D. P. Dobkin [1] (Princeton), H. Edelsbrunner [2] (Graz)

Ham-Sandwich Theorems Applied to Intersection Problems *)

Abstract

New data structures for two-and three-dimensional intersection problems are described. These structures store n points, segments, lines, polygons, planes, or tetrahedra in O(n) space and answer intersection queries for similar objects in sub-linear time. Ham-sandwich theorems and balanced dissections of point-sets form the mathematical backdrop of this approach.

1. Introduction

Ham-sandwich theorems are classical in topology where they concern cutting objects in two parts of equal measure [M]. We develop discrete versions of these results, that is, finite sets of points are dissected. On this basis, the existence of balanced dissections of finite point-sets can be shown. New data structures for answering various types of range queries follow from these geometric tools. We also show how to extend the structures to handle some examples of intersection problems of the following generic type:

Given a set of objects; count or report those objects that intersect a query object. n objects are stored in O(n) space and a query can be answered in sublinear time, if the objects or query object are specified as points, segments, rays, lines, triangles or polygons in E^2, or points, segments, planes, tetrahedra in E^3. The novelty of our solution is the space efficiency which contrasts previously suggested solutions [EKM,C,CY].

*) Research of the first author was partially supported by the National Science Foundation under Grant MCS83-03926.

1) Dep.Comp.Sci. and El.Eng., Princeton Univ., Princeton, NJ 08544, USA.
2) Inst. für Informationsverarbeitung, Techn. Univ. Graz, Schießstattg. 4a, A-8010 Graz, Austria.

The organisation of the paper is as follows: Section 2 describes the geometric results which are used to develop data structures for range search in E^2 and E^3 in Section 3. Applications to intersection problems are discussed in Section 4, before Section 5 addresses general issues. Proofs and details are omitted throughout. They can be found in [EW,DE1,DE2].

2. Ham-Sandwich Theorems with Extensions

Let us start with the introduction of some geometric formalism: E^d designates the d-dimensional Euclidean space and a hyperplane h in E^d is an affine subspace of dimension d-1; so h is a line in E^2 and a plane in E^3. It is convenient to write h as

$$\langle v,x\rangle = a,$$

for $x = (x_1,\ldots,x_d)$, $v = (v_1,\ldots,v_d)$ some vector, a some real, and $\langle v,x\rangle = \sum_{i=1}^{d} v_i x_i$ the scalar product of v and x. We consider h as being oriented by v and call the half-spaces

$$h^+ : \langle v,x\rangle > a \quad \text{and}$$
$$h^- : \langle v,x\rangle < a$$

the positive and negative sides of h, respectively.

Let now P be a set of n points in E^d. The following definition is crucial for the rest of this paper: A hyperplane h bisects P if

$$|P\cap h^+| \leq \tfrac{n}{2} \quad \text{and}$$
$$|P\cap h^-| \leq \tfrac{n}{2}.$$

In this case h is a bisector of P. It seems worthwhile to remark that points in h are counted on neither side; so if h happens to contain at least $\frac{n}{2}$ points then it is guaranteed to bisect P. In fact, if n is odd, every bisector contains at least one point of P.

The discrete ham-sandwich theorem is now:

Theorem 1: Let $P_1,\ldots,P_d$ be finite sets of points in E^d. There is a plane that bisects each of the d sets.

A full proof is given in [E]. In E^2, Theorem 1 can be used to show

Theorem 2: Let P be a set of n points in E^2.

(i) There are lines h_1 and h_2 such that each open region defined contains no more than $\frac{n}{4}$ points of P.

(ii) A dissection as in (i) even exists if h_1 is a bisector of P that is fixed in advance.

(iii) There are not necessarily three lines such that each region of the dissection contains at most $\frac{n}{7}$ points of P.

(ii) is a strengthening of (i) and follows from Theorem 1 since h_1 bisects P into two sets that can be bisected simultaneously. (iii) shows that (i) is best-possible in the sense that three or more non-concurrent and non-parallel lines cannot always produce a balanced dissection. The case of many points on a circle shows (iii).

There is a generalization of Theorem 2 to E^3, although not a straightforward one:

Theorem 3: Let P be a set of n points in E^3.

(i) There are three planes h_1,h_2,h_3 such that each (open) cell contains at most $\frac{n}{8}$ points of P.

(ii) A dissection as in (i) exists even if h_1 is a fixed bisector of P.

(iii) There might not be a dissection as in (i) when h_1 and h_2 are fixed but such that each of the four cells defined contains at most $\frac{n}{4}$ points of P.

