On the Complexity of Approximating and Illuminating Three-Dimensional Convex Polyhedra

(Preliminary Version)

GAUTAM DAS* Univ. of Memphis

MICHAEL T. GOODRICH† Johns Hopkins Univ.

Abstract

We show that several well-known computational geometry problems involving 3-dimensional convex polyhedra are NP-hard or NP-complete. One of the techniques we employ is a linear-time method for realizing a planar 3-connected triangulation as a convex polyhedron.

1 Introduction

Convex polyhedra are fundamental geometric structures (e.g., see [14]). Moreover, due to a beautiful theorem of Steinitz [14, 23], they provide a strong link between computational geometry and graph theory, for Steinitz shows that a graph forms the edge structure of a convex polyhedra if and only if it is planar and 3-connected.

Unfortunately, algorithmic problems dealing with 3-dimensional convex polyhedra seem to be much harder than their 2-dimensional counterparts. In addition, this difficulty goes beyond simple notions of running time; it also impacts our notions of efficiently-representable structures. For example, although the published proofs of Steinitz's theorem can easily be converted to algorithms running in $O(n^3)$ time in the real-RAM model, these algorithms produce polyhedra that may require an exponential number of bits to represent.

In this paper we formally establish that several natural problems on convex polyhedra are provably difficult, including several problems involving the approximation and illumination of convex polyhedra. Interestingly, a key ingredient in our proofs is a linear-time method for realizing any 3-connected planar triangulation as a convex polyhedron using a polynomial number of bits.

We describe this realization method in the section that follows. In the subsequent section we establish the NP-completeness of the problem of finding the minimum number of vertex lamps needed to illuminate a convex polyhedron, which is a problem studied by Grünbaum and O'Rourke and featured in O'Rourke's book on "art gallery" theorems [20], where they show that, for a convex polyhedron P with f faces in $\mathbb{R}^3$, $\lfloor (2f-4)/2 \rfloor$ vertices are sometimes necessary and always sufficient to see the exterior of P. Finally, in Section 4, we show that finding an optimal decision tree in $\mathbb{R}^3$ is NP-complete, as is the problem of finding a minimum-facet convex polyhedron lying between two polyhedra in $\mathbb{R}^3$ is NP-complete, which fixes a "gap" in a proof by Das and Joseph [6, 7].

*This research supported in part by NSF Grant CCR-9306822. Math Sciences Dept., Univ. of Memphis, Memphis, TN 38152. E-mail: dasg@next1.msci.memst.edu.

†This research supported by the NSF under Grants IRI-9116843 and CCR-9300079. Department of Computer Science, Johns Hopkins University, Baltimore, MD 21218, USA. E-mail: goodrich@cs.jhu.edu

2 Realizing 3-Connected Triangulations in $\mathbb{R}^3$

In this section we show how to realize a 3-connected planar triangulation as a convex polyhedron in linear time. Our algorithm constructs a polyhedron that can be represented using a polynomial number of bits in the rational-RAM model.

Theorem 2.1 *Given a n-vertex 3-connected planar triangulation $G = (V, E)$, one can realize G as a convex polyhedron P with a bit complexity that is polynomial in n. The running time is $O(n)$ in the rational-RAM model.*

Before we prove this theorem we present the following graph-theoretic lemma, which has been proven in various forms (e.g., see [10, 17]).

Lemma 2.2 *Given a n-vertex planar graph $G = (V, E)$, one can compute in linear time an independent set with at least $n/18$ vertices such that each vertex has degree ≤ 8.*

The overall idea of our algorithm is as follows. We compute a large independent set of G, and "compress" each vertex in this set with one of its neighbors along a common incident edge. We show that one can always choose a neighbor so that this results in a smaller planar triangulation that is still 3-connected; hence, we can recursively construct an equivalent polyhedron P' for the compressed graph G'. To construct P, we then "expand" the previously compressed edges appropriately so that convexity is maintained. Although this approach seems fairly straightforward, implementing it in $O(n)$ time is not so easy.

Since we compress a constant fraction of the vertices in each level, there are $O(\log n)$ levels of recursion. Our algorithm ensures that at each level the number of bits required to represent each added vertex is within a constant multiple of the number of bits required to represent a vertex of the previous level. Thus, the total bit complexity of representing P is polynomial in n.

