BIT 31 (1991), 2–14

A DUAL APPROACH TO DETECT POLYHEDRAL INTERSECTIONS IN ARBITRARY DIMENSIONS

OLIVER GÜNTHER and EUGENE WONG

FAW-AI Laboratory,
University of Ulm,
Postfach 2060, 7900 Ulm,
Germany

Department of Electrical Engineering
and Computer Sciences,
University of California,
Berkeley, CA 94720, USA

Abstract.

This paper presents a dual approach to detect intersections of hyperplanes and convex polyhedra in arbitrary dimensions. In d dimensions, the time complexities of the dual algorithms are $O(2^d \log n)$ for the hyperplane-polyhedron intersection problem, and $O((2d)^{d-1} \log^{d-1} n)$ for the polyhedron- polyhedron intersection problem. These results are the first of their kind for $d > 3$. In two dimensions, these time bounds are achieved with linear space and preprocessing. In three dimensions, the hyperplane-polyhedron intersection problem is also solved with linear space and preprocessing; quadratic space and preprocessing, however, is required for the polyhedron-polyhedron intersection problem. For general d, the dual algorithms require $O(n^{2^d})$ space and preprocessing. All of these results readily extend to unbounded polyhedra.

CR categories: E.1, F.2.2.

Keywords: computational geometry, intersection detection, geometric duality, multidimensional search, hyperplane-polyhedron intersection, polyhedron-polyhedron intersection, arbitrary dimensions, linear programming.

1. Introduction.

Detecting and computing intersections is a fundamental problem in computational geometry [1]. Fast solutions for intersection problems are desirable in a wide range of application areas, including linear programming [2], hidden surface elimination [3], or spatial databases [4]. In many of these applications, the dimension of the intersection problems may be greater than three. This is partic-1ularly obvious in linear programming; another example is database applications where geometric objects are used to represent predicates [5].

This is an extended and revised version of a paper presented at the 25th Annual Allerton Conference on Communication, Control, and Computing (October 1987). Our work was sponsored by the U.S. Army Research Office (research contract DAAG29-85-0223) and, in the case of the first author, by graduate fellowships from the IBM corporation and the German National Scholarship Foundation.

Received December 1989. Revised August 1990.

It was first noted by Chazelle and Dobkin [6] that it is often easier to *detect* the intersection of two suitably preprocessed geometric objects rather than to actually *compute* it. In the detection problem, one only asks if two objects intersect or not; also, it is allowed to preprocess each of the given objects separately.

In this paper, we present algorithms to solve the intersection detection problem in arbitrary dimensions for hyperplanes and convex polyhedra. A (*d-dimensional, convex*) *polyhedron* P in d-dimensional Euclidean space $\boldsymbol{E}^d$ is defined to be the intersection of some finite number of closed halfspaces in $\boldsymbol{E}^d$, such that the dimension of the smallest affine subspace containing P is d. If $a \in \boldsymbol{E}^d - \{0\}$ and $c \in \boldsymbol{E}^1$ then the $(d-1)$-dimensional set $H(a,c) = \{x \in \boldsymbol{E}^d\colon x \cdot a = c\}$ defines a hyperplane in $\boldsymbol{E}^d$. $H(a,c)$ is orthogonal to a and its distance from the origin is $c/|a|$. A hyperplane $H(a,c)$ defines two closed halfspaces $1 \cdot H(a,c) = \{x \in \boldsymbol{E}^d\colon x \cdot a \geq c\}$ and $-1 \cdot H(a,c) = \{x \in \boldsymbol{E}^d\colon x \cdot a \leq c\}$. A hyperplane $H(a,c)$ *supports* a polyhedron P if $H(a,c) \cap P \neq \emptyset$ and $P \subseteq 1 \cdot H(a,c)$, i.e., if $H(a,c)$ embeds parts of P's boundary. If $H(a,c)$ is any hyperplane supporting P then $P \cap H(a,c)$ is a *face* of P. The faces of dimension 1 are called *edges*; those of dimension 0 *vertices*. A supporting hyperplane is called a *boundary hyperplane* if the face $H(a,c) \cap P$ is of dimension $d-1$. The faces of P that are a subset of some supporting hyperplane $H(a,c)$ with $a_d \leq 0$, form the *upper hull* of P. Similarly, the faces with $a_d \geq 0$ form the *lower hull* of P. Throughout this paper, we assume that P is given by a list of its faces and the corresponding adjacency relations.

So far, the intersection detection problem has only been considered in two and three dimensions. In their original paper, Chazelle and Dobkin [6] solve the d-dimensional hyperplane-polyhedron intersection problem in time $O(\log n)$ $(d = 2)$ and $O(\log^2 n)$ $(d = 3)$, and the polyhedron-polyhedron intersection problem in time $O(\log n)$ $(d = 2)$ and $O(\log^3 n)$ $(d = 3)$. Here n denotes the maximum number of vertices of any given polyhedron. Both problems require $O(n)$ $(d = 2)$ and $O(n^2)$ $(d = 3)$ space and preprocessing. A revised version of that paper has been published recently [7]. In the three-dimensional case, $O(n \log n)$ space and preprocessing are also sufficient [8], in which case the running times given above have to be multiplied by a $\log n$ factor.

