Discrete Comput Geom 5:505-526 (1990)

Discrete & Computational
Geometry

Triangulating a Nonconvex Polytope*

Bernard Chazelle and Leonidas Palios

Department of Computer Science, Princeton University,
Princeton, NJ 08544, USA

Abstract. This paper is concerned with the problem of partitioning a three-dimensional nonconvex polytope into a small number of elementary convex parts. The need for such decompositions arises in tool design, computer-aided manufacturing, finite-element methods, and robotics. Our main result is an algorithm for decomposing a nonconvex polytope of zero genus with n vertices and r reflex edges into $O(n + r^2)$ tetrahedra. This bound is asymptotically tight in the worst case. The algorithm requires $O(n + r^2)$ space and runs in $O((n + r^2) \log r)$ time.

1. Introduction

This work is concerned with the problem of partitioning a polytope in $\mathfrak{R}^3$ into a small number of elementary convex parts. The general problem of decomposing an object into simpler components has been the focus of much attention in recent years. In two dimensions, computer graphics and pattern recognition have been the main source of motivation for this work. Beginning with the papers of Feng and Pavlidis [15] and Schachter [26], the problem of rewriting a simple polygon as a collection of simple parts has been exhaustively researched; see O'Rourke's book [22] and the survey article by Chazelle [8]. In higher dimensions, however, results have been few and far between. It is known from [7] that a polytope of n vertices can always be partitioned into $O(n^2)$ convex pieces, provided that Steiner points are allowed in the decomposition, and that this bound is tight in the worst case. Ruppert and Seidel [25] proved recently that if no Steiner points are allowed, the problem of deciding whether a given polytope is decomposable into tetrahedra (whose vertices have to be vertices of the polytope) is NP-complete, even if

* This research was supported in part by the National Science Foundation under Grant CCR-8700917.

restricted to the class of star-shaped polytopes. The algorithm to carry out the partitioning described in [7] is based on the exact RAM model of computation. The issue of robustness while dealing with finite precision arithmetic is considered in [2], where a numerically robust algorithm to compute a convex decomposition of nonconvex polytopes of arbitrary genus is presented.

On a related problem, Aronov and Sharir [1] have shown that the cells of an arrangement of n triangles in 3-space can be partitioned into a total of $O(n^2\alpha(n) + h)$ tetrahedra, where h is the number of faces in the arrangement, and $\alpha(n)$ is the inverse Ackermann function. For fixed arbitrary dimension d, Edelsbrunner *et al.* [14] have given an optimal algorithm for computing the partition of d-space induced by a collection of hyperplanes. The stratification and triangulation of real-algebraic varieties and related issues are discussed in [6], [9], [11], [24], [27] and [29].

The specific problem of partitioning a three-dimensional polytope into simple parts arises in mesh generation for finite-element methods, computer-aided design and manufacturing, and automated assembly systems and robotics [3], [16], [28]. The problem comes under various guises, depending on the desired shape of partitioning elements: convex, simplicial, star-shaped, monotone, rectangular, isothetic, etc. In general, the quest for minimal partitions seems destined to be frustrated. For example, finding minimum convex decompositions is NP-hard [19]. In practice, however, good approximation algorithms may be just as attractive, especially if they are fast and robust and the decompositions produced are free of pathological features. Indeed, a minimum partition can sometimes be so contrived that a finer, yet more regular, decomposition is preferable.

How difficult is it to triangulate a polytope (that is, subdivide it into a collection of tetrahedra)? In practice, a "good" triangulation algorithm should not only guarantee $O(n^2)$ pieces in the worst case, but it should also make the size of the triangulation dependent on both n, the size of the polytope, and r, the number of reflex edges. The polytopes arising in standard application areas tend to be almost convex, and this fact should be used to one's advantage. For example, a triangulation of quadratic size would be disastrous if, say, the polytope is convex. When both n and r are taken into account, the lower bound on the triangulation size becomes $\Omega(n + r^2)$ (as is easily derived from [7]). By this criterion, the algorithm described in this paper is optimal: A polytope of n vertices and r reflex edges is triangulated into $O(n + r^2)$ pieces. The running time is $O((n + r^2) \log r)$. We believe that the algorithm is practical. An implementation is under way, and the plan is to test it on actual problems arising in the use of finite-element methods in aerospace engineering.

The triangulation algorithm consists of two parts. In the *pull-off phase* the size of the polytope is reduced to $O(r)$. To achieve this we identify vertices of small degree that are not *hindered* by other vertices and remove them one by one, much like we would pull a ski hat off someone's head. Next, we enter the *fence-off phase*, which involves erecting vertical fences from each edge of the polytope. In Section 2 we set our notation and move a number of technicalities out of the way. Section 3 describes the triangulation algorithm proper.

2. Cups, Crowns, Domes, and Other Widgets

We begin by recalling some standard terminology and introducing some of our own. In $\Re^3$ a polytope is a piecewise-linear 3-manifold with boundary, which is homeomorphic to a closed k-holed torus for arbitrary k. Its boundary consists of a collection of relatively open sets, the *faces* of the polytope, which are called *vertices*, *edges*, or *facets*, if their affine closures have dimension 0, 1, or 2, respectively. A polytope is *simple* if no two faces share a point. Note that, for each point p on the boundary of a simple polytope, there exists a ball B_p centered at p, which intersects only the facets that contain p in their closure. We define a simple polytope Q to be *nondegenerate* if the boundary of Q divides the ball B_p (defined as above) centered at a point p on Q's boundary in exactly two nonempty regions, one in Q's interior and the other in Q's complement (Fig. 1). The class of polytopes with which we deal in this paper is the class of nondegenerate simple polytopes of genus 0; thus self-intersecting, dangling, or abutting faces, as well as handles, are ruled out.

Let P be a nondegenerate simple polytope. An edge e of P is said to be *reflex* if the (interior) dihedral angle formed by its two incident facets exceeds π. By extension, we say that a vertex is *reflex* if it is incident upon at least one reflex edge, and that it is *flat* if all its incident facets lie in at most two distinct planes. Finally, a vertex is *pointed* if it is neither flat nor reflex (Fig. 2). It is easy to see that a pointed vertex cannot be incident upon two collinear edges, although it can be incident upon two coplanar (adjacent) facets of P. As usual, the number of edges incident upon a vertex is referred to as its *degree*. Next, we define the *cone* of a pointed vertex v as the unbounded convex polyhedron spanned by the edges incident upon v. More precisely, *cone*(v) is the locus of points $v + \sum_{1 \le i \le k} \alpha_i(w_i - v)$, where $w_1, \ldots, w_k$ are the vertices of P adjacent to v and the α_i's are arbitrary nonnegative reals. We are now ready to introduce the key notion of a *cup*. The cone of a pointed vertex v contains a number of vertices of P distinct from v. Some lie on the boundary of the cone; others may lie strictly inside. Let H and H^- be the convex hulls of all the vertices of P lying in *cone*(v) and *cone*$(v)\backslash\{v\}$, respectively. We define *cup*(v) as the simple polytope formed by the closure of $H\backslash H^-$.

