SIAM J. COMPUT.
Vol. 13, No. 3, August 1984

003

CONVEX PARTITIONS OF POLYHEDRA: A LOWER BOUND AND WORST-CASE OPTIMAL ALGORITHM*

BERNARD CHAZELLE†

Abstract. The problem of partitioning a polyhedron into a minimum number of convex pieces is known to be NP-hard. We establish here a quadratic lower bound on the complexity of this problem, and we describe an algorithm that produces a number of convex parts within a constant factor of optimal in the worst case. The algorithm is linear in the size of the polyhedron and cubic in the number of reflex angles. Since in most applications areas, the former quantity greatly exceeds the latter, the algorithm is viable in practice.

Key words. Computational geometry, convex decompositions, data structures, lower bounds, polyhedra

1. Introduction. The general problem of decomposing complex structures into simpler components has received a great deal of attention recently [1], [4], [5], [8]. The reason for this concern comes partly from the impossibility of applying many of people's favorite geometric algorithms to nonconvex structures. Often, decomposing the structures into convex parts and applying the algorithms to each part is one way to overcome this difficulty. For example, intersection [2] and searching problems [9] can be solved efficiently by means of convex decompositions. One of the forefathers of decomposition algorithms is Garey et al.'s algorithm [4] for partitioning an n-gon into triangles in $O(n \log n)$ time. Minimality considerations were addressed later on in [1], where an $O(n+N^3)$ time algorithm was given for decomposing an n-gon with N reflex angles into a *minimum* number of nonoverlapping convex pieces. Several variants of this problem were shown to be NP-hard [8]; in particular, the generalization of the problem to polygons with *holes* [5]. This result was to be used as a stepping stone to prove that the following problem was NP-hard.

Given a three-dimensional polyhedron P, what is the smallest set of pairwise disjoint convex polyhedra, whose convex union is exactly P?

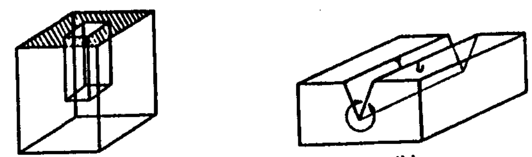
This paper is devoted to this problem, and is organized along the following lines: in § 2, we present the basic concepts and outline an effective method for decomposing an arbitrary polyhedron into convex pieces. Let n and N designate respectively the size of the input and the number of reflex angles into the polyhedron. We prove that the algorithm never produces more than approximately $N^2/2$ convex pieces. We show in § 3 that this figure is optimal in the worst case up to within a constant factor. To do so, we exhibit a polyhedron P with an arbitrary number of reflex angles N and $n = O(N)$ vertices, and we prove that P necessarily has $\Omega(N^2)$ convex parts. Of course, by a trivial output size argument, this result also establishes a quadratic lower bound on the time complexity of the decomposition problem. Finally in § 4, we give the details of the algorithm outlined at the beginning.

Before proceeding, we shall set our notation. We define a three-dimensional polyhedron as a finite, connected set of simple plane polygons, such that every edge of each polygon belongs to exactly one other polygon. To exclude degenerate cases (e.g., two cubes connected by a single vertex), we also require that the polygons surrounding each vertex form a simple circuit [3, p. 4]. Note that this definition does

* Received by the editors May 17, 1982, and in revised form July 21, 1983. This work was partly supported by a Yale fellowship and by the Defense Advanced Research Projects Agency under contract F33615-78-C-1551.
† Department of Computer Science, Brown University, Providence, Rhode Island 02912.

not prevent faces from having holes (Fig. 1a). A face with k holes is said to be of *genus k*. Similarly, polyhedra may have holes (i.e., *handles*), and we define the genus of a polyhedron as the genus of the surface formed by its boundary [6]. It follows from the definition that a polyhedron may not have interior boundaries.

Let P be a polyhedron with n vertices $v_1, \cdots, v_n$, p edges $e_1, \cdots, e_p$ and q faces $f_1, \cdots, f_q$. A necessary condition for vertices, edges, and faces to be adjacent is to have at least one point in common. For simplicity, however, we will say that a face and an edge or two faces are adjacent if and only if they share an entire line segment. If T and U are two adjacent faces intersecting in a segment L, we define the angle (T, U) as the angle between two segments lying respectively on T and U and perpendicular to L. Recall that there is no natural orientation of angles in Euclidean space. Thus, to avoid ambiguity, the angle (T, U) will always be measured between 0 and 360 degrees with respect to a given side of the pair T, U. Noticing that each face of P has an outer and an inner side, we define a *notch* of P as an edge with its adjacent faces forming a reflex angle (i.e. >180 degrees) with respect to their inner side (Fig. 1b). Let $g_1, \cdots, g_N$ denote the notches of P.



(a) (b)

FIG. 1. a) *A face of a polyhedron with a hole in the middle.* b) *A notch of a polyhedron.*

2. The basic method. It is easy to see that the presence of notches in a polyhedron is characteristic of its nonconvexity [3, p. 4]. Thus we can view a convex decomposition of P either as a partition of P into convex polyhedra or as a set of *cuts* performed through P in order to resolve the reflex angles at its notches. This suggests a naive decomposition algorithm, which we proceed to describe next.

2.1. The naive decomposition. Informally, a notch can be removed by cutting along a plane adjacent to it so as to resolve the reflex angle between its adjacent faces. More precisely, let g be a notch of the polyhedron P with f_i and f_j its adjacent faces, and let T be a plane which contains g and resolves its reflex angle, i.e., such that both angles (f_i, T) and (T, f_j), as measured from the inner side of f_i and f_j, are not reflex. The intersection of T and P is in general a set of polygons. These polygons may have holes and the holes may themselves contain other polygons (Fig. 1a). Let S be the unique polygon containing g. We call S a *cut* of the naive decomposition. It is clear that cutting along S will remove the notch g. Note that, in general, this operation will break P into two pieces. If P has a nonzero genus, however, removing a notch may simply cut a handle of P and preserve its connectivity. In this case, the polyhedron obtained has two distinct faces with the same geometric location (Fig. 2a). Other intriguing effects may be observed and it is worthwhile to mention some of them.

If the polygon S has holes, removing g may create a handle in either of the two parts produced (Fig. 2b). Therefore the added genus of all the pieces produced thus far will increase by one. We also observe that the operation may produce one piece, while removing a handle and creating another handle (Fig. 2c). We will thus treat the more general case where the polyhedron P may have arbitrary genus, since the naive

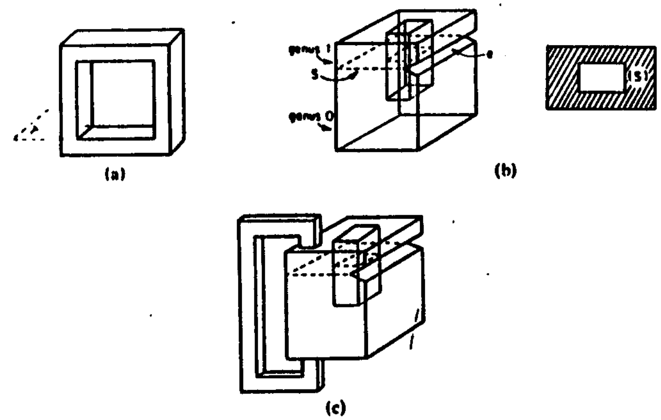


FIG. 2. *Removing a notch.*

decomposition may produce intermediate objects of higher genuses. In spite of these intricacies, we can easily show that repeating the cutting process on each remaining nonconvex part will eventually produce a convex decomposition in a finite number of steps. To find out how many convex parts such a decomposition may generate, we first observe that, at any time, any notch of a part is either a notch of P or the subsegment of a notch of P, called a *subnotch*. This follows from the fact that a cut may intersect other notches, thus *duplicating* them (Fig. 3). Note, however, that no new notch is ever created. At worst, each cut may intersect all of the other notches or subnotches present in the polyhedron considered. If $f(N)$ is the maximum number of cuts which a complete decomposition may necessitate, we have $f(0)=0$, and

$$f(N) \leq 2f(N-1)+1.$$

Therefore, at most 2^N-1 cuts are needed, which shows that the procedure will always converge and produce at most 2^N convex parts. Unfortunately, as shown in [1], this scheme may indeed produce an exponential number of pieces, so an alternate method is in order.

