A Deterministic Linear Time Algorithm for Geometric Separators and its Applications

David Eppstein* Gary L. Miller† Shang-Hua Teng‡

Abstract

We give a deterministic linear time algorithm for finding a "good" sphere separator of a k-ply neighborhood system Φ in any fixed dimension, where a k-ply neighborhood system in $\mathbb{R}^d$ is a collection of n balls such that no points in the space is covered by more than k balls. The separating sphere intersects at most $O\left(k^{1/d}n^{1-1/d}\right)$ balls of Φ and divides the remaining of Φ into two parts: those in the interior and those in the exterior of the sphere, respectively, so that the larger part contains at most δn balls ($(d+1)/(d+2) < \delta < 1$). This result improves the $O(n^2)$ time deterministic algorithm of Miller and Teng [30] and answers a major algorithmic open question posed by Miller, Teng, Thurston, and Vavasis [23, 26].

The deterministic algorithm hinges on the use of a new method for deriving the separator property of neighborhood systems. Using this algorithm, we devise an $O(kn + n\log n)$ time deterministic algorithm for computing the intersection graph of a k-ply neighborhood system. We give an $O(n\log n)$ time algorithm for constructing a linear space, $O(\log n)$ query time search structure for a geometric query problem associated with k-ply neighborhood systems, and we use this data structure in an algorithm for approximating the value of k within a constant factor in time $O(n\log n)$. We also develop a deterministic linear time algorithm for finding an $O\left(k^{1/d}n^{1-1/d}\right)$-separator for a k-nearest neighborhood graph in d dimensions.

1 Introduction

Many problems in scientific computing and computational geometry require efficiently computing a small separator of the underlying graph [6, 7, 9, 10, 11, 18, 20, 28, 29, 30]. By small separator we mean a relatively small subset of vertices whose removal divides the rest of the graph into two disconnected pieces of approximately equal size [19].

In a series of papers [25, 27, 22, 26, 23, 30, 34], Miller, Teng, Thurston, and Vavasis have developed a geometric characterization of graphs (embedded in some fixed dimension) that have a small separator. Their characterization is based on a notion of a *k-ply neighborhood system*, which is a collection of n balls in a fixed dimensional space such that no points in the space is covered by more than k balls. Using this characterization, they gave randomized linear time and processor-efficient randomized NC algorithms for finding a *good* sphere separator – a sphere

*Department of Information and Computer Science, University of California, Irvine, CA 92717.

†School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213.

‡Department of Mathematics and Lab. for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139.

that intersects at most $O\left(k^{1/d}n^{1-1/d}\right)$ balls of the neighborhood system and has a constant splitting ratio.

Unfortunately, these previous results require the use of randomization—they are not deterministic. As part of his doctoral dissertation, Teng [30] devised an $O(n^2)$ time deterministic algorithm for finding a good sphere separator. But for many applications in scientific and geometric computing, an $O(n^2)$ time algorithm is too slow to be used. The result has been that the randomized linear time algorithm is used, and in general this is the only part in the application program that uses randomness [30]. For various applications such as sparse matrix solving in physical simulation and many of other ones in scientific computing, it is desirable to use a *repeatable* program: each run of the program on the same input produces the same output. To this end, it is important to have an efficient deterministic geometric separator algorithm.

In this paper, we give the first known deterministic linear time algorithm for finding a good sphere separator of a k-ply neighborhood system. Our algorithm works in any fixed dimension, and solves a major algorithmic open question posed by Miller, Teng, Thurston, and Vavasis [26, 23]. We first provide a deterministic linear time reduction from the problem of finding sphere separators to an interesting geometric problem, of finding a point covered by a small subset of a system of annuli. We then develop a deterministic linear time algorithm for the second problem using ϵ-cuttings [21] as the basis for a novel geometric prune-and-search technique.

Our algorithm can be applied to several problems in computational geometry. The new algorithm removes the necessity of randomness from the (sphere separator based) divide and conquer paradigm developed in [30, 7]. More specifically, we devise an $O(kn + n\log n)$ time deterministic algorithm for constructing the intersection graph of a k-ply neighborhood system, improving and greatly simplifying the (quad-tree based) deterministic $O(kn\log n)$ time construction of [30]. We give a deterministic $O(n\log n)$ time algorithm for constructing a linear space, $O(\log n)$ query time search structure for a geometric query problem associated with k-ply neighborhood systems. We use this data structure in an algorithm for approximating the value of k within a constant factor in time $O(n\log n)$. Furthermore, we give an $O(nlogn + nk^{d-1}logk)$ time algorithm for computing the exact value of k. Our algorithm improves the best previously known algorithm that finds the exact value of k by constructing a sphere arrangement, and takes time $O(n^d\log n)$. We also develop a deterministic linear time algorithm for finding an $O\left(k^{1/d}n^{1-1/d}\right)$-separator for the k-nearest neighborhood graph in d dimensions.

The outline of this paper is as follows. Section 2 introduces the basic concepts and reviews the techniques and results of our previous works [23, 26, 30] that will be used later in the paper. Section 3 presents a new approach for deriving the separator property of neighborhood systems and reduce the problem of finding 'good' sphere separator to a problem of finding a point in few annuli. Section 4 gives a deterministic linear time algorithm for this problem. Section 5 presents some applications of the new result in computational geometry. Section 6 gives an efficient deterministic NC algorithm for finding good sphere separator. Section 7 asks some open questions.

2 The Background

A d-dimensional *neighborhood system* $\Phi = \{B_1, \ldots, B_n\}$ is a finite collection of balls in $\mathbb{R}^d$. Let p_i be the center of B_i $(1 \leq i \leq n)$ and call $P = \{p_1, \ldots, p_n\}$ *centers* of Φ. For each point $p \in \mathbb{R}^d$, let the *ply* of p, denoted by $\text{ply}_\Phi(p)$, be the number of balls from Φ that contains p. Φ is a k-*ply* neighborhood system if for all p, $\text{ply}_\Phi(p) \leq k$.

Each $(d-1)$-dimensional sphere S in $\mathbb{R}^d$ partitions Φ into three subsets: $\Phi_I(S)$, $\Phi_E(S)$, and $\Phi_O(S)$, those balls that are in the interior of S, in the exterior of S, and that intersect S, respectively. The cardinality of $\Phi_O(S)$, denoted by $|\Phi_O(S)|$, is called the *intersection number* of S.

Notice that the removal of $\Phi_O(S)$ splits Φ into two subsets: $\Phi_I(S)$ and $\Phi_E(S)$, such that no ball in $\Phi_I(S)$ intersects any ball in $\Phi_E(S)$ and vice versa. In analogy to separators in graph theory, $\Phi_O(S)$ can be viewed as a separator of Φ. We call S a *sphere separator* of Φ. Let f be a positive function and δ be a constant such that $0 < \delta < 1$.

Definition 2.1 (Sphere Separators) *A $(d-1)$-sphere S is an $f(n)$-**sphere separator** that δ-**splits** a neighborhood system Φ if $|\Phi_O(S)| \leq f(n)$ and $|\Phi_I(S)|$, $|\Phi_E(S)| \leq \delta n$.*

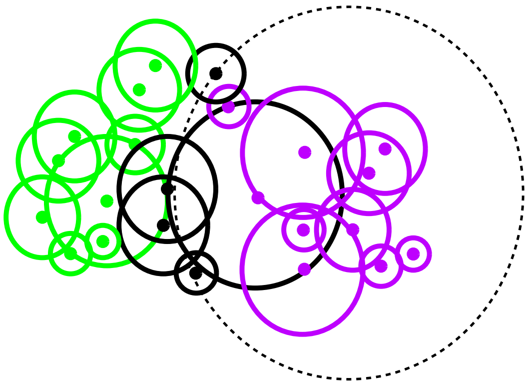

Figure 1: A sphere separator

With these definitions, Miller, Teng, Thurston, and Vavasis [23, 26] proved the following theorem.