In a later paper, Dobkin and Kirkpatrick [9] improve the running times of Chazelle and Dobkin for the three-dimensional case by a factor of $\log n$. The new upper bounds are $O(\log n)$ and $O(\log^2 n)$ for the hyperplane-polyhedron and the polyhedron-polyhedron problems, respectively. As the algorithms of Chazelle and Dobkin, their algorithms require $O(n^2)$ space and processing. Again, the results of Dobkin and Munro [8] can be used to reduce the space and preprocessing requirements in three dimensions to $O(n \log n)$, in which case the running times increase by a $\log n$ factor.

We obtain our results by means of a geometric duality transformation in d-dimensional Euclidean space $\boldsymbol{E}^d$ that is an isomorphism between points and hyperplanes [10, 11, 1, 12, 13]. In d dimensions, our upper time bounds are $O(2^d \log n)$ to detect the intersection of a hyperplane and a polyhedron, and $O((2d)^{d-1} \log^{d-1} n)$ to detect the intersection of two polyhedra. These are the first results of their kind for

$d > 3$, and match the time bounds given by Dobkin and Kirkpatrick [9] for $d = 2$ and $d = 3$. Furthermore, our results seem to be the first of their kind that extend to unbounded polyhedra as well.

For $d = 2$ and for the hyperplane-polyhedron intersection problem in $d = 3$, the space and preprocessing requirements of the dual representation scheme are $O(n)$ and therefore optimal. For the three-dimensional hyperplane-polyhedron intersection problem, this represents an improvement over the results of Dobkin and Kirkpatrick [9] by a factor of n. The three-dimensional polyhedron-polyhedron problem takes quadratic space and preprocessing, as does the algorithm of Dobkin and Kirkpatrick. For general d, the scheme requires $O(n^{2^d})$ space and preprocessing.

Section 2 introduces the dual representation scheme for convex polyhedra. Sections 3 and 4 show how the hyperplane-polyhedron and the polyhedron-polyhedron intersection detection problems can be solved efficiently using the dual scheme, and section 5 contains our conclusions.

2. The dual representation scheme.

If the hyperplane $H(a, c)$ is non-vertical (i.e. $a_d \neq 0$), then H intersects the dth coordinate axis in a unique and finite point and can be represented by an equation

$$x_d = b_1 x_1 + \ldots + b_{d-1} x_{d-1} + b_d$$

where $b_i = -a_i/a_d$ $(i = 1 \ldots d-1)$ and $b_d = c$. F_H denotes the function whose graph is H, i.e.

$$F_H: \boldsymbol{E}^{d-1} \to \boldsymbol{E}^1$$

$$F_H(x_1 \ldots x_{d-1}) = b_1 x_1 + \ldots + b_{d-1} x_{d-1} + b_d.$$

A point $p = (p_1 \ldots p_d)$ lies *above* (*on*, *below*) H if $p_d > (=, <) F_H(p_1 \ldots p_{d-1})$.

Brown [11] defines a duality transformation D in $\boldsymbol{E}^d$ that maps hyperplanes into points and vice versa. Slightly modifying these definitions, we define the dual $D(H)$ of a hyperplane H as the point $(b_1 \ldots b_d)$ in $\boldsymbol{E}^d$. Conversely, the dual $D(p)$ of a point p is the hyperplane defined by the equation

$$x_d = -p_1 x_1 - p_2 x_2 - \ldots - p_{d-1} x_{d-1} + p_d.$$

LEMMA 1. *A point p lies above* (*on, below*) *a hyperplane H if and only if the dual D*(*H*) *lies below* (*on, above*) *D*(*p*).

PROOF. see [4]. ■

A hyperplane H intersects a bounded polyhedron P if and only if there are two vertices v and w of P such that H lies between v and w (i.e., v lies on or above H and w lies on or below H, or vice versa). According to Lemma 1, this is the case if and only if the dual $D(H)$ lies between the duals $D(v)$ and $D(w)$.

This observation leads to a new representation schema for bounded convex polyhedra. Consider the functions TOP^P, BOT^P: $E^{d-1} \to E^1$ that are defined for a convex polyhedron P as follows. Here, V_P denotes the set of vertices of P.

$$TOP^P(x_1 \ldots x_{d-1}) = \max_{v \in V_P} F_{D(v)}(x_1 \ldots x_{d-1})$$

$$BOT^P(x_1 \ldots x_{d-1}) = \min_{v \in V_P} F_{D(v)}(x_1 \ldots x_{d-1}).$$

Obviously, both functions are piecewise linear, continuous, and TOP^P is convex (or concave-up), whereas BOT^P is concave (or concave-down) [15]. With this notation, a non-vertical hyperplane H intersects P if and only if $D(H)$ lies between TOP^P and BOT^P. More formally, the hyperplane H, given by the equation $x_d = b_1x_1 + \ldots + b_{d-1}x_{d-1} + b_d$, intersects P if and only if $BOT^P(b_1 \ldots b_{d-1}) \leq b_d \leq TOP^P(b_1 \ldots b_{d-1})$. Two- and three-dimensional examples of polyhedra P and their corresponding functions TOP^P and BOT^P are given in figures 1 and 2.

