Efficient parallel computation of arrangements of hyperplanes in d dimensions*

Torben Hagerup
Fachbereich 10, Informatik, Universität des Saarlandes, D-6600 Saarbrücken

Hermann Jung
Sektion Mathematik, Humboldt-Universität Berlin, PF 1297, DDR-1086 Berlin

Emo Welzl
Institut für Informatik, Fachbereich Mathematik, Freie Universität Berlin,
Arnimallee 2-6, D-1000 Berlin 33

Abstract

We propose the first optimal parallel algorithm computing arrangements of hyperplanes in E^d ($d \geq 2$). The algorithm is randomized and computes the arrangement of n hyperplanes within expected logarithmic time on a CRCW-PRAM *with $O(n^d/\log n)$ processors.*

1 Introduction

A finite set H of n hyperplanes in E^d defines a dissection of E^d into at most $O(n^d)$ connected components of various dimensions. This dissection is called arrangement $\mathcal{A}(H)$ of H. Arrangements of hyperplanes are fundamental in combinatorial geometry (especially in the setting of cell complexes, see e.g. [15]) as well as in computational geometry, due to numerous applications [10].

For $d = 2$, a topological sweep technique yields an optimal sequential algorithm [11] which computes the arrangement of n hyperplanes in time $O(n^2)$ and working space $O(n)$. For $d > 2$, the only known algorithm, computing the arrangement within optimal $O(n^d)$ time, is based on an incremental approach [13]. Both algorithms are inherently sequential.

So far, there is no known deterministic parallel algorithm which computes $\mathcal{A}(H)$ in $o(n^d)$ time with optimal time-processor product[1] $O(n^d)$. We propose the first randomized parallel algorithm which achieves simultaneously a nontrivial time speed up and an optimal time-processor product.

There are only few known techniques in the design of efficient parallel algorithms. These are, in particular, the reduction to parallel prefix computation [14,17], divide-and-conquer [1] and the sequential subsets technique [8], in which one stops a divide-and-conquer recursion when the subproblems are of small size, and solves all the subproblems sequentially, one processor per subproblem.

Applying the divide-and-conquer approach or the sequential subsets technique in the design of randomized parallel algorithms is not as easy as might appear at first glance. Randomization has been successfully used in a wide number of applications and has recently been applied to design efficient sequential algorithms in computational geometry [6,7,16]. Many of these algorithms randomly choose a subset of the input set that is used to partition the problem into smaller ones, where the expected size of the subproblems can be well bounded from above. Crucial to the parallelization of this randomized approach is that the expected parallel time is bounded only by the sum of the maxima of the expected time bounds of the processors in each level of recursion, while the expected time of a sequential recursive algorithm is the sum of expected time of all individual steps. This is the main reason why Reif and Sen [18] introduced the so called polling technique, which allows to choose among several random samples the most balanced one, thus decreasing the probability of a bad sample.

We propose another approach to circumvent the problems of the recursive divide-and-conquer technique.

*Research partially supported by the DFG, SFB 124, TP B2, VLSI Entwurfsmethoden und Parallelität and by the ESPRIT II Basic Research Actions Program of EC under contract no. 3075 (project ALCOM)

[1] Recently, we have been informed that R.Anderson, P.Beame and E. Brisson achieve deterministic time $O(\log n \log^* n)$ with $O(n^d/\log n)$ processors, "Parallel Algorithms for Arrangements," *these proceedings.*

We randomly choose a "large" subset of the input set and solve the problem on this subset, applying a suboptimal parallel algorithm, which yields a subdivision of the problem into a large number of very small subproblems with total complexity proportional to the complexity of the original one. Moreover, all these subproblems can be solved independently and can be easily composed to the general solution. The size of the random sample depends directly on the factor the time-processor product of the known suboptimal parallel algorithm is apart from the optimum.

Thus, the proposed new algorithm mainly consists of a deterministic suboptimal parallel algorithm and an optimal sequential algorithm, applied to each of the subproblems. This approach yields algorithms which are considerably easier to implement then those following the recursive divide-and-conquer method of Reif and Sen.

The optimal parallel computation of arrangements leads to randomized optimal parallel algorithms for several other problems. We claim that the above approach can be applied not only to the computation of arrangements but to a number of other problems.

The paper is organized as follows. Basic geometrical and combinatorial definitions are introduced in Section 2. Section 3 outlines the algorithms and their analysis, and Section 4 discusses applications and open problems.

2 Preliminaries and Basic Definitions

We assume some familarity with basic notions about arrangements of hyperplanes in E^d, see, e.g., [10] for a general treatment of arrangements. Hence we only briefly recall some definitions.

A set H of n hyperplanes in E^d dissects the d-dimensional Euclidean space into several connected components, the faces of the arrangement $\mathcal{A}(H)$. The dimension $\dim(f)$ of a face f is the smallest dimension of an affine subspace containing f. f is said to be a k-face of $\mathcal{A}(H)$ if f is a face of $\mathcal{A}(H)$ with $\dim(f) = k$. 0-faces and 1-faces are called vertices and edges, resp., whereas $(d-1)$-faces and d-faces are the facets and cells, resp., of an arrangement $\mathcal{A}(H)$.