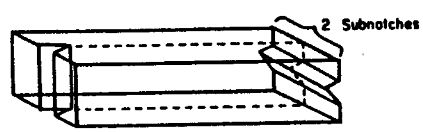


FIG. 3. *The duplication of notches.*

LEMMA 1. *There exist two constants a, b and a class of polyhedra $P(n)$ with $O(n)$ vertices, such that for any $n>a$, the naive decomposition applied to P produces at least 2^{bn} convex parts.*

Proof. See [1]. □

2.2. The naive decomposition revisited. To avoid an exponential blow-up in the number of pieces, we will remove all the subnotches of each notch with coplanar cuts. This will ensure that all the cuts used in the removal of a notch duplicate a total of at most $N-1$ other notches, leading to an $O(N^2)$ upper bound on the number of convex parts. More precisely, let us define for each notch g_i a plane T_i that resolves its reflex

angle. We proceed as before, with the additional requirement that the cuts of each subnotch of g_i should be coplanar with T_i.

THEOREM 2. *The revised naive decomposition algorithm applied to P yields at most $N^2/2+N/2+1$ convex parts.*

Proof. We can assume that all the subnotches of a notch are removed consecutively. Since the cuts corresponding to the subnotches of g_i are coplanar, their union intersects every other notch in at most one point. It follows that, at the ith step, each remaining notch will have been broken up into at most $i+1$ subnotches, and step $i+1$ will introduce at most $i+1$ polyhedra into the decomposition. □

In the last section of this paper, we will describe an effective method for carrying out the naive decomposition. But first we will establish a lower bound on the size of any convex decomposition.

3. A quadratic lower bound on the number of convex parts.

3.1. Introduction. The algorithm described above produces $O(N^2)$ convex parts, thus saving us from an exponential blow-up. We may yet wonder whether $O(N)$ parts is not always achievable, as is the case in two dimensions [1]. We next tackle this problem and prove that this $O(N^2)$ upper bound is indeed tight. To achieve our goal, we must exhibit a class of polyhedra which cannot be decomposed into fewer than cN^2 parts. The technique used to derive this lower bound is based on volume considerations. We define a portion Σ of the polyhedron P and, observing that a decomposition of P also realizes a partition of Σ, we study the contribution of each convex part to this partitioning. The crux is to show that a convex part can only have a small piece lying in Σ, and therefore lots of convex parts are needed to fill up Σ. To realize this condition, we must carefully design Σ, giving it a warped shape so that its intersection with any convex object can never occupy too much space. The fact that Σ must be defined by means of straight lines suggests giving it the shape of a hyperbolic paraboloid. Recall that this surface can be generated by two sets of orthogonal lines [11, p. 649].

The main idea can be summarized as follows: Σ has thickness ε so that its volume is approximately εN^2. The warpness of a hyperbolic paraboloid will then ensure that since Σ is bounded by notches, the "chunk" of Σ removed by any convex piece can only be very small, i.e. have volume ε. As a result at least $\Omega(N^2)$ convex parts will be necessary to decompose Σ.

3.2. Description of the polyhedron *P*. P is essentially a rectangular parallelepiped with a series of $N+1$ notches cut through the lower face and $N+1$ similar notches cut through the upper face (Fig. 4a, b). The two faces adjacent to any notch form a very small angle and, for our purposes, can be regarded as a single vertical quadrilateral. Thus, we have $N+1$ such quadrilaterals emanating from the lower face, all of which are vertical, parallel to the plane Oxz, and equidistant. The upper edges of these quadrilaterals are called the *bottom notches* of the polyhedron P, and are designated $BOT0, \cdots, BOTN$ in ascending Y-value. To achieve the desired warping, all the bottom notches lie on the hyperbolic paraboloid $z=xy$. The $N+1$ quadrilaterals emanating from the upper face of P are parallel to the plane Oyz and satisfy the same specifications. Similarly, their lower edges are called the *top notches* of P and are designated $TOP0, \cdots, TOPN$ in increasing X-order. All these notches lie on the hyperbolic paraboloid $z=xy+\varepsilon$. We now give a more precise definition of P by characterizing its significant vertices with the system of axes indicated in Fig. 4b. Note that the origin O is the intersection of $BOT0$ with the vertical plane passing through $TOP0$. The upper face of the parallelepiped lies on the plane $z=2N^2$ and its lower face, on the plane $z=-2N$. This ensures that all bottom and top notches fit strictly

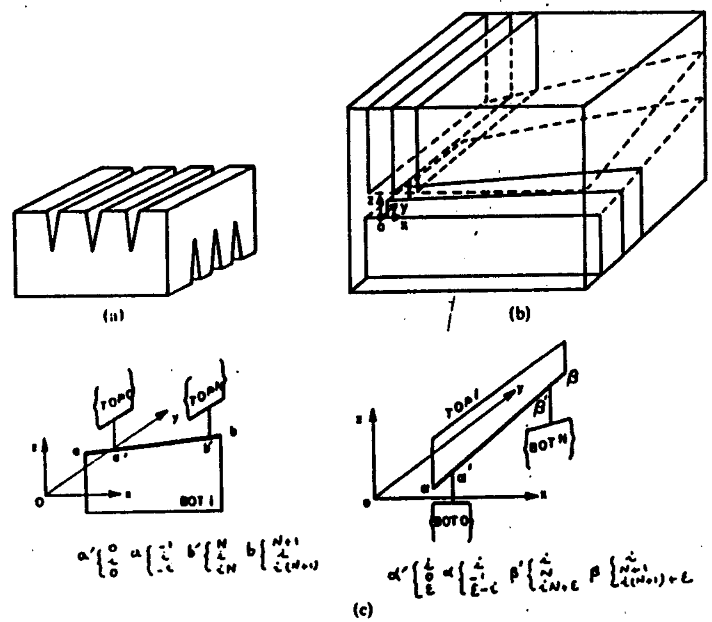


FIG. 4. *The polyhedron P.*

between these two faces. Also the parallelepiped has a depth and width of $N+2$. Fig. 4c gives all the coordinates of the top and bottom notches.

3.3. Decomposing P into convex parts. We define Σ as the portion of P comprised between the two hyperbolic paraboloids $z=xy$ and $z=xy+\varepsilon$ and the four planes $x=0$, $x=N$, $y=0$, $y=N$. Σ is a cylinder parallel to the z-axis, of height ε, whose base is the region of the hyperbolic paraboloid $z=xy$ with $0\le x, y\le N$ (Fig. 5). Let $Q_1, \cdots, Q_m$ be any convex decomposition of P and let Q_i^* denote the intersection of Q_i and Σ. Since Σ lies inside P, the set of Q_i^* forms a partition of Σ. Note that Q_i^* may consist of 0, 1, or several blocks, most of which are likely not to be polyhedra. Our goal is to prove that $m\ge cN^2$ for some constant c, by showing that the volume of Q_i^* cannot be too large. By volume of Q_i^*, we mean the sum of all the volumes of the blocks composing Q_i^*. We first characterize the shape and the orientation of the large Q_i^*'s, which permits us to derive an upper bound on their maximum volume.