The two functions TOP^P and BOT^P can be viewed as mappings that map any slope $(b_1 \ldots b_{d-1})$ of a non-vertical hyperplane into the maximum (TOP^P) or minimum (BOT^P) intercept b_d such that the hyperplane given by $x_d = b_1x_1 + \ldots + b_{d-1}x_{d-1} + b_d$ intersects the polyhedron. It can easily be shown that each convex

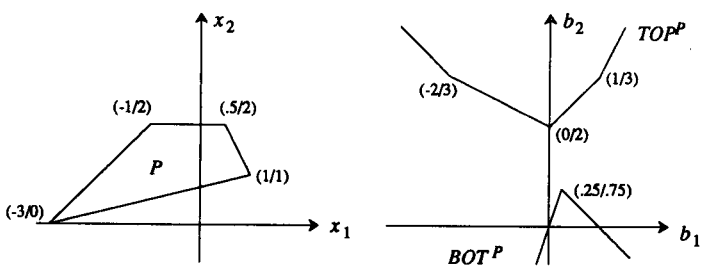

Fig. 1. A polygon P with corresponding functions TOP^P and BOT^P.

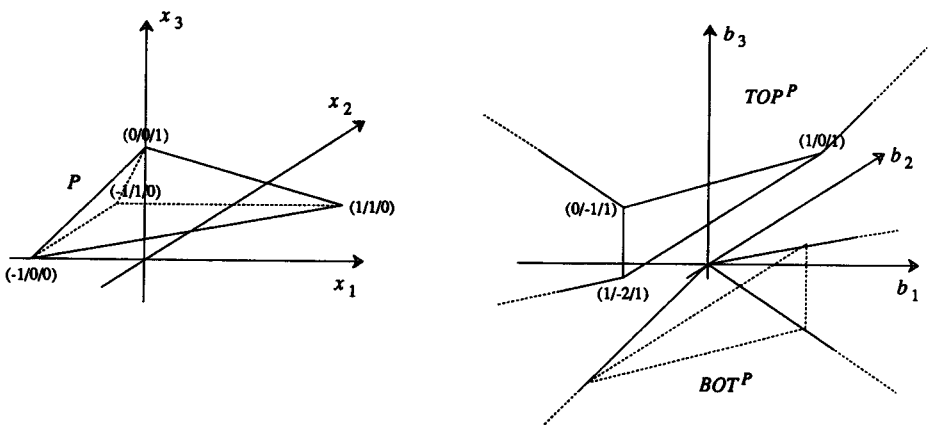

Fig. 2. A polyhedron P with corresponding functions TOP^P and BOT^P.

polyhedron P corresponds to exactly one pair of functions (TOP^P, BOT^P), and conversely [4].

3. Hyperplane-polyhedron intersection detection.

A non-vertical hyperplane H, given by $x_d = b_1x_1 + \ldots + b_{d-1}x_{d-1} + b_d$ intersects a bounded polyhedron P if and only if $BOT^P(b_1 \ldots b_{d-1}) \leq b_d \leq TOP^P(b_1 \ldots b_{d-1})$. Moreover, an intersecting hyperplane H supports P if and only if $b_d = BOT^P(b_1 \ldots b_{d-1})$ or $b_d = TOP^P(b_1 \ldots b_{d-1})$. Therefore, the intersection detection problem can be solved by obtaining the functional values $TOP^P(b_1 \ldots b_{d-1})$ and $BOT^P(b_1 \ldots b_{d-1})$. It follows from the definition of TOP^P and BOT^P that these values can be found in time $O(d \cdot n)$ by computing $F_{D(v)}(b_1 \ldots b_{d-1})$ for each vertex $v \in V_P$. With some preprocessing, however, one can obtain logarithmic time bounds as follows.

It follows from [11] that there is the following isomorphism between the upper hull of the polyhedron P and the graph of TOP^P. Each k-dimensional face (or k-face) f of the upper hull of P corresponds to exactly one $(d - k - 1)$-face $D(f)$ of TOP^P's graph, and vice versa. Furthermore, if two faces f_1 and f_2 of P's upper hull are adjacent, then so are the faces $D(f_1)$ and $D(f_2)$ of TOP^P's graph. The same isomorphism exists between P's lower hull and the graph of BOT^P. Hence, the graphs of TOP^P and BOT^P are polyhedral surfaces in E^d, consisting of no more than n convex $(d - 1)$-faces and no more than $m = O(n^2)$ $(d - 2)$-faces.