We now give more details of our algorithm. Let the exterior face of the input triangulation contain the vertices u, v and w. At every level of the recursion, along with other properties, we will also ensure that u, v and w are on the xy-plane ($u = (0, 0, 0)$, $v = (2, 0, 0)$ and $w = (1, 2, 0)$), and the remaining vertices are above this plane, but strictly within the vertical "tube" whose horizontal cross section is congruent to the triangle uvw.

Case 1 ($n = 4$): Let the four vertices be u, v, w and t. In this case we construct a tetrahedron by positioning u at $(0, 0, 0)$, v at $(2, 0, 0)$, w at $(1, 2, 0)$), and t at $(1, 1, 1)$, which completes the construction.

Case 2 ($n > 4$): Using the method of Lemma 2.2, we compute a large independent set I of G. Let $I_1 = I \setminus \{u, v, w\}$, so that I_1 contains only interior vertices. Then, we repeat the following for each vertex s in I_1. Let s be incident to the vertices $s_1, s_2, \ldots s_l$, where $l \leq 8$. We choose one of the vertices s_j and compress the edge (s, s_j), removing any parallel edges this produces. We cannot choose just any vertex, however, for compressing s with some s_j's may violate 3-connectivity of the resulting planar graph. Consider the face $f = s_1 s_2 \ldots s_l$ that would result if we were to remove the edges incident to s, and mark the edges $(s_1, s_2), (s_2, s_3), \ldots, (s_l, s_1)$ as *peripheral* edges. The vertex s_j is selected

as follows. If there are no edges connecting two non-adjacent vertices of f, then any vertex of f may be selected, say s_1. If, on the other hand, there are indeed such "exterior" edges, then there has to be an edge (s_i, s_k) such that the closed region defined by (s_i, s_k) and the boundary of f does not further contain such exterior edges. Consider the relevant boundary of f between s_i and s_k. It has to contain at least one intermediate vertex, and we select this to be s_j.

Let the resultant graph after all the edge compressions are performed be G'. We recursively construct an equivalent polyhedron P' for this graph. We know that the vertices of P' other than u, v and w are strictly confined within a vertical tube with cross section congruent to uvw.

Recall that some edges of P' have been marked as peripheral. We compute for each peripheral edge e a plane $p(e)$ as follows. If $e \in \{(u,v),(v,w),(w,u)\}$, then $p(e)$ is the vertical plane tangential to P' at e, otherwise $p(e)$ is any plane tangential to P' at e. Let s' be some vertex of P' created by the compression of some s and s_j in G, and let f be the cycle of edges marked as peripheral by this compression. For each edge $e \in f$, consider the half-space defined by $p(e)$ which includes P'. The intersection of these halfspaces defines a "pyramid" over f. Next, consider the half-spaces not containing P corresponding to each face of P' within f. If we intersect these half-spaces with the above pyramid, the resulting region will be convex. We show in the full version that if we expand s' into s_j and s, where the point[1] s_j remains at s', and s is selected inside this convex region, then P (which is the convex hull of s and P') will be convex. We can find s so that its resulting bit complexity (using rational arithmetic) is at most a constant factor larger than the bit complexity needed to represent each vertex of f. Performing this edge expansion for each s' that resulted from an edge compression, then, completes the construction.

Implementing the compression algorithm

In this subsection we show how to implement a single recursive level in our edge-contraction algorithm in $O(n)$ time. Since the size of the graph decreases by a constant-factor with each recursive level, this will establish that the total running time of our drawing algorithm is $O(n)$.

The important step in our procedure is identifying, for a particular node s in our independent set I_1, an adjacent node s_j such that the edge (s, s_j) can be compressed without violating 3-connectivity. The crucial condition for this to be possible is that s and s_j cannot already be members of a separating triangle, for then merging them would create a separating pair (and the graph would no longer be 3-connected). As observed above, the set of adjacencies for s define a face f, whose edges we call the peripheral edges. Since the graph is triangulated, the crucial condition for s to to be mergeable with s_j is equivalent to the condition that s_j cannot be adjacent to another vertex of f through a non-peripheral edge (i.e., an edge external to f). We say that such an adjacency *disqualifies* the merge of s and s_j. It is not immediately clear, however, how we can efficiently test this condition for each candidate s_j around f during the compression step for s, since some of these s_j's may have a large number of adjacencies in the graph.