Theorem 2.2 (Geometric Separators) *Suppose $\Phi = \{B_1, \ldots, B_n\}$ is a k-ply neighborhood system in $\mathbb{R}^d$. Then there is an $O(k^{1/d} n^{1-1/d})$-sphere separator S that $(d+1)/(d+2)$-splits Φ.*

We now reviews some of the basic concepts and lemmas used in [25, 26, 23, 27, 30] for proving Theorem 2.2. Proofs to lemmas in this section can be found in [23, 30].

A *density function* in $\mathbb{R}^d$ is a real valued nonnegative function $f(x)$ defined on $\mathbb{R}^d$ such that f^k is integrable for all $k = 1, 2, 3, \ldots$. The f-*measure* of a $(d-1)$-dimensional sphere S, denoted by $f(S)$ (by abuse of notation), is given by

$$f(S) = \int_{v \in S} (f(v))^{d-1} (dv)^{d-1}$$

The total volume of the function f, denoted by Total-Volume(f), is given by

$$\text{Total-Volume}(f) = \int_{v \in \mathbb{R}^d} (f(v))^d (dv)^d$$

Density functions can also be defined on the surface of a sphere. To be consistent with the discussion of subsequent sections, we focus on the unit d-sphere. Let U_d be the unit d-sphere in $\mathbb{R}^{d+1}$ whose center is at the origin o. Suppose f is a real valued nonnegative function defined on the surface of U_d such that f^k is integrable for all $k = 1, 2, 3, \ldots$. We call f a *density function* of U_d. The *total volume* of f is defined to be

$$\text{Total-Volume}(f) = \int_{v \in U_d} (f(v))^d (dv)^d$$

A *great sphere* of U_d is the intersection of U_d with a hyperplane passing through the center of U_d. The *f-measure* of a great sphere $\mathcal{G}$ of U_d, denoted by $f(\mathcal{G})$, is given by

$$f(\mathcal{G}) \leq \int_{v \in \mathcal{G}} (f(v))^{d-1} (dv)^{d-1}$$

Let avg(f) be the average f-measure over all great spheres of U_d when the normal vectors of the great spheres are chosen uniformly according to the Euclidean measure. Using Hölder's inequality [14], Miller *et al* [24, 25, 30, 26] proved the following lemma.

Lemma 2.3 *Let A_d denote the "Euclidean" surface volume of U_d. Let f be a density function on U_d. Then*

$$\text{avg}(f) \leq (A_{d-1}/A_d^{1-1/d}) \left((\text{Total-Volume}(f))^{1-1/d} \right).$$

To use Lemma 2.3, we use conformal map to lift points of $\mathbb{R}^d$ onto U_d. Rigid motions and dilations are examples of conformal maps. Another conformal map is the stereographic projection which maps $\mathbb{R}^d$ together with a point at infinity onto U_d [16, 25]. The composition of any two conformal maps is also conformal, so we can apply a sequence of conformal maps to obtain a desirable conformal map. Conformal maps preserve angles and hence preserve the density function in every dimension simultaneously [16, 4, 32]. As shown by Miller et al (see Section 6 of [24]), we can define a new density function f' on U_d such that Total-Volume(f') = Total-Volume(f) and $f'(S) = f(\mathcal{G})$, for each great sphere $\mathcal{G}$ of U_d, where S is the pre-image of $\mathcal{G}$ in $\mathbb{R}^d$ of a conformal map.

Conformal maps have another nice property: the pre-image of each great sphere $\mathcal{G}$ of U_d itself is a $(d-1)$-sphere in $\mathbb{R}^d$. Moreover, the interior and the exterior of the pre-image sphere are mapped to the two hemispheres of U_d defined by $\mathcal{G}$, respectively. It simply follows from Lemma 2.3 [25, 26, 30] that,

Lemma 2.4 *Suppose f is a density function of $\mathbb{R}^d$ and π is a conformal map from $\mathbb{R}^d$ to U_d. Let*

$$\mathcal{S}(\pi) = \{S : S \text{ is a pre-image of some great sphere } \mathcal{G}, \;\; i.e., \, \mathcal{G} = \pi(S)\},$$

and $avg_\pi(f)$ be the average f-measure with respect to all projected great spheres in $\mathcal{S}(\pi)$. Then

$$\text{avg}_\pi(f) \leq A_{d-1} \left((\text{Total-Volume}(f))^{1-1/d} \right).$$

Suppose $P = \{p_1, \ldots, p_n\}$ is a set of points in $\mathbb{R}^d$ and π is a conformal map from $\mathbb{R}^d$ to U_d. Let $\pi(P) = \{\pi(p_1), ..., \pi(p_n)\}$. It follows from the above discussion of conformal mapping that for each great sphere $\mathcal{G}$ of U_d, if S is the pre-image of $\mathcal{G}$ then S δ-splits P iff $\mathcal{G}$ δ-splits $\pi(P)$.

Therefore, to ensure that each sphere of $\mathcal{S}(\pi)$ δ-splits P for some constant $0 < \delta < 1$, the conformal map π needs to satisfy the condition that all hyperplanes containing o δ-split $\pi(P)$. We make use of center points of a point set.

Definition 2.5 *Let Q be a point set in $\mathbb{R}^d$ and let δ be a constant such that $0 < \delta < 1$. A point $c \in \mathbb{R}^d$ is a* δ-center point *of a set of points Q if every hyperplane containing c δ-splits Q.*

It follows from [30, 21, 33] that a δ-center point $(d/(d+1) < \delta < 1)$ can be computed in deterministic linear time.

Miller *et al* [24, 23, 31] showed that there exists a conformal map such that o is a $(d+1)$-dimensional δ-center point of $\pi(P)$. In particular, suppose c is a δ-center point of the stereographic projections of P. Let ST be the stereographic projection from $\mathbb{R}^d$ to U_d. Let $\pi_1 : U_d \rightarrow U_d$ be a rotation of U_d that maps c to c_1, a point on the diameter between the south and the north poles, say $c_1 = (0, ..., 0, r)$ and let π_2 be the dilation of $\mathbb{R}^d$ by a factor of $\sqrt{(1-r)/(1+r)}$. We can use $\pi = ST \circ \pi_2 \circ ST^{-1} \circ \pi_1 \circ ST$. The correctness of the mapping is given in [24]. Summarizing, we have

Lemma 2.6 *A conformal map π such that o is a $(d+1)$-dimensional δ-center point of $\pi(P)$ can be computed in deterministic linear time.*

Suppose $\Phi = \{B_1, \ldots, B_n\}$ is a k-ply neighborhood system in $\mathbb{R}^d$. let r_i be the radius of B_i and let $\gamma_i = 2r_i$, define

$$f_i(x) = \begin{cases} 1/\gamma_i & \text{if } ||x - p_i|| \leq \gamma_i \\ 0 & \text{otherwise} \end{cases}$$

We call f_i the *local density function* of B_i.