Without loss of generality, we show how to obtain only $TOP^P(b_1 \ldots b_{d-1})$. The projection of TOP^P's graph on the $(d - 1)$-dimensional hyperplane J: $b_d = 0$ subdivides J into no more than n convex $(d - 1)$-dimensional polyhedral partitions with no more than m $(d - 2)$-faces. Any given partition $E \subseteq J$ corresponds to a vertex $v(E)$ of P's upper hull, such that $TOP^P(p_1 \ldots p_{d-1}) = F_{D(v(E))}(p_1 \ldots p_{d-1})$ for any point $(p_1 \ldots p_{d-1}) \in E$. Hence, $TOP^P(b_1 \ldots b_{d-1})$ can be obtained by a $(d - 1)$-dimensional point location in J to find the partition E that contains the point $(b_1 \ldots b_{d-1})$, followed by a computation of $F_{D(v(E))}(b_1 \ldots b_{d-1})$.

For $d = 2$ and $d = 3$, the computation of $F_{D(v(E))}(b_1 \ldots b_{d-1})$ takes only constant time. For $d = 2$, the point location degenerates to a binary search, which takes time $O(\log n)$ with $O(n)$ space and preprocessing. For $d = 3$, the point location can be performed in time $O(\log n)$, using the algorithm of Edelsbrunner *et al.* [16] for point location in a monotone subdivision. Because the given partitions are convex and therefore monotone, space and preprocessing requirements are only $O(n)$ as well. The total time complexity to detect the intersection of a hyperplane and a polyhedron in two or three dimensions is therefore $O(\log n)$ with $O(n)$ space and preprocessing.

For $d > 3$, it takes time $O(d)$ to compute the functional value $F_{D(v(E))}(b_1 \ldots b_{d-1})$. Dobkin and Lipton [17] solve a $(d - 1)$-dimensional point location problem with m $(d - 2)$-faces recursively as follows. In a preprocessing step, they compute the

$O(m^2)$ $(d-3)$-faces in the partition and project them on some $(d-2)$-dimensional hyperplane K. For each partition K_i of K that is formed by these $(d-3)$-faces, one maintains a sorted list L_i of those $(d-2)$-faces that project onto K_i. This way, the point location problem can be solved by a point location in K, which yields a partition K_i, followed by a binary search of the corresponding list L_i of $(d-2)$-faces. The time complexity of this point location algorithm is

$$\begin{aligned}
&TPL(d-1,m)\\
&\le TPL(d-2,m^2)+(d-1)\lfloor \log m+1\rfloor\\
&\le \dots\\
&\le TPL(2,m^{2^{d-3}})+\textstyle\sum_{i=1}^{d-3}(d-i)\lfloor 2^{i-1}\log m+1\rfloor\\
&\le TPL(2,m^{2^{d-3}})+2^{d-1}\log m+d^2\\
&= O(2^d\log n).
\end{aligned}$$

We obtain a total time complexity of $O(d+2^d\log n)$ or $O(2^d\log n)$. Note that we assume that it takes time $O(d)$ to determine on which side of a given hyperplane a point is located. Dobkin and Lipton [17] assume in their analysis that this can be done in constant time (while still obtaining the same bound of $O(2^d\log n)$).

For $d>3$, the space requirements of the dual algorithm are as follows. The equations of the $O(n)$ $(d-1)$-faces require space $O(dn)$. The space requirements to store a convex subdivision of $\boldsymbol{E}^2$ with m edges, $SP(2,m)$, is $O(m)$ [16]. For a subdivision of $\boldsymbol{E}^{d-1}$ with m $(d-2)$-faces, one has to store a subdivision of the $(d-2)$-dimensional projection hyperplane K with m^2 $(d-3)$-faces and a sequence of no more than m $(d-2)$-faces for each of the partitions. The number of partitions is no more than $m^{2(d-2)}$ [18]. Therefore,

$$\begin{aligned}
&SP(d-1,m)\\
&\le SP(d-2,m^2)+m^{2(d-2)}m(d-1)\\
&\le \dots\\
&\le SP(2,m^{2^{d-3}})+\textstyle\sum_{i=1}^{d-3}m^{2^i(d-i-1)}m^{2^{i-1}}(d-i)\\
&\le SP(2,m^{2^{d-3}})+m^{2^{d-1}}\\
&= O(n^{2^d}).
\end{aligned}$$

We obtain a total space complexity of $O(dn+n^{2^d})$ or $O(n^{2^d})$.

