Linear time approximation of 3D convex polytopes

Mario A. Lopez a, Shlomo Reisner b,∗

a Department of Mathematics and Computer Science, University of Denver, Denver, CO 80208, USA
b Department of Mathematics and Department of Mathematics Education (Oranim), University of Haifa, Haifa 31905, Israel

Received 13 November 2001; received in revised form 25 March 2002; accepted 3 June 2002
Communicated by J. Zaks

Abstract

We develop algorithms for the approximation of a convex polytope in \( \mathbb{R}^3 \) by polytopes that are either contained in it or containing it, and that have fewer vertices or facets, respectively. The approximating polytopes achieve the best possible general order of precision in the sense of volume-difference. The running time is linear in the number of vertices or facets.

© 2002 Elsevier Science B.V. All rights reserved.

1. Introduction

In this paper we develop algorithms for the approximation of convex polytopes in \( \mathbb{R}^3 \) by simpler polytopes. The term “simpler” means the following: we present inner approximation algorithms that replace a convex polytope \( P \) in \( \mathbb{R}^3 \) having \( n \) vertices, by a convex polytope \( Q \) contained in \( P \) with \( k < n \) vertices. The outer approximation algorithms replace a convex polytope \( P \) with \( n \) facets, by a convex polytope \( R \) containing \( P \), that has \( k < n \) facets. The precision of the approximation is measured by the difference of volume between \( P \) and \( Q \), or, respectively, between \( R \) and \( P \). In both cases we achieve an order of precision which is best possible as a general estimate. Namely, if we normalize \( P \) to have volume equal to 1, then the above volume-difference is not greater than \( c (\frac{k}{n} - \frac{1}{n}) \), where \( c \) is a positive constant which is independent of \( P \) (the remark following Theorem 2.4 explains why this estimate is best possible). The procedures of the inner and outer approximations are very similar in their formal algorithmic aspects. Therefore, we describe in full detail the inner approximation, while only the main points concerning the outer approximation are explained. The algorithms we present are based

∗ Corresponding author.
E-mail addresses: mlopez@cs.du.edu (M.A. Lopez), reisner@math.haifa.ac.il (S. Reisner).
on recent mathematical results of [17], concerning convex polytopes. We present both randomized and deterministic algorithms with $O(n)$ running time, where $n$ is the number of vertices or facets of $P$.

The analogous two-dimensional problem, of approximating convex polygons by simpler polygons, has been investigated fairly intensively. In contrast, for $\mathbb{R}^3$, one finds very few algorithmic results in the literature concerning the approximation of polytopes by polytopes. Methods for the approximation of general convex bodies in $\mathbb{R}^d$ by convex polytopes can be found in the mathematical literature (see [7,8] for surveys, as well as [6]). These methods can be applied to polytope approximation, but no serious attempt (except for [13] in the special case of ellipsoids) to refine these methods into algorithms and to investigate their efficiency has been made so far. Algorithmic results which can be interpreted as approximation results analogous to ours, with the Hausdorff distance as a measure of precision, were obtained in [4] where a randomized algorithm is given which, for 3D polytopes, runs in $O(k^2 n \log k \cdot \log(n/k))$ time in the worst case ($n$ and $k$ as above). In [1] a deterministic algorithm based on similar ideas is presented, but its running time is significantly higher. While these algorithms produce approximating polytopes with precision rate which is (almost) a multiple of the best possible for the specific polytope $P$ (in the Hausdorff distance), they are slower and harder to implement than our algorithms, which give best possible order of precision in the worst case. Let us mention that much related work has been done in the area of surface simplification, some of the algorithms proposed as part of this work also provide theoretical bounds. See for example [10,11].

The paper is structured as follows. In Section 2 we bring, in full detail, the algorithms for inner approximation. Section 3 explains how algorithms for outer approximation are constructed along the same lines. We also discuss in Section 3 the use of our approach in dimensions other than three, and show that our algorithms adapt efficiently only for two and three dimensions. In Section 4 we compute upper bounds for the constants involved in the precision estimates. These upper bounds are important not only for the sake of estimating the precision, but are also needed in the implementation of the algorithms.