Cells can be easily understood as intersections of halfspaces, defined by the hyperplanes in H. A k-face f $(0 \leq k < d)$ is contained in the intersection of (at least) $d-k$ hyperplanes of H. Following these definitions, faces of $\mathcal{A}(H)$ are relatively open sets.

An arrangement $\mathcal{A}(H)$ will be represented by its combinatorial structure, the incidence graph of $\mathcal{A}(H)$. The incidence graph G of $\mathcal{A}(H)$ is a directed graph with the faces of $\mathcal{A}(H)$ as nodes and an edge between faces f and g if f is on the boundary of g and $\dim(f) = \dim(g) - 1$. In this case we say that f is a subface of g, g is a superface of f, or f and g are incident. The incidence graph will be given by the list of (marked) nodes and, for each node, by the lists of its subfaces and superfaces.

For the sake of simplicity, we assume in the sequel that the hyperplanes in H are in general position, by which we mean that

1. any d hyperplanes intersect in a common point, but no $(d+1)$ do, and
2. no pair of vertices in $\mathcal{A}(H)$ is contained in a horizontal hyperplane.

Moreover, we consider a simplex s_0 with the set H_0 of bounding hyperplanes, such that all vertices of $\mathcal{A}(H)$ are contained in s_0 and $H \cup H_0$ is in general position. From now on, whenever we talk about an arrangement $\mathcal{A}(H')$ $(H' \subseteq H)$ we mean the arrangement $\mathcal{A}(H' \cup H_0)$, ignoring the part outside s_0 . Clearly the part inside s_0 contains already all incidence information about $\mathcal{A}(H')$. s_0 is added only to ensure that every face, ever considered in the algorithms, is bounded and contains a lowest vertex. The general position assumption is no restriction due to perturbation techniques [12], [20]. The following upper bound is basic to the complexity analysis of the algorithms. For proofs we refer to [13] or to the textbook of Edelsbrunner [10].

Lemma 1 *Given a set H of n hyperplanes in E^d, we can bound the number of nodes and edges in the incidence graph of $\mathcal{A}(H)$ by $O(n^d)$. Furthermore, the number of faces of $\mathcal{A}(H)$, visible from any fixed hyperplane is bounded by $O(n^{d-1})$.*

In order to motivate further definitions, let us first outline the general scheme of the algorithm, described in detail in Section 3.

Given a set H of n hyperplanes in general position,

(1) choose a set $R \subseteq H$ of r hyperplanes at random;

(2) compute the arrangement $\mathcal{A}(R)$ and the arrangement $\mathcal{A}(H)$, restricted to facets in $\mathcal{A}(R)$, by a suboptimal algorithm, and distribute the hyperplanes in $H \setminus R$ to the cells in $\mathcal{A}(R)$;

(3) triangulate $\mathcal{A}(R)$ and distribute the hyperplanes in $H \setminus R$ to the cells of the triangulated arrangement $\mathcal{A}(R)$;

(4) compute the arrangement $\mathcal{A}(H)$ in each of the above cells independently by an optimal sequential algorithm.

Following this scheme, we have to define the subdivision, computed in (2), and we have to specify the triangulation, needed in (3).

The arrangement $\mathcal{A}(H)$ restricted to facets in $\mathcal{A}(R)$ (plus the cells in $\mathcal{A}(R)$) will be denoted by $\mathcal{A}(H/R)$. Formally, the set of faces F of $\mathcal{A}(H/R)$ is defined as follows:

$$\begin{aligned} F = \{&f : f \text{ is a face in } \mathcal{A}(H) \text{and} \\ &f \subseteq h, \text{for some hyperplane } h \text{ in } R\} \\ \cup\{&f : f \text{ is a cell in } \mathcal{A}(R)\}. \end{aligned}$$

Two faces f, g of $\mathcal{A}(H/R)$ are incident in $\mathcal{A}(H/R)$ if

(i) f is incident upon g in $\mathcal{A}(H)$ or

(ii) f is a facet in $\mathcal{A}(H)$, lying on the boundary of cell g in $\mathcal{A}(R)$.

Obviously, for $R = H$ we have $\mathcal{A}(H/R) = \mathcal{A}(H)$ while, in general, $\mathcal{A}(H)$ is a refinement of the subdivision $\mathcal{A}(H/R)$. Furthermore, we define the *skeleton* $SK(H/R)$ of $\mathcal{A}(H/R)$ to be the undirected graph induced by the vertices and by the edges in $\mathcal{A}(H/R)$. The skeleton of $\mathcal{A}(H/R)$ contains the complete combinatorial structure of $\mathcal{A}(H/R)$, provided we add the information about the set of defining hyperplanes to each vertex v.