The preprocessing requirements of this algorithm are as follows. Each $(d-2)$-face of the subdivision is obtained from the original polyhedron P in time $O(d)$ by dualization and projection. As there are $m=O(n^2)$ $(d-2)$-faces, it takes time $O(dn^2)$ to obtain all of them. The preprocessing requirements to solve a point location problem in a convex subdivision of $\boldsymbol{E}^2$ with m edges, $PRP(2,m)$, are $O(m)$ [16]. For a subdivision of $\boldsymbol{E}^{d-1}$ with m $(d-2)$-faces, one has to intersect eachof

pair $(d-2)$-faces and project the resulting $O(m^2)$ $(d-3)$-faces on the $(d-2)$-dimensional hyperplane K. As mentioned above, these $(d-3)$-faces partition K into no more than $m^{2(d-2)}$ partitions. For each partition K_i, one has to sort the $O(m)$ $(d-2)$-faces that project onto K_i. Finally, one has to do the necessary preprocessing for the subdivision of K. Therefore,

$$\begin{aligned}
&PRP(d-1,m)\\
&\leq PRP(d-2,m^2) + m^{2(d-2)}m\lfloor \log m + 1\rfloor \cdot 2(d-1)\\
&\leq \ldots\\
&\leq PRP(2,m^{2^{d-3}}) + \textstyle\sum_{i=1}^{d-3} m^{2^i(d-i-1)}m^{2^{i-1}}\lfloor \log m^{2^{i-1}} + 1\rfloor \cdot 2(d-i)\\
&\leq PRP(2,m^{2^{d-3}}) + 2m^{2^{d-1}}\\
&= O(n^{2^d}).
\end{aligned}$$

We obtain a total preprocessing time of $O(dn^2 + n^{2^d})$ or $O(n^{2^d})$. Theorem 2 summarizes our results for the hyperplane-polyhedron intersection detection problem.

THEOREM 2. *Given a non-vertical $(d-1)$-dimensional hyperplane H and a d-dimensional convex polyhedron P, H and P can be tested for intersection in time $T(d,n)$ with $S(d,n)$ space and $PP(d,n)$ preprocessing:*

$P \cap H = \emptyset$?	$T(d,n)$	$S(d,n)$	$PP(d,n)$
$d=2$	$O(\log n)$	$O(n)$	$O(n)$
$d=3$	$O(\log n)$	$O(n)$	$O(n)$
$d>3$	$O(2^d \log n)$	$O(n^{2^d})$	$O(n^{2^d})$

PROOF: follows from the preceding discussion. ■

A generalization of these results to vertical hyperplanes is possible [4]. Note that our results for the two- and three-dimensional case are much better than the formulas for the general case would indicate. This discrepancy is due to the fact that in those cases, dual representations are isomorphic to planar graphs. Because of Euler's formula [14], this results in a much lower number of faces to be stored and processed.

4. Polyhedron–polyhedron intersection detection

Two convex polyhedra P and Q do not intersect if and only if there is a separating non-vertical hyperplane between them. Any such hyperplane H does not intersect

either P or Q, but there are hyperplanes H' and H'' parallel to H, such that H' is above and H'' below H, and either H' intersects P and H'' intersects Q, or vice ver-6sa. More formally, a non-vertical hyperplane H, given by the equation $x_d = b_1x_1 + \ldots + b_{d-1}x_{d-1} + b_d$, separates the polyhedra P and Q if and only if

$$TOP^P(b_1 \ldots b_{d-1}) < b_d < BOT^Q(b_1 \ldots b_{d-1}),$$

or

$$TOP^Q(b_1 \ldots b_{d-1}) < b_d < BOT^P(b_1 \ldots b_{d-1}).$$

Therefore, two polyhedra P and Q intersect if and only if

(i) $\min_{(x_1 \ldots x_{d-1}) \in E^{d-1}}(TOP^P - BOT^Q)(x_1 \ldots x_{d-1}) \geq 0,$

and (ii) $\min_{(x_1 \ldots x_{d-1}) \in E^{d-1}}(TOP^Q - BOT^P)(x_1 \ldots x_{d-1}) \geq 0.$

If both conditions are only met as equalities, then only the boundaries of P and Q intersect, but not their interiors. See figures 3 and 4 for several two- and three-dimensional examples. Note that both $TOP^P - BOT^Q$ and $TOP^Q - BOT^P$ are the difference of a convex and a concave function and therefore convex.

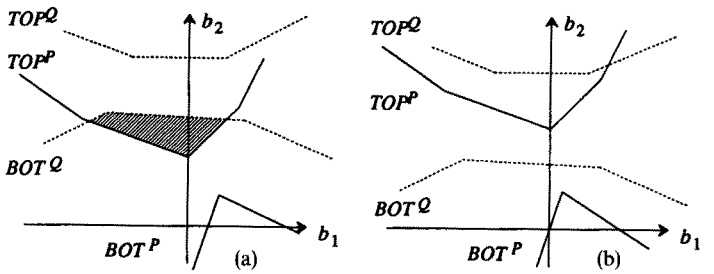

Fig. 3. (a) No intersection: the points in the shaded area are the duals of the separating hyperplanes. (b) Intersection: for all $x \in E^1$: $TOP^P(x) \geq BOT^Q(x)$ and $TOP^Q(x) \geq BOT^P(x)$.

