Convex covers in higher dimensions

Patrice Belleville

School of Computing Science
Simon Fraser University
Burnaby, B.C., Canada, V5A 1S6

Abstract

A *convex cover* of a set S in d-dimensional euclidean space is a finite collection of convex sets whose union equals S. The *convex k-cover problem* is the problem of determining if a given set admits a convex cover with at most k pieces. In the plane, the convex k-cover problem can be solved in linear time for each $k \leq 3$; its complexity remains open for fixed values of $k \geq 4$, and it is NP-hard when k is part of the input. In this paper, we show that the convex 2-cover problem for simple polyhedra in three dimensions can be solved in $O(n \log n)$ time, but that the convex 3-cover problem is NP-hard for star-shaped polyhedra in three or more dimensions.

1 Introduction

Let E^d denote the d-dimensional euclidean space. A point x of a subset S of E^d is said to be *visible* from a point y of S, or equivalently x *sees* y, if the line segment $\overline{xy}$ is contained in S. Problems related to visibility have been studied extensively in computational geometry; they have practical applications in several domains, such as graphics, pattern recognition, robotics and motion planning, etc. One type of visibility problem that has received a lot of attention is the Π *k-cover* problem:

> Given a set S, a property Π, and an integer k, are there k sets with property Π whose union equals S?

The first covering problem for simple polygons was posed in 1973 by Victor Klee: "What is the minimum number of guards required to cover any art gallery of n vertices?" An art gallery can be modeled as a simple polygon, and each guard as a star-shaped subset of the polygon (a polygon P is *star-shaped* if it contains a point from which every other point of P is visible; the set of all such points is called the *kernel* of P and denoted by $Kr(P)$). The answer to this question, first proved by Chvátal [Chv75], is that $\lfloor n/3 \rfloor$ guards are always sufficient and sometimes necessary. The question has become known as the *Art Gallery Problem*, and the answer as the *Art Gallery Theorem*. Since then, a large number of variations on the art gallery problem have been considered [O'R87, She92].

Unfortunately, in most cases finding minimum size covers of simple polygons with polygons that have property Π is NP-hard. Culberson and Reckhow [CR88], and independently Shermer [She89], proved that this also holds when Π is the convexity property.

These negative results do not, however, preclude the existence of polynomial time algorithms for fixed values of k. In the case where Π is the convexity property, the convex 1-cover problem (determining whether a polygon is convex) can be solved using a trivial linear time algorithm. Shermer [She93] gave a linear time algorithm to solve the convex 2-cover problem for simple polygons, and Belleville [Bel95] gave a linear time algorithm to solve the convex 3-cover problem. The cases where $k \geq 4$ remain open.

These algorithms give insight into the complexity of the convex cover problem, and help us understand why this problem is difficult. This knowledge will in turn allow us to design approximation algorithms that are as efficient as possible. Since solutions to the convex cover problem with guaranteed error bounds are needed in practice, it is thus highly desirable to study the convex k-cover problem for small values of k.

In this paper, we consider the complexity of the convex k-cover problem in higher dimensions, for fixed values of k. The convex 2-cover problem is solvable in $O(n \log n)$ time in three dimensions; this algorithm is described in Section 2. On the other hand, the convex 3-cover problem for star-shaped polyhedra in three (or higher) dimensional space is NP-hard; this is shown in Section 3. Finally we present conclusions and future research directions in Section 4.

2 Convex 2-cover

In this section, we outline an $O(n \log n)$ time and $O(n)$ space algorithm to solve the convex 2-cover problem in three dimensional euclidean space. This algorithm proceeds by partitioning the input polyhedron P, and then defining on the components of this partition a graph G that admits a proper 2-coloring if and only if P admits a convex 2-cover.

