\left(\frac{f_2(p_i n)}{p_i}\right).$$

Time Analysis. We want to verify that in the i-th round, the computation for all $\square \in G_{i-1}$ can be performed in time $O(K_i \log r_i)$. Step 1 requires time $O(n_\square \log(1/\epsilon_i))$ for $\square \in G_{i-1}$. Since $\log(1/\epsilon_i) = O(\log r_{i-1} + \log t_{i-1})$, we see that $\log t_{i-1} = O(\log r_{i-1})$ and assume that from now on. Thus, Step 1 requires $O(K_{i-1} \log r_{i-1})$ work in total. The computation of $R_\square$ from $A_\square$ and $R_{i-1,\square}$ in Step 2 requires time $O(|A_\square|^c)$, for some constant c. Since the size of $A_\square$ is $O(1/\epsilon_i^{2+\delta})$, the total time is $O(r_{i-1}^c |T_{i-1,\square}|)$, for a new constant c. Added over all $\square \in G_{i-1}$, this is $O(r_{i-1}^c J_{i-1})$, and hence $O(K_{i-1})$ for r_{i-1} with $r_{i-1}^c = O(n_{i-1})$, which is always the case as long as $r_i = O(n_{i-1}^\alpha)$ for small α. Step 3 requires time $O(|R_{i,\square}|^c + n_\square \log r_{i-1} + |T_{i,\square}[S]|)$,using linearization,

for $\square \in G_{i-1}$, hence $O(K_{i-1} \log r_{i-1} + K_i)$. Step 4 can be performed in linear time [17], $O(|T_{i,\square}|)$ for $\square \in G_{i-1}$, and hence $O(J_i)$ in total (but a less efficient polynomial algorithm would suffice). Replacing $K_i = O(f_2(p_i n)/p_i)$ in the bound $O(K_i \log r_i)$, we obtain the following.

Theorem 5 *The lower envelope of n functions of 2 variables, belonging to a class with a decomposition satisfying the assumptions stated above, and with lower envelope of worst-case complexity $O(f_2(n))$, can be computed deterministically in time $O(f_2(n) \log n)$, in general, or $O(f_2(n))$ when $f_2(n) = \Omega(n^{1+\epsilon})$.*

4 Concluding Remarks

The idea of clustering to reduce the usual size "blow-up" of divide-and-conquer algorithms based on random sampling seems quite useful and we expect further applications. In fact, some of our previous work on computing segment intersections [5] can be improved using this approach and this has been reported in [6]. It remains to investigate further applications, and the possibility of applying the technique to higher dimensional problems. For the latter, a major difficulty is the lack of results guaranteeing the existence of separators.

References

[1] P. K. Agarwal, H. Edelsbrunner, O. Schwarzkopf and E. Welzl. Euclidean minimum spanning trees and bichromatic closest pairs. *Discrete Comput. Geom.* **6** (1991), 407–422.

[2] P. K. Agarwal and J. Matoušek. On range searching with semialgebraic sets. *Discrete Comput. Geom.* **11** (1994), 393–418.

[3] P. K. Agarwal, O. Schwarzkopf and M. Sharir . The Overlay of Lower Envelopes in Three Dimension and its Applications. In *Proc. 11th Annu. Symp. Comput. Geom.*, 182–189, 1995.

[4] N. M. Amato, M. T. Goodrich, and E. A. Ramos. Parallel algorithms for higher-dimensional convex hulls. In *Proc. 35th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS 94)*, 683–694, 1994.

[5] N. M. Amato, M. T. Goodrich and E. A. Ramos. Computing faces in segment and simplex arrangements. In *Proc. 26th Annual ACM Sympos. Theory Comput.*, 672–682, 1995.