Again (ii) is a strengthening of (i) and (iii) shows that it cannot be strengthened any further. A proof of (ii) is given in [DE1]. The following configuration implies (iii):

$h_1 : y = 0$, $h_2 : x = 0$, and
P is the union of

$\{(1,1,\frac{i}{m})\}$, $\{(-1,-1,\frac{i}{m})\}$,

$\{(1,-1,-\frac{i}{m})\}$, $\{(-1,1,-\frac{i}{m})\}$,

for $m = \frac{n}{4}$ and $i = 1,\ldots,m$.

3. Data Structures

Theorems 2 and 3 lead to efficient data structures that store finite sets of points and answer certain types of range queries. We consider the two-dimensional case first:

Let P be a set of n points in E^2 and let h_1 bisect P. The C-tree $\mathcal{B}$ of P (and h_1) is a binary tree. Unless P is empty, the root of $\mathcal{B}$ stores h_1, possibly $|P|$, and the sorted array for $P \cap h_1$. Let $P^+ = P \cap h_1^+$ and $P^- = P \cap h_1^-$, and let h_2 bisect both P^+ and P^-. The C-tree of P^- (and h_2) is the left subtree of the root, and the C-tree of P^+ (and h_2) is the right subtree. Consider Fig. 1 which depicts a point-set, the dissection by lines, and the associated C-tree.

The following simple observation implies that certain types of range queries can be answered efficiently with the C-tree.

Observation 4: Let h_1 and h_2 be two lines in E^2. Another line h intersects at most three of the four regions defined by h_1 and h_2.

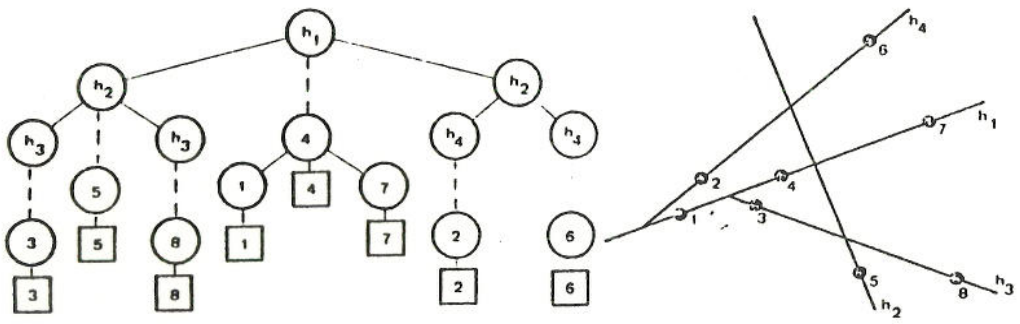

Figure 1: Point-set and C-tree.

For v a node in a C-tree, define the <u>domain</u> $dom(v)$ of v as follows:

If v is the root then $dom(v) = E^2$.
Otherwise let v be the left (right) son of node w, and let w store line h.
Then $dom(v) = dom(w) \cap h^-$ ($dom(v) \cap h^+$).
Intuitively, $dom(v)$ is the area in which the points below v can lie. Observation 4 implies that if a line h intersects $dom(v)$ then it does not intersect the domain of at least one grandson of v. Let $i(n)$ be the maximal number of nodes v in a C-tree for n points such that

$$dom(v) \cap h \neq \emptyset,$$

for some line h. Then

$$i(n) \leq i(\tfrac{n}{2}) + i(\tfrac{n}{4}) + 2.$$

Since $0.695 \approx \log_2 x$ with $x^2 - x - 1 = 0$, we have

<u>Lemma 5</u>: $i(n) = O(n^{0.695})$.

Let q be some convex range (i.e. area) in E^2. The query with q asks to compute $|P \cap q|$. Using the C-tree, it can be shown that there is a procedure that visits a node only if its father's domain intersects q. Furthermore, each visited node v takes $O(\log n)$ time:

<u>Case 1</u>: $dom(v)$ intersects the boundary of q. Then visit v's sons after inspecting the points in the array of v.

<u>Case 2</u>: $dom(v)$ does not intersect the boundary of q. If $dom(v) \subseteq q$ then add the number of points below v to some global variable.

This implies

<u>Theorem 6</u>: The C-tree stores a set P of n points in E^2 in $O(n)$ space such that $|P \cap q|$ can be determined in $O(n^{0.695})$ time if q is the intersection of a fixed number of half-planes.

We now turn to three dimensions:
Using Theorem 3 (ii), we define the D-tree for a set P of n points in E^3 and a bisector h_1 of P: The root of the <u>D-tree</u> stores h_1, possibly $|P|$, and the C-tree for $P \cap h_1$. Let h_2 and h_3 complete h_1 to a dissection with each (open) cell containing at most $\frac{n}{8}$ points of P. The left son L (right son R) of the root stores h_2, possibly $|P \cap h_1^-|$ ($|P \cap h_1^+|$), and the C-tree for $P \cap h_1^- \cap h_2$ ($P \cap h_1^+ \cap h_2$). The left (right) subtrees of L and R are the D-trees for $P \cap h_1^- \cap h_2^-$ ($P \cap h_1^- \cap h_2^+$) and $P \cap h_1^+ \cap h_2^-$ ($P \cap h_1^+ \cap h_2^+$), respectively (see Fig. 2).