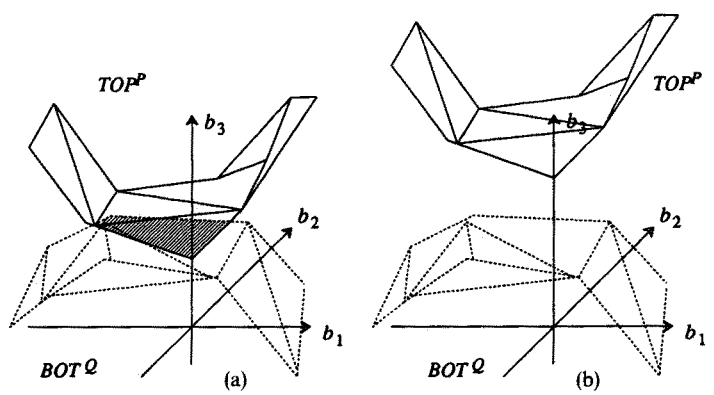

Fig. 4. (a) No intersection: the points in the shaded area are the duals of the separating hyperplanes. (b) Possibly an intersection: for all $x \in E^2$: $TOP^P(x) \geq BOT^Q(x)$. If $TOP^Q(x) \geq BOT^P(x)$ for all $x \in E^2$ holds as well, then P intersects Q.

Both of the above conditions can be tested by solving linear programming problems as follows. As described above, TOP^P and BOT^Q are defined as

$$TOP^P(x_1 \dots x_{d-1}) = \max_{v \in V_P} F_{D(v)}(x_1 \dots x_{d-1})$$

$$BOT^Q(x_1 \dots x_{d-1}) = \min_{w \in V_Q} F_{D(w)}(x_1 \dots x_{d-1}).$$

Furthermore, let x_{TOP} and x_{BOT} be defined as $TOP^P(x_1 \dots x_{d-1})$ and $BOT^Q(x_1 \dots x_{d-1})$, respectively . Condition (i) can now be tested by solving the following linear programming problem.

$$\min x_{TOP} - x_{BOT}$$

$$F_{D(v)}(x_1 \dots x_{d-1}) - x_{TOP} \leq 0 \qquad (v \in V_P)$$

$$-F_{D(w)}(x_1 \dots x_{d-1}) + x_{BOT} \leq 0 \qquad (w \in V_Q).$$

A similar problem exists for condition (ii). According to Megiddo [19], the time complexity to solve a linear programming problem with no more than $2n$ constraints is bounded by $2^{O(2^d)} \cdot 2n$. Hence, the conditions can be tested in linear time $O(n)$ if the dimension is fixed. With some preprocessing, however, the conditions can be tested in polylogarithmic time as follows.

Without loss of generality, we show how to test only condition (i). We present a multidimensional search technique that finds the minimum of a convex piecewise linear function in arbitrary dimensions. The technique is recursive; it solves a d-dimensional problem by solving $O(d \log n)$ $(d-1)$-dimensional problems, and so on.

In the two-dimensional case, condition (i) can be tested by a variation of Dobkin and Kirkpatrick's algorithm [9] to detect the intersection of two polygons. The graphs of TOP^P and BOT^Q are monotone convex polygonal chains with edges $t_1 \dots t_k$ and $u_1 \dots u_l$ $(k + l \leq 2n)$; see also figure 1. The relative position and the slopes of the edges $t_{\lfloor k/2 \rfloor}$ and $u_{\lfloor l/2 \rfloor}$ give enough information to eliminate half of the edges of one (or both) chains from further consideration without missing a negative value if one exists. The algorithm proceeds recursively, eliminating at least one quarter of the remaining edges at each recursion level. Therefore, the existence of a negative value is decided in time $O(\log n)$ without any preprocessing or extra space.

In order to solve the d-dimensional problem, we solve $O(d \log n)$ $(d-1)$-dimensional problems. Due to the fact that $TOP^P - BOT^Q$ is piecewise linear, its global minimum occurs at some vertex of the graph of $TOP^P - BOT^Q$, i.e., at some vertex of TOP^P's graph or BOT^Q's graph [2]. Let TG and BG denote the sets of vertices of TOP^P's graph and BOT^Q's graphs, respectively, and let $(v_1 \dots v_{|TG|})$ denote the sequence of vertices in TG, sorted by increasing x_1-coordinate. We consider the vertex $v_{\lfloor |TG|/2 \rfloor}$ and its x_1-coordinate $\bar{b}_1$, and compute the minimum point of $TOP^P - BOT^Q$ on the hyperplane $x_1 = \bar{b}_1$. This is a $(d-1)$-dimensional minimization problem and can be solved recursively. Due to the convexity of $TOP^P - BOT^Q$, we

can determine the position of the global minimum relative to the minimum on the hyperplane $x_1 = \bar{b}_1$ from the local slope of $TOP^P - BOT^Q$ as follows.

