J. Symbolic Computation (1986) **2**, 171–178

Computing a Ham-sandwich Cut in Two Dimensions

H. EDELSBRUNNER† AND R. WAUPOTITSCH‡

† *Department of Computer Science, University of Illinois at Urbana-Champaign, 1304 West Springfield Avenue, Urbana, Illinois* 61801, *U.S.A.*

‡ *Institutes for Information Processing, Technical University of Graz, Schiesstattgasse 4a, A-8010, Graz, Austria*

(*Received* 25 *September* 1985)

Let B be a set of n_b black points and W a set of n_w white points in the Euclidean plane. A line h is said to bisect B (or W) if, at most, half of the points of B (or W) lie on any one side of h. A line that bisects both B and W is called a ham-sandwich cut of B and W. We give an algorithm that computes a ham-sandwich cut of B and W in $O((n_b+n_w)\log(\min\{n_b, n_w\}+1))$ time. The algorithm is considerably simpler than the previous most efficient one which takes $O((n_b+n_w)\log(n_b+n_w))$ time.

1. Introduction

Let B be a set of n_b black points and W a set of n_w white points in the Euclidean plane E^2. A line h is called a *bisector of* $B(W)$ if at most half of the points of $B(W)$ lie on any one side of h; points on h are counted on neither side. If n_b is odd, then any bisector of B contains at least one point of B. If h bisects both B and W, then h is called a *ham-sandwich cut of* B and W. It is not too hard to prove the existence of a ham-sandwich cut for any two finite sets in E^2.

The case where B and W can be separated by a line was treated several times in the literature of computational geometry (Willard, 1982; Edelsbrunner & Welzl, 1986; Cole *et al.*, 1984) before Megiddo (1985) presented an optimal algorithm which runs in $O(n_b+n_w)$ time. For the non-separated case, Cole (1984) described a general method which implies an $O((n_b+n_w)\log(n_b+n_w))$ algorithm—it might be hard to implement it, however, to say the least. Applications of these algorithms to other problems can be found in Atallah (1983) and Dobkin & Edelsbrunner (1984).

In this paper, we present a generalisation of Megiddo's algorithm to the case of non-separated point-sets. To allow a more intuitive presentation of the ideas, we treat the problem in its dual setting

> For any point $p=(p_x, p_y)$, we let $D(p): y=p_x x-p_y$ be its dual line and vice versa, that is, $D(D(p))=p$.

Write $H_1=\{h|h=D(p)$ and p in $B\}$ and $H_2=\{h|h=D(p)$ and p in $W\}$. For $h: y=ax+b$ a non-vertical line and $p=(p_x, p_y)$ a point, we say that p is *above*, *on*, or *below* h if p_y is greater than, equal to, or less than ap_x+b, respectively. The transform D preserves incidences as well as order relations:

0747–7171/86/020171+08 $03.00/0

OBSERVATION 1.1. Let p be a point and h a non-vertical line in E^2. Then p is above, on, or below h if and only if point $D(h)$ is above, on, or below line $D(p)$, respectively.

If a non-vertical line h bisects point-set B (or W), then point D(h) lies above at most half of the lines in $H_1(H_2)$ and below at most half of the lines in $H_1(H_2)$. If we restrict ourselves to sets B and W with odd numbers of points, then the dual points of bisecting lines define piecewise linear and continuous functions from x to y. The problem of finding a ham-sandwich cut thus boils down to finding a point of intersection of the function for H_1 and the function for H_2.

The organisation of this paper is as follows: Section 2 explains how we can avoid boring and tedious special cases like collinear points, etc., in our investigations as well as in the implementation. Sections 3 and 4 give the details of the algorithm that finds a ham-sandwich cut in time

$$O((n_b+n_w)\log(\min\{n_b, n_w\}+1)).$$

2. Getting Rid of Degenerate Cases

To simplify the presentation, sections 3 and 4 will assume that the points in B and W, respectively the lines in H_1 and H_2, are in *general position*, that is

(i) no three points in $S = B \cup W$ are collinear (so no three lines in $H = H_1 \cup H_2$ intersect in a common point);

(ii) no two points in S lie on a common vertical line (so no two lines in H are parallel); and

(iii) no two lines connecting respective two points of S are parallel or intersect on a vertical line through a point of S (so any two pairs of lines from H neither intersect on a common vertical line nor on a line parallel to another line in H).

