The Strange Complexity of Constrained Delaunay Triangulation

Nicolas Grislain
École Normale Supérieure de Lyon
Lyon, France
ngrislai@ens-lyon.fr

Jonathan Richard Shewchuk
Computer Science Division
University of California at Berkeley
Berkeley, California, USA 94720
jrs@cs.berkeley.edu

Abstract

The problem of determining whether a polyhedron has a constrained Delaunay tetrahedralization is NP-complete. However, if no five vertices of the polyhedron lie on a common sphere, the problem has a polynomial-time solution. Constrained Delaunay tetrahedralization has the unusual status (for a small-dimensional problem) of being NP-hard only for degenerate inputs.

1 Introduction

Suppose we wish to find a tetrahedralization T of a polyhedron P. P is not necessarily convex, and is not necessarily homeomorphic to a ball (i.e. it may have holes and handles, and be of arbitrary genus). We use the most common definition of *tetrahedralization*, in which T is a simplicial complex whose *underlying space* (the union of all the simplices in T) is P, and the vertices of T are precisely the vertices of P, with no extra vertices permitted. Observe that a simplicial complex T is not a tetrahedralization of P if T leaves any portion of P uncovered. Unfortunately, many polyhedra have no tetrahedralization [4], and Ruppert and Seidel [3] show that it is NP-hard to determine whether a polyhedron is tetrahedralizable.

A *constrained Delaunay tetrahedralization* (CDT) is a specific tetrahedralization of a polyhedron (or of a more general input called a *piecewise linear complex*) that has several favorable properties, notably that it helps to control interpolation error [8, 7] because it minimizes the radius of the largest min-containment sphere of a tetrahedron [2] (compared to all other constrained tetrahedralizations of the polyhedron), and it is an ingredient that helps Delaunay refinement algorithms to reliably generate good tetrahedral meshes [5]. A polynomial-time algorithm exists to construct a CDT of a polyhedron if one exists [6] (and therefore to check whether a polyhedron has a CDT) in $\mathcal{O}(n_v n_s)$ time, where n_v is the number of vertices of the polyhedron, and n_s is the number of simplices in its CDT. However, the algorithm is only guaranteed to work correctly if no five vertices of the polyhedron lie on a common sphere.

For convenience, we say that a polyhedron is *degenerate* if it has five (or more) vertices that lie on a common sphere (with a warning that we are abusing the term). A nondegenerate polyhedron has a unique CDT if it has a CDT at all, whereas some degenerate polyhedra have many CDTs. A degenerate polyhedron can be converted into a nondegenerate polyhedron by an infinitesimal perturbation of its vertices [1], but a perturbation might convert a polyhedron with a CDT (or many CDTs) into a nondegenerate polyhedron with no CDT. Similarly, an infinitesimal perturbation might convert a tetrahedralizable polyhedron into a polyhedron with no tetrahedralization at all.

Supported in part by the National Science Foundation under Awards ACI-9875170, CMS-9980063, CCR-0204377, and EIA-9802069, and by a gift from the Okawa Foundation. This work was done while the first author was visiting the University of California at Berkeley.

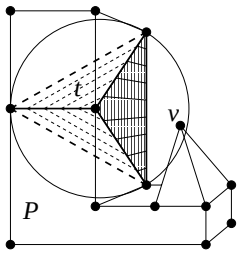

Figure 1: A constrained Delaunay tetrahedron t inside a polyhedron P.

This note shows that determining whether a degenerate polyhedron has a CDT is NP-hard. (It is easy to show the problem is in the class NP, so it is NP-complete.) This is an odd result for a small-dimensional problem, because the existence and identity of the (unique) CDT of a nondegenerate polyhedron is easily determined in polynomial time.

Note that the Ruppert–Seidel result [3] is not relevant to CDTs; their construction cannot be embodied in constrained Delaunay tetrahedra. Our construction is entirely different.

2 Constrained Delaunay Tetrahedralizations

Let P be a polyhedron. Two points p and q are *visible* to each other if the line segment pq lies in P. A tetrahedron t is *constrained Delaunay* if all four vertices of t are vertices of P, the tetrahedron t lies entirely in P, and the circumsphere of t (the unique sphere that passes through all four vertices of t) encloses no vertex of P that is visible from any point in the interior of t. (Any number of vertices is permitted *on* the sphere, though.) Figure 1 depicts a constrained Delaunay tetrahedron t. Although the circumsphere of t encloses a vertex v, v is not visible from any point in the interior of t, so t is constrained Delaunay.