LEMMA 3: *Let* $M = (M_1 \dots M_d)$ *denote the global minimum of* $TOP^P - BOT^Q$, *and let* $m = (m_1 = \bar{b}_1, m_2 \dots m_d)$ *denote the minimum point of* $TOP^P - BOT^Q$ *on the hyperplane* $x_1 = \bar{b}_1$. *Then* $M_1 > (<) m_1$ *holds if and only if there is an* $\varepsilon_0 > 0$, *such that for all* ε *with* $0 < \varepsilon < \varepsilon_0$

$$\begin{aligned} &(TOP^P - BOT^Q)(m_1 - \varepsilon, m_2 \dots m_{d-1}) \\ &> (<)\ (TOP^P - BOT^Q)(m_1 \dots m_{d-1}) \\ &> (<)\ (TOP^P - BOT^Q)(m_1 + \varepsilon, m_2 \dots m_{d-1}). \end{aligned}$$

Otherwise, m is a global minimum of $TOP^P - BOT^Q$.

PROOF: see [4]. ■

Therefore, looking up the functional values $(TOP^P - BOT^Q)(m_1 \pm \varepsilon, m_2 \dots m_{d-1})$ for some suitable $\varepsilon > 0$ gives us enough information to eliminate half of the vertices in TG (and some vertices in BG) from the search without missing the global minimum. If the search among the vertices in TG does not yield a global minimum, one continues with a similar search among the remaining vertices in BG. Hence the global minimum is obtained in no more than $\lfloor \log |TG| + \log |BG| + 2 \rfloor$ iterations.

The analysis of this algorithm obviously depends on the cardinalities of TG and BG. Each vertex in TG (BG) corresponds to a vertex of the graph of TOP^P (BOT^Q), and therefore to a $(d-1)$-face of P (Q). Because both P and Q have no more than n vertices, they have no more than n^{j+1} j-faces for any $j \leq d$ (see [20], p. 174). Hence, the cardinalities of both TG and BG are no more than n^d and the number of iterations, $\lfloor \log |TG| + \log |BG| + 2 \rfloor$, is no more than $2 \log n^d + 2$. Each iteration involves a $(d-1)$-dimensional minimization and the four function lookups necessary to obtain $(TOP^P - BOT^Q)(m_1 \pm \varepsilon, m_2 \dots m_{d-1})$. In the $(d-1)$-dimensional minimization problem, which is solved recursively, each vertex in TG (BG) corresponds to an *edge* of TOP^P (BOT^Q) and therefore to a $(d-2)$-face of P (Q). Hence, the cardinalities of both TG and BG are no more than n^{d-1}. Regarding the four function lookups, it can be deduced from the analysis of time complexity TPL (see section 3) that each lookup can be carried out in no more than $2^d \log n^2$ steps. We obtain a total time complexity

$$\begin{aligned} &T(d, n) \\ &\leq (2 \log n^d + 2) \cdot (4 \cdot 2^d \log n^2 + T(d-1, n)) \\ &\leq \dots \\ &\leq 2^{d+3} \log n \textstyle\sum_{i=1}^{d-2} (d \log n + 1)^i (i+1) + (2d \log n + 2)^{d-2} T(2, n) \\ &\leq 2^{d+3} \log^{d-1} n \cdot 2(d+1)^{d-1} + (2d \log n + 2)^{d-2} T(2, n) \\ &= O((2d)^{d-1} \log^{d-1} n). \end{aligned}$$

Of course, in practice one might be able to solve the intersection detection problem much faster by checking at various stages if

$$(TOP^P - BOT^Q)(x_1 \dots x_{d-1}) < 0, \text{ or } (TOP^Q - BOT^P)(x_1 \dots x_{d-1}) < 0,$$

rather than continuing the search until the minimum has been found. However, in the worst case it may well be necessary to proceed until the minimum has been reached (fig. 5).

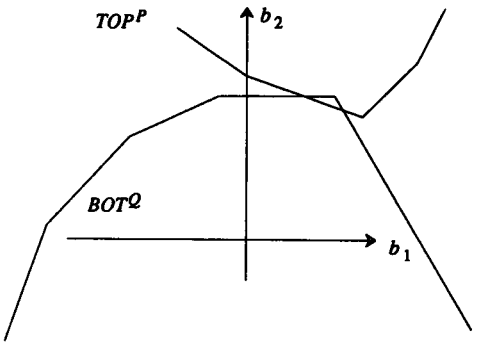

Fig. 5. The minimum of $TOP^P - BOT^Q$ is the only vertex where the difference is negative.

For $d = 3$, the space and preprocessing requirements of this algorithm are as follows. The equations of the $O(n)$ faces of P and Q require space $O(n)$. For the multidimensional binary search one has to store (a) a subdivision of the x_1-axis into no more than $n + 1$ partitions, and (b) a sequence of $O(n)$ edges for each one of the partitions. The total space requirements are therefore $O(n^2)$. The preprocessing can be done in time $O(n^2)$ by means of a plane sweep as described in [14].