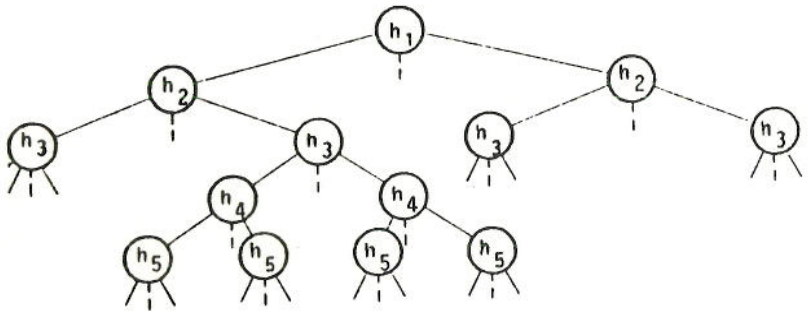

Figure 2: D-tree

Note that only every other level of a D-tree contains roots of D-trees. Now Observation 4 generalizes to

Observation 7: Let h_1, h_2, h_3 be three planes in E^3. Another plane h intersects at most seven of the open cells defined.

If the notion of a domain is naturally extended to D-trees, and $j(n)$ is the maximal number of nodes v in a D-tree for n points such that $dom(v) \cap h \neq \emptyset$, for some plane h, then we have

$$j(n) = 3j(\tfrac{n}{4}) + 2j(\tfrac{n}{16}) + 5,$$

and therefore

Lemma 8: $j(n) = O(n^{0.917})$.

It follows that D-trees accommodate certain range queries in E^3. We have

Theorem 9: The D-tree stores a set P of n points in $O(n)$ space such that $|P \cap q|$ can be computed in $O(n^{0.917})$ time if q is the intersection of a fixed number of halfspaces.

It seems worthwhile to note that our restriction to computing cardinalities is not essential. Rather it can be replaced by any other mapping into a semigroup which allows constant time operations. Also the points in $P \cap h$ can be reported which costs $O(n^{0.695} + |P \cap h|)(O(n^{0.917} + |P \cap h|))$ time.

4. Intersection Search

C- and D-trees and combinations can be used to store sets of geometric objects for intersection queries. As an example for such query problems, we demonstrate a solution for the case that the objects as well as the query object are segments in E^2:

Let $S = \{s_1, \ldots, s_n\}$ be a set of segments in E^2. Store S such that $|\{s \in S \mid s \cap q \neq \emptyset\}|$, for q any query segment, can be determined efficiently.

We develop the solution in three steps: First, S is replaced by a set of lines. Second, the case when q is a line is discussed. Third, the two solutions obtained are combined.

4.1 Segment Intersecting Lines

Let $h : y = ax + b$ be a non-vertical line in E^2, and let q be a segment with endpoints $A = (a,b)$ and $B = (c,d)$. The following is trivial but crucial.

Observation 10: $h \cap q \neq \emptyset$ if and only if A and B lie on different sides of h.

A dual transform is used to obtain a restatement of the problem which lends itself towards an application of C-trees.

D : $h : y = ax + b \rightarrow D(h) = (a,-b)$.

$p = (p_1,p_2) \rightarrow D(p) : y = p_1x - p_2$.

We say that p is above, on, below h, if p_2 is greater, equal, less than $ap_1 + b$, respectively. The transform D satisfies the following nice property:

Lemma 11: Point p is above, on, below line h if and only if point $D(h)$ is above, on, below line $D(p)$.

Let now $S = \{h_i : y = a_ix + b_i | i = 1,\ldots,n\}$ be a set of non-vertical lines, and q a segment with endpoints $A = (a,b)$ and $C = (c,d)$. Let $D(q)$ be the region of points p such that p is above or on $D(A)$ and below or on $D(C)$, or p is below or on $D(C)$ and above or on $D(A)$.

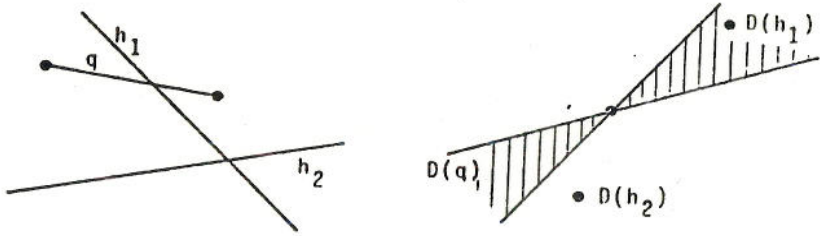

Figure 3: Double-wedge of segment q.

