Separating Collections of Points in Euclidean Spaces

Ralph P. Boland[1] and Jorge Urrutia[2]

Abstract

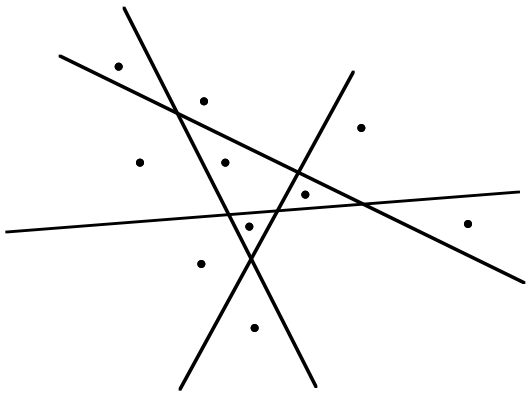
Given two disjoint convex sets A and B in E^d, a hyperplane h in E^d separates them if A lies on one of the half spaces defined by h while B lies on the complementary half space. Given a collection F of convex sets in E^d we say F is separated by a set of hyperplanes H if every pair of elements of F is separated by some hyperplane of H. We deal here with the case that the convex sets are all points. Let f(n,d) be the minimum number of hyperplanes always sufficient and occasionally necessary to separate n points in general position in E^d. We prove that $\lceil (n-1)/d \rceil \le f(n,d) \le \lceil (n-2^{\lceil \log(d) \rceil}) / d \rceil + \lceil \log(d) \rceil$ When d is even the lower bound can be improved to $\lceil n/d \rceil$ In the planar case this gives us $f(n,2) = \lceil n/2 \rceil$ We prove the upper bound by presenting an algorithm that generates a separating family of hyperplanes H that satisfies the upper bound. In dimension 2 the algorithm has a time complexity of O(n log(n)). Finally, we show that in the planar case H can be stored such that retrieving a line in H that separates a given pair of points from P can be found in O(1) expected time and worst case time of $O(\log^2(n))$.

Keywords: Algorithms, combinatorial problems, computational geometry, information retrieval, point sets, separability

1 Introduction

Let E^d represent the d-dimensional Euclidean vector space. Given two disjoint convex sets A and B in E^d, a hyperplane h in E^d separates them if A lies on one of the half spaces defined by h while B lies on the complementary half space. Given a collection F of convex sets in E^d, we say F is separated by a set of hyperplanes H if, for each possible pair of elements of F, there is a hyperplane of H that separates them.

There are many results in the literature concerning the separability of convex sets [1],[2],[3],[4],[6], and [11]. For instance , it is known that any family of n disjoint plane convex sets can always be separated with at most 3n-6 lines. In this paper we deal with the separability of point sets in general position in E^d. For any two points $a,b \in E^d$, we denote the open line segment joining them seg(a,b) and the closed line segment joining them seg[a,b]. We say that a set of points P in E^d is *separated* by a set of hyperplanes H in E^d if for every pair of points $a,b \in P$, there is a hyperplane h of H not containing a or b that intersects seg(a,b). We further say that h *separates* a from b. See Figure 1. We say that the points of P are in general position in E^d if no d + 1 points of P are in the same hyperplane in E^d. This restriction is important since if the elements of P can be located arbitrarily then we can trivially show that n - 1 hyperplanes are needed simply by placing n points in a straight line. All points sets considered here are assumed to be in general position.



A set of ten points separated by four lines
Figure 1

Let f(n,d) be the minimum number of hyperplanes always sufficient and occasionally necessary to separate n points in E^d where $n \ge d$. We show that $\lceil (n-1)/d \rceil \le f(n,d) \le \lceil (n-2^{\lceil \log(d) \rceil}) / d \rceil + \lceil \log(d) \rceil$ for d odd and that $\lceil n/d \rceil \le f(n,d) \le \lceil (n-2^{\lceil \log(d) \rceil}) / d \rceil + \lceil \log(d) \rceil$ for d even. From now on we assume that $n \ge d$. Theorem 3 will show that if $n < d$ then $f(n,d) = \lceil \log(n) \rceil$. The upper bounds of f(n,d) are determined by construction: we present algorithms that generate a solution within these upper bounds. In dimension 2 the algorithm can be made to run in O(n log(n)) time.

[1] School of Computer Science, Carleton University, Ottawa Ontario Canada.