Miller *et al* [26, 30] introduced the density function below and proved the following two important lemmas:

$$\begin{aligned} f(x) &= L_{d-1}(f_1(x), ..., f_n(x)) \\ &= \left(\sum_{i=1}^{n} |f(x)|^{d-1} \right)^{1/(d-1)}. \end{aligned}$$

Lemma 2.7 ([26]) *Suppose $\Phi = \{B_1, \ldots, B_n\}$ is a k-ply neighborhood system in $\mathbb{R}^d$. Then there are constants c_1 and c_2 depending only on d such that for each $(d-1)$-sphere S, $|\Phi_O(S)| \leq c_1 k + c_2 f(S)$.*

Lemma 2.8 *Suppose $\Phi = \{B_1, \ldots, B_n\}$ is a k-ply neighborhood system in $\mathbb{R}^d$. Then there is a constant c_3 depending only on d such that*

$$\text{Total-Volume}(f) \leq c_3 k^{1/(d-1)} n.$$

3 Great Belts and Sphere Separators

As shown in Section 2, each d-dimensional neighborhood system $\Phi = \{B_1, \ldots, B_n\}$ can be conformally mapped to a unit d-sphere U_d so that each d-dimensional hyperplane containing the center of U_d δ-splits Φ $((d+1)/(d+2) < \delta < 1)$. Notice that each B_i is mapped to a patch D_i on U_d, whose boundary C_i has the shape of a (d-1)-sphere (see Figure 3). The radius of D_i is defined to be the radius of C_i. Clearly, the ply of $\{D_i : 1 \leq i \leq n\}$ is k iff the ply of Φ is k. Notice that the number of patches with surface volume greater than a half of the surface volume of U_d is bounded by $O(k)$. Thus the number of such patches that intersect any sphere separator is no more than k. For the purpose of bounding the size of a sphere separator, we can ignore those patches whose surface volumes are greater than a half of the surface volume of U_d. Thus, without loss of generality, we assume that each patch has a surface volume no more than a half of the surface volume of U_d. Let r_i be the radius of C_i. In the remaining of this section, we will identify B_i with D_i, and assume that $\Phi = \{B_1, \ldots, B_n\}$ is given on the unit d-sphere.

3.1 Great belts

Each great sphere $\mathcal{G}$ can be identified with the pair of points $p_{\mathcal{G}}$ and $q_{\mathcal{G}}$ on U that lie on the normal to $\mathcal{G}$ (see Figure 2 for an example in 2 dimensions). There is a dual relation between points on U_d and their great spheres:

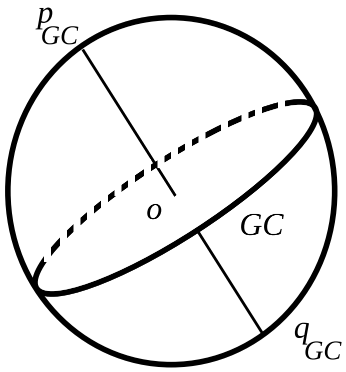

Figure 2: G and its $p_{\mathcal{G}}$, $q_{\mathcal{G}}$

Proposition 3.1 (Duality) *For each pair of great spheres $\mathcal{G}$ and $\mathcal{G}'$ of U_d, $\mathcal{G}$ contains $p_{\mathcal{G}'}$ (and hence $q_{\mathcal{G}'}$ as well) if and only if $\mathcal{G}'$ contains $p_{\mathcal{G}}$ (and hence $q_{\mathcal{G}}$).*

Define a *great belt*[1] be the set of points of U_d that lie between a pair of parallel hyperplanes symmetric to the center of U_d. The *width* of a great belt is then the distance between its two hyperplanes. Notice that a great sphere is a great belt with width 0. Because we assume that all patches have surface volumes less than a half of the surface volume of U_d, the notion of great belts and the duality between patches and great belts are well defined.

It simply follows from Proposition 3.1 that

[1] In [30], a great belt is called a great ring.

Lemma 3.2 *Suppose $\Phi = \{B_1, \ldots, B_n\}$ is a neighborhood system on U_d. Then for each $1 \leq i \leq n$, there is a great belt R_i such that a great sphere $\mathcal{G}$ intersects B_i iff $p_{\mathcal{G}}$ and $q_{\mathcal{G}}$ are contained in R_i. Moreover, the width of R_i is equal to $2r_i$, where r_i is the radius of B_i (see Figure 3).*

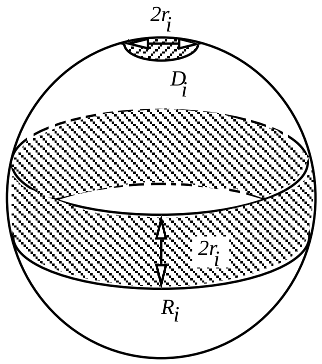

Figure 3: The great belt induced by B_i

Suppose $\Phi = \{B_1, \ldots, B_n\}$ is a neighborhood system on U_d. Let R_i be the great belt defined by B_i and $R = \{R_1, ..., R_n\}$. For each point x on U_d, let $\text{ply}(x)$ be the number of the great belts from R that contain x. By Lemma 3.1, $\text{ply}(x)$ is equal to the number of balls in Φ that intersect the great sphere $\mathcal{G}(x)$ associated with x.

Define

$$\text{ply}(U_d) = \frac{1}{A_d}\left(\int_{x \in U_d} \text{ply}(x)(dx)^d\right).$$

It follows from Proposition 3.1, we have

Proposition 3.3 *The above defined $\text{ply}(U_d)$ is equal to the expected intersection number of when a great sphere U_d is chosen uniformly at random according to the Euclidean measure.*

Lemma 3.4 *Suppose $\Phi = \{B_1, \ldots, B_n\}$ is a neighborhood system on U_d. Let R_i be the great belt defined by B_i. Then*

$$\text{ply}(U_d) = \frac{1}{A_d}\left(\sum_{i=1}^{n} \text{volume}(R_i)\right)$$

Proof: For each $1 \leq i \leq n$, for each $x \in U_d$, Let $\Phi_i(x)$ be the function which takes value 1 if $\mathcal{G}(x)$ intersects B_i and 0 otherwise.

$$\begin{aligned}
\text{ply}(\Phi) &= \frac{1}{A_d}\left(\int_{x \in U_d} \text{ply}(x)(dx)^d\right) \\
&= \frac{1}{A_d}\left(\int_{x \in U_d} [|\{B_i : \mathcal{G}(x) \cap B_i \neq \emptyset\}|]\,(dx)^d\right) \\
&= \frac{1}{A_d}\left(\sum_{i=1}^{n}\left[\int_{x \in U_d} \Phi_i(x)(dx)^d\right]\right)
\end{aligned}$$

$$= \frac{1}{A_d}\left(\sum_{i=1}^{n}[\text{volume}(R_i)]\right)$$

The first equality follows from the definition of ply(x); the second equality is obvious; and the third equality follows from Lemma 3.2. □