We now present some terminology and notation. A convex polytope $P$ in the Euclidean space $\mathbb{R}^d$ is the convex hull of a finite set of points, in the present paper we always assume that $P$ is full-dimensional, that is, it has a non-empty interior. An extreme point of $P$ is called a vertex. The set of vertices of $P$ is $\text{vert}(P)$. Equivalently, $P$ is a convex polytope in $\mathbb{R}^d$ if and only if it is a bounded set (with non-empty interior) which is the intersection of a finite set of half-spaces bounded by hyperplanes. The $(d-1)$-dimensional faces of a $d$-dimensional polytope are its facets. For a set $A$ we denote by $|A|$ the cardinality of $A$. The volume of a (measurable) set $K \subset \mathbb{R}^3$ is denoted by $\text{vol}(K)$. The convex hull of a set $M$ is $\text{conv}(M)$.

2. Inner approximation

In this section we introduce algorithms for approximating a convex polytope $P$ in $\mathbb{R}^3$, having $n$ vertices, by a convex polytope $Q$ contained in $P$, which has $k$ vertices. The first algorithm that we present is a randomized algorithm that runs in $O(n)$ expected time ($O(n-k)$ if $\text{vol}(P)$ is known). Then, a deterministic algorithm is presented, which runs in $O(n)$ time. In these algorithms the constant which is involved in the $O(n)$ estimate is strongly influenced by the degree of precision of the approximation which the user expects.
We make here the remark that these estimates assume that the polytope $P$ is given together with its “convex hull”, by which we mean that all the adjacency relations between vertices, edges and facets of $P$ are given. Computation of the convex hull of $P$ requires $O(n \log n)$ time (see [2,3,16], for example).

The algorithms operate by repeatedly removing a carefully chosen vertex from the current polytope. Thus, the basic step selects a vertex $v$ from the current polytope $P_i$, and replaces $P_i$ by \( \text{conv}(\text{vert}(P_i) \setminus \{v\}) \), the convex hull of all vertices of $P_i$ except $v$. When repeated $n - k$ times, vertex removal yields a polytope of the desired size. We refer to the part removed, i.e., $P_i \setminus \text{conv}(\text{vert}(P_i) \setminus \{v\})$, as the cap of $v$.

The algorithms differ on the method (and time) used to select the next vertex to remove. We now elaborate on how to implement this step efficiently.

Let $u$ and $v$ be distinct vertices of $P_i$. We say that $u$ and $v$ are adjacent if they are connected by an edge of $P_i$. The number of vertices adjacent to $v$ is the degree of $v$, denoted by $\deg(v)$.

To compute $\text{conv}(\text{vert}(P_i) \setminus \{v\})$, let $N_i(v)$ denote the set of vertices adjacent to $v$ in $P_i$. A facet $g$ of $\text{conv}(N_i(v))$ is said to be visible from $v$ if $\text{conv}(N_i(v))$ and $v$ are on opposite sides of the supporting plane of $g$ (or if $\text{conv}(N_i(v))$ is 2-dimensional, in which case it is considered a visible face).

**Lemma 2.1.** The facets of $\text{conv}(\text{vert}(P_i) \setminus \{v\})$ are of three kinds:

(a) Facets $f$ of $P_i$ not incident to $v$.
(b) Facets $g$ which are the convex hull of the vertices, other than $v$, of a facet $f$ of $P_i$ such that $f$ contains $v$ and has more than three vertices.
(c) Facets $h$ of $\text{conv}(N_i(v))$ which are visible from $v$.

Each facet $g$ of type (b) can be computed in constant time by removing $v$ from the corresponding facet $f$ of $P_i$. This is done by replacing the two edges of $f$ incident on $v$ with a single edge that connects the two vertices of $f$ that are adjacent to $v$. Of course, there are at most $m = \deg(v)$ such facets. Computing the convex hull of $N_i(v)$ can be done in $O(m \log m)$ time. All facets of type (c) can then be found in $O(m)$ additional time from $\text{conv}(N_i(v))$. Thus the removal of a vertex $v$ of degree $m$ from $P_i$ can be done in $O(m \log m)$ time.

Also, as discussed later, all algorithms require the computation of the volume lost to the removal, i.e., $\text{vol}(P_i) - \text{vol}(\text{conv}(\text{vert}(P_i) \setminus \{v\}))$. Once $\text{conv}(N_i(v))$ has been computed, this can be done in time proportional to the number of edges in $\text{conv}(N_i(v))$, i.e., $O(|N_i(v)|)$ time, by adding the volume of all pyramids with common apex $v$ and whose bases are the visible facets of $\text{conv}(N_i(v))$.