[2] Department of Computer Science, University of Ottawa, Ottawa Ontario Canada.

For the planar case we can construct a data structure for storing the lines generated in the above algorithm that allows us to retrieve a line that separates two given points in worst case time $O(\log^2(n))$ and expected case time $O(1)$. This data structure is constructed in linear additional time.

2 Determination of Lower Bounds for f(n,d)

To determine a lower bound for $f(n,d)$ consider the moment curve

$$M_d = \{(t, t^2,..., t^d) \mid t \in R\} \quad (1)$$

and denote the point $(t, t^2,..., t^d)$ by $M_d(t)$. Let $\Gamma_n(d) = \{M_d(1), ..., M_d(n)\}$ be a set of n points on M_d. We determine how many times a hyperplane can intersect the moment curve. A hyperplane in E^d can be represented by an equation of the form:

$$a_d x_d + a_{d-1}x_{d-1} + \cdots + a_1 x_1 + a_0 = 0 \quad (2)$$

Combining (2) and (1) we get the equation

$$a_d t^d + a_{d-1}t^{d-1} + \cdots + a_1 t^1 + a_0 = 0 \quad (3)$$

Equation (3) is a polynomial of degree d and thus has at most d roots. This proves the following well-known result. See, for example [7], [8].

Lemma 1. Any hyperplane h in E^d intersects the moment curve M_d in at most d points. If h intersects M_d in exactly d points then h does not intersect M_d tangentially.

Using Lemma 1 we establish lower bounds for $f(n,d)$.

Theorem 2. $f(n,d) \geq \lceil (n-1)/d \rceil$ for d odd and $f(n,d) \geq \lceil n/d \rceil$ for d even.

Proof. Consider $\Gamma_n(d) = \{M_d(1),..., M_d(n)\}$. Let $M(i,j)$ be the section of M_d between points $M_d(i)$ and $M_d(j)$, $i < j$. Let e_i denote $seg[M_d(i), M_d(i+1)]$ and let $E = \{e_i \mid 1 \leq i < n\}$. Let H be any set of hyperplanes that separates $\Gamma_n(d)$. Observe that every $e_i \in E$ has to be crossed by some hyperplane of H.

We determine the maximum number of segments of E that any hyperplane $h \in H$ can intersect. Assume h intersects an element e_i of E. Then h separates $M_d(i)$ from $M_d(i+1)$. Therefore, since $M(i,i+1)$ is an arc joining $M_d(i)$ and $M_d(i+1)$, h intersects $M(i,i+1)$. (Note that we need not consider that h intersects $M(i,i+1)$ at points $M_d(i)$ or $M_d(i+1)$ since h is to separate these points.) Now, by Lemma 1, h intersects M_d at most d times; therefore h crosses at most d elements of E. Since $|E| = n - 1$, we have that $f(n,d) \geq \lceil (n-1)/d \rceil$.

If d is even then this result can be improved slightly. We observe that the line segment $f = seg[M_d(1), M_d(n)]$ must cross the hyperplane $h \in H$ that separates $M_d(1)$ from $M_d(n)$. If h intersects $M(1,n)$ d times then each intersection is a crossing of $M(1,n)$ by Lemma 1 and since d is even $M_d(1)$ and $M_d(n)$ lie on the same side of h, contradicting that h separates them. Therefore h crosses $M(1,n)$ at most $d - 1$ times. Therefore, when d is even, $f(n,d) \geq \lceil n/d \rceil$.

3 An Algorithm for Separating Points by Hyperplanes

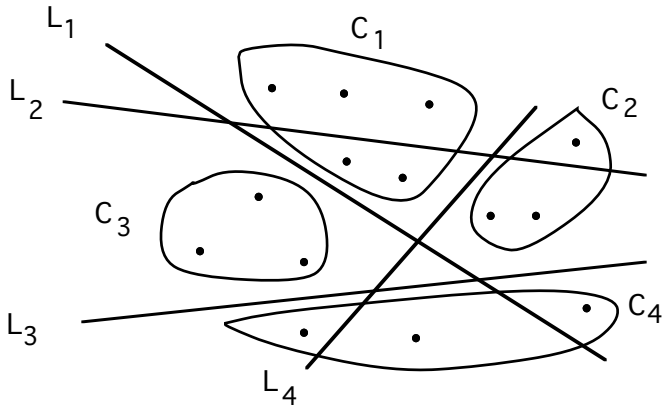
We now show that $f(n,d) \leq \lceil (n-2^{\lceil \log(d) \rceil}) / d) \rceil + \lceil \log(d) \rceil$ by constructing an algorithm that separates a point set P in E^d with $\lceil (n - 2^{\lceil \log(d) \rceil}) / d) \rceil + \lceil \log(d) \rceil$ hyperplanes. If a set of hyperplanes H partitions P into t points sets of cardinality $\lceil n/t \rceil$ or $\lfloor n/t \rfloor$ then we say that H *t-partitions* P. The first step of our algorithm will use the following result.