If degeneracies as described occur, then we make use of a method called the simulation of simplicity (Edelsbrunner, 1986), that resolves the degeneracies according to the following rules which make essential use of indices 1 to n (with $n = n_b + n_w$) assigned to the n points or lines:

For $H = \{h_1, \ldots, h_n\}$ let h_i be given by $y = a_i x + b_i$. For any positive real number ε, we define

$$a_i(\varepsilon) = a_i + \varepsilon^{2^{2i}}, \quad b_i(\varepsilon) = b_i + \varepsilon^{2^{2i-1}}, \quad h_i(\varepsilon): y = a_i(\varepsilon)x + b_i(\varepsilon),$$

and

$$H(\varepsilon) = \{h_i(\varepsilon) | 1 \leqslant i \leqslant n\}.$$

Straightforward calculations imply that the lines in $H(\varepsilon)$ are in general position, provided ε is sufficiently small, and that lines in $H(\varepsilon)$ behave like the corresponding lines in H in all non-degenerate cases. In particular, a ham-sandwich cut h computed for the perturbed dual points corresponds to a ham-sandwich cut h' for B and W: if line h contains point $D(h_i(\varepsilon))$, then point $D(h_i)$ belongs to h' and if $D(h_i(\varepsilon))$ is above (below) h, then $D(h_i)$ is not below (above) h'. The perturbation is never computed but only used conceptually, that is, all decisions about the relative positions are based on the lines in $H(\varepsilon)$, for $\varepsilon > 0$ but sufficiently small.

To put the simulation of simplicity into use, we demonstrate how we can deal with lines in $H(\varepsilon)$ without ever specifying ε or even calculating exponents of ε. We do this by discussing procedures for three primitive operations needed in the algorithm outlined in sections 3 and 4:

(1) given three different indices i, j, k from $\{1, \ldots, n\}$, decide whether lines $h_i(\varepsilon)$ and $h_j(\varepsilon)$ intersect above or below line $h_k(\varepsilon)$;

(2) given four different indices i, j, k, l between 1 and n, decide whether the x-coordinate of $h_i(\varepsilon) \cap h_j(\varepsilon)$ is smaller or greater than the one of $h_k(\varepsilon) \cap h_l(\varepsilon)$, and

(3) given five indices i, j, k, l, m, with i, j, k, l pairwise distinct, decide whether the distance of $h_i(\varepsilon) \cap h_j(\varepsilon)$ from $h_m(\varepsilon)$ is greater or smaller than the distance of $h_k(\varepsilon) \cap h_l(\varepsilon)$ from $h_m(\varepsilon)$ (where the distance is taken positive above $h_m(\varepsilon)$ and negative below $h_m(\varepsilon)$).

We discuss the first primitive in detail and sketch the other two. W.l.o.g. assume that $a_i(\varepsilon) < a_j(\varepsilon)$, that is, $a_i < a_j$ or $a_i = a_j$ and $i > j$. Then $h_i(\varepsilon) \cap h_j(\varepsilon)$ lies above (below) $h_k(\varepsilon)$ if and only if

$$\Delta_1 = \det \begin{bmatrix} a_i(\varepsilon) & b_i(\varepsilon) & 1 \\ a_j(\varepsilon) & b_j(\varepsilon) & 1 \\ a_k(\varepsilon) & b_k(\varepsilon) & 1 \end{bmatrix}$$

is positive (negative). To determine the sign of Δ_1, we assume $i < j < k$; otherwise, exchange rows and remember the number of exchanges. Consequently, Δ_1 can be expressed in terms of the coefficients of h_i, h_j, and h_k as below

$$\begin{aligned} \Delta_1 = \det \begin{bmatrix} a_i & b_i & 1 \\ a_j & b_j & 1 \\ a_k & b_k & 1 \end{bmatrix} - \varepsilon^{2^{2i-1}} \det \begin{bmatrix} a_j & 1 \\ a_k & 1 \end{bmatrix} + \\ + \varepsilon^{2^{2i}} \det \begin{bmatrix} b_j & 1 \\ b_k & 1 \end{bmatrix} + \varepsilon^{2^{2j-1}} \det \begin{bmatrix} a_i & 1 \\ a_k & 1 \end{bmatrix} + \\ + \varepsilon^{2^{2i}+2^{2j-1}} + \ldots . \end{aligned} \tag{1}$$