Then $h_i \cap q \neq \emptyset$ if and only if $D(h_i)$ is contained in $D(q)$. $|D(S) \cap D(q)|$, however can be determined by two angular range queries in the C-tree of $D(S)$. This implies a solution with $O(n)$ space and $O(n^{0.695})$ query time.

4.2 Line Intersecting Segments

Let $S = \{s_1,\ldots,s_n\}$ be a set of segments in E^2, and let $q : y = ax + b$ be a query line. We specify s_i by its endpoints $A_i = (a_i,b_i)$ and $C_i = (c_i,d_i)$, for $1 \leq i \leq n$. Using Observation 10, the segments that intersect q can be found by two conjunctive halfplanar range queries:

Invoke a query to determine points A_i above (below) or on q. From the points found, identify those with C_i below (above) or on q.

Two so-called levels of C-trees permit this kind of query: The primary C-tree stores $\{A_i | i = 1,\ldots,n\}$. Each node v of this tree has a secondary C-tree attached which stores $\{C_i | A_i$ lies in dom(v)$\}$. Whenever dom(v) is found to be contained in the query range, a "secondary" query in the attached C-tree computes the number of segments stored in v's subtree that intersect q. It is not hard to see that the described structure needs $O(n\log n)$ space and $O(n^{0.695}\log n)$ query time. Improvements to $O(n)$ space and $O(n^{0.695})$ query time can be found in [DE2].

4.3 Segment Intersecting Segments

The solutions given in Sections 4.1 and 4.2 can be applied since we have

Observation 12: Let s_1 and s_2 be two segments in E^2 with endpoints A_1,C_1, and A_2,C_2, respectively. Let h_1 and h_2 be the lines that support s_1 and s_2, respectively. Then $s_1 \cap s_2 \neq \emptyset$ if and only if $s_1 \cap h_2 \neq \emptyset$ and $s_2 \cap h_1 \neq \emptyset$.

Let $S = s_1,\ldots,s_n$ be a set of segments and q be a query segment. The segments intersecting q are determined in two steps:

Using the C-tree for the dual points, the lines of support that intersect q are identified.
Each node v of this tree has two levels of C-trees attached that store $(s_i$ in $S|D(h_i)$ in $dom(v)$, for h_i the line that supports $s_i)$ (see Section 4.2). These levels are used to compute $|\{s_i|h_i \cap q \neq \emptyset\}|$.
The resulting three-level structure requires $O(n\log^2 n)$ space and $O(n^{0.695}\log^2 n)$ query time. Improvements described in [DE2] yield:

Theorem 13: A set of n segments in E^2 can be stored in $O(n)$ space such that those which intersect a query segment can be counted in $O(n^{0.695+\varepsilon})$ time, for any $\varepsilon > 0$.

5. Discussion

This paper outlines a new approach to intersecting problems in two and three dimensions which builds on discrete versions of classical Ham-Sandwich Theorems. In contrast to previously suggested solutions [EKM,C,CY], our approach yields data structures that require only linear space.

Because of space limitations, only one two-dimensional intersection problem is described here. Others can be found in [DE2] where also three-dimensional problems are discussed. The general idea towards a solutions is the "concatenation" of D-trees as shown for C-trees in Section 4. Interestingly, this method is not applicable to the following problem:

Store a set of lines in E^3 such that those that intersect a query line can be counted.

It is open whether or not there is a linear space structure that allows the counting in sublinear time.

References

[C] Chazelle,B.M. Fast computation of segment intersections. Rep. CS-83-11, Dep. Comp. Sci., Brown Univ., Providence, RI, 1983.

[CY] Cole,R. and Yap,C.K. Geometric retrieval problems. Proc. 24th FOCS (1983), 112-121.

[DE1] Dobkin,D.P. and Edelsbrunner,H. Organizing points in two and three dimensions. Rep. F130, IIG, TU Graz, 1984.

[DE2] Dobkin,D.P. and Edelsbrunner,H. Space searching for intersecting objects. Manuscript, 1984.

[E] Edelsbrunner,H. Arrangements and geometric computation. Book in preparation.

[EKM] Edelsbrunner,H., Kirkpatrick,D.G., and Maurer,H.A. Polygonal intersection searching. Inform. Proc. Lett. 14 (1982), 74-79.

[EW] Edelsbrunner,H. and Welzl,E. Halfplanar range search in linear space and $O(n^{0.695})$ query time. Rep. F111, IIG, TU Graz, 1983.

[M] Mendelson,B. Introduction to topology. Allyn & Bacon, Boston, 1962.