[6] N. M. Amato, M. T. Goodrich, and E. A. Ramos. Computing the Arrangement of Curve Segments: Divide-and-Conquer Algorithms via Sampling. Manuscript, 1999. (Short abstract submitted to SODA 00.) Available from `http://www.mpi-sb.mpg.de/~ramos`

[7] N. M. Amato and E. A. Ramos. On computing Voronoi diagrams by divide-prune-and-conquer. In *Proc. 12th Annual ACM Sympos. Comput. Geom.*, 672–682, 1996.

[8] S. Bespamyatnikh. An efficient algorithm for the three-dimensional diameter problem. In *Proc. 9th Annu. ACM-SIAM Sympos. Discrete Algorithms (SODA 98)*, 137–146, 1998.

[9] H. Brönnimann, B. Chazelle, and J. Matoušek. Product range spaces, sensitive sampling, and derandomization. In *Proc. 34th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS 93)*, 400–409, 1993.

[10] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geom.* **9** (1993), 145–158.

[11] B. Chazelle, H. Edelsbrunner, L. Guibas and M. Sharir, Diameter, width, closest line pair, and parametric searching, *Discrete Comput. Geom.* **10** (1993), 183–196.

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

[13] F. Dehne, X. Deng, P. Dymond, A. Fabri and A. Khokbar. A randomized parallel 3D convex hull algorithm for coarse grained multicomputers. In *Proc. ACM Sympos. Parallel Algorithms Architectures (SPAA 95)*, 27–33, 1995.

[14] X. Deng and B. Zhu. A randomized algorithm for the Voronoi diagram of line segments on coarse grained multiprocessors. *Algorithmica* **24** (1999), 270–286.

[15] D. P. Dobkin and D. G. Kirkpatrick. Fast detection of polyhedral intersection. *Theoret. Comput. Sci.* **27** (1983), 241–253.

[16] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph.* **9** (1990), 66–104.

[17] M. T. Goodrich. Planar separators and parallel polygon triangulation. *J. Comput. Syst. Sci.* **51** (1995), 374–389.

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

[19] A. Heppes, Beweis einer Vermutung von A. Vázsonyi, *Acta Math. Acad. Sci. Hungar.* **7** (1956), 463–466.

[20] U. Kühn. Lokale Eigenschaften in der algorithmischen Geometrie mit Anwendungen in der Parallelverarbeitung. Inaugural-Dissertation, Fachbereich Mathematik und Informatik, Westfälische Wilhelms-Universität Münster. 1998.

[21] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM J. Comput.* **36** (1979), 177–189.

[22] J. Matoušek. Approximations and optimal geometric divide-and-conquer. In *Proc. 23rd Annu. ACM Sympos. Theory Comput. (STOC 91)*, 505–511, 1991. Also in *J. Comput. Syst. Sci.* **50**, 203–208 (1995).

[23] J. Matoušek. Efficient partition trees *Discrete Comput. Geom.* **8** (1992), 315–334.

[24] J. Matoušek. Derandomization in computational geometry. *Journal of Algorithms* **20** (1996), 545–580.

[25] J. Matoušek and O. Schwarzkopf. A deterministic algorithm for the three-dimensional diameter problem. *Proc. 25th ACM Sympos. Theory Comput. (STOC 93)*, 478–484, 1993. Revised version in: *Computational Geometry: Theory and Applications.* **6** (1996), 253–262.

[26] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms.* Prentice Hall, Englewood Cliffs, NJ, 1994.

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

[28] P. Raghavan. Probabilistic construction of deterministic algorithms: Approximating packing integer programs. *J. Comput. Syst. Sci.* **37** (1988), 130–143.

[29] E. A. Ramos. An algorithm for intersecting equal radius balls in $\mathbb{R}^3$. Technical Report UIUCDCS-R-94-1851, Dept. of Computer Science. University of Illinois at Urbana-Champaign, 1994. Journal version in *Computational Geometry: Theory and Applications.*

[30] E. A. Ramos Constructing 1-d lower envelopes and applications. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, 57–66, 1997.