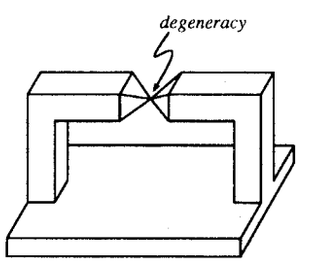

Fig. 1. A degenerate polytope.

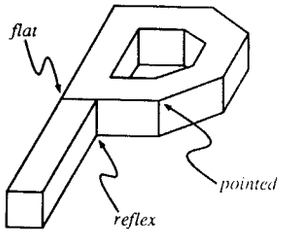

Fig. 2. The different types of vertices.

A number of simple properties follow readily from the definition. A cup is the closure of the difference between the convex hull of a finite point-set $A \sqcup \{v\}$ and the convex hull of A. Since v does not belong to A, its cup is a simple star-shaped polytope whose kernel contains v (Fig. 3). Its boundary contains a number of polygons incident upon v which are glued to the convex hull of A. The gluing border is a closed simple polygonal curve, called the *crown* of v, and can be centrally projected onto a plane so as to appear as the boundary of a convex polygon. The crown acts as a Jordan curve on the boundary of the cup, which it separates into one piece on the boundary of the cone and a concave (with respect to the cup) polyhedral patch, which we call the *dome* of v. (If a facet of the dome contains a vertex or an edge of P then it is refined to include these additional features.) Edges and vertices of the dome that are not in the crown are called *internal*. Obviously, the internal edges of the dome are the only edges of the cup which are reflex (with respect to the cup). To conclude this string of definitions, we refer to the pointed vertex v as the *apex* of the cup of v.

We now investigate the relationship between P and the cup of v. All cup vertices are vertices of P though, obviously, the same cannot be said of cup edges. A more interesting observation is that the cup lies inside P. This follows from the fact, to be proven below, that the facets of the cup that are not in the dome lie on the boundary ∂P of P. Thus, it is impossible for a facet or an edge of P to intersect the

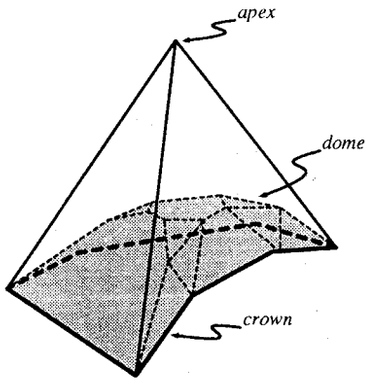

Fig. 3. A cup.

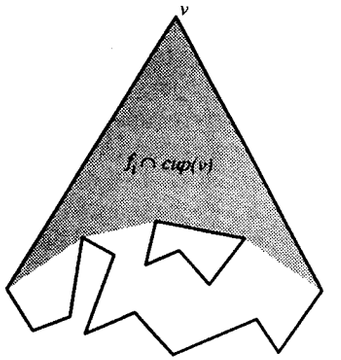

Fig. 4. A facet of a cup.

interior of the cup, unless a vertex of P does. But that, of course, is ruled out by the very definition of a cup. This establishes our claim. Let us now prove the premise of this reasoning, which is that a facet of the cup that is not in the dome lies in ∂P. It suffices to show that the crown lies entirely in ∂P. Let $g_1, \ldots, g_\ell$ be the facets of P incident upon v and let us (mentally) merge any pair of coplanar facets. This produces superfacets $f_1, \ldots, f_k$ (given in either circular order around v), such that the dihedral angle between two adjacent f_i's is strictly less than π. Now, for each $i = 1, \ldots, k$, let H_i (resp. H_i^-) be the two-dimensional convex hull of the vertices of P lying in f_i (resp. $f_i \backslash \{v\}$). The closure of $H_i \backslash H_i^-$ is the polygon formed by intersecting the cup of v with the plane supporting f_i: it is the two-dimensional equivalent of a cup (Fig. 4). Its boundary consists of a two-edge convex chain followed by a concave chain (possibly reduced to a single edge). By a convex (resp. concave) chain we mean a piece of a polygon's boundary which always turns strictly right (resp. left) when traversed clockwise. The construction works as desired because v is pointed, and therefore exhibits an angle less than π in f_i. The crown of v is the closed curve obtained by concatenating the concave chains in sequence. This proves our claim that the crown lies in ∂P. In addition, no vertices but the endpoints of such concave chains can be pointed vertices of P. Therefore, since all edges of the cup adjacent to the apex are also edges of P, a pointed vertex can be on the crown of another pointed vertex only if they are adjacent in P. Note, however, that the converse is not always true. In particular, if two pointed vertices are connected by an edge which is incident upon two coplanar facets, none of them lies on the crown of the other.

Let us summarize the various types of faces which a cup may have. The following statements are to be understood with respect to the cup and *not* with respect to P. The edges incident to the apex as well as the edges of the crown are nonreflex. Actually, none of them can be incident to two coplanar facets. The reason is that the convex hull operation merges coplanar facets. As a result, although each facet of the cup incident upon v lies in ∂P, it does not necessarily lie within any given facet of P. Returning to our classification, we should note that the internal edges of the dome are all reflex.

We close this section with a few technical lemmas which hold the key to understanding the whys and wherefores of the pull-off phase. Our goal in the pull-off phase is to bring the size of the polytope down to proportional to the number of its reflex edges. The idea is to identify pointed vertices of the polytope, and to remove them by pulling their cups off. The selection of the pointed vertices to be removed, however, must be done carefully, if we do not want to increase the genus of the polytope or compromise its nondegeneracy. We say that a dome is *hindered* if it contains (i) an internal vertex, or (ii) an internal edge that is also an edge of P. The removal of a pointed vertex whose dome is not hindered ensures that neither the genus of the polytope will change, nor any degeneracies will be introduced.

In the following, we assume that P is a nondegenerate simple polytope of zero genus with n vertices and m edges, exactly r of which are reflex. We also assume that P does not have any flat vertices.

Lemma 2.1. *Let v and v' be two distinct nonadjacent pointed vertices of P. No point can be an internal vertex of the domes of both v and v'. Similarly, no line segment can be an internal edge of both domes.*

Proof. Let z be a vertex internal to the domes of v and v'. Let us first assume that the intersection Q of the interiors of the cups of v and v' is nonempty. The closure of Q must have at least one vertex outside the dome of v, otherwise it would have empty interior. The only such vertex can be the apex v, however, since the interior of a cup is free of vertices. Thus, v lies in the cup of v'. Since it can neither coincide with v' nor be an internal vertex of the dome of v', the vertex v must lie on the crown of v'. But this is not possible, since v and v' are assumed nonadjacent.

So, we can now assume that the intersection of the interiors of the two cups is empty. Since z is internal to the dome of v, there exists a small open half-ball centered at z that lies entirely within the cup of v. A similar statement holds for v' as well, and since the two half-balls are nonintersecting, the domes of both v and v', locally around z, have to lie on the plane separating the two half-balls, which contradicts the simplicity of P.