Let $P$ be a polytope in $\mathbb{R}^3$ with $n$ vertices. In assessing the quality of the approximation we use a recent result of [17]. When applied to 3D, this result implies the existence of constants $c_0$ and $c_1$ such that for every $0 < \varepsilon < 1/2$ and $n \geq c_0/\varepsilon$ there exist at least $(1 - 2\varepsilon)n$ vertices $x$ of $P$ which satisfy

$$\frac{\text{vol}(P) - \text{vol}(\text{conv}(\text{vert}(P) \setminus \{x\}))}{\text{vol}(P)} \leq c_1 \varepsilon^{-2} n^{-2}. \tag{2.1}$$

In the present section we accept the above constants $c_0$ and $c_1$ as given quantities. In Section 4 we shall elaborate on upper bounds for these constants and other constants that appear later in the paper and depend on them. In the sequel, we say that a vertex $x$ of $P_i$ is $\varepsilon$-useful if (2.1) is satisfied for $x$ and $P_i$. When $\varepsilon$ is clear from the context we may simply call a vertex useful instead of $\varepsilon$-useful.
2.1. A randomized algorithm

Lemma 2.2. Let \( P \) be a polytope in \( \mathbb{R}^3 \) with \( n \) vertices. For any \( 0 < \beta < 1 \) there are at least \( \beta n \) vertices of \( P \) with degree less than \( \frac{3(2-\beta)}{(1-\beta)} \).

Proof. Let \( S \) be the set of vertices of \( P \) with degree less than \( 3(2-\beta)/(1-\beta) \). Euler’s theorem guarantees that \( \sum_{v \in P} \deg(v) \leq 6n - 12 \). If \( |S| \leq \beta n \) then \( |S^c| \geq (1-\beta)n \). So, we have a set of \( [(1-\beta)n] \) vertices for which the degree is at least \( 3(2-\beta)/(1-\beta) \). The remaining set of vertices counts \( \lfloor \beta n \rfloor \) vertices. Since every vertex has degree at least 3 and \( 3(2-\beta)/(1-\beta) > 3 \) we have

\[
\sum_{v \in P} \deg(v) \geq 3|\beta n| + [(1-\beta)n] \frac{3(2-\beta)}{(1-\beta)}
\]

\[
\geq 3|\beta n| + (1-\beta)n \frac{3(2-\beta)}{(1-\beta)} > 6n - 3 > 6n - 12,
\]

a contradiction. \( \square \)

The above implies, for example, that at least 75% of the vertices of \( P \) must have degree no more than 14. Combined with the fact that usefulness occurs frequently we now establish the abundance of useful vertices of low degree.

Lemma 2.3. Let \( P \) be a polytope in \( \mathbb{R}^3 \) with \( n \) vertices. Then for any \( 6 < r < n \) and \( \varepsilon < \frac{n-6}{2\varepsilon} \), if \( n \geq c_0/\varepsilon \) then there exist at least \( \sigma n \) \( \varepsilon \)-useful vertices of degree less than \( r \), where \( \sigma = \frac{n-6}{r-3} - 2\varepsilon \) (note that \( \sigma > 0 \)).

Proof. Let \( S_r \) denote the set of vertices of \( P \) with degree less than \( r \), and \( S_u \), the set of vertices that are \( \varepsilon \)-useful. For \( r > 6 \) Lemma 2.2 implies \( |S_r| \geq \frac{n-6}{r-3} n \). Similarly, from [17] we know that \( |S_u| \geq (1-2\varepsilon)n \). It follows that \( |S_r \cap S_u| \geq (\frac{n-6}{r-3} - 2\varepsilon)n = \sigma n \), thus the claim is established. \( \square \)

Note that the required relation between \( r \) and \( \varepsilon \) which is stated in Lemma 2.3, can be reversed to give the same \( \sigma \) provided \( 0 < \varepsilon < 1/2 \) and \( r > \frac{6(1-\varepsilon)}{1-2\varepsilon} \).