For $d > 3$, the data structures required to do the search are essentially the same as the ones required to do the point location as described in section 3. Therefore, the space and preprocessing requirements are the same as for the hyperplane-polyhedron intersection detection problem. We obtain

THEOREM 4: *Given two d-dimensional convex polyhedra P and Q, P and Q can be tested for intersection in time $T(d, n)$ with $S(d, n)$ space and $PP(d, n)$ preprocessing:*

$P \cap H = \emptyset$?	$T(d, n)$	$S(d, n)$	$PP(d, n)$
$d = 2$	$O(\log n)$	$O(n)$	$O(n)$
$d = 3$	$O(\log n^2)$	$O(n^2)$	$O(n^2)$
$d > 3$	$O((2d)^{d-1} \log^{d-1} n)$	$O(n^{2^d})$	$O(n^{2^d})$

PROOF: follows from the preceding discussion. ■

An extension of these results to unbounded polyhedra is possible [4].

5. Conclusions.

We showed that in arbitrary, but fixed dimensions, the hyperplane-polyhedron and the polyhedron-polyhedron intersection detection problems can be solved in logarithmic and polylogarithmic time, respectively. For dimensions larger than three, these results are the first of their kind. There are two reasons why, as of now, these results are of only limited practical use. First, the coefficient which is exponential in d becomes prohibitively high for higher dimensions. Second, the space and preprocessing requirements make it difficult to apply these algorithms in practice. It is subject to further research to improve these results in order to achieve practical algorithms for intersection detection in higher dimensions. In particular, we suspect that lower bounds for space and preprocessing may be achieved at the expense of slightly higher time bounds for the detection algorithms. It should be noted though that our results are worst-case bounds. Recent experiments indicate that the average space and preprocessing requirements of our algorithms are much lower.

REFERENCES

[1] Lee, D. T. and Preparata, F. P., *Computational geometry – a survey*, IEEE Trans. on Computers C-33, 12 (Dec. 1984), pp. 1072–1101.

[2] Dantzig, G. B., *Linear Programming and its Extensions*, Princeton University Press, Princeton, NJ, 1963.

[3] Newman, W. M. and Sproull, R. F., *Principles of Interactive Computer Graphics*, McGraw-Hill, New York, NY, 1979.

[4] Günther, O., *Efficient Structures for Geometric Data Management*, Lecture Notes in Computer Science No. 337, Springer-Verlag, Berlin, 1988.

[5] Stonebraker, M., Sellis, T. and Hanson, E., *An analysis of rule indexing implementations in data base systems*, in *Proc. of the 1st International Expert Data Base Systems Conference*, April 1986.

[6] Chazelle, B. and Dobkin, D. P., *Detection is easier than computation*, in *Proc. 12th Annual ACM Symposium on Theory of Computing*, 1980, pp. 146–153.

[7] Chazelle, B. and Dobkin, D. P., *Intersection of convex objects in two and three dimensions*, J. ACM 34, 1 (Jan. 1987), pp. 1–27.

[8] Dobkin, D. P. and Munro, J. I., *Efficient uses of the past*, in *Proc. 21st Annual Symposium of Foundations of Computer Science*, Syracuse, NY, 1980, pp. 200–206.

[9] Dobkin, D. P. and Kirkpatrick, D. G., *Fast detection of polyhedral intersection*, Theoret. Comput. Sci. 27 (1983), pp. 241–253.

[10] Preparata, F. P. and Muller, D. E., *Finding the intersection of a set of N half-spaces in time $O(N \log N)$*, Theoret. Comp. Sci. 8 (1979), pp. 45–55.

[11] Brown, K. Q., *Geometric transformations for fast geometric algorithms*, Ph.D. dissertation, Carnegie-Mellon University, Pittsburgh, Pa., Dec. 1979.

[12] Edelsbrunner, H., *Algorithms in Combinatorial Geometry*, Springer, Berlin, 1987.

[13] Dobkin, D. P. and Souvaine, D. L., *Computational geometry – A user's guide*, in *Algorithmic and geometric aspects of roboticss*, Advances in robotics, Vol. 1, C. Yap and J. Schwartz (eds.), Lawrence Erlbaum Assoc., Hillsdale, N.J. 1987.

[14] Preparata, F. P. and Shamos, M. I., *Computational Geometry*, Springer, New York, NY, 1985.

[15] Rockafellar, R. T., *Convex Analysis*, Princeton University Press, Princeton, NJ, 1970.

[16] Edelsbrunner, H., Guibas, L. J. and Stollfi, J., *Optimal point location in a monotone subdivision*, SIAM J. Comput. 15, 2 (1986), pp. 317–340.

[17] Dobkin, D. P. and Lipton, R. J., *Multidimensional searching problems*, SIAM J. Comput. 5, 2 (June 1976), pp. 181–186.
[18] Edelsbrunner, H., O'Rourke, J. and Seidel, R., *Constructing arrangements of lines and hyperplanes with applications*, SIAM J. Comput. 15, 2 (1986), pp. 314–363.
[19] Meggido, N., *Linear programming in linear time when the dimension is fixed*, J. ACM 31, 1 (Jan. 1984), pp. 114–127.
[20] Grünbaum, B., *Convex Polytopes*, John Wiley, London, 1967.