A tetrahedralization T of P is a *constrained Delaunay tetrahedralization* (CDT) of P if all the tetrahedra in T are

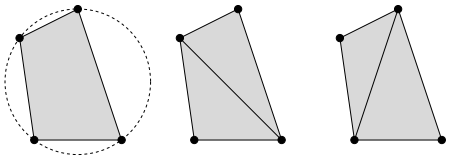

Figure 2: A bit (left), whose triangulation indicates a value of true or false.

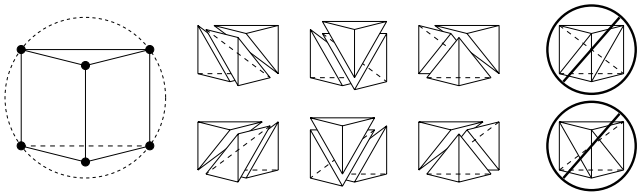

Figure 3: There are two configurations the S gate does not allow: the inputs are true and the output false; or the inputs are false and the output true.

constrained Delaunay. A set of five or more vertices lying on a common sphere, with no vertex inside it, might admit several different tetrahedralization, all composed of constrained Delaunay tetrahedra.

3 The Reduction

Here we show how to embody simple circuits and gates in a polyhedron P, so any satisfiability problem can be reduced to finding a CDT. Let $p(v_1, v_2, \ldots, v_n)$ be a predicate of length m on n variables $v_1, v_2, \ldots, v_n$.

One basic component of our circuits is a *bit*, realized as four coplanar vertices that lie on a common sphere, with no other vertex of P inside or on the sphere. Because the four vertices are coplanar, their convex hull is a quadrilateral (which is not necessarily rectangular), depicted in Figure 2. The interior of the quadrilateral lies in the interior of P. Because the sphere neither touches nor encloses any other vertex, in any CDT of P, the quadrilateral must be triangulated in one of two ways, which we interpret as "true" or "false." The edge that divides a bit into two triangles is called a *diagonal*.

In our construction, some bits represent variables, some bits represent the values of subexpressions, and most bits are simply used to propagate a signal (variable or subexpression) from one place to another.

Gates are simulated by modular volumes that have bits on their surfaces and can be tetrahedralized in several different ways. The surface triangulations, and therefore the bits, must conform to the gate tetrahedralizations. Circuits are built by gluing gates together along their bits. Unfortunately, our gates are weak, because their outputs are not fixed for all possible inputs. Sadly, we do not even know how to build a proper wire that propagates both true and false signals, but we will not need one.

We build a gate called an *S gate*, inspired by Schönhardt's untetrahedralizable polyhedron [4]. Place six vertices on a common sphere so that their convex hull is, topologically, a triangular prism. (None of its faces need be parallel, though.) Let S be a sphere, and let p_1, p_2, and p_3 be three planes such that $S \cap p_1$, $S \cap p_2$, and $S \cap p_3$ are pairwise intersecting circles, and $p_1 \cap p_2 \cap p_3$ lies outside S or does not exist. For each distinct i and j, $S \cap p_i \cap p_j$ is two points, yielding six points total. An S gate is the convex hull of these six points, as illustrated in Figure 3.

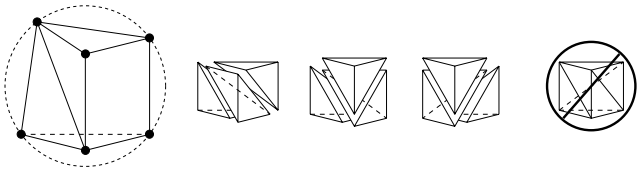

Figure 4: A W gate does not allow the output to be false if the input is true. Alternatively, by flipping the fixed diagonal, we have a W gate that propagates false signals (but not necessarily true signals).

An S gate has three quadrilateral faces, each of which is a bit, and two triangular faces. Every S gate we use in our circuit is included in the polyhedron P, so any tetrahedralization of P must cover each S gate. There are six ways to tetrahedralize an S gate, and if no vertex lies inside S, all these tetrahedralizations use only Delaunay tetrahedra. There are two configurations of the three bits for which there is no corresponding tetrahedralization. If we arbitrarily treat two of the bits as inputs and one as an output, the S gate enforces the following truth table.

S gate

Input 1	Input 2	Output
false	false	false
false	true	?
true	false	?
true	true	true

A second gate, called a *W gate* (for *wire*), is like an S gate except that one input is fixed. A W gate has only two bits; the third quadrilateral face is replaced by two triangular faces that meet at a diagonal reflex edge, as illustrated in Figure 4. These two triangular faces lie in the boundary of the polyhedron P, so they are fixed parts of any tetrahedralization of P.