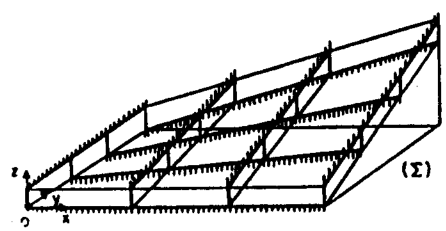


FIG. 5. *The warped region Σ.*

For all i between 0 and N, let $BOTi^*$ (resp. $TOPi^*$) denote the vertical projection of $BOTi$ (resp. $TOPi$) on the plane Oxy. The set of all $BOTi^*$ and $TOPi^*$ forms a regular square grid of N^2 cells, each cell being itself a one-by-one square. Consider the two points A: (x_A, y_A, z_A) and B: (x_B, y_B, z_B) lying in Q_i^*. We will investigate their possible positions when their vertical projections on the grid lie on two parallel lines which are at a distance 2 of each other. Wlog, we will assume that $x_A\le x_B$. We have the following result.

LEMMA 3. *Let A and B be two points of Q_i^*.*

1. *If x_A is an integer i with $0\le i\le N-2$ and $x_B=x_A+2$, then $y_B-y_A\le 2\varepsilon$.*
2. *If y_A is an integer i with $2\le i\le N$ and $y_B=y_A-2$, then $x_B-x_A\le 2\varepsilon$.*

Proof. Recall that the lines supporting $BOTi$ and $TOPi$ are defined respectively by $(y=i, z=ix)$ and $(x=i, z=iy+\varepsilon)$.

1. Let the coordinates of A and B be respectively $(x_A=i, y_A, z_A)$ and $(x_B=i+2, y_B, z_B)$ with $0\le i\le N-2$. Let T be the middle point of the segment AB, $(x_T=i+1, y_T=(y_A+y_B)/2, z_T=(z_A+z_B)/2$, and consider the point C on $TOPi+1$ with coordinates $(x_C=x_T, y_C=y_T, z_C=x_Cy_C+\varepsilon)$. Since Q_i is convex, the whole segment AB lies in Q_i and T lies inside P, therefore $z_T\le z_C$. Also, since A and B lie in Σ, $x_Ay_A\le z_A$ and $x_By_B\le z_B$, therefore $(x_Ay_A+x_By_B)/2\le z_T$. Combining these results yields $(x_Ay_A+x_By_B)/2\le z_C$, therefore

$$iy_A+(i+2)y_B\le 2(\varepsilon+(i+1)(y_A+y_B)/2),$$

hence

$$y_B-y_A\le 2\varepsilon.$$

2. The proof is very similar. The coordinates of A and B are respectively (x_A, i, z_A) and $(x_B, i-2, z_B)$ with $2\le i\le N$. The middle point of AB is now defined by T: $(x_T=(x_A+x_B)/2, y_T=i-1, z_T=(z_A+z_B)/2)$ and lies right above the point of $BOTi-1$, C: $(x_C=x_T, y_C=y_T, z_C=x_Cy_C)$, therefore $z_C\le z_T$. Since both A and B belong to Σ, $z_A\le x_Ay_A+\varepsilon$ and $z_B\le x_By_B+\varepsilon$, therefore

$$2(i-1)(x_A+x_B)/2\le 2\varepsilon+ix_A+(i-2)x_B$$

and

$$x_B-x_A\le 2\varepsilon$$

which completes the proof. □

When A is now any point in Σ with $0\le x_A\le N-2$ and $2\le y_A\le N$, we can still use the previous result to delimit the region where B cannot lie. The shaded area in Fig. 6 represents the forbidden area. Assume that $x_B-\lceil x_A\rceil>2$ and let A' and B' be the two points on the segment AB with $x_{A'}=\lceil x_A\rceil$ and $x_{B'}=x_{A'}+2$. Since A' and B' lie in Q_i^*, we can apply the result of Lemma 3 on these two points. It follows that $y_{B'}-y_{A'}\le 2\varepsilon$, therefore

$$\frac{y_B-y_A}{x_B-x_A}=\frac{y_{B'}-y_{A'}}{x_{B'}-x_{A'}}\le\varepsilon.$$

This shows that B must lie under the line $y=y_A+\varepsilon(x-x_A)$ as indicated in Fig. 6. Similarly, we can show that if [illegible] >2, B must lie on the left-hand side of the line $x=x_A+\varepsilon($[illegible]$)$.

We can now attack our main problem, that is, evaluating the maximum [illegible] Q_i^*. Recall that Q_i^* may be [illegible] of several blocks [illegible]
Q_i^* with maximum [illegible]

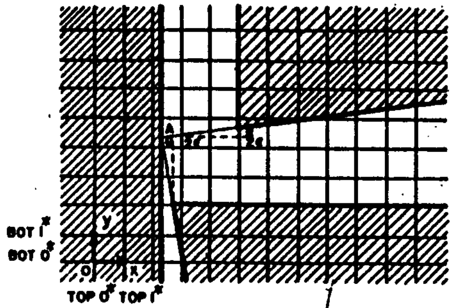


FIG. 6. *The forbidden area.*

*BOT*0 or *TOPN* in order to have the points B and C of Fig. 7 well defined. More precisely, we require that

$$0 < x_A < N-2, \qquad 2 < y_A < N-3\varepsilon.$$

Fig. 7 is only a reproduction of Fig. 6, specifying the regions of interest with respect to A. Note that VA, VB, and VC really denote the intersection of Σ with the vertical cylinders whose bases are represented by the shaded areas in Fig. 7. We know that Q_f^* lies entirely in the union of VA, VB, and VC. So we can partition Q_f^* into 3 parts, $VA1$, $VB1$, and $VC1$, defined respectively as the intersection of Q_f^* with VA, VB, and VC.

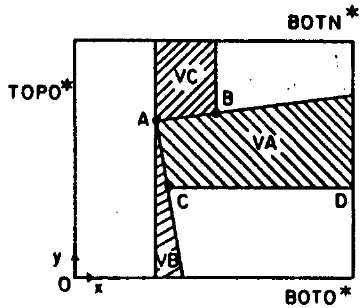


FIG. 7. *Restricting the domain where Q_f^* has to be computed.*

1) *Evaluating the volume of* $VA1$. When there is no ambiguity, we will refer to a three-dimensional object and to its volume by using the same symbol (in this case $VA1$). To derive an upper bound on the volume $VA1$, we integrate a vertical section of $VA1$ along a direction "almost" parallel to Y-axis. This permits us to exploit the warping of Σ in order to bound the area of the section, while having a very short interval of integration. More precisely, let P_w be the vertical plane (P_w: $y = x \tan\theta + w$), and $S(\theta, w)$ the area of the cross section formed by the intersection of P_w and $VA1$. The volume of $VA1$ can be computed by integrating $S(\theta, w)$ along a line normal to the planes P_w.