RandApprox1\((P,n,k,\varepsilon,r)\)
Input: a polytope \( P \) with \( n \) vertices, \( c_0/\varepsilon \leq k < n \).
Output: a polytope \( Q \subset P \) with \( k \) vertices.
1. Compute \( \text{vol}(P) \) and let \( P_0 = P \)
2. for \( i \leftarrow n \) downto \( k+1 \) do
3. Repeatedly select a vertex of \( P_i \) at random (no duplicates) until a vertex \( v \) which is both useful and has degree \( \leq r \) is found.
4. Let \( P_{i-1} \leftarrow \text{conv}(\text{vert}(P_i) \setminus \{v\}) \), compute \( \text{vol}(P_{i-1}) \)
5. return \( Q = P_k \)

Analysis. The computation of \( \text{vol}(P) \) in step 1 takes \( O(n) \) time. We now fix \( i, k+1 \leq i \leq n \), and estimate the time needed for step 3. For a vertex \( v \) of \( P_i \), the time needed to compute the convex hull of
Assume that the number of useful vertices having degree at most \( r \) of \( P_i \) is \( \beta i \) (then, by Lemma 2.3, \( \beta \geq \sigma \)). Let \( v_1, \ldots, v_{j-1} \) be vertices of \( P_i \), which are either non-useful or have degree greater than \( r \), and let \( v_j \) be a useful vertex of degree at most \( r \). Let \( A(v_1, \ldots, v_{j-1}, v_j) \) be the event that in step 3, \( v_1, \ldots, v_{j-1} \) are sampled (in this order) before \( v_j \) is selected to be removed. For convenience of the estimation we allow repetitions in the sampling of \( v_1, \ldots, v_{j-1}, v_j \) (this will give an upper bound on the expected time without repetitions). Hence in the estimates below we should allow \( j \) to be larger than \( i \), in fact, arbitrarily large. We check each one of \( v_1, \ldots, v_{j-1}, v_j \) first to see if its degree is greater than \( r \); this check takes a constant time and if the answer is positive, we ignore this vertex and move on to the next one. If the degree is not greater than \( r \), then we check for usefulness, but in this case the time for this check, as well as the time to remove \( v_j \) is bounded by a constant times \( r \log r \). So, the time required by the event \( A(v_1, \ldots, v_{j-1}, v_j) \) is \( O(jr \log r) \).

Thus, the expected time of step 3 for \( P_i \) is bounded by a constant (depending on \( r \)) times the expected number of attempts in this step, which is

\[
\sum_{j=1}^{\infty} j\beta(1-\beta)^{j-1} = \frac{1}{\beta} \leq \frac{1}{\sigma}.
\]

As \( \sigma \) depends on user defined constants \( \varepsilon \) and \( r \), this expectation is \( O(1) \). Finally, the computation of \( \text{vol}(P_{i+1}) \) is practically done already in step 3. All this implies that the expected time for the algorithm is \( O(n) \).

**Remark.** If \( \text{vol}(P) \) is known beforehand, then the running time of the algorithm is \( O(n - k) \).

### 2.2. A deterministic algorithm

We now describe a deterministic algorithm that computes the approximating polytope in linear time. The idea is to repeatedly remove as many vertices as possible without having to update the caps of vertices adjacent to the removed vertices. To this end, we repeatedly identify and eliminate a set of useful vertices which constitute an independent set in the 1-skeleton of the current polytope. During this process we are essentially computing a hierarchical representation of the input polytope, an idea proposed earlier in [5,12] (see also Section 7.10 of [15]), in the context of point location, polytope separation and other problems.

We start with an auxiliary algorithm that identifies and removes one independent set of vertices. Assume, initially, that \( k \) satisfies the following precondition:

\[
k > \left( 1 + \frac{2\varepsilon}{r} - \frac{r - 6}{r(r - 3)} \right) n = n - O\left( \frac{1 - 2\varepsilon}{r} n \right)
\]

(\( \varepsilon \) and \( r \) are as above).

Clearly, the removal of a vertex \( v \) does not alter the cap of a vertex \( w \) if \( v \) and \( w \) are not adjacent (this can be concluded with the help of Lemma 2.1). Thus, we identify a subset \( R \subset \text{vert}(P) \) of size \( n - k \) such that for all \( v \in R \): \( v \) is useful and \( \text{deg}(v) < r \) and \( v \) is not adjacent to \( w \) for any other \( w \in R \). The following algorithm, based on this simple idea, finds an approximating polytope \( Q \), provided that \( k \) satisfies (2.2) above.
**Auxiliary** \((P, n, k, \epsilon, r)\)

*Input*: a polytope \(P\) with \(n\) vertices, \(c_0/\epsilon \leq k < n\), \(k\) satisfies (2.2).