3.2 The total volume of great belts

Lemma 3.5 *Suppose $\Phi = \{B_1, \ldots, B_n\}$ is a k-ply neighborhood system on U_d. Let R_i be the great belt defined by B_i. Then*

$$\sum_{i=1}^{n} \text{volume}(R_i) = O\left(k^{1/d} n^{1-1/d}\right).$$

Proof: Let π be the stereographic projection which maps $\mathbb{R}^d$ onto U_d. Let Φ' be the pre-image of Φ in $\mathbb{R}^d$, i.e., $\Phi' = \{B'_1, \ldots, B'_n\}$ such that $\pi(B'_i) = B_i$. Clearly, Φ' is a neighborhood system in $\mathbb{R}^d$, whose density is also k. Let f'_i be the local density function defined on B'_i as in Section 2 (immediately after Lemma 2.6) and let $f' = L_{d-1}(f'_1, \ldots, f'_n)$. It follows from Lemma 2.8 that

$$\text{Total-Volume}(f') = O\left(k^{1/(d-1)} n\right).$$

Let f be the density function on U_d obtained from f' via the stereographic projection π, after changing of variables. We have

$$\text{Total-Volume}(f) = \text{Total-Volume}(f') = O\left(k^{1/(d-1)} n\right).$$

It follows from Lemma 2.3, that $\text{avg}(f) = O\left(k^{1/d} n^{1-1/d}\right)$. Recall that $\text{avg}(f)$ denotes the average f-measure of great spheres of U_d. We now relate $\text{avg}(f)$ with $\text{avg}(\Phi)$.

For each great sphere $\mathcal{G}$ of U_d, let S be the pre-image of $\mathcal{G}$ in $\mathbb{R}^d$, i.e., S is a $(d-1)$-sphere in $\mathbb{R}^d$ such that $\pi(S) = \mathcal{G}$. By the definition of f and the fact that π is conformal, $f'(S) = f(\mathcal{G})$. Furthermore, the intersection number of $\mathcal{G}$ over Φ is equal to the intersection number of S' over Φ'. It follows from Lemma 2.7 that $\text{ply}(p_{\mathcal{G}}) \leq c_1 k + c_2 f(\mathcal{G})$. In other words, for each point $x \in U_d$, $\text{ply}(x) \leq c_1 k + c_2 f(\mathcal{G}(x))$. Therefore,

$$\begin{aligned}
\text{ply}(U_d) &= \frac{1}{A_d}\left(\int_{x\in U_d} \text{ply}(x)(dx)^d\right) \\
&\leq \frac{1}{A_d}\left(\int_{x\in U_d} [c_1 k + c_2 f(\mathcal{G}(x))]\,(dx)^d\right) \\
&= \frac{1}{A_d}\left(c_1 A_d k + c_2 A_d \text{avg}(f)\right) \\
&= c_1 k + c_2 \text{avg}(f) \\
&= O\left(k^{1/d} n^{1-1/d}\right)
\end{aligned}$$

The lemma then follows from Lemma 3.4.

Notice that there might be at most k B_i's which cover the north pole so that the inverse of stereographic projection maps them to the complements of balls in $\mathbb{R}^d$. For technical simplicity, we can put them all into separator and hence can assume that there is not B_i that covers the north pole. $\square$

In summary, we show that $\text{ply}(U_d)$ is small, i.e., is bounded by $O(k^{1/d}n^{1-1/d})$. Hence, there must be some point x for which $\text{ply}(x)$ is small, and the dual great sphere $\mathcal{G}(x)$ gives a good separator. In the next section, we will show how to find such a point in deterministic linear time.

4 The Linear Time Algorithm

From the last section, we have a system of belts with total volume $\sum_{i=1}^{n} \text{volume}(R_i) \leq O(k^{1/d}n^{1-1/d})$. The *average ply* of the system is found simply by dividing this total volume by the volume of the sphere itself, which changes the result by a constant factor. Hence, there is a point on U_d which is contained in at most $O(k^{1/d}n^{1-1/d})$ great belts from $\{R_1, \ldots, R_n\}$, and in fact randomly choosing a point uniformly according Euclidean measure will lead to that value as the expected number of belts containing it. Our task now is to find such a point deterministically.

Let $\text{ply}(p)$ denote the number of great belts from $\{R_1, \ldots, R_n\}$ containing $p \in U_d$. For a given subset S of U_d, let $\text{ply}(S)$ be the average value of $\text{ply}(p)$ over those points in S; that is, $\text{ply}(S) = \sum_{i=1}^{n} \text{volume}(R_i \cap S)/\text{volume}(S)$. Then as above $\text{ply}(U_d) = O(k^{1/d}n^{1-1/d})$.

Now by using the reductions described in previous sections, the problem of computing a sphere separator of small intersection number is reduced to the following geometry problem:

- Given a set of great belts $\{R_1, \ldots, R_n\}$ on the unit sphere U_d, compute a point p on U_d such that $\text{ply}(p) \leq \text{ply}(U_d)$.

In [30], an $O(n^2)$ time deterministic algorithm is given for this problem. We now show a deterministic linear time algorithm. The basic idea is the following. We imbed the sphere U_d in $\mathbb{R}^{d+1}$. The $2n$ $(d-1)$-spheres forming the boundaries of great belts can then be extended to d-dimensional hyperplanes. We reduce the portion of the sphere under consideration, starting from the whole sphere, and proceeding through intersections of U_d with smaller and smaller simplices in $\mathbb{R}^{d+1}$. We maintain the invariant that in each simplex S the average covering number $\text{ply}(S \cap U_d)$ is small (i.e., $\text{ply}(S \cap U_d) \leq \text{ply}(U_d)$). As we proceed through smaller and smaller simplices, the simplices will be crossed by fewer and fewer hyperplanes corresponding to belt boundaries. Eventually we will reach a simplex F crossed by no belt boundaries at all: then every belt of our system will either cover $F \cap U_d$ or miss the simplex entirely. At such a point, the ply within $F \cap U_d$ will be a constant function and our desired output point p can be chosen as any point in $F \cap U_d$.

We find the sequence of simplices using a construction known as an ϵ-cutting [15, 21, 33]. This is a partition of space normally used for geometric divide-and-conquer techniques in problems such as simplex range searching [15, 21]. In our application, we instead use ϵ-cuttings as the basis of a prune-and-search technique, in which we reduce the problem to smaller and smaller simplices in which fewer and fewer great belts are involved. A *cutting* is a partition of

$\mathbb{R}^d$ (or in our case $\mathbb{R}^{d+1}$) into simplices. Given a set of n hyperplanes in $\mathbb{R}^d$, an ϵ-cutting is a cutting for which each simplex is crossed by at most ϵn hyperplanes. Matoušek [21] showed that an ϵ-cutting involving $O(\epsilon^{-d})$ simplices can be computed in time $O(n\epsilon^{-d-1})$. The number of simplices is within a constant factor of optimal, and the time bound is tight if one wants to enumerate the hyperplanes crossing each simplex. In our application, we compute an ϵ-cutting for the $2n$ hyperplanes in $\mathbb{R}^{d+1}$ coming from great belt boundaries, with ϵ some fixed constant to be determined later.

For any given simplex S in $\mathbb{R}^{d+1}$, we can compute the average number of great belts covering each point, $\mathrm{ply}(S \cap U_d)$, by finding the spherical polytopes formed by intersecting each great belt with the simplex, adding the volumes of all such polytopes, and dividing by the volume of the spherical polytope $S \cap U_d$. Each such polytope is an intersection of at most $d + 4$ disks in U_d (one per facet of S, two for the great belt boundaries) so its volume can be computed in constant time. There are $O(\epsilon^{-d})$ simplices for which we need to compute this quantity, and Matoušek's algorithm lists the $O(n\epsilon^{-d-1})$ incidences between simplices and the belt boundaries that cross them, so this computation takes $O(n\epsilon^{-d-1})$ time.