[31] M. Sharir and P.K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications.* Cambridge University Press, 1995.

[32] A. C. Yao. On constructing minimum spanning trees in k-dimensional space and related problems. *SIAM J. Comput.* **11** (1982), 721–736.

[33] A. C. Yao and F. F. Yao. A general approach to D-dimensional geometric queries. In *Proc. 17th Annu. ACM Sympos. Theory Comput.*, 163–168, 1985.

[34] C. K. Yap. A geometric consistency theorem for a symbolic perturbation scheme. *J. Comput. Syst. Sci.* **39** (1989), 236–246.

A Proof of Claim 4

Proof. The analysis here is similar to one used in [9]. To simplify notation, we omit the subindex i in p_i and $\tilde{p}_i$. For a *region* $\square$ (it can be a brick or a *cylinder* below), and a function $f(t)$, let

$$\Sigma_\square f = \sum_{v\in \mathrm{vert}(S,\square)} p^3\cdot f(t_v),\quad \tilde{\Sigma}_\square f = \sum_{v\in \mathrm{vert}(A,\square)} \tilde{p}^3\cdot f(t_v),$$

$$\text{and}\quad \tilde{\Sigma}_\square \tilde{f} = \sum_{v\in \mathrm{vert}(A,\square)} \tilde{p}^3\cdot f(\tilde{t}_v).$$

We want to bound $|\Sigma_\Delta f - \tilde{\Sigma}_\Delta \tilde{f}|$ for $f(t)=e^{-t}$. For this, we separate the error into two parts:

$$|\Sigma_\Delta f - \tilde{\Sigma}_\Delta \tilde{f}| \le |\Sigma_\Delta f - \tilde{\Sigma}_\Delta f| + |\tilde{\Sigma}_\Delta f - \tilde{\Sigma}\tilde{f}|.$$

First, we deal with the second part. Because A_Δ is an ϵ-approximation for $S_{|\Delta}$, it follows that $|\tilde{t}_v - t_v| \le \epsilon r_\Delta$, and so, using the mean value theorem,

$$\begin{aligned} |f(\tilde{t}_v) - f(t_v)| &\le |\tilde{t}_v - t_v|\cdot \max_{t\in[t_v,\tilde{t}_v]} |f'(t)| \\ &\le \epsilon r_\Delta \cdot \max_{t\in[t_v,\tilde{t}_v]} |f'(t)| \end{aligned}$$

and

$$\begin{aligned} |\tilde{\Sigma}_\Delta \tilde{f} - \tilde{\Sigma}_\Delta f| &\le \tilde{p}^3 \epsilon r_\Delta \cdot \sum_{v\in \mathrm{vert}(A_\Delta,\Delta)} \max_{t\in[t_v,\tilde{t}_v]} |f'(t)| \\ &\le \tilde{p}^3 \epsilon r_\Delta |\mathrm{vert}(A_\Delta,\Delta)|\cdot \max_{v\in\mathrm{vert}(A_\Delta,\Delta)} \max_{t\in[t_v,\tilde{t}_v]} |f'(t)| \\ &= O\left(\epsilon r_\Delta^4 \cdot \max_{v\in\mathrm{vert}(A_\Delta,\Delta)} \max_{t\in[t_v,\tilde{t}_v]} |f'(t)|\right). \end{aligned}$$

To deal with the first part, it is convenient to consider an ϵ'-net N_Δ and the decomposition Ξ of its arrangement into *cylinders*, and consider the error in each $\diamond \in \Xi$. Since $|N_\Delta| = O((1/\epsilon')\log(1/\epsilon'))$, then $|\Xi| = O(1/\epsilon'^7)$ (with a good margin). Let $f_\diamond^M$ and $f_\diamond^m$ be the max and the min of $f(t_v)$ over $v\in \mathrm{vert}(S,\diamond)$, and let $h_\diamond(f) = f_\diamond^M - f_\diamond^m$. We have,