*Output*: a polytope \(Q \subset P\) with \(k\) vertices.

1. Compute a list \(L\) of all useful vertices of \(P\) of degree < \(r\)
2. Mark all vertices of \(L\) as clean
3. Let \(P_n = P\)
4. for \(i \leftarrow n\) downto \(k + 1\) do
   5. find the next clean vertex \(v\) in \(L\)
   6. mark \(v\) and all vertices adjacent to it in \(L\) as dirty
   7. Let \(P_{i-1} \leftarrow \text{conv}(\text{vert}(P_i) \setminus \{v\})\)
5. return \(Q = P_k\)

**Analysis.** Step 1 requires \(O(nr \log r)\) time. Since each vertex in the list is scanned at most once, the total time spent in step 5 is \(O(n)\). Each execution of steps 6 and 7 requires \(O(r)\) and \(O(r \log r)\) time, respectively. Thus, the total time is bounded by \(O(nr \log r)\).

Clearly, if \(n - k\) is too large then \(R\), as specified above, may not exist. We now establish a lower bound on the maximum possible value of \(n - k\). From Lemma 2.2 we know there are at least \(\beta n\) vertices of \(P\) that have degree less than \((2 - \beta)/(1 - \beta)\) and at least \((1 - 2\epsilon)n\) vertices of \(P\) are useful. Thus, by Lemma 2.3, at least \(\beta n + (1 - 2\epsilon)n - n = (\beta - 2\epsilon)n\) vertices are both useful and have small degree. Since the removal of a (clean) vertex produces at most \(r\) dirty vertices and we want to remove \(n - k\) (clean) vertices, to be on the safe side, we require

\[
(\beta - 2\epsilon)n > (n - k)r
\]

that is,

\[
n - k < \frac{\beta - 2\epsilon}{r} n.
\]

Using Lemma 2.2 we can express this constraint in terms of \(r\) and \(\epsilon\) in the form of (2.2) above. Let \(\gamma\) be any positive number less than \((r-6)/r(r-3)\) and \(\delta < \gamma\). We shall be interested in the quantity \(1/(\gamma - \delta)\), note that we can get this quantity to be, for example, as small as \(2r(r-3)/(r-6-2\epsilon(r-3))\).

**DetApprox2** \((P, n, k, \epsilon, r)\)

*Input*: a polytope \(P\) with \(n\) vertices, \(\max(c_0/\epsilon, 1/(\gamma - \delta)) \leq k < n\).

*Output*: a polytope \(Q \subset P\) with \(k\) vertices.

1. \(n_0 \leftarrow n\)
2. \(Q_0 \leftarrow P\)
3. \(i \leftarrow 0\)
4. repeat
   5. \(n_{i+1} \leftarrow \max(k, [(1 - \gamma)n_i])\)
   6. \(Q_{i+1} \leftarrow \text{Auxiliary}(Q_i, n_i, n_{i+1}, \epsilon, r)\)
   7. \(i \leftarrow i + 1\)
5. until \(n_i = k\)
6. return \(Q = Q_i\)
Analysis. As long as $n_i$, the number of vertices in the polytope $Q_i$ used as the input in the $i$th iteration ($n_0 = n$) has not reached the value $k$, we have $(1 - \gamma)n_i \leq (1 - \delta)n_i$. In fact, the left hand side of the last inequality is clear, while the right hand side is derived as follows:

$$n_i = \lceil (1 - \gamma)n_{i-1} \rceil \leq (1 - \delta)n_{i-1},$$

this is because

$$(1 - \delta)n_{i-1} - (1 - \gamma)n_{i-1} \geq (\gamma - \delta)k \geq 1.$$

Now, for

$$i_0 = \left\lceil \frac{\log(n/k)}{\log(1/(1 - \delta))} \right\rceil$$

we get $(1 - \delta)i_0 n \leq k$, hence there are at most $i_0$ steps of the iteration. The $(i + 1)$th step (steps 5–7) requires $O(n_i r \log r)$ time. Summing up, the total time is bounded by

$$r \log r \sum_{i=0}^{i_0-1} n_i < r \log r \cdot n \sum_{i=0}^{\infty} (1 - \delta)^i < nr \log r \cdot \frac{1}{\delta}.$$

Thus, by the choice of $\delta$, the running time is $O(r^2 \log r)$.