This proves the first part of the lemma. The second part is a trivial corollary: simply introduce an artifical vertex at the midpoint of the internal edge. □

Lemma 2.2. *Any reflex vertex of P which is internal to a dome has at least three reflex edges of P incident upon it.*

Proof. Let w be an internal vertex of the dome of some pointed vertex. There exists a small open half-ball centered at w that lies entirely inside P. Therefore, w is a vertex of the convex hull of the point-set consisting of w and all its adjacent vertices in P. Moreover, since P is simple, w will be incident upon at least three edges of the convex hull. The lemma follows from the observation that the convex hull edges incident upon w are also edges of the polytope, and are in fact reflex. □

Lemma 2.3. *A reflex vertex of P can contribute internal vertices to at most three distinct domes.*

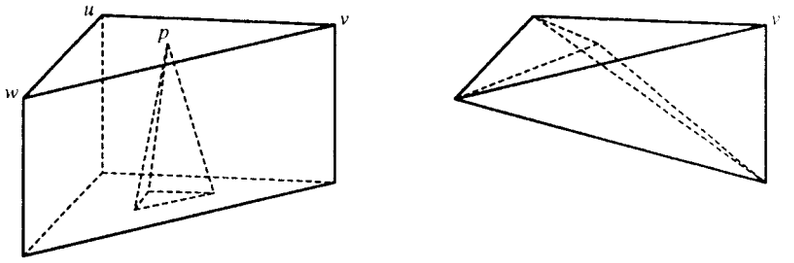

Fig. 5. The reflex vertex p hinders the cups of u, v, and w (on the right, the cup of v).

Proof. Figure 5 shows that this bound is tight. Now, suppose, for contradiction, that a reflex vertex p is internal to the domes of four pointed vertices u, v, w and z of P. It follows from Lemma 2.1 that all four apexes must be adjacent to each other, thus forming a tetrahedron T whose six edges all lie in ∂P. This tetrahedron cannot have empty interior, otherwise one of the apexes would be either reflex or flat. Note also that the cup of a pointed vertex s contains any vertex t adjacent to s such that the (interior) dihedral angle around st is strictly less than π. Therefore, the crown of each of u, v, w, z contains the other three apexes as vertices. Furthermore, since p is an internal vertex of all four domes, and the edges of T are nonreflex, p must lie in T. However, it cannot lie on any of T's edges, otherwise there would be two nonadjacent apexes.

Since p is internal to the domes of all four apexes, there exists a small ball centered at p that lies entirely in each of their cones. As P is nondegenerate, this ball intersects P's complement, and it contains consequently a point q outside P that avoids each of the six planes defined by p and any two of the four apexes. Then p must lie in the interior of one of the four tetrahedra defined by q and any three of the apexes. All four vertices of that tetrahedron, however, lie in the cone of the fourth apex. Thus, p cannot be a vertex of that apex's dome, which gives us a contradiction. □

Lemma 2.4. *Given an edge pq of P, there are at most two pointed vertices u and v, such that pq is internal to the domes of both u and v, and p and q lie on the crowns of both domes.*

Proof. Suppose that pq is internal to the domes of u, v, and w, and that p and q lie on all three crowns. Then all six line segments pu, qu, pv, qv, pw, qw lie in ∂P, and all three triangles puq, pvq, pwq lie entirely in P. Furthermore, it follows from Lemma 2.1 that u, v, and w are adjacent to each other, and, therefore, each of them lies on the crowns of the other two, making it impossible for the boundary of each of the triangles puq, pvq, pwq to contain more than one of u, v, and w. We can thus establish an ordering of these triangles clockwise around pq; let it be puq-pvq-pwq.

Let us denote by $[x, l, y]$ the dihedral angle that is bounded by the two half-planes passing through the line l and containing the points x and y, respectively, and is swept when the former half-plane rotates clockwise around l until it

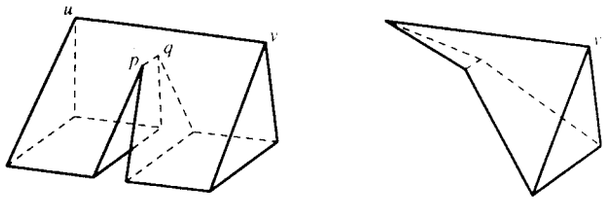

Fig. 6. Internal edge pq has its endpoints on the crowns of u and v (on the right, the cup of v).

coincides with the latter one. From the definition of the dome, and the fact that the edge pq is internal to the dome of u, we conclude that there is a plane S passing through pq such that u lies in one of the open half-spaces defined by S while v and w lie either in the other one or on S. Since the same argument applies to v and w as well, the dihedral angles $[u, pq, v]$, $[v, pq, w]$, and $[w, pq, u]$ cannot be larger than π. None of them can be equal to π either, however, because of the adjacency of the apexes.

Therefore, all three dihedral angles $[u, pq, v]$, $[v, pq, w]$, $[w, pq, u]$ are less than π. As a result, the union of the three tetrahedra defined by pq and any two of the apexes completely contains a small open ball centered at the midpoint of pq. Since the edge pq is a reflex edge of P, there is a point z outside P that lies in that ball and avoids all three supporting planes of puq, pvq, and pwq. Then the interior of one of the triangles uvz, vwz, wuz intersects pq. If this were uvz, then pq cannot possibly be internal to the dome of w, as the relative interior of uvz lies outside the cup of w.

Note that the statement of the lemma is tight, as shown in Figure 6. □

Summarizing the results of the previous lemmas, we derive:

Lemma 2.5. *The polytope P contains at most $2r$ pointed vertices whose domes are hindered.*

Proof. We partition the reflex edges of P into three classes:

(1) those with at least one endpoint being an internal vertex of some dome,
(2) internal edges of a dome with both endpoints on the crown, and
(3) all remaining reflex edges.

Let us prove by contradiction that classes 1 and 2 are disjoint. Assume that there exists a reflex edge e of P, internal to the dome of a pointed vertex u, such that one of its endpoints, say p, lies on the crown of u and is internal to the dome of v. Then there exists a small open half-ball centered at p that lies entirely in the cup of v, and hence in P. Since e is internal to the dome of u, it is not collinear with pu. With pu nonreflex, it follows that the unique plane defined by e and pu intersects the half-ball in a half-disk centered at p. The internal angle between e and pu is strictly less than π however, therefore the half-disk cannot lie entirely in P: a contradiction. We conclude that no vertex can be internal to the dome of a pointed vertex and lie on the crown of another one.

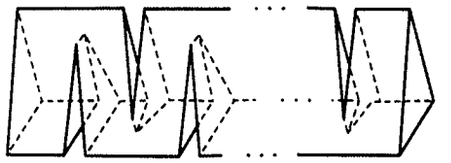

Fig. 7.