$$\begin{aligned} |\Sigma_\diamond f - \tilde{\Sigma}_\diamond f| &\le |\Sigma_\diamond f - \Sigma_\diamond f^M| + |\Sigma_\diamond f^M - \tilde{\Sigma}_\diamond f^M| \\ &\qquad + |\tilde{\Sigma}_\diamond f^M - \tilde{\Sigma}_\diamond f| \\ &\le p^3|\mathrm{vert}(S,\diamond)|h_\diamond(f) + |p^3|\mathrm{vert}(S,\diamond)| - \\ &\qquad \tilde{p}^3|\mathrm{vert}(A_\Delta,\diamond)|\,|f_\diamond^M + \tilde{p}^3|\mathrm{vert}(A,\diamond)|h_\diamond(f) \\ &\le |p^3|\mathrm{vert}(S,\diamond)| - \tilde{p}^3|\mathrm{vert}(A_\Delta,\diamond)|\,|f_\diamond^M + \\ &\qquad \left(p^3|\mathrm{vert}(S,\diamond)| + \tilde{p}^3|\mathrm{vert}(A,\diamond)|\right) h_\diamond(f). \end{aligned}$$

Using the mean value theorem and the net property, we have:

$$h_\diamond(f) \le \epsilon' r_\Delta \cdot \max_{v\in\mathrm{vert}(S,\diamond)} \max_{t\in[0,t_v]} |f'(t)|,$$

and using the approximation property for vertices in [9], we have:

$$\left|\, \tilde{p}^3\cdot|\mathrm{vert}(A_\Delta,\diamond)| - p^3\cdot|\mathrm{vert}(S,\diamond)|\,\right| \le \epsilon r_\Delta^3.$$

So, we obtain:

$$\begin{aligned} |\Sigma_\Delta f - \tilde{\Sigma}_\Delta f| &\le \sum_{\diamond\in\Xi} |\Sigma_\diamond f - \tilde{\Sigma}_\diamond f| \\ &\le \sum_{\diamond\in\Xi} \epsilon r_\Delta^3 f_\diamond^M + p^3 \sum_{\diamond\in\Xi} |\mathrm{vert}(S,\diamond)|\cdot h_\diamond(f) + \\ &\qquad \tilde{p}^3 \sum_{\diamond\in\Xi} |\mathrm{vert}(A,\diamond)|\cdot h_\diamond(f) \\ &\le C\frac{\epsilon}{\epsilon'^7} r_\Delta^3 f_\Delta^M + C\epsilon' r_\Delta^4 \cdot \max_{v\in\mathrm{vert}(S,\Delta)} \max_{t\in[0,t_v]} |f'(t)|. \end{aligned}$$

For $f(t)=e^{-t}$, $f_\Delta^M \le 1$,

$$\max_{v\in\mathrm{vert}(S,\Delta)} \max_{t\in[0,t_v]} |f'(t)| \le 1 \quad\text{and}$$

$$\max_{v\in\mathrm{vert}(A_\Delta,\Delta)} \max_{t\in[t_v,\tilde{t}_v]} |f'(t)| \le 1,$$

and so

$$\begin{aligned} |\Sigma_\Delta f - \tilde{\Sigma}_\Delta \tilde{f}| &\le |\Sigma_\Delta f - \tilde{\Sigma}_\Delta f| + |\tilde{\Sigma}_\Delta f - \tilde{\Sigma}\tilde{f}| \\ &\le C\left(\frac{\epsilon}{\epsilon'^7} r_\Delta^3 + \epsilon' r_\Delta^4 + \epsilon r_\Delta^4\right) \\ &= O(\beta), \end{aligned}$$

taking $\epsilon' = O(\beta/r_i^4)$ and $\epsilon = O(\beta^8/r_i^{31})$. This gives the error inside a brick Δ, To obtain the error in a cluster $\square$, we multiply by $O(t_{i-1})$, the size of a cluster. ∎