The following theorem summarizes our discussion. Note that the estimate (2.4) in it is obtained by summing up the inequalities (2.1) associated with the polytopes $P_i$ of the algorithm (see [14] and [17] for detail).

**Theorem 2.4.** Let $\varepsilon < 1/2$ and $r > \frac{\alpha_1}{1 - 2\varepsilon}$ be arbitrary user-defined constants. There exists an algorithm that runs in $O(n)$ time which, given a polytope $P$ with $n$ vertices and $k$ such that $\alpha_1 \leq k < n$, where $\alpha_1$ is a constant depending on $\varepsilon$ and $r$, finds a polytope $Q \subset P$ with $k$ vertices such that

$$\text{vol}(P) - \text{vol}(Q) \leq \alpha_1 \varepsilon^2 \left( \frac{1}{k - 1} - \frac{1}{n} \right) \text{vol}(P).$$

(2.4)

$a_1$ is a constant independent of $P$, $n$, $k$, $\varepsilon$ or $r$.

Also, there exists a randomized algorithm that runs in expected $O(n)$ time and produces a polytope $Q$ with $k$ vertices that satisfies (2.4).

In practice, a slightly better polytope $Q$ may be obtained by a less efficient but simpler to implement algorithm. The idea is to construct and maintain a heap, based on cap volume, on all vertices of $P$ of small degree, independent of whether such vertices are useful or not. We then repeatedly remove the cheapest vertex (which is guaranteed to be useful), re-evaluate the degree and update the heap for each of its adjacent vertices. The process, which is repeated $n - k$ times, runs in $O((n - k) \log n + n)$ time and results in a polytope $Q$ with $k$ vertices that satisfies (2.4).

**Remark.** The precision rate (2.4) is best possible as a general estimate (up to the constants involved), this is seen by the following argument: there is a positive constant $c$, such that for every polytope $Q$ in $\mathbb{R}^3$ with $j$ vertices which is contained in the Euclidean unit ball $B$, the following inequality holds:

$$\text{vol}(B) - \text{vol}(Q) \geq \frac{c}{j} \text{vol}(B)$$
Given \( j \), let \( P \subset B \) be a polytope with \( n \) vertices, \( n > j \), \( n \) big enough to approximate \( B \) to any desired precision (in the volume-difference sense). Then for any constant \( \gamma < c \) we get, if \( n \) is big enough,

\[
\text{vol}(P) - \text{vol}(Q) \geq \frac{\gamma}{j} \text{vol}(P)
\]

for any polytope \( Q \subset P \) with \( j \) vertices. This shows that a general estimate better than (2.4) is impossible.

## 3. Other applications

In this section we consider applications of the techniques described in Section 2 to other problems, namely the construction of outer approximating polytopes and the approximation of polygons.

The two algorithms presented earlier work in a complete analogy for outer approximation. Here we consider a convex polytope \( P \) in \( \mathbb{R}^3 \), having \( n \) facets and the result of the algorithm is a convex polytope \( R \) containing \( P \), which has \( k \) facets \( (k < n) \) and approximates \( P \), in the volume-difference sense, with best possible order of precision in general, in terms of \( k \). The new algorithms are again based on a mathematical result from [17] whose 3-dimensional case is dual to (2.1). Using this result instead of (2.1) we derive algorithms which are “dual” to those of Section 2. In these algorithms the roles of vertices and facets are exchanged and the removal of a facet is accomplished by replacing \( P \) with the intersection of the remaining facet half-spaces. This results in the following theorem (in which the constant \( \alpha_2 \) depends on its parameters in the same way as \( \alpha_1 \) of Theorem 2.4):

**Theorem 3.1.** Let \( \varepsilon < 1/2 \) and \( r > \frac{\varepsilon(1-\varepsilon)}{1-2\varepsilon} \) be arbitrary user-defined constants. There exists an algorithm that runs in \( O\left( \frac{n}{\varepsilon^2} \log n \right) \) time which, given a polytope \( P \) with \( n \) facets and \( k \) such that \( \alpha_2 \leq k < n \), where \( \alpha_2 \) is a constant depending on \( \varepsilon \) and \( r \), finds a polytope \( R \supset P \) with \( k \) facets such that

\[
\text{vol}(R) - \text{vol}(P) \leq \alpha_2 \varepsilon^{-2} \left( \frac{1}{k} - \frac{1}{n} \right) \text{vol}(P),
\]

where \( \alpha_2 \) is a constant independent of \( P, n, k, \varepsilon \) or \( r \).