[1] We ask the readers indulgence into this abuse of notation so that s (resp., s_j) can denote a vertex in G and its corresponding point on P.

Our implementation is to break this computation into a batch component, which we perform in advance for all the s's in our independent set, and an on-line component, which we perform for each s in turn as we perform our edge compressions. Our batch computation is as follows:

1. We identify, for each s in I_1, and each vertex s_j adjacent to s, all the candidate adjacencies that would disqualify our being able to merge s and s_j. There are $d(s)(d(s)-2) = O(1)$ such adjacencies for each s in I_1, where $d(s)$ denotes the degree of s; hence, the total number of all such candidate adjacencies is $O(n)$. We label each such candidate adjacency between s_j and some s_i on f as (s_i, s_j, s) meaning "adjacency (s_i, s_j) would disqualify the merging of s_j and s."

2. We then radix sort into a list L all the labels computed in the previous step together with all the existing adjacencies in G, lexicographically. This takes $O(n)$ time.

3. For any match of a real adjacency (s_i, s_j) with a candidate disqualifying adjacency (s_i, s_j, s) we mark the edge (s_j, s) as "disqualified." We remove all the (s_i, s_j) and (s_i, s_j, s) labels from the sorted list L for each such match. This step also takes $O(n)$ time.

4. Finally, we group together in one list $L_{i,j}$ each sublist of the sorted list L that identify the same candidate disqualifying adjacency (s_i, s_j) (for several different s's in our independent set). We store a pointer to the list $L_{i,j}$ in the records of each s in I_1 that contributes an element to $L_{i,j}$. The total number of such fields is $O(1)$ for any such s and the total space needed for all the $L_{i,j}$'s is clearly $O(n)$.

The meaning for each list $L_{i,j}$ is that this is a disqualifying adjacency that currently does not exist in G, but may exist at some point during the compression phase. Thus, for the compression computation for a node s in I_1, we choose an edge (s_j, s) that is not marked "disqualified" and compress it. For each new adjacency (s_i, s_j) this creates, we consult the list $L_{i,j}$ (if it exists), and for each (s_i, s_j, s') label in $L_{i,j}$ (with $s' \neq s$) we mark the edge (s_j, s') as "disqualified." We then discard the list $L_{i,j}$.

We have already argued why there will always be some edge incident upon s that is not marked "disqualified;" hence, the above computation can always proceed to the next s in I_1. The total time needed is $O(n)$ for the preprocessing step, and then an additional $O(n)$ time during the compression step (for once an $L_{i,j}$ list is consulted it is then discarded). Therefore, we can complete a recursive step in our 3-d drawing algorithm in $O(n)$ time, as claimed.

Since we can perform each level in the recursion in $O(n)$ time, by Lemma 2.2, this results in a linear-time algorithm for drawing G as a convex polyhedron. Moreover, the fact that there are only $O(\log n)$ levels in this recursion implies that our method produces a polyhedron that can be represented using a polynomial number of bits (using rational arithmetic).

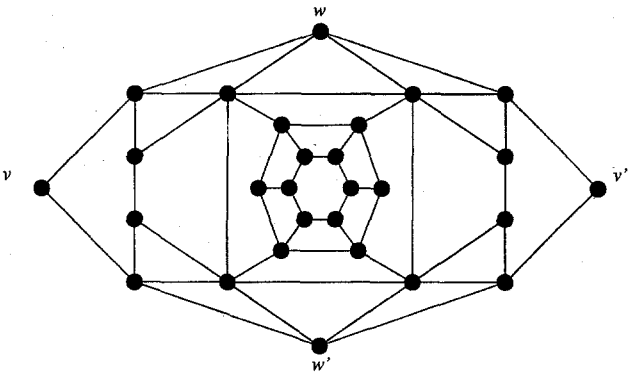

Figure 1: The cross-over gadget.

3 Polytope Illumination

In this section we prove that polyhedron illumination is NP-complete. We begin by showing that the well-known vertex cover problem remains NP-complete even for 3-connected planar graphs, and show how this can be used to further extend the NP-completeness of vertex covering to convex polyhedra in $\mathbb{R}^3$. This extends the previous results of Garey and Johnson [11] and Garey, Johnson, and Stockmeyer [13], which showed that the vertex cover problem remains NP-complete for planar graphs with degree at most three.