Theorem 3. Any set P of n points in E^d can be $2^{\lceil \log(d) \rceil}$-partitioned using at most $\lceil \log(d) \rceil$ hyperplanes.

This theorem is easily proven by repeated application of the following theorem [5], [12].

Theorem 4. Let $P_1, \ldots, P_d$ be d finite sets of points on E^d. There exists a hyperplane h that simultaneously bisects $P_1, \ldots, P_d$.

To explain the rest of our algorithm we will need a number of definitions and observations. Let P be any set of points in E^d and H be any set of hyperplanes in E^d. A cell CL of H is a connected subset of E^d such that, for any two points $a,b \in CL$, $seg(ab)$ does not intersect any hyperplane of H. We say that a subset C of P is a *cluster* of H if C is contained in a cell CL determined by some subset of H while $P - C$ is contained in the complement of CL. See figure 2. A partitioning $S = \{P_1, \ldots, P_k\}$ of P into disjoint subsets such that each $P_i \in P$ is a cluster of H is called a *cluster set* of (P,H). Note that if all the elements of S are singletons, then H separates P. We say that C is *separated* if $|C| = 1$ and *crowded* if $|C| \geq 2$. See figure 2.



A possible cluster set for a set of fourteen points P and four lines $H = \{L_1, L_2, L_3, L_4\}$. C_1 is a cluster of (P,H) determined by $\{L_1, L_4\}$.

Figure 2.

Given a cluster set S_i of (P,H) we say that a hyperplane h (not in H) refines S_i if there is a set of crowded clusters $C = \{C_1, \ldots, C_k\}$, $k \geq 1$, of S_i such that h splits each element C_i of C into two smaller non-empty clusters C_{i1}, C_{i2}. We then say that h is a *refine* of S_i and denote by $S_{i+1} = (S_i - C) \cup \{C_{1,1}, C_{1,2}, \ldots, C_{k,1}, C_{k,2}\}$. Note that S_{i+1} is now a cluster set of $(P, H \cup \{h\})$. Note also that C does not necessarily contain every cluster of S_i that can be split by h.

The ability to do refines, where as many as d clusters are split simultaneously, in dimension d is key to our point separation algorithm so we develop an algorithm that does this first. We observe that for any set of 2k points in E^d, $k \leq d$, grouped into k pairs $S = \{s_i \mid 1 \leq i \leq k\}$ in E^d, there exists a hyperplane h that simultaneously intersects the interior of each of the k line segments determined by each pair of S. It is easy to show that such a hyperplane can be found in time $O(d^3)$, i.e., in constant time for any fixed d. An example, when $k = d$, is the hyperplane determined by the midpoints of these line segments or a small perturbation of it. See Figure 3 (b,c). This observation gives us a simple algorithm for carrying out a refine, that splits k clusters, $k \leq d$, in E^d.

Algorithm Refine

Input: A cluster set S_j of a point set P in E^d, and k clusters $\{C_1, \ldots, C_k\}$ of S_j, $1 \leq k \leq d$, $|C_i| \geq 2$, $1 \leq i \leq k$.

Output: 1) A cluster set S_{j+1} that results from a refine of S_j in which each C_i, $1 \leq i \leq k$, is split into two clusters by a hyperplane h and 2) Hyperplane h.

1) For each C_i, $1 \leq i \leq k$, select any two points $p_{i1}, p_{i2} \in C_i$.
2) h := the hyperplane determined by the midpoints of $seg(p_{i1}, p_{i2})$, $1 \leq i \leq k$, and d-k arbitrarily chosen points (or a small perturbation of this hyperplane).
3) For each i, $1 \leq i \leq k$, partition C_i into two clusters C_{i1} and C_{i2} according to which side of h each point of C_i is on.
4) Return($(S_{j+1} := S_j - \{C_i, 1 \leq i \leq k\} \cup \{C_{ij}, 1 \leq i \leq k, 1 \leq j \leq 2\})$, h).