The terms in (1) are ordered such that the exponents of ε increase from left to right. Consequently, $\Delta_1 > 0$ if and only if there is a positive integer m such that the first $m-1$ coefficients of the ε-terms vanish and the mth coefficient (including its sign) is positive. Since the fifth coefficient equals one, we can restrict m to $1 \leqslant m \leqslant 5$. In other words, a decision can be found by evaluating one three-by-three determinant and at most three two-by-two determinants of the form above. It seems worthwhile to note that already the first or second two-by-two determinant does not vanish if h_i, h_j and h_k are assumed to be pairwise distinct.

The second primitive can be decided by computing the sign of

$$\Delta_2 = \det \begin{bmatrix} a_j(\varepsilon) - a_i(\varepsilon) & b_j(\varepsilon) - b_i(\varepsilon) \\ a_l(\varepsilon) - a_k(\varepsilon) & b_l(\varepsilon) - b_k(\varepsilon) \end{bmatrix}$$

that is, the x-coordinate of $h_i(\varepsilon) \cap h_j(\varepsilon)$ is less (greater) than the one of $h_k(\varepsilon) \cap h_l(\varepsilon)$ if and only if $\Delta_2 < 0$ ($\Delta_2 > 0$), for ε small enough. The computation can, again, be based solely on the coefficients of lines h_i, h_j, h_k, and h_l and on indices i, j, k, and l.

For the third primitive, we may compute the sign of

$$\Delta_3 = \frac{1}{a_i(\varepsilon)-a_j(\varepsilon)} \det \begin{bmatrix} a_i(\varepsilon) & b_i(\varepsilon) & 1 \\ a_j(\varepsilon) & b_j(\varepsilon) & 1 \\ a_m(\varepsilon) & b_m(\varepsilon) & 1 \end{bmatrix}$$
$$- \frac{1}{a_k(\varepsilon)-a_l(\varepsilon)} \det \begin{bmatrix} a_k(\varepsilon) & b_k(\varepsilon) & 1 \\ a_l(\varepsilon) & b_l(\varepsilon) & 1 \\ a_m(\varepsilon) & b_m(\varepsilon) & 1 \end{bmatrix}$$

that is, $h_i(\varepsilon) \cap h_j(\varepsilon)$ lies above (below) the line parallel to $h_m(\varepsilon)$ through $h_k(\varepsilon) \cap h_l(\varepsilon)$ if and only if $\Delta_3 > 0$ ($\Delta_3 < 0$), for ε small enough.

3. Testing Against a Line

This section examines two variants of the key subproblem occurring in the main algorithm outlined in the section to follow. First, we will take some care to specify tl subproblem and then show how to solve it algorithmically.

Let G_1 and G_2 be non-empty sets of m_1 and m_2 lines in general position in E^2, and let k_1 and k_2 be integers such that $1 \leqslant k_1 \leqslant m_1$ and $1 \leqslant k_2 \leqslant m_2$. For a point p in E^2, we let $a_i(p)$, $o_i(p)$, and $b_i(p)$ denote the number of lines in G_i such that p is below, on, and above, respectively, for $i = 1, 2$. We define the *k_i-level* $L_{k_i}(G_i)$ *of* G_i as the set of points p in E^2 with $a_i(p) \leqslant k_i - 1$ and $b_i(p) \leqslant m_i - k_i$; it follows that $o_i(p)$ is either 1 or 2. Figure 1 shows the three levels of two sets of respectively 5 lines in E^2.

The procedure discussed in this section accepts as input G_1, G_2, k_1, k_2, a possibly unbounded open convex polygon P with non-empty interior given by a sorted list of non-redundant halfplanes, and a line t called the test-line. The requirements on this input are

(1) Either P is unbounded to the left or there is a vertical edge bounding P at its left; the same holds for the right-hand side of P.

(2) If P is bounded by a vertical edge to the left (right), then $L_{k_1}(G_1)$ intersects this edge; otherwise $P \cap L_{k_1}(G_1)$ is unbounded to the left (right).