Let r_i $(i = 1, 2, 3)$ be the cardinality of the ith class above. According to Lemma 2.3, an edge in class 1 may hinder at most three domes by contributing internal vertices through one given endpoint q (so the total might be as high as six). The endpoint q, however, will be incident upon at least two additional reflex edges of P in class 1 (Lemma 2.2). Therefore, the r_1 edges in class 1 can hinder at most $(2 \times 3)/3\, r_1$ domes. Additionally, from Lemma 2.4, each edge in class 2 may hinder at most two domes. The lemma follows readily. The stated bound is in fact achievable, as shown in Fig. 7. □

Lemma 2.5 implies that a polytope of n vertices and r reflex edges has at least $n - 4r$ pointed vertices whose domes are not hindered. The cups of these vertices can be removed one by one, and be decomposed into tetrahedra, resulting in a polytope of at most $4r$ vertices and a collection of $\sum_i (d_i - 2)$ tetrahedra, where d_i is the degree of the apex of the ith cup removed. The removal, however, must be done in a systematic way if we want to guarantee that a linear number of tetrahedra are produced. Let us consider the convex polytope of $2k$ vertices shown in Fig. 8. Note that if the cups of the vertices $u_k, u_{k-1}, \ldots, u_2$ are removed in that order, the total number of tetrahedra extracted is $(k-1)^2$. If, however, the cups of the same vertices are removed in the reverse order, we end up with only $2(k-2) + k - 1$ tetrahedra. This observation and the fact that the degree of a pointed vertex v of P is an upper bound on the number of the vertices of *crown*(v) if the boundary of P is triangulated suggest that we should try to remove the cups of the pointed vertices of small degree first, or even those of degree bounded by some constant d. In an

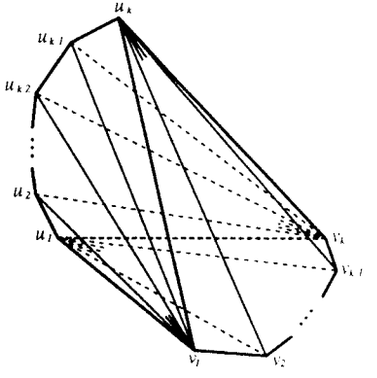

Fig. 8.

implementation of the algorithm this constant may have been set by the programmer ahead of time, or be provided by the user. The larger d is, the more vertices will be removed, and, therefore, the more time the pull-off phase will take and the skinnier the polytope (resulting after the removal of these vertices) will be. Clearly, by restricting our attention to pointed vertices of degree bounded by a constant, we can guarantee that the removal and triangulation of their cups will result in a linear number of tetrahedra. On the other hand, we must also ensure that the removal will in fact bring the size of the polytope down to linear to the number of its reflex edges. This is established by means of the following lemma.

Lemma 2.6. *Let $d \geq 6$ be a fixed integer, and let $c < 1$ be a prespecified positive real. If $r \leq c(d-5)n/(4d-4)$, then P contains at least $(1-c)(d-5)n/(d-2)$ pointed vertices of degree at most equal to d whose domes are unhindered.*

Proof. Since a pointed vertex is incident upon nonreflex edges only, the sum of the degrees d_i of all pointed vertices is

$$\sum_i d_i \leq 2(m-r), \tag{1}$$

where m is the total number of edges of P. Let us denote by N and n' the number of pointed vertices of degree at most d, and the number of reflex vertices, respectively. We can then write

$$\sum_i d_i = \sum_{3 \leq d_i \leq d} d_i + \sum_{d_i > d} d_i \geq 3N + (n - n' - N)(d+1). \tag{2}$$

The combination of (1) and (2) yields $3N + (n-n'-N)(d+1) \leq 2(m-r)$ or

$$N \geq \frac{(d+1)(n-n') - 2(m-r)}{d-2}. \tag{3}$$

Since each reflex vertex is incident upon at least one reflex edge, we have $n' \leq 2r$, while the formula relating the number of edges, and the number of vertices of a polytope of genus 0 provides us with the bound $m \leq 3n-6$. Substituting these values in (3), we derive

$$N \geq \frac{(d+1)(n-2r) - 2(3n-6-r)}{d-2} = \frac{(d-5)n - 2dr + 12}{d-2}.$$

Among these vertices, at most $2r$ can have their domes hindered (Lemma 2.5). So, in order to guarantee the presence of pointed vertices with unhindered domes, it suffices that

$$\frac{d-5}{d-2}n - \frac{2d}{d-2}r + \frac{12}{d-2} > 2r.$$

Table 1. Values of $c(d-5)/(4d-4)$ and $(1-c)(d-5)/(d-2)$.

	$d=6$	$d=7$	$d=8$	$d=9$	$d=10$	$d=11$	$d=17$	$d=25$
$c=\frac{1}{2}$	1/40	1/24	3/56	1/16	5/72	3/40	3/32	5/48
	1/8	1/5	1/4	2/7	5/16	1/3	2/5	10/23
$c=\frac{2}{3}$	1/30	1/18	1/14	1/12	5/54	1/10	1/8	5/36
	1/12	2/15	1/6	4/21	5/24	2/9	4/15	20/69
$c=\frac{3}{4}$	3/80	1/16	9/112	3/32	5/48	9/80	9/64	5/32
	1/16	1/10	1/8	1/7	5/32	1/6	1/5	5/23
$c=\frac{4}{5}$	1/25	1/15	3/35	1/10	1/9	3/25	3/20	1/6
	1/20	2/25	1/10	4/35	1/8	2/15	4/25	4/23
$c=\frac{5}{6}$	1/24	5/72	5/56	5/48	25/216	1/8	5/32	25/144
	1/24	1/15	1/12	2/21	5/48	1/9	2/15	10/69
$c=\frac{6}{7}$	3/70	1/14	9/98	3/28	5/42	9/70	9/56	5/28
	1/28	2/35	1/14	4/49	5/56	2/21	4/35	20/161

The number of such vertices will then be at least

$$\frac{d-5}{d-2}n - \frac{2d}{d-2}r + \frac{12}{d-2} - 2r = \frac{(d-5)n - (4d-4) + 12}{d-2} \geq \frac{(1-c)(d-5)n + 12}{d-2}.$$

□

What Lemma 2.6 states is that, provided that the number of reflex edges of P is "relatively" small, the number of pointed vertices of small degree whose domes are unhindered is at least a constant fraction of the total number of vertices of P. Consequently, by removing the cups of these vertices, and reiterating on the resulting polytope, we will eventually end up with a polytope whose size will be at least equal to a multiple of r, as desired.

In Table 1, we present the values of $c(d-5)/4d-4)$ and $(1-c)(d-5)/(d-2)$ for a small sample of values of c and d. To clarify the meaning of these numbers, let us consider a concrete example. Let us select, for instance, c equal to $\frac{3}{4}$. The table indicates that if we pick d to be equal to 6, i.e., we intend to remove the cups of pointed vertices of degree at most 6, then if the ratio of reflex edges over vertices of the polytope is at most 3/80, at least one out of 16 vertices will qualify. If instead, d is chosen equal to 11, then for a ratio of reflex edges over vertices no less than 9/80, the ratio of "qualified" vertices increases to 1 out of 6. The figures in the table persuade us that our algorithm can indeed be useful in practice.

3. The Triangulation Algorithm

Given a nondegenerate simple polytope P with n vertices and r reflex edges, we show how to partition P into $O(n+r^2)$ tetrahedra. The algorithm requires $O(n \log r + r^2 \log r)$ time and $O(n+r^2)$ space. Up to within a constant factor, the number

of tetrahedra produced is optimal in the worst case. This follows from a lower bound of $\Omega(m^2)$ on the number of convex parts needed to partition a certain polytope of m vertices, which is a member of an infinite family $\{P_m\}$ [7]. Indeed, we simply add dummy nonreflex edges to P_r until we have a polytope of n vertices with r reflex edges. Although not all realizable pairs (n, r) might be obtained in this way, enough of them are to justify our claim that $\Theta(n + r^2)$ is a tight worst-case bound on the number of tetrahedra needed to triangulate a polytope with n vertices and r reflex edges.