Vertex Cover for 3-connected planar graphs

Our reduction is actually a chain of reductions, starting from the (standard) vertex cover problem. So, let $G = (V, E)$ and k be the graph and integer parameter defining an instance of the vertex cover problem. Without loss of generality, we can assume that $|V| \geq 4$. We begin our chain of transformations by augmenting G by adding three new vertices v_1, v_2, and v_3 that we define to be adjacent to all the vertices in G. Clearly, the resulting graph G' is 3-connected.

Claim 3.1 *G has a vertex cover of size $k < n$ if and only if G' has a vertex cover of size $k + 3$.*

Proof: Omitted in this preliminary version. ■

Thus, the vertex cover problem remains NP-complete for 3-connected graphs. So, let us now use G and k to together denote an instance of vertex cover with G being 3-connected. We will reduce this version of vertex cover to the version of the problem where the graph is 3-connected and *planar*. Our reduction is an adaptation of the proof of Garey *et al.* [13], who give a reduction from general graphs to planar graphs that does not preserve 3-connectivity. We begin by drawing G in the plane so as to have $c = O(n^2)$ edge crossings (e.g., using a simple straight-line strategy). We replace each edge crossing by the "gadget" illustrated in Figure 1 as illustrated in Figure 2. Performing all these replacements results in a 3-connected planar graph G'.

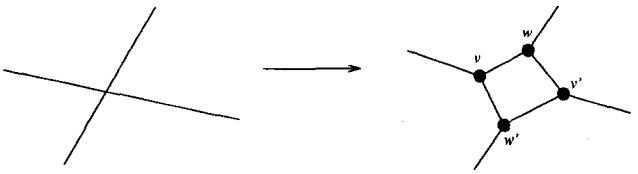

Figure 2: The way the cross-over gadget replaces an edge crossing.

Claim 3.2 *G has a vertex cover of size k if and only if G' has a vertex cover of size $16c + k$.*

Proof: A close inspection of the gadget we use to replace each edge crossing shows that the edges of the gadget can be covered with 16 nodes only if we include at most one member of $\{v, v'\}$ and at most one member of $\{w, w'\}$. Thus, if there is a vertex cover of size k in G, we can create a vertex cover of size $16c + k$ by including the 16 nodes in each crossover gadget so as to also cover each of the edges joining crossover gadgets (and original vertices of G). Suppose, conversely, that G' has a vertex cover of size $16c + k$. As we have already observed, each cross-over gadget can be covered with 16 nodes only if at most 16 nodes only if we include at most one member of $\{v, v'\}$ and at most one member of $\{w, w'\}$. That is, covering each gadget with 16 nodes establishes a "parity" along any chain of gadgets derived from a single edge in G. Thus, by a counting argument, which is similar to one given by Garey *et al.* [13], we can conclude that G must have a vertex cover of size k. ■

Therefore, the vertex cover problem remains NP-complete for 3-connected planar graphs. We can further restrict our graphs, however, and the problem still remains NP-complete.

Polytope Vertex Cover

Given an embedded 3-connected planar graph G, define the *stellation* of a face f in G as the insertion of a vertex in the interior of f that we then make adjacent to each vertex on f. Moreover, if f is a triangle, then we also allow any of edges of f to be subsequently removed, so long as we still preserve the 3-connectivity of G. (See Figure 3.) Define a *stellation* of the entire graph G to be the result of performing a collection of independent, non-interfering face stellations on a subset of the faces of G. Further define the *t-stellation* of G to be the result of performing t consecutive stellations on G.

An interesting property of stellations is that they have a natural analogue with respect to convex polyhedra. In particular, if a 3-connected planar graph G is represented as a convex polyhedron in $\mathbb{R}^3$, then the stellation of a face f of G can be accomplished geometrically by introducing a point p "above" f so that the convex full of p unioned with P results in the updated graph G'. Indeed, the proof of Steinitz's theorem (e.g., see [14]), showing that a graph can be drawn as a convex polyhedron in $\mathbb{R}^3$ if and only if it is 3-connected and planar, is essentially equivalent to showing that any 3-connected planar graph (or polyhedron) can be constructed from a planar embedding of K_4 (or tetrahedron) in a series of