(3) $P \cap L_{k_1}(G_1)$ is connected.

(4) P contains an odd number of intersections between $L_{k_1}(G_1)$ and $L_{k_2}(G_2)$ in its interior (so at least one), but no intersection on its boundary.

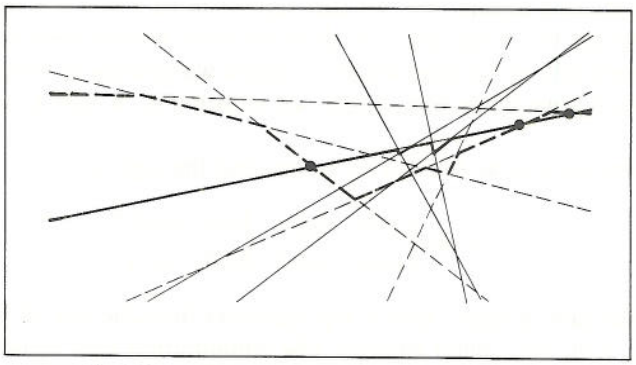

Fig. 1. Two 3-levels which intersect in three points.

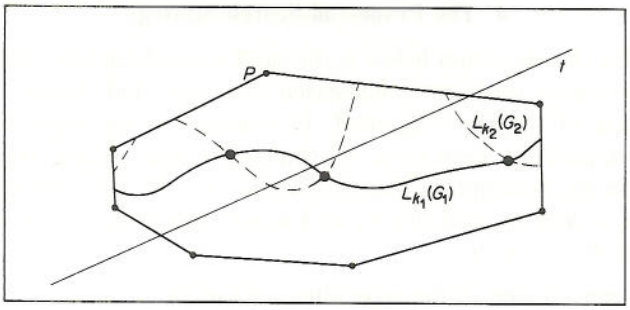

Fig. 2. Convex polygon, two levels, test line.

Figure 2 shows a valid input situation.

In addition to the above input, we pass a boolean variable *above* which is true if the leftmost point of $L_{k_1}(G_1)$ in P lies above $L_{k_2}(G_2)$, and false, otherwise. (If *above* = true, then by condition (4), the rightmost point of $L_{k_1}(G_1)$ in P is below $L_{k_2}(G_2)$.)

We are asked to decide whether t contains an intersection of $L_{k_1}(G_1)$ and $L_{k_2}(G_1)$ in P and, if not, which one of the open halfplanes bounded by t contains an odd number of such intersections. We consider the somewhat easier case when t is vertical first.

If t does not intersect P, then the side of t that contains P also contains the desired odd number of intersections, and we are finished. Otherwise, let the lines in G_i intersect t in points $q_{i,m}$, for $1 \leqslant m \leqslant m_i$ and $i = 1, 2$. Then the k_i-level of G_i intersects t in point q_i with k_i-largest y-coordinate among these points. Using the classic algorithm for finding the k-largest of an unsorted set of numbers (see, e.g. Aho *et al.*, 1974), we can decide in $O(m_1 + m_2)$ time whether

(i) $L_{k_1}(G_1)$ and $L_{k_2}(G_2)$ intersect on t;

(ii) $L_{k_1}(G_1)$ intersects t above $L_{k_2}(G_2)$; or

(iii) $L_{k_1}(G_1)$ intersects t below $L_{k_2}(G_2)$.

In the first case we are finished. In case (ii), the open halfplane to the left of t contains an odd number of intersections if *above* = false; the same is true in case (iii) if *above* = true. Otherwise, the open halfplane to the right of t is the desired one.