$$VA1 = \int S(\theta, w)\cos\theta \, dw.$$

If we choose θ larger than (Ox, AB) (Fig. 7), all values of $S(\theta, w)$ will be null outside of A and D, that is, for:

$$w > w_A = y_A - x_A \tan\theta$$

and

$$w < w_D = y_D - x_D \tan\theta.$$

Letting $S(\theta)$ be the maximum value of $S(\theta, w)$ for all w, we have

$$VA1 \leq (w_A - w_D)S(\theta)\cos\theta$$

and from $y_A - 3 \leq y_D$ and $x_D = N$, we derive

(1) $$VA1 \leq (3 + N\tan\theta)S(\theta)\cos\theta.$$

The condition on θ is easily expressed as

(2) $$\varepsilon < \tan\theta.$$

We are now reduced to establishing an upper bound on $S(\theta, w)$. We will find it more convenient to change the system of coordinates so that the point $(0, w, 0)$ becomes the new origin and the line ($z = 0$, $y = x\tan\theta + w$) becomes the new X-axis. We express the old coordinates (x, y, z) of any point in terms of the new coordinates (X, Y, Z) as follows:

$$x = X\cos\theta - Y\sin\theta,$$
$$y = w + X\sin\theta + Y\cos\theta,$$
$$z = Z.$$

The hyperbolic paraboloid $z = xy$ is now described by the equation:

$$Z = (X\cos\theta - Y\sin\theta)(w + X\sin\theta + Y\cos\theta)$$

and the intersection of P_w with Σ is a strip in the plane $(Y = 0)$ comprised between the two parabolas:

$$(f)\colon Z = X^2\sin\theta\cos\theta + Xw\cos\theta,$$
$$(g)\colon Z = X^2\sin\theta\cos\theta + Xw\cos\theta + \varepsilon.$$

Before proceeding further, we will prove a technical result about areas covered by parabolas. Suppose that we have two parabolas of the previous type, described by $f(x) = ax^2 + bx$ with $a > 0$, and $g(x) = f(x) + \varepsilon$. Let $T(x)$ be the area comprised between the parabola f and the tangent to g at x (Fig. 8). We can show the following

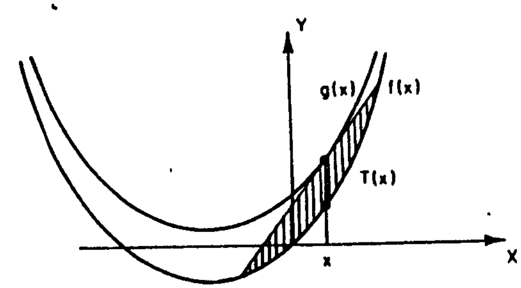


FIG. 8. *The function $T(x)$.*

LEMMA 4. *$T(x)$ is a constant function equal to $4\varepsilon\sqrt{\varepsilon/a}/3$.*

Proof. The tangent to g at x has the equation:

$$Y=(2ax+b)(X-x)+ax^2+bx+\varepsilon$$

and intersects the parabola f at the points with X-coordinates x_1 and x_2, solutions of

$$(2ax+b)(X-x)+ax^2+bx+\varepsilon=aX^2+bX$$

that is,

$$aX^2-2axX+ax^2-\varepsilon=0$$

yielding $x_1=x-\sqrt{\varepsilon/a}$ and $x_2=x+\sqrt{\varepsilon/a}$. It is now straightforward to evaluate $T(x)$.

$$T(x)=\int[(2ax+b)(t-x)+ax^2+bx+\varepsilon-at^2-bt]\,dt$$

that is,

$$T(x)=(x_2-x_1)(\varepsilon-ax^2+ax(x_1+x_2)-a(x_1^2+x_1x_2+x_2^2)/3)$$

therefore

$$T(x)=4\varepsilon\sqrt{\varepsilon/a}/3,$$

which establishes the proof. □

We will now take a closer look at the structure of the parabolic strip formed by the intersection of Σ and P_w which, we know, contains $S(\theta,w)$. Here again, $S(\theta,w)$ designates both the surface and its area. Recall that $S(\theta,w)$ may consist of several disconnected pieces. The intersection of P_w and Σ is a connected strip enclosed between two vertical lines $X=a$, $X=b$ (the exact values of a and b are irrelevant for our purposes). Also, as illustrated in Fig. 9a, the upper parabola of this strip, g, intersects the top notches, *TOPk*, at regular intervals of length $1/\cos\theta$. Let F denote the convex polygon formed by the intersection of Q_i and P_w. Assuming that F is not empty, we distinguish two cases:

1) No point of F lies above the parabola g (Fig. 9b).

Since F is convex, there exists a line L separating g and F. Since L', the tangent to g parallel to L, also separates g and F, the X-coordinate, u, of the tangent point satisfies $S(\theta)\le T(u)$.

2) There exists a point M in F lying above g (Fig. 9c).

Using the notation of Fig. 9c, it is clear that $S(\theta,w)$ lies totally in $L\cup C\cup R$. Since the areas of L and R are dominated by $T(X_k)=T(X_{k+1})$, and the area of C is exactly $\varepsilon/\cos\theta$, we have

$$S(\theta,w)\le 2T(X_k)+\varepsilon/\cos\theta.$$

From Lemma 4, it follows that

$$S(\theta,w)\le\varepsilon/\cos\theta+\tfrac{8}{3}\varepsilon\sqrt{\varepsilon/\sin\theta\cos\theta}.$$

And from (1), we derive

$$VA1\le\varepsilon(3+N\tan\theta)(1+\tfrac{8}{3}\sqrt{\varepsilon/\tan\theta}).$$

II) *Evaluating the volume of* $VC1$. Since the hyperbolic paraboloids are symmetric about x and y, the same computation will give an upper bound on $VC1$. Note that now, no condition like (2) must be set on the angle giving the direction of

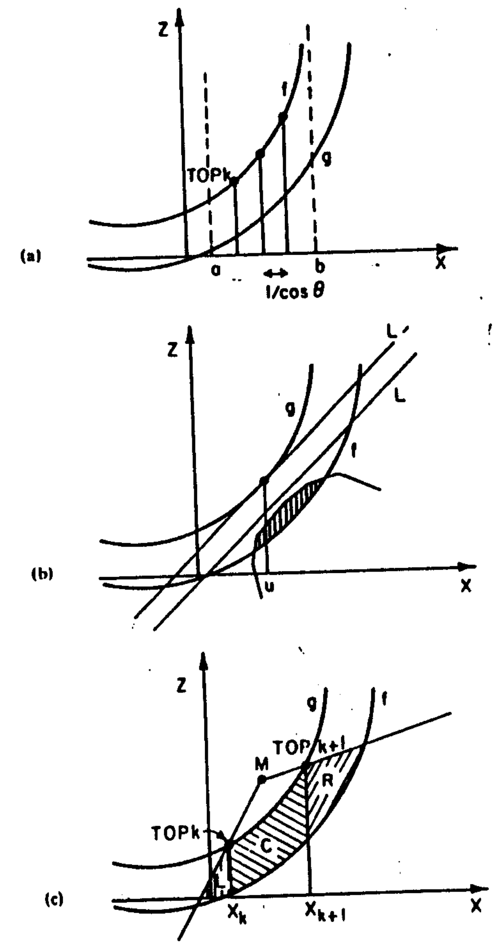


FIG. 9. *Evaluating $S(\theta,w)$.*

integration. For convenience, we will take it equal to θ, however. Thus, we have

$$VC1\le\varepsilon(3+N\tan\theta)(1+\tfrac{8}{3}\sqrt{\varepsilon/\tan\theta}).$$