First consider an arbitrary convex 2-cover $\mathcal{C} = \{C_1, C_2\}$ of a simple polyhedron P. Let $C_1' = CH(C_1 \cup Kr(P))$ and $C_2' = CH(C_2 \cup Kr(P))$. Because every point of $Kr(P)$ sees every other point of P, the set $\{C_1', C_2'\}$ is a convex 2-cover of P such that $C_1' \cap C_2' = Kr(P)$. Hence it suffices to decide whether P admits a convex 2-cover in which the intersection of the two cover elements is $Kr(P)$. This can be done by determining how $P \setminus Kr(P)$ is partitioned between the cover elements. The following property simplifies this decision.

Property 2.1 *Let P be a simple polyhedron, and Q be a connected component of $P \setminus Kr(P)$. In every convex 2-cover of P, there is a cover element that contains Q, and no point of Q belongs to the other cover element.*

Let $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$ be the set of connected components of $P \setminus Kr(P)$. It suffices to determine which elements of $\mathcal{Q}$ belong to each cover element. This can be done by constructing a graph G whose vertices represent the elements of $\mathcal{Q}$. An edge joins two vertices of G if the corresponding elements Q_i, Q_j of $\mathcal{Q}$ are not completely visible, i.e. there is a point of Q_i and a point of Q_j that do not see each other. We note that every two points of a given element Q_i of $\mathcal{Q}$ must be visible, but that Q_i need not be convex (for instance, let the polyhedron P be the planet Saturn, and Q_i be the rings of Saturn). We will call Q_i *pseudo-convex* if every two points of Q_i are visible.

Property 2.2 *If $\mathcal{Q}'$ is a subset of $\mathcal{Q}$ whose elements are individually pseudo-convex and pairwise completely visible, then $Kr(P) \cup (\cup_{Q_i \in \mathcal{Q}'} Q_i)$ is a convex subset of P.*

Property 2.2 implies that every proper 2-coloring of G yields a unique convex 2-cover of P in which the intersection of the two cover elements is $Kr(P)$, and that each convex 2-cover of P with this property corresponds to exactly one proper 2-coloring of G. The algorithm thus starts by computing $Kr(P)$, and the set $\mathcal{Q}$. It then computes G from $\mathcal{Q}$, and attempts to two-color G. If G does not admit a proper 2-coloring, then the algorithm aborts. Otherwise, it computes the cover of P induced by a proper 2-coloring of G.

Let $N_\epsilon(x)$ denote the set of all points of E^3 whose distance to a point x is less than or equal to ϵ. A point of *local non-convexity* of P is a point x of $bd(P)$ such that, for every $\epsilon > 0$, there are points y and z of $P \cap N_\epsilon(x)$ that are not visible. To compute G, we use the following fact:

Property 2.3 *If P admits a convex 2-cover, and two connected components Q_i, Q_j of $P \setminus Kr(P)$ are not completely visible, then there is a point x of $bd(P)$ such that for every $\epsilon > 0$, there is a point x_i of $Q_i \cap N_\epsilon(x)$ that does not see a point x_j of $Q_j \cap N_\epsilon(x)$.*

Hence every pair of vertices of G joined by an edge in G correspond to elements of $\mathcal{Q}$ that are adjacent along an edge of P, or incident upon a same vertex of P. Only $O(n)$ edges join elements of $\mathcal{Q}$ that share an edge of P, but there may be $\Omega(n^2)$ edges that join elements of $\mathcal{Q}$ sharing a vertex. We can however show that G has the following property:

Property 2.4 *Let G^* be the subgraph of G induced by the k vertices corresponding to k elements of $\mathcal{Q}$ with a common vertex of P. If P admits a convex 2-cover, then $k-2$ vertices of G^* are incident upon at most 4 edges of G^* each.*

Hence if P admits a convex 2-cover, then G contains only $O(n)$ edges. We can show how to determine these edges in $O(n)$ time. Hence, instead of computing G, we obtain a graph G' using this procedure and compute the cover of P corresponding to G'. Finally we verify that each cover element is convex. If G' does not admit a proper 2-coloring, or if the cover we obtain is not a convex 2-cover of P, then it is because P does not admit a convex 2-cover. This implies that the convex 2-cover problem in E^3 can be solved in $O(n \log n)$ time.