Our separation algorithm is also based on the following simple but important observation.

Observation 5. Let $S_0 = \{P\}$ be a cluster set of a point set P and $H = \{h_1, h_2, \ldots, h_{n-1}\}$ be a family of hyperplanes such that h_i is a refine of S_i, $i = 1, \ldots n-1$ that splits only one cluster C_i of S_i. Then H separates P.

Theorem 3, algorithm k-refine, and observation 5 are the tools we need to write our point separation algorithm.

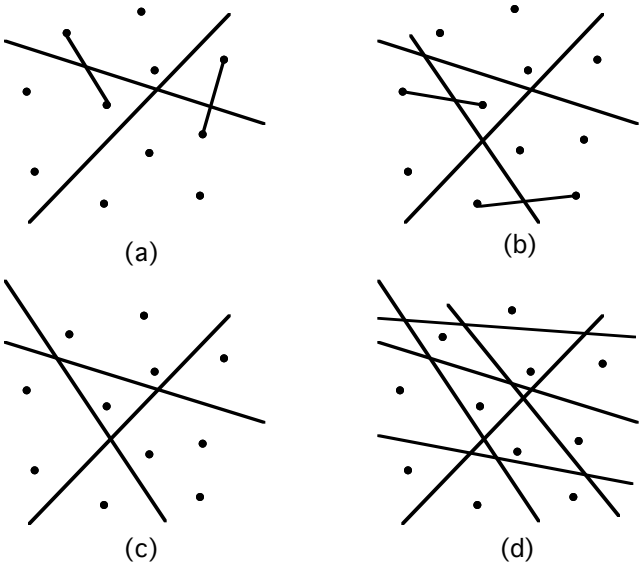
Algorithm Separate.

Input: A point set P in general position in E^d.

Output: A set of hyperplanes H that separates P.

1) $H_{init} := \{h_1, \ldots h_{\lceil \log(d) \rceil}\}$ where $\{h_1, \ldots h_{\lceil \log(d) \rceil}\}$ is a set of hyperplanes that $2^{\lceil \log(d) \rceil}$ partitions P into initial clusters C_i, $0 \leq i < 2^{\lceil \log(d) \rceil}$ labeled such that $|C_i| \geq |C_{i+1}|$, $0 \leq i < 2^{\lceil \log(d) \rceil} - 1$. The initial clusters are put in a queue Q such that C_i precedes C_{i+1}, $0 \leq i < 2^{\lceil \log(d) \rceil} - 1$ in Q.
2) $S_0 := \{C_i, 0 \leq i < 2^{\lceil \log(d) \rceil}\}$, $H_0 := H_{init}$, $k := 1$.
3) While there is a crowded cluster D_j, $1 \leq j \leq d$ contained in each of the first d initial clusters in Q do
   3.1) Use algorithm Refine to find a hyperplane h_k that simultaneously splits D_j, $1 \leq j \leq d$ and then split them.
   3.2) $S_k = S_{k-1} - \{D_j, 1 \leq j \leq d\} \cup \{$the new clusters created by the Refine$\}$
   3.3) Move the first d clusters in Q to the back of Q, maintaining their relative order.
   3.4) $H_k := H_{k-1} \cup h_k$, $k := k + 1$.
4) If there are m crowded clusters D_j, $1 \leq j \leq m$, remaining, then split each of them into two clusters using Refine. $H_k := H_{k-1} \cup h_k$ where h_k is the hyperplane produced by this Refine.

5) return H. See figure 3.



The sequence of the first three 2-splits ((a), (b), and (c)) of P and the final separating set of lines of P (d).
Figure 3

Theorem 6. Algorithm Separate separates a set of n points P in E^d by a set of hyperplanes H where $|H| = \lceil (n-2^{\lfloor \log(d) \rfloor}) / d) \rceil + \lfloor \log(d) \rfloor$