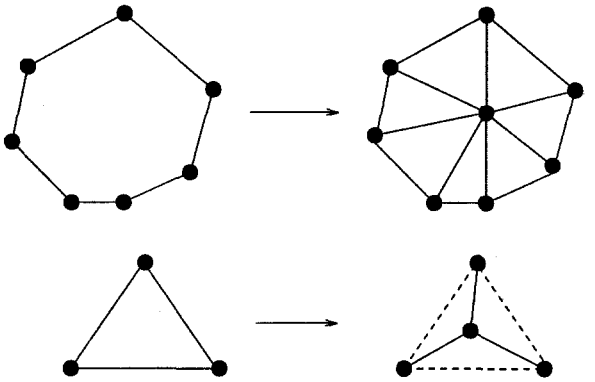

Figure 3: The stellation of a face.

$O(n^3)$ stellations or "inverse" stellations. We show that the vertex cover problem remains NP-complete for c-stellations of 3-connected 3-regular planar graphs, for any constant $c \geq 4$.

We have shown, in Section 2, that any 3-connected planar triangulation can be drawn as a convex polyhedron in $\mathbb{R}^3$ using a polynomial number of bits. By a simple duality argument, this immediately implies that any 3-connected 3-regular planar graph can also be drawn as a convex polyhedron in $\mathbb{R}^3$ using a polynomial number of bits. Since performing a stellation of a convex polyhedron in $\mathbb{R}^3$ will increase the bit complexity of its representation by at most a constant factor, this also implies that the c-stellations of a 3-connected 3-regular planar graph can be drawn as a convex polyhedron in $\mathbb{R}^3$ using a polynomial number of bits if c is a constant. Thus, by showing that vertex cover remains NP-complete for c-stellations of 3-connected 3-regular planar graphs we will establish the NP-completeness of the Polytope Vertex Cover problem, where we are given a convex polyhedron P and an integer k and asked if there is a subset V of the vertices on P such that each edge on P has at least one end in V.

Our reduction will be from the vertex cover problem for 3-connected planar graphs. So, let G be a 3-connected planar graph and let k be a given integer parameter. Our reduction is a modification of an argument of Garey and Johnson [11], who showed that vertex cover remains NP-complete for planar graphs with degree at most 3. For each vertex v in G, we replace v by a cycle C_v of size $d(v)$, where $d(v)$ denotes the degree of v, so that each vertex on C_v retains exactly one adjacency of v. Clearly, the graph that results from this transformation will be a 3-connected 3-regular graph. We stellate each face defined by the interior of a C_v by introducing a new vertex v' in its interior. We furthermore stellate each triangle T incident on v' so as to eliminate all the edges of T. (See Figure 4.) The resulting graph, G', is a c-stellation of a 3-connected 3-regular graph (the last step can be accomplished by first stellating the odd-numbered triangles around v' and then doing the even-numbered ones, with possibly one more to do after that if the number of triangles is odd).

Claim 3.3 *$G = (V, E)$ has a vertex cover of size k if and only if G' has a vertex cover of size $k + 2|E|$.*

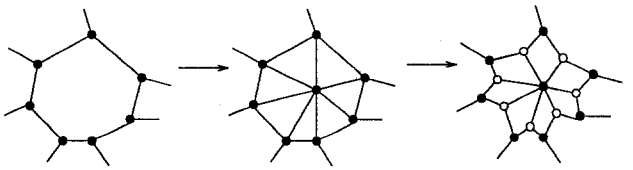

Figure 4: The stellations forming the subgraph of G' associated with a face in G.

Proof: Suppose G has a vertex cover of size k. For any v in G in this cover, we can put in a cover for G' all the vertices in the cycle C_v we created for v, together with the interior vertex v' (these vertices are shown in black in Figure 4). If v is not in the cover for G, then we can cover the subgraph of G' associated with v by using the vertices introduced in the stellation of each triangle incident on v' (these vertices are shown in white in Figure 4). The set of all such vertices will clearly form a cover of G'. We use $d(v)+1$ vertices for each vertex v in the cover for G and $d(v)$ vertices for each vertex v not in the cover, where $d(v)$ denotes the degree of v; hence, the total size of this cover is $k + \sum_{v \in G} d(v) = k + 2|E|$.