III) *Evaluating the volume of* $VB1$. The shaded area of Fig. 7 corresponding to VB has a maximum area of $\varepsilon N^2/2$, therefore the volume of VB is dominated by $\varepsilon^2N^2/2$. This yields an upper bound on $VB1$

$$VB1\le\varepsilon^2N^2/2.$$

3.4. The lower bound on the number of convex parts. We can now prove our main result.

THEOREM 5. *There exist a constant c and a class of polyhedra involving an arbitrarily large number of vertices such that each polyhedron cannot be decomposed into fewer than cn^2 convex parts, where n is the number of vertices.*

Proof. Recall that the volumes computed in the previous section are only relevant for the points A satisfying

$$0 < z_A < N-2 \quad \text{and} \quad 2 < y_A < N-3\varepsilon.$$

Let V be the corresponding portion of Σ. We have

$$V = (N-2)(N-3\varepsilon-2)\varepsilon.$$

Since no Q_i can contribute more than $VA1 + VB1 + VC1$ to the volume V, we can derive the following lower bound on the number m of convex parts Q_i,

$$m \geq \frac{V}{VA1 + VB1 + VC1}.$$

Assume that N is large enough and that $\varepsilon < \sin\theta < \tan\theta < 1/N^2$. Relation (2) is then satisfied, and we have

$$VA1,\ VC1 < (1+\tfrac{3}{5})(3+1/N)\varepsilon < 16\varepsilon.$$

Also, since

$$V > \varepsilon N^2/2$$

it follows that

$$m > \varepsilon N^2/2(32\varepsilon + \varepsilon^2 N^2/2),$$

hence

$$m > N^2/66$$

which completes the proof. □

4. The decomposition algorithm. We give a precise description of the decomposition algorithm outlined in § 2. We will show that it is possible to decompose P into $O(N^2)$ pieces in $O(nN^2(N + \log n))$ time, using $O(nN^2)$ storage. We will also indicate that at the price of added complication, we can reduce the running time to $O(nN^3)$.

The first issue to investigate is the mode of representation used for describing a polyhedron. Since many practical problems involve dealing with faces rather than edges or vertices, we may assume that the edges enclosing a given face are readily available. More precisely, we require the data structure chosen to provide three types of lists:

1. *Edge-to-face lists*: contain the names of the two faces adjacent to each edge.
2. *Face-to-edge lists*: give the sequence of edges enclosing each face in clockwise order.
3. *Adjacency lists*: provide a set of the vertices adjacent to each vertex.

Note that the faces of a nonconvex polyhedron may be polygons with holes. In that case, each face-to-edge list should provide clockwise descriptions of the outer as well as of the inner boundaries. We call a *graph representation* of a polyhedron any representation providing the above lists. We may notice that these representations are redundant, but they are chosen to be so for the sake of simplicity. These lists reflect the *size* of the polyhedron accurately, however, since they clearly require $O(p)$ storage. Recall that p is the number of edges in P.

Because decomposing P consists essentially of dividing it up with successive cuts, we first consider the problem of computing graph representations for the two polyhedra P_1 and P_2 into which a cut S breaks up P. For the time being, we will assume P to be of genus 0. In the following, we will successively show I) how to compute the intersection of T and P, II) how to obtain S from it, and III) how to compute the two polyhedra P_1 and P_2. But before proceeding we need to take a closer look at the problem and prove a preliminary result.

Let e be the edge through which the cut is performed. We first compute W, the intersection of P with the plane supporting S. W may consist of a set of polygons with holes, which may themselves contain polygons of the same nature. We identify S as the unique polygon which contains the edge e (Figs. 2, 11). Whereas it is immediate to compute a description of the outer boundary of S, obtaining the inner boundaries (if any) requires more work. Viewing W as a set of nonintersecting boundaries, we first determine all the boundaries in W which lie inside the outer boundary of S, thus forming a set W^*. Next, we keep all the *maxima* of W^*. A boundary is said to be a *maximum* if it is not contained in any other boundary. We can show that the two problems are very closely related, and that an algorithm for solving one can easily be modified to handle the other.

LEMMA 6. *All the maxima of a set W of boundaries can be found in $O(n \log n)$ time, if n is the total number of vertices in W.*

Proof. To begin with, we should note that the nonintersection of the boundaries of W implies that W always has at least one maximum. The method which we will describe is inspired from Shamos and Hoey's algorithm for intersecting pairs of segments [10]. The crucial observation to make is that the intersection of a vertical line L with the maxima of W forms a set (possibly empty) of disjoint segments. The endpoints of each segment lie on some edges of W, and the vertical line L induces a total ordering R on the set E of these edges. E consists exactly of all the edges of maxima which intersect L (Fig. 10a). We say that two edges of E, consecutive with respect to R, are *linked* if the vertical segment joining them lies in a maximum of W. Note that consecutive pairs of edges in R are alternately linked and not linked. For any point v of L, we define $h(v)$ (resp. $l(v)$) as the first edge in E above v (resp. below v). If no such edge exists, $h(v)$ or $l(v)$ is 0 (Fig. 10a). The notion of *above* and *below* is, of course, defined with respect to the vertical line L. Similarly, the order of two edges of W is defined with respect to a common intersecting vertical line. Actually, this order is the same for any vertical line since the edges of W can intersect only at their endpoints. If v is the leftmost vertex of a polygon P of W, P is a maximum if and only if $h(v)$ and $l(v)$ are not linked. This condition is clearly necessary since, if $h(v)$ and $l(v)$ are linked, they belong to the same polygon, which cannot be P since v is its leftmost vertex. To see that it is sufficient, assume that P is not a maximum; then there is a unique maximum Q in W which contains P, and Q must intersect the vertical line passing through v, therefore the intersection is a segment containing v and the pair $h(v)$, $l(v)$ must be linked.

The algorithm proceeds as follows: we sweep a vertical line from left to right, passing through each vertex v in W. The vertices are maintained in sorted order (by X-values) in a set O. We first check if v is the leftmost vertex of a polygon P of W. If it is, we can decide immediately if P is a maximum by finding whether $h(v)$ and $l(v)$ are linked. If they are, P is not a maximum and all its vertices are deleted from O. Otherwise, P is a maximum. Actually, since nonmaxima are removed as soon as their leftmost vertex is encountered, the polygon containing v is a maximum in all the other cases (i.e., when v is not a leftmost vertex). Then we can simply update the ordering R with the functions *insert* and *delete*, as well as the linked pairs with the