This leads to the definition of the *extended skeleton graph* of $\mathcal{A}(H/R)$ to be an undirected graph with the set of nodes $\{(v, f) : v$ is a vertex in $\mathcal{A}(H/R)$ and f is a k-face of $\mathcal{A}(H/R)$ with v on its boundary$\}$, and $\{(u, f), (v, g)\}$ is an edge in this graph if $f = g$ and u and v are connected by an edge in the skeleton of $\mathcal{A}(H/R)$.

Finally let us specify the triangulation, we are going to compute. This triangulation is called *bottom vertex triangulation* (*bv-triangulation*, for short) and is uniquely defined as the result of the following algorithm:
for k=2 **to** d **do**
begin
triangulate all k-faces f by selecting the lowest (with respect to the x_d-coordinate) vertex v on the boundary of f and creating all k-simplices $\text{conv}(\{v\} \cup s)$, over all $(k-1)$-simplices in the (already triangulated) boundary of f which do not contain v itself on their boundary
end,
where $\text{conv}(\{v\} \cup s)$ denotes the convex hull of the set $\{v\} \cup s$. The above construction is unique and well-defined, if we recall that we consider an arrangements inside s_0 only, and $H \cup H_0$ is in general position.

3 The Algorithm

First we describe a suboptimal parallel algorithm computing an arrangement $\mathcal{A}(H)$ ($\mathcal{A}(H/R)$, resp.) of hyperplanes in E^d within logarithmic parallel time. The algorithm consists of the following steps ($n = \text{card}(H)$ and $r = \text{card}(R)$):

(1) compute all $O(n^{d-1})$ lines $l = h_1 \cap \ldots \cap h_{d-1}$, with $h_i \in H$, and all $O(n^{d-1}r)$ vertices of $\mathcal{A}(H/R)$;

(2) sort the vertices of $\mathcal{A}(H/R)$ along each of the lines, which yields the skeleton of $\mathcal{A}(H/R)$;

(3) compute the extended skeleton graph of $\mathcal{A}(H/R)$;

(4) compute the connected components of the extended skeleton graph of $\mathcal{A}(H/R)$, which yields the set of faces of $\mathcal{A}(H/R)$;

(5) compute the incidences between faces in $\mathcal{A}(H/R)$.

It is obvious how to compute (1) and (2) efficiently in parallel. The first step takes constant time on a PRAM with $O(n^{d-1}r)$ processors and the second step $O(\log n)$ time with the same number of processors.

Given any vertex v of $\mathcal{A}(H/R)$ and its set $H(v)=\{h_1, \ldots, h_d\}$ of defining hyperplanes , i.e. $\{v\} = \bigcap\{h : h \in H(v)\}$, it is easy to distinguish the $2^k\binom{d}{k}$ k-faces ($1 \leq k \leq d$) which have v on their boundary. Although we cannot easily compute the boundaries of these faces, it is possible to decide the equivalence between faces having vertices u and v on their boundary, which are connected by a edge in $\mathcal{A}(H/R)$. The computation time depends only on d, not on n, which is a consequence of the general position assumption. Hence, step (3) can be done within constant time using $O(n^{d-1}r)$ processors.

Applying the Shiloah/Vishkin algorithm [19], the connected components of the extended skeleton graph of $\mathcal{A}(H/R)$ can be computed within logarithmic time on a CRCW-PRAM with $O(n^{d-1}r)$ processors. These connected components define the faces of $\mathcal{A}(H/R)$.

Since we have already computed the incidence relation between vertices and k-faces ($k \geq 1$) of $\mathcal{A}(H/R)$ it is straightforward to extend this relation to compute the incidences between k- and $(k+1)$-faces ($0 \leq k < d$) with work $O(1)$ per vertex of $\mathcal{A}(H/R)$. Since the incidences between vertices and k-faces are given by lists of vertices on the boundary of a common face, and lists of k-faces, having a vertex as the intersection of their boundaries, it is easy to compute the lowest vertex on the boundary of for each face in $\mathcal{A}(H/R)$, and lists of hyperplanes h, intersecting the cells c in $\mathcal{A}(H/R)$, for all cells.

Both can be done within time $\log n$ and processor number $O(n^{d-1}r)$. Hence we can conclude:

Proposition 1 *Let H be a set of n hyperplanes in E^d in general position, and let R be a subset of H with $r = \text{card}(R)$.*

(i) The above algorithm computes $\mathcal{A}(H/R)$ within $O(\log n)$ time, using a CRCW-PRAM *with $O(n^{d-1}r)$ processors.*

(ii) The bv-triangulation of $\mathcal{A}(R)$ can be computed on a CRCW-PRAM *in time $O(\log n)$, using $O(r^d)$ processors.*

(iii) The hyperplanes in $H \setminus R$ can be distributed to all intersected cells in $\mathcal{A}(H/R)$ on an $O(n^{d-1}r)$ processor PRAM *in time $O(\log n)$.*

Now we can describe the whole scheme of the final algorithm. It is a two level application of the scheme proposed in Section 2. We need the second level to decrease the expected complexity (in terms of intersecting hyperplanes) of the remaining simplices, which corresponds directly to expected parallel computation time.