Let S be a sphere, and let p_1 and p_2 be two planes such that $S \cap p_1$ and $S \cap p_2$ are intersecting circles. A W gate has six vertices, of which two are $S \cap p_1 \cap p_2$, two more lie on $S \cap p_1$, and two more lie on $S \cap p_2$. We have some flexibility in choosing the latter four, and we select them so that they are not coplanar and the W gate has one reflex edge.

If the non-coplanarity of these four vertices makes circuit layout inconvenient, we can instead construct a W gate by attaching to an S gate an *anchor*: a tetrahedron whose apex vertex lies on the circumsphere S and is placed so that it cannot see any vertex outside the anchor, as Figure 5 illustrates.

There are two types of W gates, determined by the choice of reflex edge. One type of W gate always has a true output if the input is true. However, if the input is false, the output may be either true or false. The complementary type of W gate propagates false signals from input to output, but may

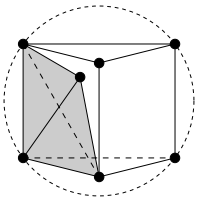

Figure 5: An alternative W gate formed by pasting an anchor (shaded) onto an S gate. The anchor forces one of the quadrilateral faces to be triangulated along a fixed diagonal. The apex of the anchor, like the other vertices, lies on the circumsphere S. The apex cannot see any vertex of the gate other than the vertices of the anchor.

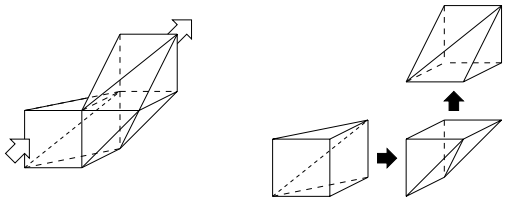

Figure 6: A **half-not** gate (left) fashioned from three W gates (right).

or may not propagate true signals. By chaining W gates together, we build wires that propagate true signals (or false signals, but not both) from one place to another.

W gates are quite flexible in shape, so chains of W gates can easily run signals along circuitous paths through three-dimensional space. This flexibility makes it easy to simulate a **half-not** gate (which maps a true to a false and a false to a "don't care," or vice versa) by rotating the diagonals as needed. Figure 6 depicts one way to fashion a **half-not** gate from three W gates.

Our most powerful gate, called a *G gate*, is illustrated in Figure 7. Let S be a sphere, and let p_1, p_2, and p_3 be three planes such that $S \cap p_1$, $S \cap p_2$, and $S \cap p_3$ are pairwise intersecting circles, and furthermore there is one point where all three planes and S intersect. The seven vertices of a G gate are the point $S \cap p_1 \cap p_2 \cap p_3$, one additional point from each of $S \cap p_1 \cap p_2$, $S \cap p_2 \cap p_3$, and $S \cap p_3 \cap p_1$, and one additional point chosen (with some flexibility) from each of the circles $S \cap p_1$, $S \cap p_2$, and $S \cap p_3$.

A G gate, like an S gate, has three bits. It has four triangular faces, which all lie in the boundary of P. As Figure 7 shows, a G gate is not convex, and its three bits are not all interchangeable. In the figure, the two bits y and z behave symmetrically, but the privileged bit x is different. There are five ways to tetrahedralize an S gate, and if no vertex lies inside S, all five tetrahedralizations use only Delaunay tetrahedra. In any bit, we are free to decide which diagonal represents "true" (because we can always interpose a **half-not** gate between two other gates if necessary to make them speak the same language), so we interpret the G gate as follows.

If we treat x as the input, and y and z as outputs, then the G gate serves as a *splitter*: if x is true, then both y and z must also be true. If x is false, then y and z may each independently take on any value. By reversing our

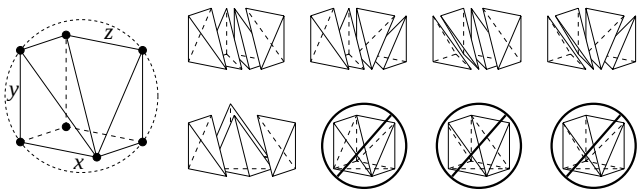

Figure 7: A G gate forces y and z to be true if x is true, and forces x to be false if y or z if false.

interpretation of the diagonals, we may instead use a G gate so that it duplicates and propagates false signals, but not necessarily true signals.

If we treat y and z as inputs, and x as the output, then the G gate serves as a **half-and** gate with the following truth table.

half-and gate

Input y	Input z	Output x
false	false	false
false	true	false
true	false	false
true	true	?

By reversing our interpretation of the diagonals, we have a **half-or** gate: if either y or z is true, then x must be true. If both y and z are false, then x may be either true or false.