Also, there exists a randomized algorithm that runs in expected \( O(n) \) time and produces a polytope \( R \) with \( k \) facets that satisfies (3.1).

We now turn our attention to the problem of approximating a \( d \)-dimensional polytope \( P \). In [14] we constructed algorithms that run in \( O((n-k) \log n + n) \) time which approximate \( n \)-gons in \( \mathbb{R}^2 \), from inside or from outside, by a \( k \)-gon. The algorithms presented here so far, adapt easily to 2D. Algorithms 1 and 2 of Section 2 result in new linear time approximation algorithms for polygons, as discussed below. Once more we make use of the results of [17]. In particular, for a polygon \( P \) with \( n \geq c_0/\varepsilon \) vertices the equivalent of (2.1) is

\[
\frac{\text{area}(P) - \text{area}(\text{conv}(\text{vert}(P) \setminus \{x\}))}{\text{area}(P)} \leq c_1 \varepsilon^{-3} n^{-3}.
\]

The fact that in the 2D case the degree of every vertex is 2, simplifies the algorithms. The randomized Algorithm 1 can be applied in 2D in the same way as before, except that in step 3 we have to look only for useful vertices and not worry about the degree. This improves the constant in the \( O(n) \) estimate of
the running time. The deterministic Algorithm 2 can also be used for the 2D case, with better constants. This is stated in the following theorems:

**Theorem 3.2.** Let \( \varepsilon < 1/2 \) be an arbitrary user-defined constant. There exists an algorithm that runs in \( O\left(\frac{n}{1-2\varepsilon}\right) \) time which, given a polygon \( P \) with \( n \) vertices and \( k \) such that \( \alpha_3 \leq k < n \), where \( \alpha_3 \) is a constant depending on \( \varepsilon \), finds a polygon \( Q \subset P \) with \( k \) vertices such that

\[
\text{area}(P) - \text{area}(Q) \leq a_3 \varepsilon^{-3} \left( \frac{1}{k^2} - \frac{1}{n^2} \right) \text{area}(P),
\]

where \( a_3 \) is a constant independent of \( P, n, k \) or \( \varepsilon \).

**Theorem 3.3.** Let \( \varepsilon < 1/2 \) be an arbitrary user-defined constant. There exists an algorithm that runs in \( O\left(\frac{n}{1-2\varepsilon}\right) \) time which, given a polygon \( P \) with \( n \) vertices and \( k \) such that \( \alpha_4 \leq k < n \), where \( \alpha_4 \) is a constant depending on \( \varepsilon \), finds a polygon \( R \supset P \) with \( k \) vertices such that

\[
\text{area}(R) - \text{area}(P) \leq a_4 \varepsilon^{-3} \left( \frac{1}{k^2} - \frac{1}{n^2} \right) \text{area}(P),
\]

where \( a_4 \) is a constant independent of \( P, n, k \) or \( \varepsilon \).

For dimension \( d > 3 \), the best possible rate of inner and outer approximation is of the order \( cd \text{vol}(P)\left(k^{2/(d-1)} - n^{-2/(d-1)}\right) \). While our algorithms can be generalized to provide an approximation rate of \( cd^3 \text{vol}(P)\left(k^{2/(d-1)} - n^{-2/(d-1)}\right) \), they do not do so efficiently. The reason for this is that for \( d > 3 \) and arbitrarily large \( n \) it is possible to construct a simplicial \( d \)-polytope \( P \) with the following properties:

1. For each vertex \( v \) of \( P \) the number of vertices that share a facet with \( v \) is \( n - 1 \). In other words, every vertex is adjacent to every other vertex.
2. For each vertex \( v \) of \( P \) the number of facets of \( \text{conv}(\text{vert}(P) \setminus \{v\}) \) that are visible from \( v \) (see definition in Section 2) is \( \Theta(n^{[d/2]}) \).

Examples of polytopes with the properties above include the so-called cyclic polytopes. Nonexistence of such phenomena in 3D and the three-dimensional Euler's formula are essential factors that enable one to produce efficient algorithms in 3D.