Lemma 4.1 *If U_d is partitioned by a cutting of $\mathbb{R}^{d+1}$ into simplices S_i, then some simplex has* $\mathrm{ply}(S_i \cap U_d) \leq ply(U_d)$.

Proof: The average number of belts covering points on the whole sphere, $\mathrm{ply}(U_d)$, can be computed as the weighted average $\mathrm{ply}(U_d) = \sum_{i=1}^{k} \mathrm{ply}(S_i \cap U_d)\mathrm{volume}(S_i \cap U_d)/\mathrm{volume}(U_d)$. The minimum is at most the average, so for some simplex S_i, $\mathrm{ply}(S_i \cap U_d) \leq \mathrm{ply}(U_d)$. □

Thus we can reduce the problem of searching for a point in U_d to that of searching for one in S_i. All but at most $2\epsilon n$ great belts (referred as *active belts*) either miss S_i entirely, or entirely cover S_i; in either case we need not account for them in future computation. Thus we have reduced the problem to one in which there are $O(\epsilon n)$ great belts.

Similarly, if we start with a simplex S with $\mathrm{ply}(S \cap U_d) \leq \mathrm{ply}(U_d)$, for which some number m of the great belts remain neither completely covering S nor completely avoiding S, we can return to $\mathbb{R}^{d+1}$, and compute an ϵ-cutting for the arrangement of the $2m$ hyperplanes that determine great belt boundaries. This cutting will partition S into $O(\epsilon^d)$ polytopes, which can be further partitioned into smaller simplices. We again compute the average ply within each smaller simplex, and a similar proof to that of Lemma 4.1 shows that some simplex has smaller average ply than that of S. Thus if we choose the simplex having the smallest average ply, we reduce the problem to a single smaller simplex S', for which $\mathrm{ply}(S' \cap U_d) \leq \mathrm{ply}(S \cap U_d) \leq \mathrm{ply}(U_d)$, and for which at most $2\epsilon m$ belts remain "active".

If we iterate this approach, we eventually reach a final simplex F in which no belts are active. Then $\mathrm{ply}(x)$ is a constant function for all points $x \in F$, and $\mathrm{ply}(x) = \mathrm{ply}(F)$. Any such point from this simplex gives the desired solution to our geometric problem, and hence (returning from dual points to their primal great spheres) the geometric separator we are seeking. We have proved the following results.

Theorem 4.2 *Suppose we are given a set R of n great belts on U_d. Then in time $O(n)$ we can compute a point $x \in U_d$, with* $\mathrm{ply}(x) \leq \mathrm{ply}(U_d)$.

Proof: At each stage in the series of reductions outlined above, the number of active belts will be reduced from some number m to $2\epsilon m$. We then choose $\epsilon = 1/4$, so that at each stage the number of belts becomes at most half the previous number. The total time for the reductions will be $\sum_{i=0}^{\log n} O(2^{-i} n \epsilon^{-d-1}) = O(n)$. □

Theorem 4.3 *Given a k-ply neighborhood system of n balls in $\mathbb{R}^d$, we can find an $O(k^{1/d} n^{1-1/d})$ sphere separator that δ-splits the system for any $(d+1)/(d+2) < \delta < 1$ in time $O(n)$.*

Proof: As outlined earlier, we lift the balls to a system of disks on a sphere in $\mathbb{R}^{d+1}$. We then compute a δ-centerpoint $((d+1)/(d+2) < \delta < 1)$ in linear time and use it to transform the system by a conformal map after which each great sphere δ-splits the system (by Lemma 2.6). As in Section 3 we now dualize the system of disks to a system of great belts; by Lemma 3.5 the average ply of this system of belts is $O(k^{1/d} n^{1-1/d})$. By Theorem 4.2, we can find a point of at most average ply in this belt system. This point is dual in our original disk system to a great sphere which cuts $O(k^{1/d} n^{1-1/d})$ disks and δ-splits the disks, so it is the separator we require.

□

5 Applications

The sphere separator theorem has many applications [30], especially in scientific computing and computational geometry [9, 10, 11, 13, 18, 28]. In this section, we focus on the application of the deterministic linear time sphere separator algorithm in computational geometry.

5.1 Separators for geometric graphs

As shown in [26, 30], the sphere separator results lead to vertex separators for various geometric graphs including the intersection graph and the k-nearest-neighborhood graph.

The *intersection graph* of a neighborhood system $\Phi = \{B_1, \ldots, B_n\}$ has vertices $V = \Phi$ and edges $E = \{(B_i, B_j) : B_i \cap B_i \neq \emptyset\}$. It simply follows from the definition of the intersection graph [30] that if S is a sphere that δ-splits the centers of a neighborhood system Φ, then $\{B_i : B_i \cap S \neq \emptyset\}$ is an $|\Phi_O(S)|$-separator that δ-splits the intersection graph of Φ. Therefore,

Theorem 5.1 *Given a k-ply neighborhood system Φ in $\mathbb{R}^d$, an $O(k^{1/d} n^{1-1/d})$-separator of the intersection graph of Φ can be computed in linear time deterministically.*

Suppose $P = \{p_1, \ldots, p_n\}$ is a set of n points in $\mathbb{R}^d$. For each $p_i \in P$, let $N(p_i)$ be a closest neighbor of p_i in P, where ties are broken arbitrarily. Similarly, for any integer k, let $N_k(p_i)$ be the set of k nearest neighbors of p_i in P; here too ties are broken arbitrarily.

A *k-nearest neighborhood graph* of $P = \{p_1, \ldots, p_n\}$ in $\mathbb{R}^d$, is a graph with vertices $V = P$, and edges $E\{(p_i, p_j) | p_i \in N_k(p_j) \text{ or } p_j \in N_k(p_i)\}$.

For each i, let B_i be the largest ball centered at p_i whose interior contains at most $k-1$ points from P. Then, $\{B_1, \ldots, B_n\}$ is called *the k-nearest neighborhood system* of P. It has been shown in [26, 30] that the k-neighborhood system of P is $\tau_d k$-ply, where τ_d is the kissing number in d dimensions, i.e., the maximum number of nonoverlapping unit balls in $\mathbb{R}^d$ that can be arranged so that they all touch a central unit ball [2].

By the definition of k-nearest neighborhood graph, there is an edge between balls B_i and B_j only if either p_i is in B_j or p_j is in B_i and hence each k-nearest neighborhood graph in $\mathbb{R}^d$ is a subgraph of a k-neighborhood system in $\mathbb{R}^d$.

Corollary 5.2 *Let P be a set of n points in $\mathbb{R}^d$. An $O(k^{1/d}n^{1-1/d})$-separator of the k-nearest neighborhood graph of P can be computed in $O(kn)$ time if the k-nearest neighborhood graph is given. Otherwise, it can be computed in $O(kn \log k + n \log n)$ time.*

The second part of the corollary follows from results of Vaidya [35] and Drysdale [3] that the k-nearest neighborhood graph of a set of points can be computed in $O(kn \log k + n \log n)$ time.

Notice that in many applications such as finite element and finite difference methods, only the graph and its embedding are given. Unlike the random algorithm of [23], we need to compute the neighborhood system before applying our deterministic linear time algorithm. If a graph G and its embedding is a subgraph of a k-nearest graph, it follows from [30] that G has linear number of edges. Hence we can use G itself to find the k-nearest neighborhood system in linear time. For each node, we need to perform a selection operation of its graph neighbor to determine who is its kth nearest neighbors. Hence in linear time we can determine the k-nearest neighborhood system of the embedding.