3 Convex 3-cover

In this section, we prove that the convex 3-cover problem is NP-hard in three dimensions. The formal specification of this problem as a decision problem is the following.

3D convex 3-cover (3D3C)

Input: A simple polyhedron P in E^3.

Output: *Yes* if P admits a convex 3-cover, *no* otherwise.

The reduction will be done from boolean 3-satisfiability (3SAT), which is formally defined as follows.

3-satisfiability (3SAT)

Input: A set C of 3-element clauses over a set U of boolean variables.

Output: *Yes* if C admits a satisfying truth assignment, *no* otherwise.

However, rather than directly reducing 3SAT to 3D3C, we will go through an intermediate problem, which we call *Delaunay subgraph 3-colorability* (DS3C). The advantage of this two-step process is that each reduction then becomes reasonably intuitive. The remainder of this section proceeds in three stages: in Section 3.1, we define the terms used thereafter and formally specify the problem DS3C. In Section 3.2, we show that DS3C is NP-complete using a reduction from 3SAT. Finally, we reduce DS3C to 3D3C in Section 3.3.

3.1 Introduction

Given a set V of points in the plane, a *triangulation* of V is a maximal set E of edges such that (V, E) is a plane graph. We note that each face of the triangulation except for the outer face is a triangle. The *Delaunay triangulation* of V is the triangulation of V in which every triangular face $\triangle vv'v''$ has the property that the interior of the circle through v, v' and v'' does not contain any point of V. A plane graph $G = (V, E)$ will be called a *Delaunay subgraph* if E is a subset of the Delaunay triangulation of V. DS3C can now be formally defined as follows.

Delaunay subgraph 3-col. (DS3C)

Input: A Delaunay subgraph G.

Output: *Yes* if G admits a proper 3-coloring, *no* otherwise.

We will also require some definitions related to euclidean geometry in three or more dimensions. A d-dimensional polyhedron can be defined recursively in term of polyhedra of lower dimension. A zero-dimensional polyhedron is a point. A one-dimensional polyhedron is a line segment. For $d \geq 2$, a d-dimensional polyhedron P is a finite set $\mathcal{F} = \{F_1, \ldots, F_n\}$ of $(d-1)$-dimensional polyhedra (called the *facets* of P) with the following property:

> Every facet of an element of $\mathcal{F}$ is a facet of exactly one other element of $\mathcal{F}$. (*)

The polyhedron P is *simply-connected* if no strict subset of $\mathcal{F}$ possesses property (*). P is *simple* if it is simply-connected, if no pair of non-adjacent facets shares a point, and if the intersection of every pair of adjacent facets is their common subfacet. Simple polyhedra have well-defined interior and exterior.

Given a simple d-dimensional polyhedron P, the *visibility graph* of a subset S of P is the graph whose nodes correspond to the elements of S, and in which two nodes are joined by an edge if and only if the corresponding points are visible.

3.2 Reducing 3SAT to DS3C

The construction of the graph used in the instance of DS3C uses several components taken from various sources and illustrated in Figure 1. From a functional point of view, subgraph G_{copy} is used to *copy* the color of a vertex, subgraph G_{merge} *merges* the colors of two vertices, and subgraph G_{exch} *exchanges* the colors of two vertices. G_{exch} was suggested by Fischer and used by Garey and Johnson to prove that planar graph 3-colorability is NP-complete [GJ79]. Finally, subgraph G_{clause} represents the clauses of the 3SAT instance; it can be found in the book by Cormen et al. [CLR89] (exercise 36-2). More formally these components have the following properties:

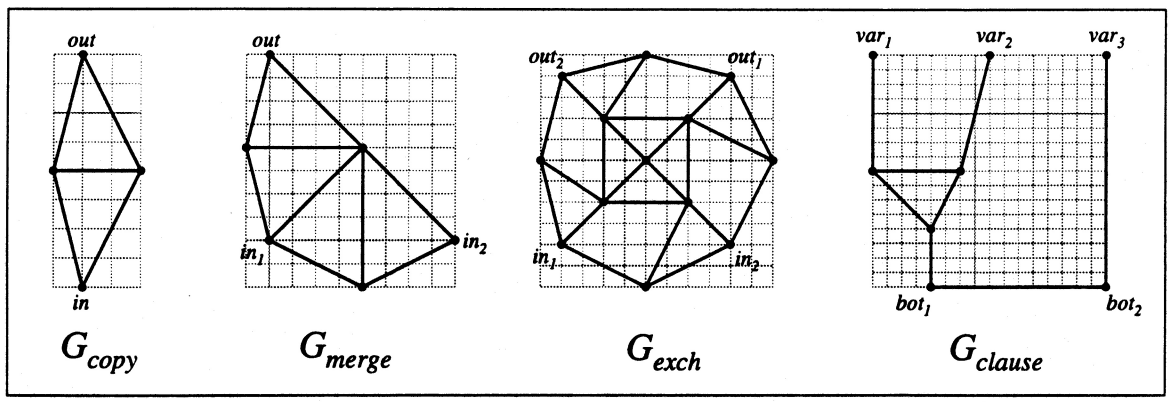

Figure 1: Illustrating the graphs G_{copy}, G_{merge}, G_{exch} and G_{clause}.

1. In every proper 3-coloring χ of G_{copy}, we have $\chi(in) = \chi(out)$.
2. Every proper 3-coloring χ of G_{merge} is such that $\chi(in_1) = \chi(in_2) = \chi(out)$.
3. Every proper 3-coloring χ of G_{exch} is such that $\chi(in_1) = \chi(out_1)$ and $\chi(in_2) = \chi(out_2)$. Also, there are proper 3-colorings χ_1, χ_2 of G_{exch} such that $\chi_1(in_1) = \chi_1(in_2)$, and $\chi_2(in_1) \neq \chi_2(in_2)$.
4. Suppose that $\chi(var_i) \in \{1,2\}$ for $i \in \{1,2,3\}$ and $\chi^*(bot_j) \neq 1$ for $j \in \{1,2\}$. We can extend χ to a proper 3-coloring of G_{clause} if and only if there is $i \in \{1,2,3\}$ such that $\chi(var_i) = 1$.

The main idea for the construction was inspired from the construction described in exercise 36–2 of the book by Cormen et al. [CLR89] and adapted to produce a subgraph of a Delaunay triangulation. The construction proceeds in several stages: a *clause* stage, a *sorting* stage, a *merging* stage, and finally a *variable* stage. These stages are illustrated in Figure 2.

During the clause stage, a vertex called *true* is first placed at the origin. Next, one copy of G_{clause} is produced for each clause in the instance of 3SAT. These subgraphs are placed from left to right in the order in which the clauses appear in the description of the instance of 3SAT, and edges are added from *true* to the vertices labeled bot_1 and bot_2. The vertices named var_1, var_2 and var_3 are labeled using the corresponding literal in the instance of 3SAT.

The sorting stage orders the labels of the vertices so that identical labels appear consecutively, and so that literals corresponding to the same variable also appear consecutively. In each step, a literal is either copied using G_{copy}, or exchanged with a neighbor using G_{exch}. This stage mimics the even-odd transposition parallel sorting algorithm used for linear arrays of processors [Akl85].

The merging stage removes duplicates from the list of literals. At each step, every sequence of two or more vertices with the same label is compressed by merging the two rightmost such vertices using G_{merge}. Once again, literals that do not participate in a merge are copied using G_{copy}.

The variable stage first places the vertices called *false* and *ignored*. It then adds an edge from each of these two new vertices to *true*, and from *false* to *ignored*. Finally, it joins each labeled vertex obtained in the last step of the merging stage to *ignored*, and adds edges between vertices whose labels complement each other.