As we alluded to earlier, the triangulation algorithm consists of two phases, figuratively termed *pull-off* and *fence-off*. In the pull-off phase we reduce the size of the polytope to $O(r)$. The idea is to locate pointed vetices of small degree whose domes are not hindered, and to remove them by pulling their cups off, and by replacing the boundary of P incident upon them by their domes. This shelling step reduces the vertex count without increasing the number of reflex edges. Then, in the fence-off phase, we erect vertical fences through each edge of P, achieving a decomposition of P into cylindrical pieces that can be easily partitioned into tetrahedra.

We assume that all the incidences in the polytope P are explicitly listed. For this purpose we can use any of the standard polyhedral representations given in the literature, e.g. winged-edge [5], doubly-connected-edge-list [21], and quad-edge [17]. We also assume that P is given to us in *normal* form, meaning that it is free of flat vertices, and that its boundary is triangulated. A simple polytope can be normalized in $O(n + r \log r)$ time. Its boundary can be triangulated using, say, Mehlhorn and Hertel's triangulation algorithm [20]. To do so, we triangulate each facet by sweeping a line across its supporting plane, stopping only at vertices exhibiting reflex angles. Since these vertices are incident upon reflex edges, there will be at most $O(r)$ sweep-line stops, each incurring a search and update cost of $O(\log r)$ time. The flat vertices can then be identified and removed in time linear to the size of the polytope. Note that the normalization may increase the number of nonreflex edges, but does not affect n or r.

3.1. *The Pull-Off Phase*

This is a form of preprocessing aimed at bringing down the number of vertices of a given polytope P to the same order as the number of its reflex edges. Since our strategy will be to remove the cups of a number of pointed vertices whose domes are not hindered, we need to be able firstly to decide whether the dome of a pointed vertex is hindered or not, and secondly to compute its cup. Let us consider a pointed vertex v of P. The boundary of the convex hull of v's crown consists of two polyhedral patches separated by the crown itself, one of which isolates the other from v. Let K_v be the polytope that contains v and is bounded by the patch in question and the cone of v. The dome of v will be unhindered if and only if every reflex vertex and every reflex edge of P lies either outside K_v or on the boundary of

K_v that is incident upon v. Furthermore, if the dome of v is unhindered, v's cup is precisely K_v.

So, in order to decide whether the dome of a pointed vertex v is hindered or not, we have to check K_v against both the reflex vertices and the reflex edges of P. We can unify the two cases using the technique applied in Lemma 2.1, i.e., replacing each reflex edge by a dummy reflex vertex at its midpoint. The reflex vertices and the midpoints of the reflex edges are collectively called *puncturing* vertices, and their number is no more than $3r$. Note that these vertices alone can determine the pointed vertices whose domes are hindered, since every hindered dome contains at least one puncturing vertex that does not lie on the crown. At the outset of the pull-off phase, we assume that every puncturing vertex is potentially hindering a dome. Local tests on the incidence structures of the vertices might allow us to weed out many candidates. Although this might be a practical step to take, it is not necessary strictly speaking. In the light of the definition of the puncturing vertices, the test for unhindered domes is stated as follows: first K_v is computed; then, if no puncturing vertex lies inside K_v or on K_v's boundary that is not incident upon v, the dome of v is not hindered; in this case, the cup of v is precisely K_v.

To achieve the desired reduction in the size of P, we will have to remove $\Omega(n)$ cups, whose domes must have been verified to be unhindered. Checking the corresponding K_i's against each of the puncturing vertices, however, amounts to an undesirable $\Omega(nr)$ time complexity for this phase. To improve on this, we assume for the moment that an oracle, the *Witness Oracle*, provides us with a set W of facets of P, after ∂P has been triangulated. The members of W are referred to as *witness* facets, and have the following properties:

(i) the set of vertices incident upon all witness facets is a superset of the pointed vertices that have hindered domes,
(ii) each witness facet f is associated with a subset of puncturing vertices that potentially hinder the cups of f's vertices, and
(iii) each puncturing vertex is associated with no more than four witness facets.

The implementation of the Witness Oracle is described in the next subsection.

The pull-off phase is an iterative process. In the general step, we are working on a polytope $P_i (i \geq 1)$, where P_1 is the given polytope P. Unless the number of vertices of P_i is not "much" larger than the number of its reflex edges (see Lemma 2.6), we remove from P_i a number of pointed vertices whose domes are not hindered, producing P_{i+1}, the next polytope to work on. The removal is carried out as follows: first, we go through all the pointed vertices of P_i, looking for those of degree no more than some fixed constant d, and we insert them in a queue of *favorable* vertices; once all the vertices have been processed, we repeat the following steps until the queue is empty.

1. Let v be the favorable vertex referenced by the top of the queue. If v is marked to be skipped, we remove v from the queue, and we reiterate. Otherwise, we go through the facets of P_i that are incident upon v, and determine v's crown, without, however, modifying the boundary of P_i. Then we compute the

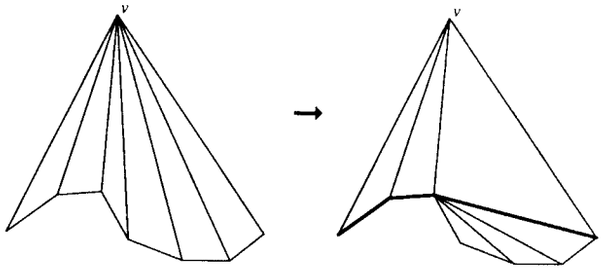

Fig. 9. The effect of the computation of v's crown on six coplanar facets incident upon v.

convex hull of the crown, and from that, the polytope K_v, as described in the opening paragraph of the pull-off phase.

2. If v is incident upon a witness facet f, we check whether the puncturing vertices associated with f lie inside K_v or on the part of K_v's boundary that is not incident upon v, thus determining whether the dome of v is hindered or not. Since the cup of v is of constant size, this can be done in time proportional to the number of puncturing vertices probed. If the dome of v is found hindered, we remove v from the queue, and proceed with step 1.
3. We are now ready to extract and triangulate the cup of v, which is in fact K_v; its size is constant, as the degree of v is no more than the fixed constant d. First, the boundary of P_i is retriangulated, so that the resulting polyhedral patch formed by the facets of P_i incident upon v is precisely $(\partial cup(v))\backslash dome(v)$ (Fig. 9). (Note that the retriangulation involves only the facets of P_i that are incident upon v.) As a result, some facets of P_i may be substituted by new ones. In case that such a facet f is a witness facet, the information in W must be updated, and the puncturing vertices associated with f must be appropriately distributed among the new facets that intersect f. After that, the cup of v can be pulled off by removing the facets of P_i that are incident upon v, and substituting them with facets of the dome of v. The dome of v is easily retrieved from K_v. Again, if any of the removed facets, say f, belongs to W, the new witness facets that result from f's removal need to be determined (these are the facets of the dome of v that lie vertically above or below f), and lists of puncturing vertices associated with them need to be set up. Finally, the cup of v, or equivalently K_v, is triangulated. The decomposition consists of the tetrahedra determined by v and each nonincident facet of K_v's triangulated boundary.
4. We remove v from the queue, and mark the vertices adjacent to v before the cup removal, so that they are skipped if met later in the queue.