### 4. Constant estimation

The constants \( c_0, c_1 \) of Section 2 and their counterparts in the outer approximation of Section 3, are important in order to decide for which values of \( n \) (number of vertices or facets) the algorithms are guaranteed to work. These constants determine also the precision rate of the algorithms. Moreover, both algorithms require verifying whether a vertex (or a facet) is useful. Now, the definition of usefulness involves the constant \( c_0 \). Hence we need estimates on the constants for the very implementation of the algorithms.

The main steps in the estimation below follow those of the proof in [17]. In dimension three, however, we obtain more accurate estimates than those of [17]. We shall refer to these steps briefly. Let us remark that what we get are upper bounds and that it is likely that a refined proof can yield smaller estimates.
The starting point is a result of [9] from 1951.

**Theorem 4.1** [9]. Let \( n \) points be located on a Euclidean sphere of radius \( R \) in \( \mathbb{R}^3 \) in such a way that the distance between each two of them is at least 1. Then

\[
\frac{\sqrt{3} n^2}{2} \leq 4 \pi R^2.
\]  

**Corollary 4.2.** Let \( 0 < \delta < 1 \) be given. There exists a \( \delta \)-net of cardinality not greater than \( (32 \pi / \sqrt{3}) / \delta^2 \) on the Euclidean sphere of radius 2 in \( \mathbb{R}^3 \).

**Proof.** Let \( \Delta \) be a maximal set of points on \( 2S \) (\( S \) is the Euclidean unit sphere), with the property that the distance between any pair of elements of \( \Delta \) is at least \( \delta \). Theorem 4.1, with \( R = 2/\delta \) shows that \( |\Delta| \leq (32 \pi / \sqrt{3}) / \delta^2 \). Maximality of \( \Delta \) implies now that every point of \( 2S \) is within distance \( \delta \) from some point of \( \Delta \). \( \square \)

Using Corollary 4.2 and proceeding as in the proof of Proposition 2.5 of [17], we get:

**Proposition 4.3.** Let \( C \) be a convex body in \( \mathbb{R}^3 \), which is contained in the Euclidean unit ball \( B \). For every \( 0 < \varepsilon < 1 \) there exists a convex polytope \( P_\varepsilon \) contained in \( C \), such that \( d_H(C, P_\varepsilon) \leq \varepsilon \) and \( |\text{vert}(P_\varepsilon)| \leq \alpha_0 / \varepsilon \), where \( \alpha_0 = 2 \sqrt{6} (\sqrt{6} - \sqrt{5}) \) (\( d_H(\cdot, \cdot) \) is the Hausdorff distance).

Note that \( \alpha_0 \leq 60.686 \). The following is a simple consequence of Proposition 4.3.

**Corollary 4.4.** For every convex body \( C \) in \( \mathbb{R}^3 \) which is contained in the Euclidean unit ball, and for \( n > \alpha_0 \), there exists a convex polytope \( P \) contained in \( C \), with at most \( n \) vertices, such that

\[
d_H(P, C) \leq \frac{\alpha_0}{n}.
\]

The rest of the computation goes exactly as in [17]. By the proof of Lemma 3.3 there, we get \( c_0 \leq 3\alpha_0 \sim 182.058 \) and by Theorem 3.2: \( c_1 \leq 9\alpha_0 \sim 546.174 \). This is also the value of \( a_1 \) (as follows from the proof of Theorem 5.1 of [17]).

From all the above it follows that \( k \) should be at least 182.058/\( \varepsilon \). If \( \varepsilon \) is taken close to 1/2 then \( k > 380 \) will do. The available bound on \( c_1 \) shows however, that in order to get a reasonable approximation \( k \) must be at least 1100. We nevertheless believe that the true values of \( c_0 \) and \( c_1 \) may be significantly smaller; one can, of course construct variants of the algorithms that will try to assume smaller values of \( c_0, c_1 \) and either proceed with them in case of success or, else, replace them with bigger values and try again.

Similar calculations, based again on Theorem 4.1, show for the outer approximation that \( a_2 \) is bounded by 1554.664, and that, at this moment, the outer approximation is guaranteed to work well for \( k > 3000 \).

**Acknowledgements**

Mario A. Lopez has been partially supported by NSF grants DMS-9626749 and DMS-0107628. Shlomo Reisner has been partially supported by NSF grants DMS-9626749 and DMS-0107628 and by NATO Collaborative Linkage Grant PST.CLG. 976356.
References