5.2 Separator Based Divide and Conquer

The sphere separator result for neighborhood systems closed under the subset operation immediately leads to divide and conquer recursive algorithms for a variety of applications [30, 7]. Frieze, Miller, and Teng [7] developed a sphere separator based divide and conquer paradigm for solving problems in geometry. At a high level, the separator based divide and conquer paradigm is very simple and intuitive. Given a neighborhood system Φ (explicitly or implicitly), it finds a sphere separator of a low intersection number and partitions the balls into two subsets of roughly equal size, and then recursively solves the problem associated with the two sub-systems; the solutions for the two sub-systems are then combined to a solution to the whole problem. Our new result removes the necessity of randomness from the separator based divide and conquer paradigm. We now illustrate some applications.

5.2.1 Neighborhood query problem

The *neighborhood query problem* is defined as: given a neighborhood system $\Phi = \{B_1, \ldots, B_n\}$ in d-space, preprocess the input to organize it into a search structure so that queries of the form "output all balls that contain a given point p" can be answered efficiently.

Like other geometry query problems, there are three costs associated with the neighborhood query problem: the *preprocessing* time $T(n)$ required to build the search structure, the *query* time $Q(n)$ required to answer a query, and the *space* $S(n)$ required to represent the search structure in memory.

The separator based divide and conquer construction for this problem is very simple [30, 7]: Given a k-ply neighborhood system Φ, a binary tree of height $O(\log n)$ is built to guide the search in answering a query. Associated with each leaf of the tree is a subset of balls in Φ, and the search structure has the property that for all $p \in \mathbb{R}^d$, the set of balls covering p can be found in one of the leaves. The binary search tree is constructed as follows:

Algorithm (Neighborhood_Querying)

1. Find a 'good' sphere separator S;
2. Let $\Phi_0 = \Phi_I(S) \cup \Phi_O(S)$ and $\Phi_1 = \Phi_E(S) \cup \Phi_O(S)$;
3. Recursively construct the search structure T_0 for Φ_0 and T_1 for Φ_1;
4. Construct a tree whose left subtree is T_0 and whose right subtree is T_1, and whose root stores the information of S, i.e., its center and its radius.

The recursive construction stops when the subsets have cardinality smaller than ck for some constant c dependent on d that is large enough so that our sphere separator algorithm can find a nontrivial separator for a set of at least ck balls.

To answer a query when given a point $p \in \mathbb{R}^d$, we first check p against S, the sphere separator associated with the root of the search tree. There are two cases: if p is either interior to S, or on S, we recursively search the left subtree of S; otherwise we search the right subtree of S. When we reach a leaf, we then check p against all balls associated with the leaf and output all those that cover p.

The result clearly is correct, because at each level of the tree, any ball that could cover p either is entirely on the same side of S as p itself, or it crosses S, and in either case it would be contained in the subproblem searched at the next level.

At each level of the data structure, the number of balls contained in a subproblem is reduced by a constant factor $\delta \approx (d+1)/(d+2)$, so the total number of levels is $O(\log_{1/\delta} n) = O(\log n)$. The time per query is $O(\log n)$ to traverse the separator tree, and $O(k)$ to select the balls that cover p among the $O(k)$ balls contained in the leaf of the tree, for a total of $O(\log n + k)$.

We next show that the space complexity of the algorithm is linear. As each internal node stores only a constant amount of information, and there are fewer internal nodes than leaf nodes, the total space will be proportional to the number of balls stored at leaf nodes. This in turn will be equal to the original number of balls, plus the number of times balls are split by the separators constructed in the course of the algorithm. We divide the construction of the data structure into stages, and count the amount of such storage added in each stage. We then show that this amount is linear for the entire algorithm.

In stage i, we consider those nodes in the data structure containing between $m_i = \delta^i n$ and $m_{i+1} = \delta^{i+1} n$ nodes. Note that each node is considered in a single stage, and the children of a node are considered in a different stage than the node itself. In stage i, for each node considered in that stage, we compute a separator of size $O(k^{1/d} m_i^{(d-1)/d})$. Thus the total storage increases in that stage by a factor of $1 + O(k^{1/d} m_i^{-1/d})$. The total storage for our data structure will end up being $n \prod_{i=0}^{O(\log n)} (1 + O(k^{1/d} m_i^{-1/d}))$. Reversing the order of the product and starting from $m_i \approx k$ gives $n \prod_{i=0}^{O(\log n)} (1 + O(\delta^{i/d}))$. Even if we remove the upper bound on this sum, we get $n \prod_{i=0}^{\infty} (1 + O(\delta^{i/d}))$. The last term is $O(1)$ for fixed $\delta < 1$, as can be seen by taking its logarithm $\sum \log(1 + O(\delta^{i/d})) = \sum O(\delta^{i/d})$ which is a geometric series. Another proof can be found in [30].

The time complexity then follows from the space complexity: the total time spent for a given ball at a given leaf of the data structure is proportional to the length of the path from that leaf to the root, which is $O(\log n)$. Since there are $O(n)$ balls stored at leaves, the total time is $O(n \log n)$.

Theorem 5.3 *A search structure for the neighborhood query problem with $Q(n) = O(k + \log n)$ and $S(n) = O(n)$ can be computed in $O(n \log n)$ time.*

5.2.2 Computing the ply

The construction above assumes that the value of k is known, but we can achieve the same bounds if we are only given a neighborhood system but not told its ply. In that case we cannot stop when $m \le ck$, since we do not know k, but we can instead find a good lower bound for k by examining the cardinality of the sphere separators produced by our algorithm. Indeed, this technique gives us a method for approximating the ply of the neighborhood system:

Theorem 5.4 *Given a neighborhood system, we can compute a number k' such that $k' \le k \le ck'$ for some absolute constant c dependent on d in time $O(n \log n)$.*

Proof: We perform the same recursion as in Algorithm Neighborhood_Querying, but modify the stopping condition so that the recursion terminates whenever the resulting separating sphere covers some set fraction $0 < \epsilon < 1$ of the points in the recursive subproblem. In this way we construct a separator tree such that the disks covering any single point will be found at some leaf of the tree, so in particular if the system has ply k, a point covered by k disks will be found in the separator tree at a leaf containing at least k of the disks. On the other hand, if the ply is k, our separator algorithm will find separators with quality better than ϵ for any set of more than ck disks, so every leaf must contain at most ck disks. Thus we can compute k' simply as the maximum number of disks contained at any leaf of the separator tree.

A similar analysis to that used to prove Theorem 5.3 also applies here and shows that the algorithm uses time $O(n \log n)$ and space $O(n)$. □

In contrast, the best previously known algorithm for finding the exact ply of a neighborhood system works by constructing the corresponding sphere arrangement, and takes time $O(n^d \log n)$. The approach above can also be used to improve this bound for exact computation of the ply of a neighborhood system.

We reduce the problem of computing the exact ply of a system of n balls to the same problem in $O(n/k)$ subproblems, each with $O(k)$ balls. We do this by finding the approximation above, and then computing the data structure of Theorem 5.3. This data structure is a separator tree with $O(n/k)$ leaves, each containing $O(k)$ balls. The ply of the system is the maximum ply in any leaf. Therefore by using the $O(n^d \log n)$ algorithm in each leaf, we have the following further result.