The correctness of the transformation follows straighforwardly from properties 1 to 4. No vertex coordinate requires more than $O(\log n)$ bits, and the graph has at most $O(n^2)$ vertices and edges, and thus the transformation can be performed in polynomial time. Finally, a rather lengthy but simple case analysis shows that the

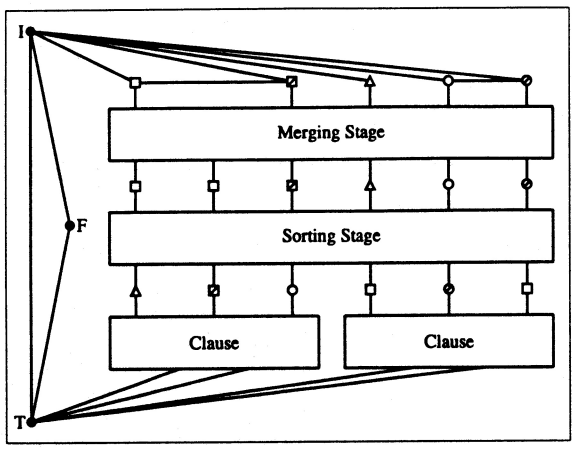

Figure 2: The stages of the reduction from 3SAT to DS3C.

graph thus generated is a subgraph of the Delaunay triangulation of its vertex set.

3.3 Reducing DS3C to 3D3C.

Let $G^* = (V^*, E^*)$ be the graph that appears in the instance of DS3C, where $V^* = \{v_1^*, \ldots, v_n^*\}$. To reduce DS3C to 3D3C, we first construct a convex polyhedron P whose vertices correspond to the vertices of G^*, and then removes pieces called *wedges* from P to obtain a polyhedron P^* with the following property:

Property 3.1 *Every reflex vertex of P^* belongs to its kernel, and the visibility graph of the set of convex vertices of P^* is the complement of G^*.*

To construct P, we consider the *inversion* function ϕ defined by $\phi(x, y) = (x, y, x^2 + y^2)$. The polyhedron P will be the convex hull of $V = \{v_1, \ldots, v_n\}$, where $v_i = \phi(v_i^*)$ for each i. Because G^* is a Delaunay subgraph, the following property follows from an observation of Edelsbrunner [Ede88]:

Property 3.2 *If (v_i^*, v_j^*) is an edge of G^*, then the line segment $\overline{v_i v_j}$ is an edge of P.*

Hence, to obtain a polyhedron P^* that satisfies property 3.1, it suffices to consider each pair of vertices that correspond to an edge of G^*, and

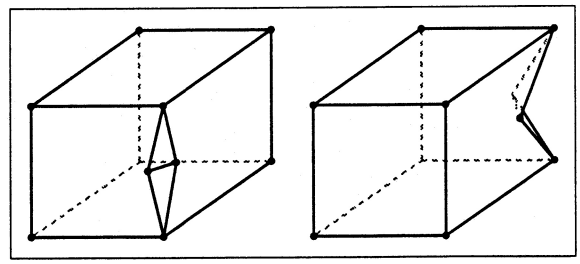

Figure 3: Two views of a cube from which a single wedge has been removed.

to prevent them from seeing each other. Because property 3.2 holds, this can be done using a *wedge*. A wedge is a small tetrahedron that has one edge along an edge of P, and that is contained in P. Figure 3 shows two different views of a cube from which a single wedge has been removed. The size of the wedge has been exaggerated for illustration purposes; real wedges will be a lot shallower.

If each wedge is shallow enough, then property 3.1 will be satisfied, and moreover no two wedges will intersect. This implies that the polyhedron P^* obtained by removing a wedge along each edge of P that corresponds to an edge of G^* will be simple.