Property (iii) of the witness facets and step 4 ensure that each puncturing vertex will not be probed more than a constant number of times during a single pass through all favorable vertices. As a result, such a pass over a polytope of n_i vertices will take a total of $O(dn_i + r)$ time. From Lemma 2.6 and the fact that at most $d + 1$

favorable vertices are removed from the queue or marked each time step 4 is executed, we derive that this process removes at least

$$\frac{1}{d+1}\frac{(1-c)(d-5)}{d-2}n_i \geq \alpha n_i$$

vertices from the polytope, where α is a fixed positive constant less than 1. Thus, we are left with a polytope of $n_{i+1} \leq (1-\alpha)n_i$ vertices and at most r reflex edges. Repeating this pruning pass until $n_i = O(r)$ takes time proportional to $\sum_i (n_i + r) = O(n + r \log n)$. Note that as the cups of the vertices are being pulled off, some reflex vertices may become pointed. Therefore, the set of puncturing vertices, and consequently the set of witness facets, may need updating. We choose, however, to make no updates at all, possibly ending up with some unnecessary work being done. In any case, the stated time complexity is not affected.

Summarizing, the total time required by the pull-off phase, the witness oracle included, is $O((n + r^2) \log r)$, taking into consideration that the time complexity of the oracle is $O(n \log r + r^2 \log r)$, and that $r \log n = O((n + r^2) \log r)$.[1] The total space needed amounts to $O(n + r^2)$. Finally, since each pulled-off cup produces at most $d - 2$ new tetrahedra, by the end of this phase, P is decomposed into a collection of $O(n)$ tetrahedra and a polytope of $O(r)$ vertices.

3.2. *The Witness Oracle*

For a given polytope P of n vertices and r reflex edges whose boundary is triangulated, the oracle comes up with a set of $O(r)$ facets of P, whose vertices form a superset of the pointed vertices that have hindered domes. The following lemma, which takes advantage of the geometry of the cup, provides us with the idea to carry out this computation.

Lemma 3.1. *Let v be a pointed vertex of a polytope P and let p be any point of $dome(v)\backslash crown(v)$. Then, for every line passing through p, at least one of the two rays from p must intersect the closure of a facet (of P's triangulated boundary) incident upon v. Moreover, the line segment delimited by p and the point of intersection lies entirely in P.*

Proof. By definition, the dome of v is a concave polyhedral patch with respect to v's cup. Therefore, either at least one of the two rays from p intersects the interior of the cup, or the line lies on the dome (in this case, the dome consists of coplanar facets) and thus intersects the crown. In either case, the line intersects the closure of one of the cup facets incident upon v, and the line segment with endpoints the point p and the point of intersection with the facet lies in the cup of v, and consequently in

[1] The function $h(x) = x/\log x$ is strictly increasing for $x > e \approx 2.718$. Thus, $h(r) < h(n + r^2)$, which directly implies that $r \log n < (n + r^2) \log r$.

P. For a given triangulation τ of ∂P, let $\pi(z, \tau)$ denote the connected polyhedral patch consisting of the facets of P incident upon z. It is easy to see that the facets of the cup of v that are incident upon the apex constitute the intersection of all the patches $\pi(v, \tau_i)$ over all possible boundary triangulations τ_i. Thus, we conclude that the line intersects the closure of a facet of the polytope incident upon v. □

The lemma implies that a set of facets of P, whose vertices form a superset of the pointed vertices that have hindered domes, can be computed, if, for each puncturing vertex p, we compile a list of the facets of P that are intersected by the vertical line through p, and, among them, we retain only those for which the line segment delimited by p and the point of intersection lies entirely in P. For convenience, we assume that no vertex of the polytope lies vertically above or below any of the puncturing vertices. Note that this assumption can be checked in $O(n \log r)$ time and can be relaxed with little extra effort. This assumption narrows down the number of facets intersected by a vertical line through a puncturing vertex to at most two per boundary crossing; as a result, the number of facets in the final list will be proportional to the number of puncturing vertices. So, our task is to determine which (triangular) facets of P are traversed by a vertical line passing through a puncturing vertex.

If we take projection on the xy-plane, we can restate the problem in terms of the facets of P as follows: Which projections of puncturing vertices (on the xy-plane) lie in a given triangle? To answer such a question, we set up a complete binary tree whose leaves are in one-to-one correspondence, from left to right, with the puncturing vertices of P, ordered by nondecreasing x-coordinates. With each internal node t of the tree, we associate the canonical strip $\{(x, y) | x_{\min} \leq x \leq x_{\max}\}$, where $x_{\min}$ and $x_{\max}$ are respectively the smallest and largest x-coordinates of the vertices associated with the leaves descending from t. Let abc be the projection of a triangular facet f of P on the xy-plane, with $a \leq b \leq c$ in x-order. Following standard segment tree partition [23], and using the x-order of the tree leaves, we partition the x-extents of ab and bc into $O(\log r)$ intervals. This, in turn, induces a partition of abc into a logarithmic number of trapezoids (or triangles), every one of which is associated with a distinct node of the tree. For each such node t, we must determine which points among those stored in t's descending leaves lie inside the corresponding trapezoid. One nice feature in this set-up is that although we have many different trapezoids, we can replace each of them in the computation by one of two fixed double wedges.

We dualize the problem using the following asymmetric transformation: a point (u, v) is mapped to the line $y = ax + b$; a line $y = kx + d$ is mapped to the point $(-k, d)$. Note that the transformation maps vertical lines to a point at infinity, and this is what makes it attractive for our purposes. The reason is that no vertical line lies entirely in any of the double wedges with which we are dealing, and therefore, the chosen transformation maps such wedges to finite length line segments. (A different transformation might map such a double wedge to a pair of collinear rays.)

The application of the duality transformation results in each node of the tree being associated with a certain arrangement of lines (the duals of the projections of

the vertices stored at the leaves below the node in question). We do not store these arrangements explicitly, except the one at the root of the tree. This takes $O(r^2)$ time and space, using the methods of Chazelle *et al.* [10] and Edelsbrunner *et al.* [14]. We further process the root arrangement to support $O(\log r)$-time point location [18], [13], which requires a linear amount of work through the arrangement. As it turns out, we also need point-location structures for all the other arrangements in the tree. We can use an economical strategy, however, based on the fact that the arrangement of a node is a portion of the arrangement of its father. More specifically, each region of a father's arrangement lies entirely within one region of either child's. Thus, we provide each region of the root arrangement with two pointers, one directed toward the enclosing region for each child. The same pointer scheme can be applied throughout the tree, although the only arrangement, and hence geometric information, stored in the tree lies at the root and at the leaves. Setting up all these pointers can be done in $O(r^2)$ time by ensuring that the work at a given node is at most proportional to the square of the number of its descending leaves. In the general step, we have at our disposal the full arrangement at a given node, and we color the lines green or red, depending on which of the left or right children inherits them. To compute, say, the green arrangement with the pointers directed to it, we begin by merging collinear green edges into edges of the green arrangement. By traversing the full arrangement, we can now collect each region lying within a given region of the green arrangement. The same process applied to the red edges completes our work.