Theorem 5.5 *We can compute the ply of a neighborhood system in $O(n \log n + k^{d-1} n \log k)$ time.*

It worthwhile to mention that for variable dimension, computing the exact ply of a neighborhood system is NP-complete (the special case of halfspaces through the origin is equivalent to OPEN HEMISPHERE [17]).

5.2.3 Constructing intersection graphs

We next consider the problem of constructing the intersection graph of a given k-ply neighborhood system. Again, we use the sphere separator based divide and conquer paradigm.

The algorithm can be described as:

Algorithm (Intersection_Graph)

1. If $k = \Omega(n)$ test each pair of neighborhoods separately, else:
2. Find a 'good' sphere separator S;
3. Let $\Phi_0 = \Phi_I(S) \cup \Phi_O(S)$ and $\Phi_1 = \Phi_E(S) \cup \Phi_O(S)$;
4. Recursively construct the intersection graphs G_0 for Φ_0 and G_1 for Φ_1;
5. Remove all parallel edges in $G_1 \cup G_2$ to obtain the intersection graph G of Φ;

Since as previously analyzed the recursion ends in $O(n/k)$ subsets of $O(k)$ neighborhoods, at most $O(kn)$ edges are produced, and the first stages of the algorithm take time $O(kn + n \log n)$. The final stage (removing parallel edges) can be done in $O(kn)$ time by bucket sort; alternately we can use the following strategy to eliminate the generation of parallel edges: we generate all edges in Φ_0, but only those edges in Φ_1 that do not have both endpoints in S. We keep track of which separators each vertex is contained in using an $O(\log n)$-bit bit-vector per vertex in each leaf set, with each bit corresponding to a level of the recursion. Each bit is set if the vertex is part of a separator at some level and clear otherwise. For each leaf set S, we also keep a bit vector, with the bit for a level set if S is part of Φ_1 at that level, and clear if S is part of Φ_0. At the bottom level of the recursion, an edge is generated only if the bitwise and of the three vectors for its two endpoints and the set S is zero. Thus we can show

Theorem 5.6 *The intersection graph of a k-ply neighborhood system Φ in $\mathbb{R}^d$ can be computed in $O(kn + n \log n)$ time.*

Proof: We compute the separator tree data structure of Theorem 5.3, augmented by the bit vectors described above. At each level of the tree, modifying the bit vectors takes constant time per ball, and does not increase the total running time of Algorithm Neighborhood_Querying by more than a constant factor. Within each of the $O(n/k)$ leaves of the separator tree, there are $O(k)$ balls, so we can construct the neighborhood graph within a single leaf by testing all pairs in $O(k^2)$ time. The total time for these tests is thus $O(kn)$, which together with the $O(n \log n)$ bound for constructing the separator tree gives the stated result. $\square$

6 Parallel Algorithms

Very recently, Goodrich [12] has given a parallel algorithm for constructing ϵ-cuttings and ϵ-approximations, which for any fixed ϵ runs in $O(\log n)$ time using $O(n/\log n)$ EREW processors. We now describe how to use his algorithm to parallelize our deterministic separator construction. Randomized parallel separator algorithms and their applications were previously studied by Frieze *et al.* [7].

Recall that our algorithm has two main computational steps: first, we must compute an approximate centerpoint, so that we can lift our disk system to U_d in such a way that any great sphere δ-splits the system, and second, we use ϵ-cuttings to reduce the number of belts in the dual system that we must consider in finding a point of average ply. The other steps, of actually performing the lifting, dualizing to a system of great belts, and computing the average ply of simplices in the cuttings, are easy to parallelize and we will not discuss them in detail.

Theorem 6.1 *For any $(d+1)/(d+2) < \delta < 1$, we can compute a δ-centerpoint of a point set $S \subset \mathbb{R}^d$, in time $O(\log n)$ using $O(n/\log n)$ EREW processors.*

Proof: We use Goodrich's algorithm to compute an ϵ-approximation to the point set. An ϵ-approximation is a subset S' such that for any halfplane h, $-\epsilon < |h \cap S'|/|S'| - |h \cap S|/|S| < \epsilon$. We use $\epsilon = \delta - (d+1)/(d+2) = O(1)$. This approximation can be found with size ϵ^c for any $c > d$, in time $O(\log n \log 1/\epsilon) = O(\log n)$ and $O(n/\log n)$ processors [12]. Thus S' has constant size and its exact centerpoint can be found in constant time (even on a single processor). If p is an exact centerpoint of S', it must be a δ-centerpoint of S, as any halfspace through p must contain at most $(d+1)|S'|/(d+2)$ points of S', so by the definition of an ϵ-approximation it must contain at most $((d+1)/(d+2)+\epsilon)|S| = \delta|S|$ points of S. □

Our other problem, finding a point of average ply in a system of great belts, was solved in Section 4 by a prune-and-search technique involving ϵ-cuttings. These can also be computed quickly in parallel by Goodrich's techniques:

Lemma 6.2 (Goodrich [12]) *Given a collection of n hyperplanes in $\mathbb{R}^d$, we can compute an ϵ-cutting with $O(\epsilon^d)$ simplices, in time $O(\log n \log 1/\epsilon)$, using $O(n/\log n)$ EREW processors.*

At each stage in the algorithm of Section 4, we computed such a cutting for $\epsilon = 1/4$, so there are $O(\log n)$ stages. Directly applying Lemma 6.2 to the methods of Section 4 thus yields an efficient $O(\log^2 n)$-time parallel algorithm.

Theorem 6.3 *Given a k-ply neighborhood system of n balls in $\mathbb{R}^d$, we can find an $O(k^{1/d}n^{1-1/d})$ sphere separator that δ-splits the system of balls, for any $(d+1)/(d+2) < \delta < 1$, in time $O(\log^2 n)$ using $O(n/\log^2 n)$ EREW processors.*

Proof: The same time bound with $O(n/\log n)$ processors follows immediately from the bound of Lemma 6.2 on the work per stage.

To reduce the processor bound to $O(n/\log^2 n)$, we apply an old and standard technique in parallel algorithm design. We simply run the algorithm with fewer processors, achieving only linear slowdown by a result of Brent [1]. At stage i, there are $n_i = O(n/4^i)$ hyperplanes in the input. If $n_i \le n/\log n$, the $O(n/\log^2 n)$ processors available will still be enough to run the algorithm of Lemma 6.2in time $O(\log n)$. If $n_i > n/\log n$, the time to run the algorithm of Lemma 6.2 with $O(n/\log^2 n)$ processors is

$$O(\log n(1+\frac{n_i}{\log n_i}/\frac{n}{\log^2 n})) = O(\log n(1+\frac{n_i\log n}{n})) = O(\log n + \frac{\log^2 n}{4^i}).$$

If we add these times over the sequence of values of i, the $O(\log^2 n/4^i)$ terms add in a geometric series to $O(\log^2 n)$, the other terms also add to $O(\log^2 n)$, and the total time is $O(\log^2 n)$. $\square$

We can do slightly better than Theorem 6.3 by using an alternate parallel ϵ-cutting algorithm, also due to Goodrich [12], that instead of the above bound takes time $O(\log n+\log^2 1/\epsilon)$. If we let $\epsilon = \exp(-\sqrt{\log n})$, we reduce the number of belts by a larger factor at each stage, and thus only need $O(\log n/\log\log n)$ stages. The overall work is higher for such an algorithm, since we must compute the ply of more triangles, and also since the alternate cutting algorithm uses work $O(n\epsilon^{-c})$ for some large unspecified constant c. However by applying the previous algorithm with $\epsilon = 1/4$ for the first $O(\sqrt{\log n})$ stages, we can reduce the problem size sufficiently that the extra work in the later stages is linear in the original problem size. Thus we can solve the problem in $O(\log^2 n/\log\log n)$ time using $O(n\log\log n/\log^2 n)$ processors. We omit details as the improvement over Theorem 6.3 is small compared to the extra complication necessary in the algorithm.