Given a set H of n hyperplanes in E^d, we compute the arrangement $\mathcal{A}(H)$ by the following steps (where c_1, $c_2 > 0$ are appropriately chosen constants, and $|s|$ denotes the cardinality of the set H_s of hyperplanes in H intersecting simplex s):

(1) select a random subset $R \subseteq H$ of $r = c_1 \frac{n}{\log n}$ hyperplanes;

(2) compute $\mathcal{A}(H/R)$ and the bv-triangulation of $\mathcal{A}(R)$ by the above suboptimal algorithm;

(3) distribute the hyperplanes in $H \setminus R$ to all d-simplices of the triangulated arrangement $\mathcal{A}(R)$;

(4) **for** all d-simplices s with $|s| > \log^{1/d} n$
do

(4.1) choose a random subset R_s of H_s of $r_s = c_2|s| \log|s| \log^{-1/d} n$ hyperplanes;

(4.2) compute $\mathcal{A}(H_s/R_s) \cap s$ and the bv-triangulation of $\mathcal{A}(R_s) \cap s$ by the above suboptimal algorithm;

(4.3) distribute the hyperplanes in $H_s \setminus R_s$ to all d-simplices of the triangulated arrangement $\mathcal{A}(R_s) \cap s$;
od

(5) **for** all d-simplices s (computed either in (2) or in (4.2)) with $|s| < \log^{1/d} n$
do

(6) compute the arrangement $\mathcal{A}(H) \cap s$ by an $O(|s|^d)$ time sequential algorithm [13];
od

(7) **for** all remaining simplices s
do

(8) compute $\mathcal{A}(H) \cap s$ applying the above suboptimal algorithm;
od

(9) compute $\mathcal{A}(H)$ by merging the subdivisions of the simplices.

Let us analyze the parallel computation time and the number of processors the implementation of the different steps takes. We do this by estimating the total amount of work $W(n)$ of the PRAM, which corresponds to the time-processor-product if the processors can be easily scheduled. This method goes back to an idea of Brent [8]. It can be proved that, given a parallel algorithm A which solves a problem on a $p(n)$ processor PRAM within time $t(n)$ and total work $w(n)$, we can design a parallel algorithm B which solves the same problem on a $q(n)$ processor PRAM within time $O(t(n) + w(n)/q(n))$. The only assumption about algorithm A is that we know the processor scheduling of A, i.e. that there is no computation overhead in scheduling the $q(n)$ processors of B to the operations , executed by $p(n)$ processors of A in one parallel time unit.

Most of the proposed algorithmic steps are either obvious to implement or are done by the mentioned other algorithms. It remains to explain, how the hyperplanes are distributed to the d-simplices (in step (3) or in step (4.3), resp.).

First we realize that the arrangement $\mathcal{A}(H/R)$ ($\mathcal{A}(H_s/R_s)$, resp.) already yields the distribution of the corresponding hyperplanes to the cells c of $\mathcal{A}(R)$ ($\mathcal{A}(R_s)$, resp.). The final distribution to the simplices is computed by testing intersection between each d-simplex s in cell c and each hyperplane h intersecting cell c. This can be clearly done in parallel with time-processor product proportional to the product of the number of d-simplices in cell c and the number of hyperplanes intersecting c.

Since the processor allocation is obvious for every single step, we analyze these steps by bounding simultaneously the total amount of work $W(n)$ and the parallel computation time $T(n)$ for each of them:

(1) $W(n) = O(n \log n)$, $T(n) = O(\log n)$, assuming the existence of appropriate random number generators.

(2) $W(n) = O(n^d)$, $T(n) = O(\log n)$, see Proposition 1.

(3) Let $\deg(c)$ denote the total number of faces on the boundary of a cell c. It is obvious that the number of simplices in a bv-triangulation of the cell c is proportional to $\deg(c)$, where the constant depends on the dimension d. Hence we can bound the total work by

$$W(n) = O\left(\sum_{h \in H \setminus R} \sum_{c \text{ cell in } \mathcal{A}(R) \atop h \cap c \neq \emptyset} \deg(c) \right)$$

$$= O\left(\sum_{h\in H\setminus R}\left(\frac{n}{\log n}\right)^{d-1}\right) \text{ (by Lemma 1)}$$
$$= O(n^d \log^{-(d-1)} n).$$

(4) Counting all intersections between the simplices and the hyperplanes in $H \setminus R$ reduces to list ranking [2,9] over all lists of hyperplanes , belonging to these simplices, i.e.

$$W(n) = O\left(\sum_{c\in\mathcal{A}(R)} |c| \deg(c)\right)$$
$$= O(n^d \log^{-(d-1)} n)$$

and $T(n) = O(\log n)$ (note that the sum in the estimate of (4) equals the sum that appears in (3)).