Proof. After Step 1 any set of hyperplanes H' that separates C_i, $0 \le i < 2^{\lfloor \log(d) \rfloor}$, together with H_{init} separates P. Let $C_{i,k}$, $0 \le i < 2^{\lfloor \log(d) \rfloor}$, denote the cluster set determined by cluster C_i and the refines that have been applied to subclusters of C_i by the first k hyperplanes produced in Separate after Step 1. Note that if the kth hyperplane does not split a subcluster of C_i then $C_{i,k} = C_{i,\,k-1}$. Let R(X) denote the number of refines used to separate cluster set X. By observation 5 $R(\{C_{i,0}\}) = |C_i|-1$, $0 \le i < 2^{\lfloor \log(d) \rfloor}$ and thus

$$R(S_0) = \sum_{i=0}^{2^{\lfloor \log(d) \rfloor}-1} |C_i|-1) = n - 2^{\lfloor \log(d) \rfloor}.$$

Therefore S_0 can be separated with $\lceil (n-2^{\lfloor \log(d) \rfloor}) / d) \rceil$ hyperplanes if each hyperplane h_k, $1 \le k \le \lceil (n-2^{\lfloor \log(d) \rfloor}) / d) \rceil$, except possibly the last, is produced by a refine which splits d clusters of S_{k-1}. This is accomplished by Separate since $| R(\{C_{i,k}\}) - R(\{C_{j,k}\}) | \le 1$, $0 \le i,j < 2^{\lfloor \log(d) \rfloor}$ after Step 1 (when k=0) and this equation remains invariant as hyperplanes are added by Separate. Thus the total number of hyperplanes produced by Separate is $\lceil (n-2^{\lfloor \log(d) \rfloor}) / d) \rceil + \lfloor \log(d) \rfloor$

Theorem 7. Not including Step 1 Algorithm Separate runs in time $O(n^2)$ in the worst case and O(n log(n)) in the expected case.

Proof. We only need to prove that after Step 1 Separate subdivides the clusters of P into smaller subclusters (until one point clusters are obtained) in a way equivalent to that of a version of Quicksort and thus has the same run time behavior.

To do this we choose a version of Partition that determines a value x to use to partition a subcollection N by randomly selecting two values $a,b \in N$, $a \le b$, and choosing x such that $a \le x \le b$. Of the values a,b let y be the one which is the furthest from the median of N. It is easily shown that Partition has expected worst case performance when x = y but that even with this bad choice for x the O(n log(n)) expected run time behavior of Quicksort is unchanged. We also note that the analysis of Quicksort is independent of the distribution of the values being sorted.

Now let C be any cluster of P generated during the execution of Separate, $|C| \ge 2$. Since $|C| \ge 2$, C must be split into two smaller subclusters of P (during an execution of Step 3.1 of Separate) using a hyperplane h_k.

Without loss of generality assume that h_k is chosen by having Refine first select a set T of d - 1 points, not in C, that are in general position with respect to the points in C. T may be chosen by an 'adversary to the algorithm' who wishes the algorithm to perform as slowly as possible. However the adversary must do so without knowledge of the next step. In the next step the algorithm determines a line segment s whose endpoints are two points selected at random from C. The algorithm now selects any point s_p on line segment s. The adversary is free to make as bad a choice for s_p as possible. Now, s_p together with the points in T determine h_k.

For each point q in E^d we associate a value $\theta(q,h_k)$ which is the angle between the hyperplane h_k and the hyperplane determined by $T \cup \{q\}$. Let $C(h_k)$ be the set of values determined by applying $\theta(p,h_k)$ to each point p in cluster C. Since the values $C(h_k)$ are totally ordered they can be partitioned by algorithm Partition. Let p_i, p_j be the endpoints of s such that $\theta(p_i, h_k) \le \theta(s_h, h_k) \le \theta(p_j, h_k)$ in this total ordering. We can assume that Partition partitions $C(h_k)$ using the value $\theta(s_p, h_k)$ since:

1) Choosing p_i, p_j randomly from C corresponds to choosing $\theta(p_i, h)$, $\theta(p_j, h)$ randomly from $C(h_k)$.

That is, for every random pair $p_i, p_j \in C$ there exists exactly one corresponding pair $\sigma(p_i, h_k), \sigma(p_j, h_k) \in C(h_k)$. and

2) s_p, on average, must be at least as good a choice as the choice with worst case expected behavior of the points on s (which is p_i or p_j).

Thus Partition partitions $C(h_k)$ into two parts $C_1(h_k)$ and $C_2(h_k)$ where C_1, C_2 are the two clusters produced by splitting C with h_k during the execution of Step 3.1 of Separate. Therefore Separate separates point sets at least as well as Quicksort sorts collections.