In dual space, the problem of finding which members of the point-set associated with a given tree node t lie in a double wedge is the same as computing which lines of a line arrangement intersect a line segment. The latter version is easily resolved using our data structure. First, we need to locate the endpoints p and q of the line segment in the line arrangement. To do that, we locate p and q in the root arrangement, and then using region-to-region pointers, we go down the tree stopping at the node t. The entire operation takes only $O(\log r)$ time. If p and q lie in the same region, no further work need be done, and the answer is the empty set. Otherwise, we pursue our search in the two children of t. This process takes us to a certain subset of the leaves, where the desired intersections can be found directly. Since a vertical line can cut only $O(r)$ facets, the total amount of time spent computing intersections between lines through puncturing vertices and facets of P is no more than $O(n \log r + r^2 \log r)$.

As mentioned earlier, of all the intersections between a vertical line through a given puncturing vertex p and the facets of P, only those for which the line segment connecting p to the point of intersection on the facet lies in P need be retained for further consideration (Lemma 3.1). Since a puncturing vertex will contribute no more than four facets, and there are at most $3r$ such vertices, the total number of selected facets is $O(r)$. Furthermore, during the selection process, pointers to the corresponding puncturing vertices are included in the records of the facets that are retained. As a result, in the end, the record of each such facet f contains a list of the puncturing vertices that potentially hinder the cups of the vertices of f. From a time complexity standpoint, the Witness Oracle accounts for $O((n + r^2) \log r)$ in the running time of the algorithm. The space needed amounts to $O(n + r^2)$.

3.3. *The Fence-Off Phase*

Our goal here is to triangulate a nondegenerate simple polytope of n vertices into $O(n^2)$ tetrahedra. This method is satisfactory if at least a fixed fraction of the edges are reflex. We begin by partitioning P into cylindrical pieces, and then we triangulate each piece separately. To build the cylindrical partition we attach vertical fences to each edge, reflex and nonreflex, one at a time. Let us say that a point p is *visible* from an edge if it can be connected to it by a vertical segment whose relative interior lies in the interior of P. The set of points visible from an edge e is easily seen to be a monotone polygon: it is called the *fence* of e and is to be attached to it (Fig. 10). Our fences are similar to the *walls* used in the *slicing theorem* of Aronov and Sharir [1]; one difference is that while fences project vertically onto their attaching edges, walls flood all over the free portion of the vertical plane passing through the edge.

The fencing operation partitions P into cylindrical pieces, each of which can be defined by

(i) specifying a horizontal base polygon,
(ii) lifting it vertically into an infinite cylinder, and
(iii) clipping the cylinder between two planes (which do not intersect inside the cylinder).

Although some fences may have exposed edges "sticking out," the hope is that in the end the cylindrical pieces will form a convex decomposition of P. Unfortunately, this is not always true. Of course, every reflex edge of P is "resolved" in the sense that the angles between its adjacent facets cease to be reflex. The problem is that new reflex edges might be created between two fences (Fig. 11). Let us examine this phenomenon in some detail. Let e be a vertical edge of a fence. Since the edge e is incident upon at least one vertex of P, and ∂P is triangulated, the edge e cannot be left exposed after the fencing. It is conceivable, however, that e coincides with an edge of another fence which results in a dihedral angle larger than π. What we can say at this point is that the fences partition P into cylindrical pieces, which are free of nonvertical reflex edges. But then, a triangulation of the base polygons of each cylindrical piece refines the partition into one consisting of cylindrical pieces whose base polygons are triangles. A decomposition of P into tetrahedra follows trivially.

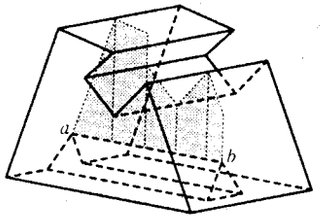

Fig. 10. The fence of ab.

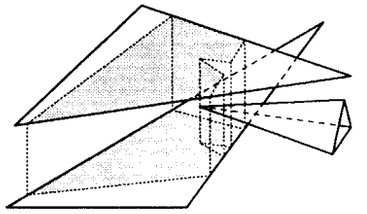

Fig. 11. A nonconvex cylindrical piece.

In general, erecting fences will result in cutting off some edges into subedges. We still treat each edge as one entity, and deal with all its subedges in one fell swoop. We give a very simple, albeit slightly inefficient, method for computing the fence of an edge e of P. Let Σ be the set of segments obtained by computing the intersection of the facets of P with the projection of e on their supporting planes. In general, Σ consists of a number of disjoint polygonal curves. Next, we compute the trapezoidal map induced by the visibility relation among the segments of Σ. This is the planar partition formed by Σ and all the vertical segments that connect endpoints to their visible segments in Σ. Since each trapezoid is bounded by a vertical segment through an endpoint, which corresponds to an edge of P, and there are $O(n)$ edges, the size of the trapezoidal map is $O(n)$. The map is computed by sorting the segments in Σ, and by sweeping them in order, to determine the visibility polygons. The entire computation for each edge can be carried out in $O(n \log n)$ time.

We do not, however, store each fence explicitly. Instead, we include each nonvertical bounding segment s of each fence in a list associated with the facet whose intersection with the fence is precisely s. After all fences have been computed, and their bounding segments have been attached to the appropriate facet records, each facet of P is scanned in turn, and its constrained triangulation is computed (the constraints are the edges incident upon the facet and the fence segments associated with it). The triangles reported are inserted in a global list of triangles. If we merge collinear segments during the triangulation, the list of triangles consists, in the end, of pairs of triangles such that

(i) the projections of the members of a pair on the xy-plane are identical, and
(ii) the cylindrical polytope that is determined and bounded by such a pair lies in P.

Note that the collection of these cylindrical pieces is nothing but the refined decomposition of P into cylindrical pieces that we mentioned earlier. The pairs of triangles will not necessarily be in order in the list. This can be achieved, however, by sorting the triangles in the list with respect to the x- and y-coordinates of their vertices (this will bring together triangles with the same projection on the xy-plane), and, in case of ties, with respect to the z-coordinate of their barycenters, which will do the final matching. Each cylindrical piece determined by such a pair of triangles can be trivially decomposed into at most three tetrahedra.

The introduction of fences results in a partition of P that involves a total of $O(n^2)$ vertices, edges, and facets. The triangulation of the facets and the extraction of tetrahedra merely adds a constant multiplicative factor to the size of the decomposition. Therefore, the description size of the final partition is $O(n^2)$, and consequently $O(n^2)$ tetrahedra are produced. As mentioned earlier, the computation of all the fences takes $O(n^2 \log n)$ time. The triangulation of a facet that is associated with k_i constraining segments can be carried out in $O(1 + k_i \log k_i)$ time, so that triangulating all the facets will take a total of $O(n^2 \log n)$ time, since $\sum_i k_i = O(n^2)$. Finally, sorting the triangles takes another $O(n^2 \log n)$ time, while processing the pairs does not take more than $O(n^2)$. Summarizing, the entire fence-off phase requires $O(n^2 \log n)$ time and $O(n^2)$ space.