We wish to convince the reader (without a fully rigorous proof) that it is straightforward to piece gates together by gluing their quadrilateral faces (bits) together, so that they can emulate arbitrary logic circuits and so that all the tetrahedralizations of each gate enumerated above, and no others, are constrained Delaunay. One way to safely route circuits is to divide space into a grid of cubes, as illustrated in Figure 8. Each gate can be laid out so its vertices lie on, or close to, the corners of one or two cubes, and so the circumsphere of each gate encloses no vertex. Therefore, for each gate, all of the tetrahedralizations enumerated above consist of constrained Delaunay tetrahedra, so any of these tetrahedralizations may be part of the CDT of P if they can be made compatible with the tetrahedralizations of the adjoining gates. Conversely, because the circumsphere of each gate touches no other vertex than the vertices of the gate, these tetrahedralizations are the *only* constrained Delaunay tetrahedralizations possible. A tetrahedron that intersects the interiors of two different gates is not constrained Delaunay, because its circumsphere must enclose a vertex of the bit shared by the two gates. We conclude that in any CDT of P, every gate is tetrahedralized by one of the configurations illustrated in Figures 3, 4, and 7.

Observe that the circumspheres of two gates meeting at a bit b determine the plane p in which b lies, because p contains the circle where the circumspheres intersect. Hence, the choice of circumspheres for the gates uniquely determines the positions of most of the gate vertices. (This is one reason why W gates with anchors are easier to lay out than the six-vertex W gates.) Some of the circumspheres must be perturbed slightly so they are not perfectly aligned with the grid, because we want every gate vertex to be a vertex of P. However, the perturbation must satisfy some constraints:

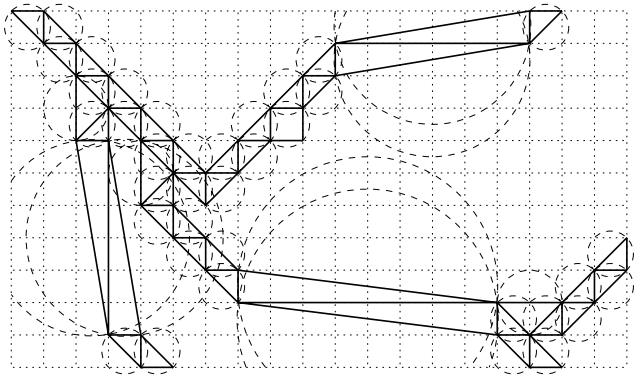

Figure 8: An illustration of how gates can be laid out so that each gate's circumsphere does not enclose vertices of other gates that are visible from the gate's interior. (The third dimension is used as well, for flexible routing and for G gates.)

wherever an S gate or G gate meets wires, there are some vertices where three or four circumspheres must meet.

Also observe that gates can be made arbitrarily small, and wires can be made arbitrarily long and thin, so wires can effectively be treated as one-dimensional objects for the purpose of circuit layout. Figure 8 illustrates several long wires. Although their circumspheres are large enough to enclose other gates, here is where the "constrained" part of constrained Delaunay comes into play: no vertex enclosed in a gate's circumsphere is visible from the interior of the gate, so any valid tetrahedralization of a gate is constrained Delaunay.

A complete formal proof of the reduction of satisfiability to constrained Delaunay tetrahedralization would include a precise mechanical procedure for laying out a circuit for any predicate. As a start, we suggest assigning each S gate and G gate a unique positive index i, and positioning the gates at integral coordinates on the *moment curve* (i, i^2, i^3). This positioning simplifies the routing of wires between gates because any two distinct line segments with their endpoints on the moment curve do not intersect, except perhaps at a shared endpoint. Thus, if the wires are made thin enough and the gates small enough, routing is a simple matter of locally attaching three wires to each S gate and each G gate so the three wires do not interfere with each other, and their bits are correctly oriented where a wire meets a gate. These local connections can be made with a constant number of W gates, so a predicate of complexity m can be represented by a polyhedron P of complexity $\mathcal{O}(m)$ whose CDT, if one exists, is also of complexity $\mathcal{O}(m)$. We hope the reader is convinced that this can always be accomplished, because a fully rigorous treatment would be tedious and unenlightening for reader and authors alike.