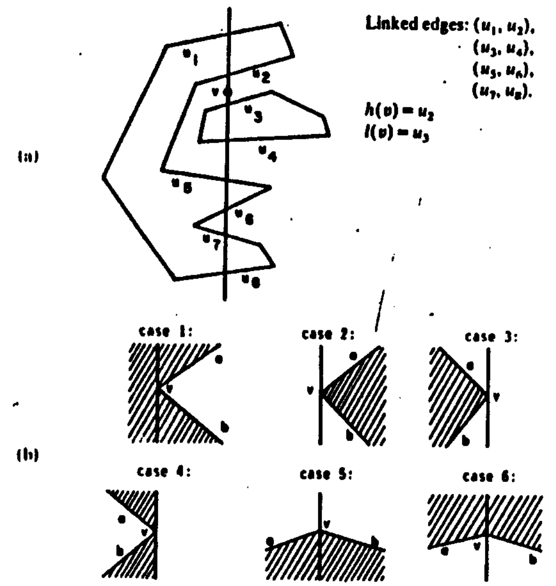


FIG. 10. a) *The ordering R.* b) *The algorithm for computing maxima.*

functions *link* and *unlink*. This is fairly straightforward and the algorithm we next present is self-explanatory.

```
MAXIMUM(W)
Q = Set of vertices in W stored in order
by x-values.
R = Ø.
for all v in Q (in ascending x-order)
  begin
   Let P be the polygon to which v belongs.
   If v is the leftmost vertex of P
     and h(v), l(v) are linked
     then "P is not a maximum"
          delete all vertices of P from Q
     else "P is a maximum"
          UPDATE(R, v)
  end
UPDATE(R, v)
Let a, b be the two edges adjacent to v.
Switch to the case corresponding to Fig. 10b.
case 1:
  insert (a), insert (b)
  unlink (h(v), l(v))
  link (h(v), a)
  link (b, l(v))
  break
```



```
case 2:
  insert (a), insert (b)
  link (a, b)
  break
case 3:
  delete (a), delete (b)
  unlink (a, b)
  break
case 4:
  delete (a), delete (b)
  unlink (h(v), a)
  unlink (b, l(v))
  link (h(v), l(v))
  break
case 5:
  delete (a), insert (b)
  unlink (a, l(v))
  link (b, l(v))
  break
case 6:
  delete (a), insert (b)
  unlink (h(v), a)
  link (h(v), b)
  break
```

Note that when the algorithm terminates, only the vertices of maxima will remain in Q, thus the maxima can be obtained from Q in $O(n)$ time. To implement the algorithm efficiently, we can store Q as a doubly-linked list with random-access to the nodes, thus allowing constant time deletions. R can be maintained as a balanced tree, so that the functions h, L, *insert*, and *delete* perform in logarithmic time. *Link*(u, v) will simply set two pointers, one from u to v, and the other from v to u, while *unlink*(u, v) will remove these pointers. With this implementation, the algorithm requires $O(n \log n)$ time. Note that all the preprocessing needed involves sorting the vertices by X-values and computing the leftmost vertices, all of which also takes $O(n \log n)$ time. □

We can now turn back to the problem of dividing up a polyhedron P. Recall that the intersection of P with the plane T supporting the cut S is in general a set of polygons. These polygons may have holes which may themselves contain other polygons of the same kind. We first compute S, from which we derive P_1 and P_2.

1) *Computing the intersection of P and T.* Consider each face F of P in turn and report all the edges of F which intersect the plane T, yet do not lie in T. This includes all the edges of the inner and outer boundaries. Let $a_1, \cdots, a_k$ denote the intersections of T with these edges, as they appear in sorted order on the line supporting the intersection of F and T. Call u_i the edge of F intersecting T at a_i. Observing that the intersection of T and F is made up of the segments $a_1a_2, \cdots, a_{k-1}a_k$ (Fig. 11b), we set two pointers for each pair (u_{2i-1}, u_{2i}); one from u_{2i-1} to u_{2i} and the other from u_{2i} to u_{2i-1}. Iterating on this process for all faces of P will eventually provide doubly-linked lists for all the boundaries of the polygons of the intersection of P and T. Let U denote this set of boundaries. Since each edge is considered at most twice, all these operations take $O(p)$ time, except for the sorts, each of which requires $O(p'_i \log p'_i)$ time, where p'_i is the number of edges intersecting T involved in the face considered. Since each edge appears on two faces, the sum of all the p'_i is less than or

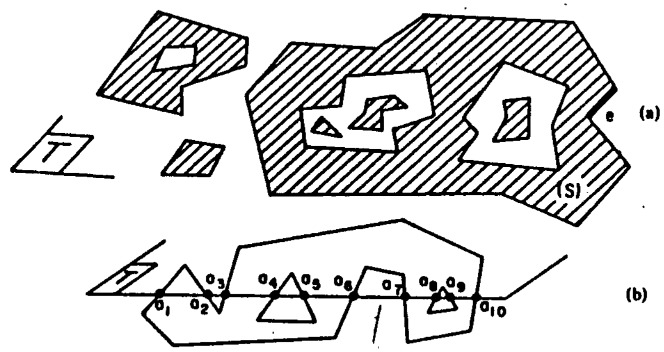


FIG. 11. a) *A cut S.* b) *The edges of S.*

equal to $2p'$, which leads to an $O(p' \log p')$ running time (similarly, p' is the number of edges of P intersecting T). Note that the conversion of the doubly-linked lists of u_i into lists of a_i is straightforward in general. Some special cases may yet be encountered, when a_i is the endpoint of u_i and several edges are adjacent to a_i. It is easy to see, however, that those cases can be handled separately without altering the total running time of the algorithm, which is $O(p+p' \log p')$.

II) *Computing S.* To begin with, we determine the outer boundary of S, denoted S^*, by identifying the boundary in U which contains the edge e. To find the inner boundaries is somewhat more involved. We first form the subset W of U consisting of all the boundaries which lie inside S^*. To do so, we can use a variant of the algorithm MAXIMUM used in the proof of Lemma 6.

Q is still the set of all vertices in U, ordered by X-values. The ordering R, however, will now involve the edges of S^* only. As before, the main loop sweeps a vertical line left-to-right passing through each vertex in Q. If v belongs to S^*, we simply maintain the ordering R with the function UPDATE defined earlier. Otherwise, we observe that the boundary in U which contains v lies inside S^* if and only if $h(v)$ and $l(v)$ are distinct from 0 and are linked. Thus, we know whether a boundary belongs to W or not as soon as we examine its leftmost vertex. To make the algorithm more efficient, we can thus delete all the vertices of the boundary from Q, after examining its first vertex. Like its look-alike, MAXIMUM, this algorithm requires $O(k \log k)$ time, where k is the total number of vertices in Q. Since each of these vertices corresponds to a distinct edge of P, the running time is $O(p' \log p')$.

```
Q = Set of vertices in U sorted by x-values.
R = W = Empty set.
for all v in Q (in ascending x-order)
  begin
  if v belongs to S*
     then UPDATE(R, v)
     else Let B be the boundary in U
          containing v.
          delete all vertices of B from Q.
          if h(v) and l(v) are not 0
            and are linked
            then "v lies inside S*"
                 W = W ∪ {B}
  end
```


We are now ready to apply the result of Lemma 6 to the set W. This will give us exactly all the inner boundaries of S, with a total running time of $O(p' \log p')$.

III) *Computing P_1 and P_2.* The last step is to compute a graph representation of P_1 and P_2. This is a trivial graph transformation, and we only sketch out the procedure. Let Adj(w) be the adjacency list of the vertex w in the graph representation of P. Also, call E the set of edges of P passing through the vertices of S. We can assume E to be readily available, since the edges in E must be determined in order to compute S. Let w be an endpoint of some edge in E. Defining P_1 as the polyhedron cut by S that contains w, we next show how to compute P_1 in $O(p)$ time.

1) *Adjacency lists of P_1.* For each edge ab of E which does not lie on T, let v be the unique vertex of S lying on ab. We can always assume that a lies on the same side of T as w, that is, is a vertex of P_1 whereas b is a vertex of P_2. If v is distinct from a, we replace b by v in the list Adj(a) and delete the list Adj(b). If $v = a$, we simply delete b from Adj(a) as well as the list Adj(b). Repeating these operations for all the edges of E which do not lie on T has the effect of disconnecting P_1 from P_2. Then, a depth-first search in the resulting graph of P, starting at w, will provide all the vertices of P_1. All the adjacency lists of the vertices common to P and P_1 have already been updated. Finally, since we have a doubly-linked-list description of the boundaries of S, we can set up the adjacency lists of the new vertices, that is, the vertices of P_1 lying on S. All these operations require $O(p)$ time.