Let $P_1,\ldots,P_k$ be disjoint point sets in R^d, $k \le d$. By Theorem 4, it is possible to find a hyperplane h that simultaneously bisects $P_1,\ldots,P_k$. In R^2 and R^3 this process can be carried out in linear time [9],[10] while in R^d, $d \ge 4$ it can be carried out in time $O(n^{d-1-a(d)})$ where $a(d) > 0$ as d goes to infinity [9].

Using the algorithms in [9],[10] in Step 1 and in Step 3.1 of Separate (instead of Refine) gives us an $O(n \log(n))$ run time performance for Separate in R^2 and R^3 while for dimension d, $d \ge 4$ we get a run time complexity of $O(n^{d-1-a(d)}\log(n))$ where $a(d) > 0$ as d goes to infinity. We call the version of Separate in which these changes have been made *Bisect-Separate*.

4 A Data Structure for Retrieval of Separating Lines

Now consider that Bisect-Separate has separated a set P of n points with a set of hyperplanes (or lines) H. We are then asked: given any two points $a,b \in P$, find a hyperplane of H that separates a,b. We now discuss data structures for storing the hyperplanes of H so that this query can be done efficiently. It is clear from the results in [4] and [9] that in the general case a query can be done in $O(\log(n))$ time. We deal here only with the planar case and show that in linear time we can construct a data structure that allows a line that separates a,b to be retrieved in worst case time $O(\log^2(n))$ and expected time $O(1)$. In both cases a linear amount of space is needed for the data structures.

Let $P = \{p_1,p_2,\ldots,p_n\}$ be a set of n points on the plane and H be a set of lines (produced during the execution of algorithm Bisect-Separate) that separates P. We will decompose the plane into n regions each bounded by a convex polygon or a convex chain constructed from portions of the lines of H in such a way that each region contains exactly one point of P. For any point $p_i \in P$ we denote the region containing it $R(p_i)$ and denote the number of edges composing the boundary of $R(p_i)$ as $|R(p_i)|$. The regions will be constructed such that $MAX(|R(p_i)|, 1 \le i \le n) \le \log(n)$ and such that $\sum_{i=0}^{n} |R(p_i)| < 4n$.

We can now find a separating line between any two points $a,b \in P$. It is easily shown that one of the edges of the convex polygons or convex chains $R(a),R(b)$ separate a from b [4]. It is also well-known that this edge can be found in time $O(\log(MAX(|R(a)|,|R(b)|)))$. This in turn leads to our claimed results.

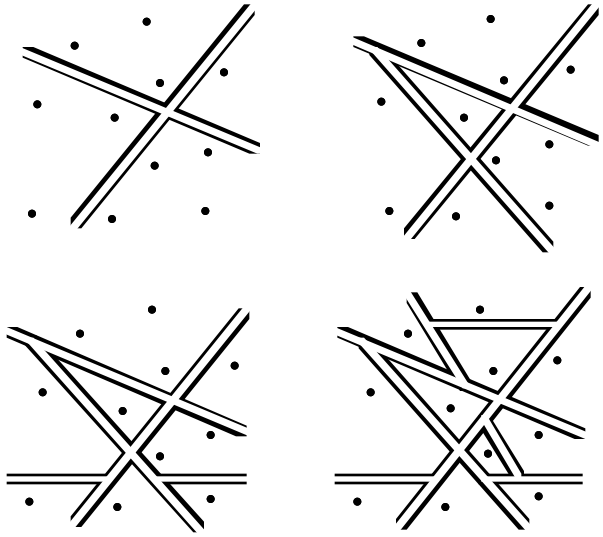
We construct our regions using a sequence of refinements, one for each line that is added to H by Bisect-Separate. Observe that each cluster C of (P,H) has a cell associated with it; that determined by the subset of H which determined C. We refer to that polygon or convex chain as $CL(C)$. Initially we have one region namely $CL(P)$. Each time Bisect-Separate refines a cluster C with a line σ, C is split into two clusters C_1,C_2. Our additional step at this time is to split $CL(C)$ into two new (possibly open) polygons $CL(C_1)$ and $CL(C_2)$ using line σ. Clearly $|CL(C)| + 4 \ge |CL(C_1)| + |CL(C_2)|$. Since $n-1$ refines of clusters are applied by Bisect-Separate to separate P, the number of edges needed to construct $CL(p_i) = R(p_i)$ for every separated point $p_i \in P$ is less than $4n$. See Figure 4.