Given a convex 3-cover of the convex vertices of P^* (or a convex 3-cover of its edges, of its boundary, or of P^* itself), we can derive a proper 3-coloring of G^* by coloring v_i^* using one of the cover elements that contains v_i. Consider now a proper 3-coloring χ of G^*. If each wedge is shallow enough, we can partition P^* as follows:

- the kernel of P^* is one piece;
- every other piece is a polyhedron that contains exactly one convex vertex of P^*.

If Q is the piece containing the convex vertex v_i of P^*, we can color Q using $\chi(v_i^*)$. For each i in $\{1, 2, 3\}$, let Q_i be the union of $Kr(P^*)$ with the set of pieces colored i. Property 3.1 implies that each Q_i is a convex subset of P^*, and so $\{Q_1, Q_2, Q_3\}$ is a convex 3-cover of P^*.

In $O(n^2)$ time, we can compute a value ε for which the statement "if each wedge is shallow enough" is equivalent to "if the depth of each

wedge is at most ε". Furthermore, using that ε, each coordinate of a vertex of P^* can be stored in a number of bit bounded by a constant times the maximum number of bits required by a coordinate of a vertex of G^*. Therefore the transformation can be done in time polynomial in the number of bits required to describe the instance of DS3C.

4 Conclusions

We gave an $O(n \log n)$ time and $O(n)$ space algorithm to solve the convex 2-cover problem for simple polyhedra in three dimensions. We also proved that the convex 3-cover problem becomes NP-hard for simple polyhedra in three dimensions. We can prove that the convex 3-cover problem is NP-hard for d-dimensional polyhedra when $d \geq 4$ by using the same reduction, and extending P^* into a cylinder along the remaining $d-3$ dimensions. This proof can then be easily modified (by adding spikes to the polyhedron) to show that the convex k-cover problem for star-shaped polyhedra in E^d is NP-hard whenever $d \geq 3$ and $k \geq 3$.

One problem that remains open is that of extending the algorithm given in Section 2 to higher dimensions. Properties 2.1, 2.2 and 2.3 remain valid, and so a similar approach will work. However we do not yet know how to use this approach efficiently in four or more dimensions. A second open problem related to Section 2 is that of improving the running time of the algorithm to $O(n)$. This would probably require a linear-time algorithm that either finds $Kr(P)$, or reports that P does not admit a convex 2-cover.

Finally, the main practical reason for studying the convex k-problem for small, fixed values of k lies in the approximation algorithms that may arise from the insights gained by such a study. The results proved here, together with Theorem 6.9 in the book by Garey and Johnson [GJ79], imply that no polynomial time approximation algorithm can provide a cover with less than 4/3 times as many pieces as the optimal cover. Finding any approximation algorithm which produces a cover with $o(n)$ time as many pieces as the optimal cover remains open.

References

[Akl85] S. G. Akl. *Parallel sorting algorithms.* Academic Press, 1985.

[Bel95] P. Belleville. On restricted boundary covers and convex three-covers. Technical report, Simon Fraser University, manuscript in preparation.

[Chv75] V. Chvátal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory, Series B*, 18:39–41, 1975.

[CLR89] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms.* MIT Press, 1989.

[CR88] J. Culberson and R. A. Reckhow. Covering polygons is NP-hard. In *Proceedings of the Twenty-Ninth IEEE Annual Symposium on the Foundations of Computer Science*, pages 601–611, 1988.

[Ede88] H. Edelsbrunner. *Algorithms in combinatorial geometry.* Springer-Verlag, 1988.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and intractability : a guide to the theory of NP-completeness.* W. H. Freeman and Company, 1979.

[O'R87] J. O'Rourke. *Art gallery theorems and algorithms.* Oxford University Press, Inc., 1987.

[She89] T. C. Shermer. *Visibility properties of polygons.* PhD thesis, McGill University, Montréal, June 1989.

[She92] T. C. Shermer. Recent results in art galleries. *IEEE Proceedings*, 80(9): 1384–1399, September 1992.

[She93] T. C. Shermer. On recognizing unions of two convex polygons and related problems. *Pattern Recognition Letters*, 14:737–745, 1993.