7 Open Questions

One interesting problem that remains open is whether we can find in $O(n)$ time a point with the smallest ply when given a set of great belts on U_d. An affirmative answer will enable us to find the optimal sphere separator of a k-ply neighborhood system with respect to a given conformal map. However even for $d = 2$ the problem is n^2-hard (meaning that it is equivalent in complexity to a number of other well-known problems for which no $o(n^2)$ time solution is known) [8]. Is there an efficient approximation algorithm for this problem?

Acknowledgments

A preliminary version of this paper appeared in the 9th ACM Symposium on Computational Geometry [5]. David Eppstein's work was supported in part by National Science Foundation grant CCR-9258355, and performed in part while visiting the Xerox Palo Alto Research Center. Gary Miller's work was supported in part by National Science Foundation grant DCR-8713489. Part of Shang-Hua Teng's work was done at the Xerox Palo Alto Research Center. His work at MIT was supported in part by AFOSR F49620-92-J-0125 and Darpa N00014-92-J-1799. We would like to thank Steve Vavasis for helpful discussions. We would also like to thank two anonymous referees for their helpful comments.

References

[1] Brent, R. P.. "The parallel evaluation of general arithmetic expressions". *J. Assoc. Comput. Mach.*, 21: 201–206, 1974.

[2] Conway, J. H., and N. J. A. Sloane. *Sphere Packings, Lattices and Groups*. Springer-Verlag, 1988.

[3] Drysdale, S. "Solving closest-point problems by searching the Delaunay diagram". In *2nd MSI Workshop on Computational Geometry*, SUNY Stony Brook, 1992.

[4] Dubrovin, B. A., A. T. Fomenko, and S. P. Novikov. *Modern Geometry – Methods and Applications*. Springer–Verlag, 1984.

[5] Eppstein, D., G. L. Miller, and S.-H. Teng. "A deterministic linear time algorithm for geometric separators and its applications". In *9th Annual ACM Symposium on Computational Geometry*, pp. 99–108, 1993.

[6] Fredickson, G. N., and R. Janardan. "Separator-based strategies for efficient message routing". In *27st Annual Symposium on Foundation of Computation Science,IEEE*, 428–237, 1986.

[7] Frieze, A. M., G. L. Miller, and S.-H. Teng. "Separator based parallel divide and conquer in computational geometry". In *4th Annual ACM Symposium on Parallel Algorithms and Architectures*, pp 420-430, 1992.

[8] Gajentaan, A., and M. H. Overmars. "n^2-Hard problems in computational geometry". Technical Report RUU-CS-93-15, Utrecht University Department of Computer Science, 1993.

[9] George, J. A., "Nested dissection of a regular finite element mesh". *SIAM J. Numerical Analysis*, 10: 345–363, 1973.

[10] George, A and J. W. H. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, 1981.

[11] Gilbert, J. R. and R. E. Tarjan. "The analysis of a nested dissection algorithm". *Numerische Mathematik*, 50(4):377–404, 1987.

[12] Goodrich, M. T. "Geometric partitioning made easier, even in parallel". In *9th Annual ACM Symposium on Computational Geometry*, pp. 73–82, 1993.

[13] Hammond, S. and R. Schreiber. "Solving unstructured grid problems on massively parallel computers". Technical Report TR 90.22, Research Institute for Advanced Computer Science, 1990.

[14] Hardy, G., J. E. Littlewood and G. Pólya. *Inequalities* Second edition, Cambridge University Press, 1952.

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

[16] Hilbert, D. and S. Cohn-Vossen. *Geometry and the Imagination*. (Translated by P. Nemenyi), Chelsea Publishing Company, New York, 1952.

[17] Johnson, D. S., and F. P. Preparata. "The densest hemisphere problem". *Theoretical Computer Science* 6: 93–107, 1978.

[18] Lipton, R. J., D. J. Rose, and R. E. Tarjan. "Generalized nested dissection". *SIAM J. on Numerical Analysis*, 16:346–358, 1979.

[19] Lipton, R. J. and R. E. Tarjan. "A separator theorem for planar graphs". *SIAM J. of Appl. Math.*, 36:177–189, April 1979.

[20] Lipton, R. J. and R. E. Tarjan. "Applications of planar separator theorem". *SIAM J. Comput*, 9(3): 615–627, August 1981.

[21] Matoušek, J. "Approximations and optimal geometric divide-and-conquer". In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*. 505–511, 1991.

[22] Miller, G. L. and S.-H. Teng. *Center Points and Point Divisions.* manuscript, School of Computer Science, Carnegie Mellon University, 1990.

[23] G. L. Miller, S.-H. Teng, W. Thurston, and S. A. Vavasis. Automatic mesh partitioning. In A. George, J. Gilbert, and J. Liu, editors, *Sparse Matrix Computations: Graph Theory Issues and Algorithms*, IMA Volumes in Mathematics and its Applications. Springer-Verlag, 1993.

[24] G. L. Miller, S.-H. Teng, W. Thurston, and S. A. Vavasis. Geometric Separators for Finite Element Meshes. submitted to *SIAM J. Scientific Computing*, 1994.

[25] Miller, G. L. and W. Thurston. "Separators in two and three dimensions". In *ACM STOC*, pages 300–309, Baltimore, May 1990.

[26] Miller, G. L., S.-H. Teng, and S. A. Vavasis. "An unified geometric approach to graph separators". In *32nd Annual Symposium on Foundations of Computer Science, IEEE*, 538–547, 1991.

[27] Miller, G. L. and S. A. Vavasis. "Density graphs and separators". In *Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 331–336, San Fransciso, January 1991. ACM-SIAM.

[28] Pan, V. and J. Reif. "Efficient parallel solution of linear systems". In *ACM STOC*, pages 143–152, Providence,RI, May 1985.

[29] Strang, G., and G. J. Fix. *An Analysis of the Finite Element Method*, Prentice-Hall, 1973.

[30] Teng, S.-H. *Points, Spheres, and Separators: a unified geometric approach to graph partitioning.* Ph.D. Thesis, Carnegie Mellon University, CMU-CS-91-184, 1991.

[31] Teng, S.-H. *Lecture Notes on Computational Geometry,* Spring 1993 LCS, MIT 1993.

[32] Thurston, W. P. *The Geometry and Topology of 3-manifolds.* Princeton University Notes, 1988.

[33] Vapnik, V. N. and A. Ya. Chervonenkis. "On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.*, 16: 264-280, 1971.

[34] Vavasis, S. A. "Automatic domain partitioning in three dimensions". *SIAM J. Sci. Stat. Comp.*, 12: 950-970, 1991.

[35] Vaidya, P. M. "An $O(n \log n)$ algorithm for the All-nearest-neighbors problem". *Discrete & Computational Geometry*,4:101–115, 1989.