To construct the boundaries in time $O(n)$ we maintain each boundary as a circular linked list of lines (each line can be thought of as an edge). Initially there are only two boundaries with each containing the initial line used to bisect P. For each cluster that is refined, we search its boundary for the edge intersection(s) with the line that is splitting the cluster, split the boundary into two new parts, and construct the new boundaries. In the worst case this cost is $O(k)$ where k is the number of edges on the boundary being split. For simplicity we assume that $|P| = n$ is a power of two. If not we can always add sufficient points to P until this is so. Let $G(n,k)$ be the set of clusters produced after each point p of P has participated in k refines. Then $|G(n,k)| = 2^k$ since each refine corresponds to a bisection of the point set containing p. Let $g(n,k)$ be the total number of edges in all the boundaries of the clusters of $G(n,k)$. Then $g(n,k) \le g(n,k-1) + 4 * 2^k$ since at most four new edges are added to cluster boundaries when a cluster is split and there are $|G(n,k-1)| = 2^{k-1}$ clusters to split in constructing $G(n,k)$ from $G(n,k-1)$. From this recurrence relation we determine that $g(n,k) < 2^{k+2}$. Let m be the total number of boundary edges to be processed during Bisect-Separate.

Then

$$m < \sum_{i=0}^{\lfloor \log(n) \rfloor} g(n,i)) = \sum_{i=0}^{\lfloor \log(n) \rfloor} 2^{i+2} < 8n.$$

Therefore the boundaries of regions for all points in P is constructed in amortized time O(n).



The polygons corresponding to each cluster after the addition of separating lines. The points and lines are as in Figure 3.
Figure 4

5 Conclusion and Open Problems

We have determined bounds on the number of hyperplanes always sufficient and occasionally necessary to separate n points in general position in E^d. For dimension two these bounds are tight. For higher dimensions the bounds are different only by a value logarithmic in the dimension. We proved the upper bound by presenting an algorithm that constructs a separating set of hyperplanes H for a given point set P satisfying the upper bound. Except for step 1, the algorithm is simple and efficient.

We also showed that in R^2 and R^3 that the time complexity of this algorithm can be reduced to O(n log(n)). In the planar case we also presented a data structure for storing H such that a line of H separating any two given points $a,b \in P$ can be found in expected time O(1) and worst case time $O(\log^2(n))$.

We conjecture that the problem of separating points in E^2 is $\Omega(n \log(n))$. However we have not been able to prove it.

References

[1] N. Alon, M. Katchalski, and W.R. Pulleyblank. "Cutting disjoint disks by straight lines", *Discrete Comput. Geom.* **4** (1989), 239-243.

[2] Avis, D. "On the partitionability of point sets in space". Proceedings of the 1st A.C.M. Symposium on Computational Geometry. 1985, 116-120.

[3] Avis, D. "Non-Partitionable Point Sets" *Information Processing Letters* **19** (1984) 125-129

[4] Boland, R. Separating Convex Sets. Masters Thesis Carleton University Ottawa, Ontario, Canada. 1992.

[5] Borsuk, K. "Drei Sätze uber die n-dimensionale euklidische Sphare". *Fund. Math.* **20** (1933).

[6] Rivera-Campo, E., Czyzowicz, J., Urrutia, J., and Zaks, J. "Separating Convex Sets On the Plane". *Discrete Comput. Geom.*, To appear.

[7] Gale, D. "Neighborly and Cyclic Polytopes" Proc. Synp. Pure Math. 7 (Convexity), p 225-232.

[8] Grünbaum, B. "Convex Polytopes", J. Wy;ey & Sons, London, New York, Sydney, 1967.

[9] Lo, C.Y., Matousek, J., and Steiger, W. "Ham Sandwich Cuts in R^d". Proceedings of the 24th Annual A.C.M. Symposium on Theory of Computing. (1992)

[10] Megiddo, N. "Partitioning with two lines in the plane". *Journal of Algorithms* **3** (1985) 430-433.

[11] Tverberg, H. "A separation property of plane convex sets", *Math. Scand.* **45** (1979), 255-260.

[12] Williard, D.E. Polygon retrieval. *SIAM Journal of Computing*. **11** (1982)