Conversely, suppose G' has a vertex cover of size $k + 2|E|$. The subgraph in G' determined by a vertex v in G can be covered with $d(v)$ vertices (using the nodes colored white in Figure 4), and $d(v)$ nodes are necessary. To cover an edge of G' outside of such a subgraph (i.e., an edge corresponding to an edge of G), however, requires that we use a vertex from some C_v (i.e., a black vertex). But if such a vertex is included in a cover for the subgraph of G' corresponding to a vertex v, then covering this subgraph now requires $d(v) + 1$ vertices. But we can cover such a subgraph using $d(v) + 1$ vertices using only the vertices of C_v and the new vertex v' (the black vertices). We can thus define a cover of G by including each vertex v whose corresponding subgraph in G has at least $d(v) + 1$ vertices and this cover will have size at most k in G. ■

As we mentioned above, given the result of Section 2 regarding drawing 3-connected 3-regular planar graphs as convex polyhedra, Claim 3.3 immediately applies to the Polytope Vertex Cover problem.

Theorem 3.4 *The Polytope Vertex Cover problem is NP-complete.*

Polytope Lamp Cover

We are now ready to prove our result regarding lamp placement on convex polyhedra. Specifically, in this problem we are given a convex polyhedron P in $\mathbb{R}^3$ and an integer k and asked if there are k vertices on P such that each point on the boundary of P can be connected to a vertex in this set by a line segment that does not intersect the interior of P. We show that deciding if a given k number of vertices suffice for P is NP-complete. Our proof is based upon a reduction from Polytope Vertex Cover.

So, suppose we are given a polyhedron Q and an integer k such that we would like to know if there is a k-node vertex cover on Q. Our reduction is to form a c-stellation of Q, where, for each face f on Q, we form a vertex F in its interior and form triangles with the nodes on f. We then perform two more stellations,

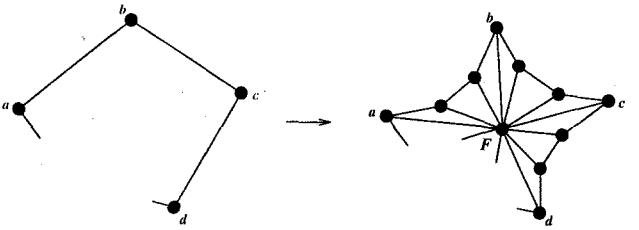

Figure 5: An example stellation used in forming P.

so as to form for each triangle $\triangle abF$ incident on F, three consecutive triangles $\triangle axF$, $\triangle xyF$, and ybF. Call this transformed polyhedron, P. (See Figure 5.)

Claim 3.5 *Q has a vertex cover of size k if and only if P has a lamp cover of size $k + \mathcal{F}$, where $\mathcal{F}$ is the number of faces on Q.*

Proof: Suppose Q has a vertex cover of size k. We can form a lamp cover for P by including each vertex in the cover for Q together with each vertex F created in the stellation of a face f of Q. This lamp cover will have size $k + \mathcal{F}$.

Conversely, suppose P has a lamp cover of size $k + \mathcal{F}$. Consider the subgraph in P associated with any face f from Q. If F is not included in a lamp cover, then illuminating this portion of P requires at least $\lceil 3e/2 \rceil$ vertices, where $e \geq 3$ is the number of edges on f. But, by including F in a lamp, we can illuminate this portion of P using just one vertex! The only other faces that are not illuminated are faces that correspond to edges of Q (that no stellation vertex F can see). Since we can assume without loss of generality that the lamp cover for P includes each stellation vertex F, we can further assume that each other vertex in the lamp cover is also a vertex in Q (for if this were not the case, we can substitute such a vertex (labeled x or y above) with a vertex that is also in Q and illuminate more faces of P). Thus, taking the vertices in the lamp cover for P that are also vertices in Q forms a vertex cover for Q of size k. ■

This immediately implies the following:

Theorem 3.6 *The Polytope Lamp Cover problem is NP-complete.*

4 Approximating Convex Polyhedra

In this section we consider the problem of *polyhedral approximations*, where one wishes to construct a polyhedron Q from a given polyhedron P, such that Q is simpler than P and may be used to replace P. This is a very loose definition, and the exact requirements which Q should satisfy of course depends upon the particular problem. For example, there have been many results describing and analyzing various approximation schemes (e.g., see [2, 1, 6, 7, 8, 19]). We are in particular concerned with the *combinatorial* simplicity of the approximate object, e.g., it should not have too many faces.