Let $\mathcal{S}$ denote the set of d-simplices in the bv-triangulation of $\mathcal{A}(R)$. Step (4) splits $\mathcal{S}$ into two subsets: $\mathcal{S}_1 = \{s : s \in \mathcal{S}, |s| > \log^{1/d} n\}$ and $\mathcal{S}_2 = \mathcal{S} \setminus \mathcal{S}_1$. Now, the analysis of (4.1)-(5) is similar to the analysis of the previous steps.

(4.1)

$$W(n) = O\left(\sum_{s\in\mathcal{S}_1} |s| \log n\right) = O(r^d \log n)$$

and $T(n) = O(\log n)$

(4.2) Following Proposition 1, we can bound

$$W(n) = O\left(\sum_{s\in\mathcal{S}_1} r_s |s|^{d-1} \log |s|\right)$$
$$= O\left(\sum_{s\in\mathcal{S}_1} |s|^d \log^2|s| \log^{-1/d} n\right)$$

and $T(n) = O(\log n)$.

(4.3) Similarly to the analysis of (3), we obtain

$$W(n) = O\left(\sum_{s\in\mathcal{S}_1} |s|^d \log^{d-1}|s| \log^{-(d-1)/d} n\right)$$

and $T(n) = O(1)$.

(5) As in the analysis of (4),

$$W(n) = O\left(\sum_{s\in\mathcal{S}_1}\sum_{c\in\mathcal{A}(R)\cap s} |c| \deg(c)\right)$$
$$= O\left(\sum_{s\in\mathcal{S}_1} r_s^{d-1}|s|\right)$$
$$= O\left(\sum_{s\in\mathcal{S}_1} |s|^d \log^{d-1}|s| \log^{-(d-1)/d} n\right)$$

and $T(n) = O(\log n)$. Let $\mathcal{S}(s)$ denote the set of d-simplices in the bv-triangulation of $\mathcal{A}(R_s) \cap s$. Again, (5) splits the sets $\mathcal{S}(s)$ into subsets $\mathcal{S}_2(s) = \{s' : s' \in \mathcal{S}(s) \text{ and } |s'| < \log^{1/d} n\}$ and $\mathcal{S}_1(s) = \mathcal{S}(s) \setminus \mathcal{S}_2$. Hence, we can bound

(6) $W(n) = O\left(\sum_{s\in\mathcal{S}_2} |s|^d + \sum_{s\in\mathcal{S}_1}\sum_{s'\in\mathcal{S}_2(s)} |s'|^d\right)$ and $T(n) = O(\log n)$.

(7) ,(8) Proposition 1 yields

$$W(n) = O\left(\sum_{s\in\mathcal{S}_1}\sum_{s'\in\mathcal{S}_1(s)} |s'|^d \log|s'|\right)$$

and $T(n) = O(\log n)$.

(9) This can be done with one processor per face in the computed subdivision. Each processor, associated with a k-face f ($0 \le k < d$) on the boundary of a d-simplex, has to remove the incidences between f and its superfaces and has to concatenate the incidence lists of superfaces and subfaces of the two $(k+1)$-faces in the neighbouring cells split by f. This merging procedure takes constant parallel time, provided we avoid conflicts in concatenating the lists by introducing dummy list elements, one for each of the $O(d)$ splitting k-faces f on the boundary of the simplex. Let the $(k+1)$-face g of $\mathcal{A}(H)$ be spread over ℓ simplices $s_1, \cdots, s_\ell$. Each simplex s_i is given by the tupel of its vertices $(p_1(i), \cdots, p_{d+1}(i))$, sorted accordingly to the lexicographic order of their coordinates. This representation of the simplices induces a lexicographic order over the set of all simplices. We avoid conflicts in merging $g_1 = g \cap s_1, s, g_\ell = g \cap s_\ell$ by obeying the rule, that the lists of superfaces and of subfaces, resp., of g_i are hooked onto an arbitrary g_j in one of the neighbored simplices s_j which are greater than s_i, with respect to this lexicigraphic ordering. Hooking simply means that the corresponding dummy list element of g_j is replaced by the list of g_i.

A final concurrent write and prefix computation removes multiple occurrences in these incidence lists and completes the computation of the arrangement. These steps take logarithmic time and a total work which is only a constant multiple of the size of the incidence graph of the subdivision computed in the previous steps (1)-(8). Beside the analysis of each single step, we have to guarantee the desired time-processor bound for the processor allocation procedures before every step. This can be easily done by list ranking [2,9] and parallel prefix computation [14,17], both in logarithmic time and processor-time product at most linear in the size of the (intermediatly) computed subdivisions and incidence graphs, resp.. The only more complicated cases

are the calls to the suboptimal parallel algorithms in (4.2), (8) and to the sequential algorithms in (6), resp.. In order to allocate the correct number of processors to the tasks (in (4.2) and in (8)) or to build blocks of tasks with evenly distributed total task length (in (6)), we have to bound the complexity of the called procedures in advance. Here we take the (in general, pessimistic) upper bounds, used in the analysis of the steps (4.2), (6), and (8), resp.. Since we have already computed the number $|s|$ of intersecting hyperplanes, these bounds are easy to derive. Hence, the computational complexity of the single steps dominates.