Because the gates and wires are so poor at signal propagation, we need one more idea to embody a circuit in a tetrahedralization. Every value that must be known or computed to evaluate the predicate p, including every variable and every subexpression, is represented by two different signals in the polyhedron P. Let s be either the value of a variable v_i, or the value of some subexpression of p (possibly p itself). Then s^t is a signal that is true if s is true, and is indeterminate (can be false, but is not guaranteed to be false) if s is false. Likewise, s^f is a signal that is false if s is false, and can be true (but is not guaranteed to be) if s is true. The signal s^t is propagated to any subexpression that depends on it via W gates oriented to propagate true inputs. Likewise, s^f is propagated via W gates oriented to propagate false inputs. Wherever it is needed, we use a G gate as a splitter to duplicate the signal s^t, and likewise we use a complementary G gate to duplicate s^f.

The predicate p is embodied in a polyhedron P as follows. Each input variable v_i is represented by a bit b_i somewhere in the interior of P. Out of each bit b_i, we run two wires—one from each side of the bit. One wire propagates v_i^t by means of W gates oriented to propagate true inputs, and one propagates v_i^f by complementary W gates.

To implement a **not** gate, pass a copy of a signal s^t through a **half-not** gate to yield $\overline{s}^f$, and pass a copy of s^f through a complementary **half-not** gate to yield $\overline{s}^t$.

To implement an **and** gate that computes $x = y \wedge z$, pass copies of y^t and z^t into an S gate to yield x^t, so that x^t must be true if both y and z are true, and x^t can be false otherwise. Pass copies of y^f and z^f into a **half-and** gate configured so its output x^f must be false if either y or z is false, and x^f can be true otherwise. Similarly, an **or** gate may be constructed from an S gate and a **half-or** gate, or equally well by De Morgan's law from an **and** gate and three **not** gates.

At the end of the circuit, the signals p^t and p^f represent the predicate p. The signal p^f must be false if p is false, so we attach p^f to one final anchor (recall Figure 5) set to enforce the true state. Hence, if p^f is false, the tetrahedralization cannot be completed. In any CDT of the polyhedron P, the bits b_i correspond to a truth assignment to the variables v_i for which $p(v_1, v_2, \ldots, v_n)$ is true.

Conversely, if p has a satisfying assignment, we construct a CDT of P as follows. Triangulate each bit b_i to match the value of v_i in the satisfying assignment. For each signal s, triangulate every bit that represents either s^t or s^f to match the value of s. Then tetrahedralize each gate in a manner conforming to its bits.

We have thus reduced the problem of circuit satisfiability to determining whether a polyhedron P has a constrained Delaunay tetrahedralization.

4 Conclusion

One unfortunate property of our construction is that P is not a simple polyhedron; rather, it has a large genus. Although the separate wires play an essential part in the proof, the holes between them do not. Perhaps we could augment P to yield a simple polyhedron by affixing additional volume to the triangular faces of the gates, or by some other method, but it remains an open problem. Nevertheless, our construction as it stands is sufficient to end any hope for a polynomial-time algorithm that constructs a CDT of any piecewise linear complex that has one.

It is interesting to speculate on whether there are any natural problems in two-dimensional geometry that are NP-hard in "degenerate" cases, but solvable in polynomial time otherwise.

References

[1] Herbert Edelsbrunner and Ernst Peter Mücke. *Simulation of Simplicity: A Technique to Cope with Degenerate Cases in Geometric Algorithms*. ACM Transactions on Graphics **9**(1):66–104, 1990.

[2] V. T. Rajan. *Optimality of the Delaunay Triangulation in* $\mathbb{R}^d$. Proceedings of the Seventh Annual Symposium on Computational Geometry, pages 357–363, 1991.

[3] Jim Ruppert and Raimund Seidel. *On the Difficulty of Triangulating Three-Dimensional Nonconvex Polyhedra*. Discrete & Computational Geometry **7**(3):227–253, 1992.

[4] E. Schönhardt. *Über die Zerlegung von Dreieckspolyedern in Tetraeder*. Mathematische Annalen **98**:309–312, 1928.

[5] Jonathan Richard Shewchuk. *Mesh Generation for Domains with Small Angles*. Proceedings of the Sixteenth Annual Symposium on Computational Geometry (Hong Kong), pages 1–10. Association for Computing Machinery, June 2000.

[6] ———. *Sweep Algorithms for Constructing Higher-Dimensional Constrained Delaunay Triangulations*. Proceedings of the Sixteenth Annual Symposium on Computational Geometry (Hong Kong), pages 350–359. Association for Computing Machinery, June 2000.

[7] ———. *What Is a Good Linear Element? Interpolation, Conditioning, and Quality Measures*. Eleventh International Meshing Roundtable (Ithaca, New York), pages 115–126. Sandia National Laboratories, September 2002.

[8] Shayne Waldron. *The Error in Linear Interpolation at the Vertices of a Simplex*. SIAM Journal on Numerical Analysis **35**(3):1191–1200, 1998.