Approximation by a Decision Tree

The first approximation problem we consider is that of defining an efficient decision tree that can be used as a discriminator for a given polyhedron (indeed, we define the problem for any point set in $\mathbb{R}^3$). It is well known, for example, that constructing a best decision tree in general settings [15, 16] or in arbitrary dimensions [3, 18] is NP-complete, but in the context of fixed-dimensional decision-tree approximations, however, each of these NP-completeness proofs fail. Still,

Theorem 4.1 *Given a set S of n points in $\mathbb{R}^3$, divided into two concept classes "red" and "blue," deciding if is there a linear decision tree T with at most k nodes that separates the red points from the blue points is NP-complete.*

Proof: First, let us observe that this problem is in NP. This is because each candidate split in a linear decision tree is determined by 3 points, hence, there are $\Theta(n^3)$ candidate splits. We can therefore guess k splits and a tree structure with one of these splits at each node, and we can then test that this decision tree separates all the red and blue points.

To prove the problem NP-hard we reduce POLYTOPE VERTEX-COVER to it. For the sake of simplicity, let us allow as input point sets where red points and blue points can "overlap". A complete classification of such a pair of points must therefore have a split that passes through this common location in space. Our reduction is based upon judiciously placing such pairs of points on the edges of Q, the Poincaré dual to P, i.e., Q is a convex polyhedron whose 1-skeleton is the graph-theoretic planar dual to the 1-skeleton of P. Thus, a *face cover* in Q corresponds immediately to a vertex cover in P. We place two red-blue pairs along each edge of Q so that the only way four such pairs can be co-planar is if they all lie on the same face of Q. Let S denote this set of red and blue points. Note that, since Q is a convex polyhedron, each face of Q contains at least six pairs of points in S.

We claim there is a k-node decision tree for S if and only if there is a k-face face-cover for Q. First, note that if there is a k-face face-cover for Q, then there must be k planes that collectively contain all the pairs in S; hence, there is a k-node decision tree for S. For the more difficult direction, suppose there is no k-face face-cover for Q; that is, any face cover requires more than k faces. This implies that any decision tree restricted to splits containing faces of P must have more than k nodes. Note, however, that each such split contains at least six pairs of points in S whereas any other type of split contains at most three pairs of points in S. Therefore, since each pair of points in S must be contained in some split, there must be more then k nodes in any decision tree that completely separates the pairs in S. ■

Polyhedral Separability

Another well-known instance of polyhedral approximation we consider is *polyhedral separability.* In the simplest sense, given two polyhedral objects, we ask whether they are separate in space or whether they intersect. If they are indeed separate, we may want to construct a *separator* between them, such as a hyperplane, or a surface with minimum number of faces. Polyhedral separability has been the subject of extensive research, e.g., see [1, 4, 9, 19, 21].

The version we address is is that one is given two concentric polyhedra, and one wishes to find a separating polyhedra with minimum faces nested between the two. The nested polyhedral separability problem that we consider was first raised by Victor Klee [21], which arose during the study of *sequential stochastic automata* [22]. In two dimensions, this problem has been solved in polynomial time by Aggarwal et al [2]. Das and Joseph [6, 7] proved the interesting result that the problem is NP-hard for three dimensions, *even for convex polyhedra.* While the combinatorial and logical aspects of the proof are correct, a certain geometric part has a flaw. More precisely, a part of the reduction requires constructing a convex polyhedron from a 3-connected planar triangulation, and the version presented in [6] does not run in polynomial time. As we show below, Theorem 2.1 can be used to correct this flaw in the proof.

Since the results in [6, 7] appeared, there have been several efforts to design good approximation algorithms for the problem. In particular, Mitchell and Suri designed an efficient algorithm in [19] which achieves an approximation ratio of $O(\log n)$. This bound was matched by a simple randomized scheme of Clarkson [5], and extended to terrains by Agarwal and Suri [1]. More recently, Brönnimann and Goodrich [4] show how to achieve an approximation ratio of $O(1)$ for the convex polyhedral case.