Assume now that t is non-vertical which is more difficult than the vertical case since t might intersect $L_{k_1}(G_1)$ as well as $L_{k_2}(G_2)$ an arbitrary number of times. To solve the problem, we consider the intersections $q_{1,m}$, $1 \leqslant m \leqslant m_1$ of t with the m_1 lines of G_1 (we assume that no line of G_1 is parallel to t which can be guaranteed if t gets assigned an index not between 1 and n, see section 2). Using the algorithm for vertical lines, we can decide in $O(m_1 + m_2)$ time whether an odd number of intersections of $L_{k_1}(G_1)$ and $L_{k_2}(G_2)$ in P lie to the left or to the right of the vertical line through some $q_{1,m}$. (If the vertical line contains an intersection we are finished.) Building on this possibility, we perform binary search in the sorted sequence of the points $q_{1,m}$ to find an index j $(0 \leqslant j \leqslant m_1)$ such that $L_{k_1}(G_1)$ and $L_{k_2}(G_2)$ intersect an odd number of times in P between the vertical lines through $q_{1,j}$ and $q_{1,j+1}$ (if $q_{1,j}$ or $q_{1,j+1}$ do not exist, then the natural dummy point at infinity can be substituted). In $O((m_1 + m_2) \log (m_1 + 1))$ time, index j can be found which decides the problem since $L_{k_1}(G_1)$ exists either only above or only below t in the computed interval.

4. The Prune-and-Search Strategy

The basic idea for the algorithm below is the same as in (Megiddo, 1985): in one step of the algorithm the region searched is decreased in a way that allows us to eliminate a constant proportion of the lines regarded. In contrast to the algorithm in (Megiddo, 1985), however, we carefully keep track of what the current region exactly is. We detail this approach under the assumption that the lines in $H = H_1 \cup H_2$ are in general position, that $n_1 \leqslant n_2$ for $n_i = card\ H_i$, and that n_1 and n_2 are odd (if n_i is even, then we delete an arbitrary line from H_i, $i = 1, 2$).

The input to a general step of the algorithm consists of

(i) sets G_1 and G_2 of m_1 and m_2 lines;

(ii) integers k_1 and k_2 with $1 \leqslant k_i \leqslant m_i$, for $i = 1, 2$; and

(iii) a possibly unbounded open convex polygon P.

Initially, $G_i = H_i$, $m_i = n_i$, $k_i = (n_i+1)/2$, for $i = 1, 2$, and $P = E^2$. The algorithm takes ca1 that the input maintains the following properties which are trivially true for the initial values of G_i, m_i, k_i, and P.

Invariance property 4.1: $L_{k_i}(G_i) \cap P = L_{r_i}(H_i) \cap P$, with $r_i = (n_i+1)/2$, for $i = 1, 2$, the interior of P contains an odd number of intersections of $L_{k_1}(G_1)$ and $L_{k_2}(G_2)$, $L_{k_1}(G_1) \cap P$ is connected, and either P is unbounded to the left (right), in which case $L_{k_1}(G_1) \cap P$ is also, or P is bounded to the left (right) by a vertical edge, in which case $L_{k_1}(G_1)$ intersects this edge. (See also section 3.)

Several actions are taken to decrease P as well as $G = G_1 \cup G_2$:

Step 1: Determine line g^* in G with median slope, that is, its slope is the $\lfloor m/2 \rfloor$-smallest among all $m = m_1 + m_2$ lines in G. Let $G_+(G_-)$ denote the set of at least as steep (less steep) lines in G.

Step 2: Match each line in G_- with an arbitrary but unique line in G_+; this leaves one line of G_+ unmatched if m is odd. Let M be the resulting set of $\lfloor m/2 \rfloor$ matched pairs of lines, or, for our purposes, of intersection points.

Step 3: Determine a vertical line v^* that bisects M. Using the methods of section 3, we decide whether v^* contains an intersection of $L_{k_1}(G_1)$ and $L_{k_2}(G_2)$ in P (in this case we are finished), or find the open halfplane v bounded by v^* that contains an odd number of intersections between $L_{k_1}(G_1)$ and $L_{k_2}(G_2)$ in P.

Step 4: Determine a line w^* parallel to g^* which bisects the set of points in M outside of v. Again, decide whether w^* contains an intersection of $L_{k_1}(G_1)$ and $L_{k_2}(G_2)$ in P (then we are finished) or determine the open halfplane w bounded by w^* that contains an odd number of these intersection points contained in v (this also yields two restricting halfplanes w' and w'' bounded by vertical lines) (see section 3).

Step 5: Define $P' = P \cap v \cap w \cap w' \cap w''$, and let G_i' contain all lines of G_i that intersect $v \cap w \cap w' \cap w''$, for $i = 1, 2$. If w^* bounds w from below, then define $k_i' = k_i$, and define $k_i' = k_i - |G_i| + |G_i'|$, otherwise, for $i = 1, 2$.