In order to complete the analysis of the described algorithm it remains to bound the expected values of the following functions:

(A) $E\left(\sum_{s\in\mathcal{S}_1}|s|^d\log^a|s|\log^{-b}n\right)$, for constants $a,b>0$;

(B) $E\left(\sum_{s\in\mathcal{S}}|s|^d\right)$;

(C) $E\left(\sum_{s\in\mathcal{S}_1}\sum_{s'\in\mathcal{S}(s)}|s'|^d\right)$;

(D) $E\left(\sum_{s\in\mathcal{S}_1}\sum_{s'\in\mathcal{S}_1(s)}|s'|^d\log|s'|\right)$.

First we introduce some more notations. Let p_1, p_2 be two vertices of the arrangement $\mathcal{A}(H\cup H_0)$, $p_1,p_2\in s_0$. W.l.o.g., let p_1 have smaller x_d-coordinate than p_2. The open line segment $\overline{p_1p_2}$ connecting p_1 and p_2 is called *bv-chord* in $\mathcal{A}(H)$, if the face in $\mathcal{A}(\{h : h\in H\cup H_0$ and $p_1\in h\})$ containing $\overline{p_1p_2}$ is disjoint from the horizontal hyperplane through p_1. Let $\mathcal{B}_H$ denote the set of all bv-chords in $\mathcal{A}(H)$. It is easy to see that $\mathcal{B}_H$ contains all edges introduced by the bv-triangulation of any of the arrangements $\mathcal{A}(R)$ for $R\subseteq H$.
Furthermore, $\mathcal{B}_H^i$ denotes the set of bv-chords in $\mathcal{A}(H)$ which are intersected by i hyperplanes in H (not counting those that determine the chord). Since all chords in $\mathcal{B}_H^0$ are edges of the bv-triangulation of $\mathcal{A}(H)$, we can conclude

Corollary 1 $\mathrm{card}(\mathcal{B}_H^0)=O(n^d)$, *where* $n=\mathrm{card}(H)$.

This corollary can be strengthened to

Lemma 2 *If H is a set of hyperplanes in E^d in general position, then* $\mathrm{card}(\mathcal{B}_H^d)=O(n^d)$.

Proof. Following the lines of Clarkson and Shor [7], we take a random sample R of H and estimate $\mathrm{card}(\mathcal{B}_H^d)$ in relation to $E(\mathrm{card}(\mathcal{B}_R^0))$. As we have seen already $\mathrm{card}(\mathcal{B}_R^0)=O(r^d)$, where $r=\mathrm{card}(R)$. We choose $r=\lfloor n/2\rfloor$. Then

$$O\left(n^d\right)=O\left(r^d\right)=E\left(\mathrm{card}(\mathcal{B}_R^0)\right)$$

$$\geq\sum_{j\geq 0}\binom{n-2d-j}{r-2d}\mathrm{card}(\mathcal{B}_H^j)\Big/\binom{n}{r}$$

$$\geq\binom{n-3d}{r-2d}\mathrm{card}(\mathcal{B}_H^d)\Big/\binom{n}{r}$$

$$=\Omega\left(\mathrm{card}(\mathcal{B}_H^d)\right).$$

(Since the general position assumption implies that the number of hyperplanes determining the two endpoints of a bv-chord is not greater than $2d$.) Hence $\mathrm{card}(\mathcal{B}_H^d)=O(n^d)$, with a constant exponential in d. □

Lemma 2 is basic to the proof of the following upper bounds.

Lemma 3 *If H is a set of n hyperplanes in E^d in general position and $R\subseteq H$ is a random subset of size r, then*

$$E\left(\sum_{B\in\mathcal{B}_R^0}|B|^d\right)=O\left(n^d\right),$$

where $|B|$ denotes the number of hyperplanes in H intersecting the chord B.