Theorem 4.2 *The problem of fitting a polyhedron with minimum faces between two concentric convex polyhedra is NP-hard.*

Proof: The original NP-hardness proof in [6, 7] is a reduction from Planar-3SAT [12]. The first part of the reduction is graph-theoretic, where an instance of Planar-3SAT is used to construct a planar triangulation G where, (1) the exterior face may not be a triangle, (2) the interior faces are marked either *fixed* or *removable.* We modify this slightly by enclosing G in a triangle, and triangulating the region between G and the enclosing triangle such that the resulting graph G' is a 3-connected planar triangulation. Let the new faces be marked *fixed.*

The next step is geometric, where two polyhedra P' and Q are constructed such that Q is inside P'. Instead of following the procedure in [6], we use the polynomial-time procedure in Theorem 2.1 to construct Q from G'. Thus Q can be expressed in a polynomial number of bits in the rational-RAM model.

We can retain the old procedure to construct P'. Recall that a *pyramid* of a face f of Q is defined to the region above f and below the intersection of the three faces adjacent to f. We construct an intermediate P, defined to be the (non-convex) polyhedral region of the union of Q with all the pyramids of the removable faces of Q. This takes polynomial time, since each pyramid only requires the computation of intersections of planes. Next, P' is constructed to be the convex hull of P, which can be performed in polynomial time. ■

Acknowledgements

We would like to thank Marek Chrobak for his encouragement and his diligence in pointing out places in previous versions of this paper that needed more details. The second author would also like to thank the U.S. Army Research Office for their support. The views, opinions, and/or findings contained in this document are those of the authors and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

References

[1] P. K. Agarwal and S. Suri. Surface approximation and geometric partitions. In *Proc. Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1994.

[2] A. Aggarwal, H. Booth, J. O'Rourke, S. Suri, and C. K. Yap. Finding minimal convex nested polygons. *Inform. Comput.*, 83(1):98–110, October 1989.

[3] A. Blum and R. L. Rivest. Training a 3-node neural net is NP-Complete. *Neural Networks*, 5:117–127, 1992.

[4] H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite VC-dimension. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 293–302, 1994.

[5] K. L. Clarkson. Algorithms for polytope covering and approximation. In *Proc. 3rd Workshop Algorithms Data Struct.*, volume 709 of *Lecture Notes in Computer Science*, pages 246–252, 1993.

[6] G. Das. *Approximation schemes in computational geometry.* Ph.D. thesis, University of Wisconsin, 1990.

[7] G. Das and D. Joseph. The complexity of minimum convex nested polyhedra. In *Proc. 2nd Canad. Conf. Comput. Geom.*, pages 296–301, 1990.

[8] G. Das and D. Joseph. Minimum vertex hulls for polyhedral domains. *Theoret. Comput. Sci.*, 103:107–135, 1992.

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

[10] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science.* Springer-Verlag, Heidelberg, West Germany, 1987.

[11] M. R. Garey and D. S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM J. Appl. Math.*, 32:826–834, 1977.

[12] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, New York, NY, 1979.

[13] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.

[14] B. Grünbaum. *Convex Polytopes.* Wiley, New York, NY, 1967.

[15] L. Hyafil and R. L. Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5:15–17, 1976.

[16] S. Judd. On the complexity of loading shallow neural networks. *J. of Complexity*, 4:177–192, 1988.

[17] D. G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM J. Comput.*, 12:28–35, 1983.

[18] N. Megiddo. On the complexity of polyhedral separability. *Discrete Comput. Geom.*, 3:325–337, 1988.

[19] J. S. B. Mitchell and S. Suri. Separation and approximation of polyhedral surfaces. In *Proc. 3rd ACM-SIAM Sympos. Discrete Algorithms*, pages 296–306, 1992.

[20] J. O'Rourke. *Art Gallery Theorems and Algorithms.* Oxford University Press, New York, NY, 1987.

[21] J. O'Rourke. Computational geometry column 4. *SIGACT News*, 19(2):22–24, 1988. Also in Computer Graphics 22(1988), 111–112.

[22] C. B. Silio, Jr. An efficient simplex coverability algorithm in E^2 with applications to stochastic sequential machines. *IEEE Trans. Comput.*, C-28:109–120, 1979.

[23] E. Steinitz and H. Rademacher. *Vorlesungen über die Theorie der Polyeder.* Julius Springer, Berlin, Germany, 1934.