Unless $|G_1'| + |G_2'| = 2$, Steps 1 to 5 are repeated for sets G_1', G_2', integers k_1', k_2', and for polygon P'. Note that Step 5 guarantees the maintenance of Invariance property 4.1. So if $|G_1'| + |G_2'| = 2$, then each one of G_1' and G_2' contain a line and these two lines intersect in a desired point. To analyse the performance of the algorithm, we show

LEMMA 4.2. *During the execution of Steps* 1 *to* 5 *for m lines, the algorithm either stops or at least* $(m-1)/8$ *lines are eliminated.*

PROOF. We assume that no condition occurs that lets the algorithm stop. If Steps 1 to 5 are executed for m lines, then set M formed in Step 2 contains $\lfloor m/2 \rfloor$ points. At least a quarter of M, so at least $\lfloor m/2 \rfloor/4$ points, lie outside of v and outside of w. By construction of M, one line of two which define a point outside of v and w does not intersect $v \cap w$. Therefore at least $\lfloor m/2 \rfloor/4$ lines are eliminated. □

Lemma 4.2. together with the results of section 3 yield the main result of this paper:

THEOREM 4.3. *Let B be a set of* n_b *and W a set of* n_w *points in* E^2. *There is an algorithm that finds a ham-sandwich cut of B and W in*

$$O((n_b+n_w)\log(\min\{n_b, n_w\}+1))$$

ne.

PROOF. If Steps 1 to 5 are executed for $m = m_1 + m_2$ lines, then all actions except for those in Step 4 can be carried out in $O(m)$ time. By the results of section 3, Step 4 takes

$$O(m \log(\min\{n_b, n_w\}+1))$$

time. Since m decreases geometrically (see Lemma 4.2 and note that $(7m+1)/8 \leqslant 8m/9$ unless $m < 9$) the total algorithm takes time

$$O\left(\sum_{i=0}^{\infty} (8/9)^i (n_b+n_w)\log(\min\{n_b, n_w\}+1)\right)$$

which is in

$$O((n_b+n_w)\log(\min\{n_b, n_w\}+1))$$

as asserted. □

5. Discussion

This paper demonstrates an algorithm that computes a ham-sandwich cut of two sets of n_b and n_w points in E^2 in

$$O((n_b+n_w)\log(\min\{n_b, n_w\}+1))$$

time. Beside the theoretical result (it is a slight improvement over the $O(n \log n)$ algorithm of Cole (1984), for $n = n_b + n_w$) it is sufficiently simple to allow a reasonable implementation. Section 2 discusses some methods that simplify the implementation of the algorithm.

The authors tried for a linear algorithm but apparently did not succeed. This leaves the open problem of whether or not a linear algorithm exists.

References

Aho, A. V., Hopcroft, J. E. & Ullman, J. D. (1974). *The Design and Analysis of Computer Algorithms*. Reading: Addison-Wesley.

Atallah, M. J. (1983). A matching problem in the plane. *J. Comp. Syst. Sci.* (in press).

Cole, R. (1984). Slowing down sorting networks to obtain faster sorting algorithms. *Proc. 25th Ann. IEEE Symp. Found. Comp. Sci.*, 225–260.

Cole, R., Sharir, M. & Yap, C. (1984). On k-hulls and related problems. *Proc. 16th Ann. ACM Symp. Theory Comput.*, 154–166.

Dobkin, D. P. & Edelsbrunner, H. (1984). Space searching for intersecting objects. *Proc. 25th Ann. IEEE Symp. Found. Comp. Sci.*, 387–392.

Edelsbrunner, H. (1986). Edge-skeletons in arrangements with applications. *Algorithmica* **1**, 93–109.

Edelsbrunner, H. & Welzl, E. (1983). Halfplanar range search in linear space and $0(n^{0.695})$ query time. *Inf. Proc. Lett.* (to appear).

Megiddo, N. (1985). Partitioning with two lines in the plane. *J. Algorithms* **6**, 430–433.

Willard, D. E. (1982). Polygon retrieval. *SIAM J. Comp.* **11**, 149–165.