Proof. For the proof method, we refer again to [7].

$$E\left(\sum_{B\in\mathcal{B}_R^0}|B|^d\right)$$

$$=O\left(E\left(\sum_{B\in\mathcal{B}_R^0}\binom{|B|}{d}\right)\right)$$

$$=O\left(\sum_{B\in\mathcal{B}_H}\mathrm{Prob}(B\in\mathcal{B}_R^0)\binom{|B|}{d}\right)$$

$$=O\left(\sum_{B\in\mathcal{B}_H}\binom{n-t(B)-|B|}{r-t(B)}\binom{|B|}{d}\Big/\binom{n}{r}\right)$$

where $t(B)$ is the number of hyperplanes in H determining the two endpoints of B,

$$=O\left((n/r)^d\sum_{B\in\mathcal{B}_H}\binom{|B|}{d}\binom{n-t(B)-|B|}{r-t(B)-d}\Big/\binom{n}{r}\right)$$

$$=O\left((n/r)^d\sum_{B\in\mathcal{B}_H}\mathrm{Prob}(B\in\mathcal{B}_R^d)\Big/\binom{n}{r}\right)$$

$$=O\left((n/r)^dE(\mathrm{card}(\mathcal{B}_R^d))\right)=O\left(n^d\right),$$

by Lemma 2. □

Since every simplex has only a constant number of edges, we can conclude

Corollary 2 *If R is a randomly choosen subset of a set H of n hyperplanes in E^d in general position, then $E(\sum |s|^d) = O(n^d)$, where the sum is over the simplices s in the bv-triangulation of $\mathcal{A}(R)$, and $|s|$ denotes the number of hyperplanes in H intersecting s.*

Corollary 2 yields an $O(n^d)$ upper bound for both (B) and (C). Crucial to (A) and (D) it remains to bound the probability of simplices s with large sets of intersecting hyperplanes. We adapt some notation from Haussler and Welzl [16], as it is needed for our purposes. Given a set H of n hyperplanes and $\epsilon > 0$, we call a subset R of H an *ϵ-net of H*, if every open line segment that is crossed[2] by no hyperplane in R, is crossed by less than ϵn of the hyperplanes in H. In our previous setting, that means that if chord B is in $\mathcal{B}_R^0$, then it is crossed by at most ϵn of the hyperplanes in H. Applying results for range spaces of finite VC-dimension [16], or Clarkson's analysis [6], we obtain

Lemma 4 *Let H be a set on n hyperplanes in E^d. Then a random subset R of H of size $\Omega(\max\{\frac{1}{\epsilon}\log\frac{1}{\delta}, \frac{d}{\epsilon}\log\frac{d}{\epsilon}\})$ is an ϵ-net of H with probability at least $1-\delta$.*

Now Lemma 4 can be applied to bound the probability that the choosen random samples are not ϵ-nets, for

$\epsilon = \log^2 n/n$, in step (1), and for

$\epsilon = \log^{1/d} n/|s|$, in step (4.1).

For appropriate constants c_1, c_2 in the above algorithm, Lemma 4 implies

(i) $E(\max\{|s| : s \in \mathcal{S}\}) = O(\log^2 n)$ and

(ii) $\text{Prob}(\max\{|s'| : s' \in \mathcal{S}(s)\} > \log^{1/d} n) < \frac{1}{\log n}$, for $s \in \mathcal{S}_1$.

Thus we can bound (A) and (D) as follows:

(A)

$$E\left(\sum_{s\in\mathcal{S}_1} |s|^d \log^a|s| \log^{-b} n\right)$$

$$= O(n^d (\log\log n)^a \log^{-b} n)$$

by (i) and Corollary 2, and

(D)

$$E\left(\sum_{s\in\mathcal{S}_1}\sum_{s'\in\mathcal{S}_1(s)} |s'|^d \log|s'|\right)$$

$$= E\left(\sum_{s\in\mathcal{S}_1}\sum_{s'\in\mathcal{S}(s)} \text{Prob}(|s'| > \log^{1/d} n)\, |s'|^d \log|s'|\right)$$

$= O(n^d)$, by (ii) and Corollary 2.

Summarizing the previous estimations we can bound the total work of the algorithm by $O(n^d)$, which is optimal with respect to the output size. Together with the logarithmic computation time and the described processor allocation schemes, Brent's Lemma implies

Theorem 1 *The arrangement of n hyperplanes in E^d can be computed on a* CRCW-PRAM *with expected parallel time $O(\log n)$ and optimal processor number $O(n^d/\log n)$.*

4 Applications and open problems

Many problems formulated for point sets are easier to approach in dual space, where the configuration of points is mapped to an arrangement of the dual hyperplanes (for more details, see [10]). For some of them, the dualized version of the problem can be easily solved in parallel, provided the arrangement of hyperplanes is given. An example is the Minimum Measure Simplicies Problem [10], which is stated as follows:
Given a set of n points in E^d, identify $d+1$ of them such that the spanned simplex has minimum measure among all such simplices.

In other cases, the computation of the dual arrangement dominates the computation time of the sequential algorithm solving the problem, but it is not easy to design an optimal parallel algorithm which solves the dual problem for the given arrangement. Only recently, Atallah, Chen and Wagener [3] proposed an optimal parallel algorithm for the visibility of a simple polygon from a point. This algorithm can be applied for the optimal parallel computation of the visibility graph of n nonintersecting line segments in the plane, provided that the endpoints are sorted with respect to their polar coordinates around each of them. It is well known that these n sorted sequences of polar angles can be easily obtained from the n sequences of vertices on the lines in the dual arrangement. Thus, our algorithm yields the first optimal parallel algorithms solving both problems, i.e.