3.4. *Putting the Pieces Together*

Given a nondegenerate simple polytope of zero genus with n vertices and r reflex edges, we start the partitioning by

(i) removing all flat vertices, and doing the obvious clean-up,
(ii) triangulating the boundary, and
(iii) applying the pull-off phase in case n greatly exceeds r.

We finish the decomposition by going through the fence-off phase. The running time of the algorithm is $O(n \log r + r^2 \log r)$. In practice, it will be important to have a robust representation of cell complexes in 3-space in order to carry out the computation successfully and efficiently. A representation of three-dimensional polyhedral subdivisions, along with the set of navigational primitives needed to carry out the required cutting operations, can be found in [12]. We summarize our results below.

Theorem 3.1. *In $O((n + r^2) \log r)$ time it is possible to partition a nondegenerate simple polytope of genus 0 with n vertices and r reflex edges into $O(n + r^2)$ tetrahedra. The time bound includes the cost of producing a full-fledged triangulation with an explicit description of its facial structure. Up to within a constant factor, the number of tetrahedra produced by the algorithm is optimal in the worst case.*

4. Closing Remarks

Of course, not every n-vertex polytope with r reflex edges necessitates $\Omega(n + r^2)$ tetrahedra to form a triangulation. Are there simple heuristics which could be used to guarantee that the triangulation size does not exceed the minimum by more than a fixed constant in *all* cases? Is there a polynomial-time algorithm for such an approximation scheme? Also, it is often desirable to avoid long, skinny tetrahedra in mesh generation. See [4] for similar concerns in two dimensions. One approach is to retriangulate the undesirable tetrahedra produced by our triangulation.

Again, are there preferred heuristics to keep the number of Steiner points as low as possible?

Acknowledgments

The authors would like to thank the anonymous referees whose comments improved the readability of the paper.

References

1. B. Aronov and M. Sharir, Triangles in Space, or Building and Analyzing Castles in the Air, *Proc. 4th Ann. ACM Symp. Comput. Geom.* (1988), 381–391.
2. C. L. Bajaj and T. K. Dey, Robust Decompositions of Polyhedra, Dept. Computer Science, Purdue University, 1989.
3. T. J. Baker, Three-Dimensional Mesh Generation by Triangulation of Arbitrary Point Sets, Dept. Mechanical Engineering, Princeton University, 1986.
4. B. S. Baker, E. Grosse, and C. S. Rafferty, Non-Obtuse Triangulation of Polygons, *Discrete Comput. Geom.* **3** (1988), 147–168.
5. B. G. Baumgart, A Polyhedron Representation for Computer Vision, *Proc. 1975 National Comput. Conf.*, AFIPS Conference Proceedings, Vol. 44, AFIPS Press, Montvale, NJ, 1975, 589–596.
6. J. F. Canny, A new Algebraic Method for Motion Planning and Real Geometry, *Proc. 28th Ann. IEEE Symp. on Found. Comput. Sci.* (1987), 39–48.
7. B. Chazelle, Convex Partitions of Polyhedra: A Lower Bound and Worst-Case Optimal Algorithm, *SIAM J. Comput.* **13** (1984), 488–507.
8. B. Chazelle, Approximation and Decomposition of Shapes, *Advances in Robotics*, Vol. 1 (J. T. Schwartz and C. K. Yap, ed.), Erlbaum, Hillsdale, NJ, 1987, 145–185.
9. B. Chazelle, H. Edelsbrunner, L. J. Guibas, and M. Sharir, A Singly-Exponential Stratification Scheme for Real Semi-Algebraic Varieties and Its Applications, *Proc. 16th ICALP*, Lecture Notes in Computer Science, Vol. 372, Springer-Verlag, Berlin, 1989, 179–193.
10. B. Chazelle, L. J. Guibas, and D. T. Lee, The Power of Geometric Duality, *BIT* **25** (1985), 76–90.
11. G. E. Collins, Quantifier Elimination for Real Closed Fields by Cylindric Algebraic Decomposition, *Proc. 2nd GI Conf. Automata Theory and Formal Languages*, Lecture Notes in Computer Science, Vol 33, Springer-Verlag, Berlin, 1975, 134–183.
12. D. P. Dobkin and M. J. Laszlo, Primitives for the Manipulation of Three-Dimensional Subdivisions, *Proc. 3rd Ann. ACM Symp. Comput. Geom.* (1987), 86–99.
13. H. Edelsbrunner, L. J. Guibas, and J. Stolfi, Optimal Point Location in a Monotone Subdivision, *SIAM J. Comput.* **15** (1986), 317–340.
14. H. Edelsbrunner, J. O'Rourke, and R. Seidel, Constructing Arrangements of Lines and Hyperplanes with Applications, *SIAM J. Comput.* **15** (1986), 341–363.
15. H. Feng and T. Pavlidis, Decomposition of Polygons into Simpler Components: Feature Generation for Syntactic Pattern Recognition, *IEEE Trans. Comput.* **24** (1975), 636–650.
16. D. A. Field, Implementing Watson's Algorithm in Three Dimensions, *Proc. 2nd Ann. ACM Symp. Comput. Geom.* (1986), 246–259.
17. L. J. Guibas and J. Stolfi, Primitives for the Manipulating of General Subdivisions and the Computation of Voronoi Diagrams, *ACM Trans. Graphics* **4** (1985), 75–123.
18. D. G. Kirkpatrick Optimal Search in Planar Subdivisions, *SIAM J. Comput.* **12** (1983), 28–35.
19. A. Lingas, The Power of Non-Rectilinear Holes, *Proc. 9th Colloq. Automata, Languages and Programming*, Lecture Notes in Computer Science, Vol. 140, Springer-Verlag, Berlin, 1982, 369–383.
20. K. Mehlhorn, *Data Structures and Algorithms*, Vol. 3, Springer-Verlag, Berlin, 1984.
21. D. E. Muller and F. P. Preparata, Finding the Intersection of Two Convex Polyhedra, *Theoret. Comput. Sci.* **7** (1978), 217–236.

22. J. O'Rourke, *Art Gallery Theorems and Algorithms*, Oxford University Press, Oxford, 1987.
23. F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.
24. D. Prill, On Approximations and Incidence in Cylindrical Algebraic Decompositions, *SIAM J. Comput.* **15** (1986), 972–993.
25. J. Ruppert and R. Seidel, On the Difficulty of Tetrahedralizing 3-Dimensional Non-Convex Polyhedra, *Proc. 5th Ann. ACM Symp Comput. Geom.* (1989), 380–392.
26. B. Schachter, Decomposition of Polygons into Convex Sets, *IEEE Trans. Comput.* **27** (1978), 1078–1082.
27. J. T. Schwartz and M. Sharir, On the "Piano Movers" Problem, II: General Techniques for Computing Topological Properties of Real Algebraic Manifolds, *Adv. in Appl. Math.* **4** (1983), 298–351.
28. W. Smith, Studies in Computational Geometry Motivated by Mesh Generation, Ph.D. Thesis, Princeton University, 1988.
29. H. Whitney, Elementary Structure of Real Algebraic Varieties, *Ann. of Math.* **66** (1957), 545–556.

Received August 15, 1989, and in revised form March 27, 1990.