2) *Face-to-edge lists of P_1.* Since the previous lists provide the set of vertices of P_1, we first remove all the faces of P made up entirely of vertices not in P_1. Then, since all the faces of P intersecting S have been previously determined, it is easy to compute a description of the parts of those faces which lie in P_1. Let F be such a face, with $a_1, \cdots, a_k$ being the vertices of S lying on F. Recall that $a_1, \cdots, a_k$ have been computed in sorted order (Step I). We may assume that the boundaries of F are represented by doubly-linked lists with the nodes representing the vertices. Letting u_i be the edge of F passing through a_i and b_i be the endpoint which lies on the same side of T as w, we first delete from the lists all the vertices lying strictly on the other side of T, then we enter the vertices a_i into the lists by linking both ways b_i and a_i as well as a_{2i-1} and a_{2i} (Fig. 12a). Note that we can always assume that u_i does not lie on T, which ensures that b_i is always well-defined. The result of these operations may produce several disconnected lists, since F may be broken up into several faces of P_1. Finally, if F has some edges lying on T, the algorithm may produce lists consisting of two vertices, and these degenerate cases should be removed in a postprocessing stage (Fig. 12b). Finally, the face-to-edge lists of S (which have already been computed) must join the set of face-to-edge lists of P_1. Once again, all these operations will take $O(p)$ time.

3) *Edge-to-face lists of P_1.* These lists can be obtained in $O(p)$ time by scanning through the face-to-edge lists once and recording the faces next to each of their boundary edges.

The computation of P_1 and P_2 is now complete. We conclude:

LEMMA 7. *A polyhedron P of genus 0 can be partitioned with a cut in time $O(p+p' \log p')$, using $O(p)$ storage, with p' being the number of edges in P intersecting the plane supporting the cut.*

We have seen that in the course of its action, the naive decomposition may produce polyhedra containing holes. For that reason, we wish to generalize the previous result to polyhedra of arbitrary genus. Now, instead of breaking P into two pieces, a cut may simply decrease its genus by one or have some of the effects described at the beginning of § 2.1 (e.g., removing a handle and creating another). To handle these cases, we may first *cut* each edge of P which intersects S, by updating the adjacency

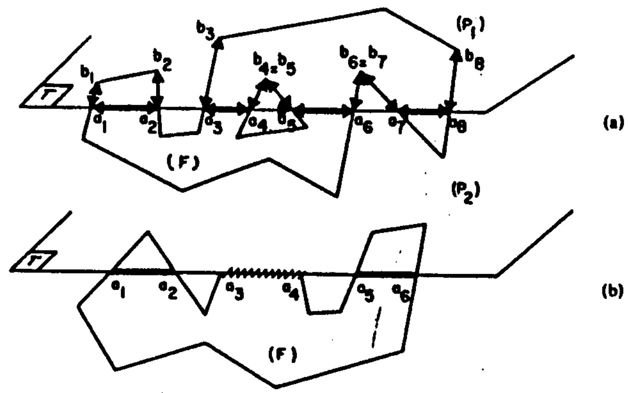


FIG. 12. *Computing the faces of* P_1.

lists accordingly. Next, we test the connectivity of the graph by doing a depth-first search with the adjacency lists. If it is no longer connected, the cut breaks P into two separate pieces P_1 and P_2 which can be computed as indicated above. Otherwise, we update the lists of the representation in a similar way; the only major difference being the introduction of two faces corresponding to the cut. We may omit the details of these operations which are very elementary.

In our analysis, we were careful to use the number of edges p and not the number of vertices as the measure of the input size. Indeed, Euler's formula, which relates the number of vertices, edges, and faces of a polyhedron has to be altered for higher genuses [6]. Consequently, the well-known inequality $p \le 3n-6$, which holds for 0-genus polyhedra, is no longer valid when it comes to polyhedra with holes, as is the case in our problem. It is, however, easy to verify that the number of edges always gives the size of the description of P, up to within a constant factor. The revised algorithm for the naive decomposition is merely a repeated application of the procedure described above. This leads to the following result.

THEOREM 8. *The naive decomposition of a polyhedron P of genus 0 can be done in $O(nN^2(N+\log n))$ time, using $O(nN^2)$ storage.*

Proof. The algorithm proceeds by removing each notch in turn. In an $O(p)$ preprocessing stage, we can assign to each notch a plane resolving its reflex angle. Then, for each notch in turn, we remove each of its subnotches with cuts lying in the plane associated with the notch. This will produce $O(N^2)$ convex parts in the end, as has been shown in Theorem 2. Each cut can be implemented with the procedure of Lemma 7 and the generalization for higher genuses which we just mentioned. Consider the partial decomposition before the notch g is removed. Let $P_1, \cdots, P_k$ be the (nonconvex) polyhedra in the current decomposition which contain a segment of g as a subnotch (we have seen that $k \le N$). Let p_i be the number of edges in P_i and p_i' the number of edges intersecting the plane supporting the cut used to remove g. From Lemma 7, we know that we can remove the subnotch of g in P_i in time $O(p_i + p_i' \log p_i')$. We next evaluate the maximum number C of edges present at any time in the decomposition. We distinguish two kinds of edges: first the edges which are pieces of edges of P. Since each edge of P can be divided into at most $N+1$ segments, the number C_1 of such edges cannot be greater than $p \times (N+1)$.

The other edges are intersections of cuts with faces (or parts of faces) of P or intersections between cuts. Since each cut lies on any of N possible planes, and all faces of P lie on q possible planes, the C_2 edges we are now considering lie on at most qN possible lines. Next we show that each of these lines supports at most $3N$ edges (we do not believe that this upper bound is tight). Let L be such a line and $u_1, \cdots, u_t$ be the edges of the decomposition that lie on L. The edges $u_1, \cdots, u_t$ form m disconnected segments $r_1, \cdots, r_m$ on L, each segment consisting of contiguous edges

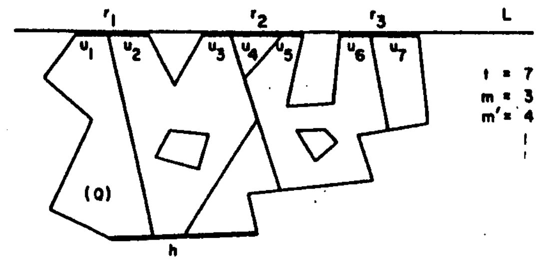


FIG. 13. *Counting the number of edges in the decomposition.*

u_i $(1 \le m \le t)$ (Fig. 13). Let m' be the number of endpoints common to two consecutive u_{ij}; we have

(1) $$m + m' = t.$$

L is the line passing through the intersection of a cut S with a face of P or the intersection of two cuts S and S'. In either case, let h be the notch passing through the cut S. The union of all the cuts used to remove h forms a polygon Q, which may possibly have holes. Moreover all the segments r_i are edges of Q and each notch of Q corresponds to a distinct notch of P. At this point, we must anticipate a little and use a result which we will prove at the end of this section (Lemma 10). This result states that the line L cannot intersect Q in more than $2N$ segments. Therefore we have

(2) $$m \le 2N.$$

Since the *interior* endpoints are all intersections of cuts with L, we also have