Corollary 3 *The following problems can be solved on a* CRCW-PRAM *in expected logarithmic time with optimal processor-time product:*

(i) computing the minimum measure simplices of n points in E^d ,

(ii) computing the visibility graph of n nonintersecting line segments in the plane.

[2] intersected by, but not contained in

We conjecture that a number of other problems can be solved efficiently in parallel, provided the arrangment of the dual hyperplanes is given. For more candidates, see [10] and [11].

Beside these applications to problems for point sets which dualize to problems formulated for arrangements of hyperplanes, we claim the proposed method of combining a suboptimal parallel algorithm with an optimal sequential one by a randomly choosen set of seperators to be a useful tool in the design of efficient parallel algorithm. Candidates for problems to be solved by similar techniques are those, where we know already about the power of (sequential) random sampling. Computing all intersections of n line segments in the plane ([5,7]) could be one such possible candidate.

Concerning the efficient parallel computation of arrangements of n hyperplanes in E^d two main problems remain open:

> Design a randomized parallel algorithm which, using $O(n^d/\log n)$ processors, terminates in time $\log n$ with probability at least $1-2^{-n^{O(1)}}$.
>
> Design a deterministic parallel algorithm which works in time $o(n^d)$, and which uses an optimal number of processors.

If we omit the second application of random sampling in our algorithm (i.e., step (4)), then we obtain in contrast to the first of the above open problems

Corollary 4 *There is a* CRCW-PRAM *with* $O(n^d/\log^4 n)$ *processors which computes the arrangement of* n *hyperplanes in* E^d *and terminates in time* $O(\log^4 n)$ *with probability* $1-2^{-n}$.

References

[1] A.Aggarwal, B.Chazelle, L.Guibas, C.O'Dunlaing, C.Yap, *Parallel Computational Geometry.* Algorithmica 3 (1988), 293-327

[2] R.J.Anderson, G.L.Miller, *Deterministic parallel list ranking.* Proc. AWOC'88, LNCS 319 (1988), 81-90

[3] M.J.Atallah, D.Z.Chen, H.Wagener, *An optimal parallel algorithm for the visibility of a simple polygon from a point.* submitted to J. ACM, 1989

[4] R.P.Brent, *The parallel evaluation of general arithmetic expressions.* J. ACM 21,2 (1974), 201-206

[5] B.Chazelle, H.Edelsbrunner, *An optimal algorithm for intersecting line segments in the plane.* Proc. 29th Ann. IEEE Symp. on Found. of Computer Sci. 1988, 590-600

[6] K.L.Clarkson, *New application of random sampling in computational geometry.* Discrete Comput. Geom. 2 (1987), 195-222

[7] K.L.Clarkson, P.W.Shor, *Applications of random sampling in computational geometry, II.* Discrete Comput. Geom. 4 (1989), 387-422

[8] R.Cole, U.Vishkin, *Deterministic coin tossing and accelerating cascades: micro and macro techniques for designing parallel algorithms.* Proc. 18th Ann. ACM Symp. Theory Comput. 1986, 206-219

[9] R.Cole, U.Vishkin, *Optimal parallel algorithms for expression tree evaluation and list ranking.* Proc. AWOC'88, LNCS 319 (1988), 91-100

[10] H.Edelsbrunner, *Algorithms in combinatorial geometry.* EATCS Monographs on Theoretical Computer Science, Springer Verlag, 1987

[11] H.Edelsbrunner, L.J.Guibas, *Topologically sweeping an arrangement.* Proc. 18th Ann. ACM Sympos. Theory Comput. 1986, 389-403

[12] H.Edelsbrunner, E.P.Mücke, *Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms.* Proc. 4th Ann. ACM Sympos. Comput. Geom. 1988, 118-133

[13] H.Edelsbrunner, J.O'Rourke, R.Seidel, *Constructing arrangements of lines and hyperplanes with applications.* SIAM J. Comput. 15 (1986), 341-363

[14] M.J.Fischer, L.Ladner, *Parallel prefix computation.* J.ACM 27, 4 (1980), 831-838

[15] B.Grünbaum, *Convex Polytopes.* John Wiley & Sons, London, 1967

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

[17] C.Kruskal, L.Rudolph, M.Snir, *The power of parallel prefix.* Proc. 1985 IEEE Int. Conf. on Parallel Proc., 180-185

[18] J.H.Reif, S.Sen, *Polling: A new randomized sampling technique for computational geometry.* Proc. 21st Ann. ACM Symp. Theory Comput. 1989, 394-404

[19] Y.Shiloach, U.Vishkin, *An O(logn) parallel connectivity algorithm.* J.Algorithms 3 (1982), 57-67

[20] C.K. Yap, *A geometric consistency theorem for a symbolic perturbation scheme.* Proc. 4th Ann. ACM Symp. Comput. Geom. 1988, 134-142