(3) $$m' \le N.$$

Combining (1)–(3) shows that $t \le 3N$, which proves our claim and implies that

$$C_2 \le 3qN^2.$$

Since each edge of P is adjacent to at most 2 faces of P while a face has at least 3 enclosing edges, we have

$$3q \le 2p$$

showing that

$$C_2 = O(nN^2)$$

since $p = O(n)$ (P is of genus 0). Our counting argument considered each u_i as the intersection of a cut or a face with a cut. Therefore each edge u_i will be counted exactly twice in $p_1 + \cdots + p_k$, hence

$$p_1 + \cdots + p_k \le C_1 + 2C_2.$$

Finally, since $C_i \geq p(N+1)$ and $p = O(n)$, we have

$$p_1 + \cdots + p_k = O(nN^2).$$

Also, since at most 2 edges intersecting a given plane in a single point can be collinear, the maximum number of edges which can intersect a given plane is bounded by the maximum number of lines L, therefore

$$p'_1 + \cdots + p'_k = O(nN).$$

It follows that all the subnotches of g can be removed in time $O(nN(N+\log n))$, using $O(nN^2)$ storage. Since N notches must be removed, the proof is now complete. □

It is possible to improve the running time of the algorithm to $O(nN^3)$, using the same amount of storage. The algorithm is too long and too complex to be presented here, given the relatively minor gain it represents. We, therefore, refer the reader to [1] for a detailed description of the method.

THEOREM 9. *The naive decomposition of P can be carried out in $O(nN^3)$ time and $O(nN^2)$ space.*

Proof. See [1]. □

We will now prove the claim made earlier that L intersects Q in at most $2N$ segments.

LEMMA 10. *Let N be the number of reflex angles in a nonconvex polygon Q with any number of holes in it. No line L can intersect Q in more than 2N segments.*

Proof. We will prove the lemma in two parts: first assume that all of Q lies on one side of L. Assume wlog that L is horizontal and that Q lies below L. Although all the vertices of Q that lie on L are collinear, we can assume that among the other vertices, no two lie on a common horizontal line. This is only desirable for the sake of simplicity and does not restrict the generality of the problem in any way. Let $s_1, \cdots, s_k$ be the segments of $Q \cap L$ in left-to-right order, and let $v_1, \cdots, v_l$ be a list of the vertices of Q lying strictly below L, sorted vertically in descending order. If we translate the line L downwards along a vertical axis in a continuous motion, we observe that the segments s_i undergo continuous transformations. New segments may appear in the process, some may vanish from L, while others may merge. Eventually all of them will disappear from L. The crucial observation is that since Q is connected, no s_i will disappear before merging at least once. Therefore there will be at least $k/2$ merges in the process (actually, it would be easy to show that there will be at least $k-1$ merges). Note that the merges can occur only when L reaches a vertex v_i. Let L_i be the corresponding position of L (i.e. the horizontal line passing through v_i). Since all the v_i have distinct Y-coordinates, at most one merge can occur at L_i. Suppose that a and b are two segments merging on L_i. The endpoint common to both segments, v_i, is clearly a notch of Q, therefore Q has at least as many notches as we have merges, i.e. $k/2$, provided that $k > 1$.

Assume now that L may intersect Q in an arbitrary fashion, and let $s_1, \cdots, s_k$ be the intersecting segments. Let us *cut* along each segment s_i. This operation partitions Q into at most $k+1$ polygons, each lying entirely on one side of L, as in the previous case. Note that we may have strictly fewer than $k+1$ polygons if Q has holes. Also, since Q is connected, each segment s_i is the edge of at least one polygon which has at least another edge collinear with L (assuming that $k > 1$). It follows that among these polygons we can find j of them, say, $Q_1, \cdots, Q_j$, such that each has at least two edges collinear with L and each s_i is an edge of at least one of them. Let N_i be the number of reflex angles in Q_i and let k_i be the number of edges collinear with L. Since Q_i has at least two edges adjacent to L, we can use the previous result to derive $k_i \leq 2N_i$. Since $k_1 + \cdots + k_j \geq k$, and all the reflex angles of Q involved in the j quantities $N_1, \cdots, N_j$ are distinct, we have $k \leq 2N$, which completes the proof. □

5. Conclusions. The contribution of this work has been to describe a heuristic for decomposing a polyhedron into a set of convex pieces, with the cardinality of this set lying within a constant factor of the minimum in the worst case. We have also established a quadratic lower bound on the complexity of the minimum convex decomposition problem in three dimensions. Refinements of the algorithm given in this paper might take into account the particular shapes that most *practical* polyhedra are likely to have. For example, it is often the case that two notches will be adjacent and can be removed with the same cut. This simple observation may reduce the number of convex parts by half. More generally, we believe that efficient special-purpose heuristics could be developed along these lines. An interesting case is to restrict the domain of polyhedra to *architectural* designs where, for example, all the edges lie on three possible perpendicular directions. Another restriction may further require that the convex parts be rectangular parallelepipeds. All these problems are highly practical, yet still open.

Only in two and three dimensions is the concept of nonconvex polyhedra totally natural. In higher dimensions, convex polyhedra are still easily expressed as intersections of halfspaces, but nonconvex polyhedra do not lend themselves to such easy descriptions. One method is to express a polyhedron as a connected union of convex polyhedra. Note that the convex polyhedra may overlap, thus do not necessarily constitute a convex decomposition of the polyhedron. This representation is common in linear programming, when the constraints are expressed by k set of inequalities, and at least one set has to be satisfied. If we can find a convex decomposition of the polyhedron into p parts with $p \ll k$, and if each convex part has relatively few faces, testing the feasibility of a point can be greatly simplified by testing its inclusion in any of the p convex parts. Here again, because of the complexity of the problem (recall that the standard version of the decomposition problem is already NP-hard), only efficient heuristics should be sought.

Acknowledgments. I wish to thank Dana Angluin for many helpful discussions, as well as the referees for their valuable help in improving the presentation of this paper.

REFERENCES

[1] B. CHAZELLE, *Computational geometry and convexity*, Ph.D. thesis, Yale Univ., New Haven, CT, 1980. Also available as Carnegie-Mellon Tech. Report, CMU-CS-80-150, Carnegie-Mellon Univ., Pittsburgh.

[2] B. CHAZELLE AND D. P. DOBKIN, *Detection is easier than computation*, Proc. 12th ACM SIGACT Symposium, Los Angeles, May 1980, pp. 146–153.

[3] H. COXETER, *Regular Polytopes*, 3rd Ed., Dover, New York, 1973.

[4] M. GAREY, D. JOHNSON, F. PREPARATA AND R. TARJAN, *Triangulating a simple polygon*, Inform. Proc. Lett., 7 (1978), pp. 175–180.

[5] A. LINGAS, *The power of non-rectilinear holes*, Proc. 9th Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science 140, Springer-Verlag, New York, 1982, pp. 369–383.

[6] W. MASSEY, *Algebraic Topology: An Introduction*, Springer-Verlag, New York, 1967.

[7] J. MUNKRES, *Topology: A First Course*, Prentice-Hall, Englewood Cliffs, NJ 1975.

[8] J. O'ROURKE AND K. J. SUPOWIT, *Some NP-hard polygon decomposition problems*, IEEE Trans. Inform. Theory, IT-29 (1983), pp. 181–190.

[9] F. PREPARATA, *A new approach to planar point location*, this Journal, 10 (1981), pp. 473–482.

[10] M. SHAMOS AND D. HOEY, *Geometric intersection problems*, 17th Annual IEEE Conference on Foundations of Computer Science, Houston, TX, Oct. 1976, pp. 208–215.

[11] G. B. THOMAS, JR., *Calculus and Analytic Geometry*, Addison-Wesley, Reading, MA, 1962.