When Can A Graph Form An Orthogonal Polyhedron?

Therese Biedl* Burkay Genc*

1 Introduction

Polyhedra are an important basic structure in computational geometry. One of the most beautiful results concerning polyhedra is Cauchy's theorem, which states that a convex polyhedron is uniquely defined by its graph, edge lengths and facial angles. (See Section 2 for definitions.) The proof of Cauchy's theorem (see e.g. [2]) unfortunately is non-constructive, and the only known algorithm to reconstruct the convex polyhedron is very slow (see also [5].)

In this paper, we study similar topics for orthogonal polyhedra. Thus, given a graph, edge lengths and facial angles, when is this the graph of an orthogonally convex polyhedron? We give an algorithm that answers this question in polynomial time, and reconstructs the polyhedron if one exists. In particular, our algorithm implies a Cauchy-type theorem for orthogonally convex polyhedra: they are determined by their graph, edge lengths and facial angles alone. We also study general orthogonal polyhedra, and show that it is NP-hard to decide whether a graph (with edge lengths and facial angles) is the graph of an orthogonal polyhedron.

Our research was motived by the question how to represent polyhedra (and especially orthogonal polyhedra) efficiently. One common way is the vertex based model, where one stores the graph and the coordinates of each vertex. For orthogonal polyhedra, it suffices to store coordinates for vertices of odd degree, see [4, 1].

The vertex based model is rather cumbersome for manipulation of polyhedra, since every translation or rotation requires an update of all coordinates. A more versatile approach is to store edge lengths, facial angles and dihedral angles only. The polyhedron is then uniquely determined by the coordinates of three vertices. The results in our paper show that for orthogonally convex polyhedra, we can omit the dihedral angles, since they are uniquely determined from the other parameters.

2 Definitions

We assume familiarity with polygons and polyhedra. The vertices, edges and faces of a polyhedron form a graph with a fixed combinatorial embedding given by the order of edges around the vertex. Furthermore, the polyhedron defines edge lengths, *facial angles* (the angle between two consecutive edges at a vertex) and *dihedral angles* (the interior angle between two faces at a common edge.)

An *orthogonal polyhedron* is a polyhedron for which all dihedral angles (and hence also all facial angles) are multiples of $\pi/2$. After a suitable rotation, every face of an orthogonal polyhedron is perpendicular to one of the coordinate axis and can be classified as X*-face*, Y*-face* or Z*-face.*

An *orthogonally convex polygon* is a polygon for which all edges are horizontal or vertical and any horizontal or vertical line intersects the polygon in at most one interval. An *orthogonally convex polyhedron* is an orthogonal polyhedron for which every intersection with a plane perpendicular to a coordinate axis is a single orthogonally convex polygon.

Given a graph with a fixed combinatorial embedding (which defines the faces), edge lengths and facial angles, we call it an *OP-graph* (*OCP-graph*) if there exists an orthogonal polyhedron (orthogonally convex polyhedron) with this graph. This paper addresses the following questions:

> Given a graph with fixed combinatorial embedding, edge lengths and facial angles, is this graph an OP-graph? Is it an OCP-graph? And if the answer is positive, can we reconstruct the polyhedron?

Our results can thus be summarized as follows: Testing whether a graph is an OP-graph is NP-hard, but testing whether it is an OCP-graph (and reconstructing the polyhedron if it is) can be done in polynomial time.

3 Recognizing OCP-graphs

In this section we study how to recognize OCP-graphs. So let G be a graph with a fixed combinatorial embedding. We may assume that G is *planar* (it can be drawn without crossing in the plane) and connected, otherwise it cannot be an OCP-graph. We will also assume that every face of G has degree 4. Our algorithm thus works for orthogonal polyhedra for which every face is a rectangle. This is not a restriction if we allow for an increased time complexity: by subdividing faces (using the edge length information), we can convert G into a graph G' with faces of degree 4 such that G is an OP-graph (OCP-graph) if and only if G' is an OP-graph (OCP-graph). Unfortunately, the size of G' may be quadratic in the size of G.[1]

The dual graph of a planar graph (which, as always in this paper, we assume to have a fixed combinatorial embedding) is obtained by taking a vertex for every face and connecting two vertices of faces if and only if the faces share an edge.

*School of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, {biedl,bgenc}@uwaterloo.ca

[1]Very recently, we have developed a different algorithm that avoids subdividing and has linear time complexity.

A graph is called a *DOP-graph* (*DOCP-graph*) if it is the dual of an OP-graph (OCP-graph). Figure 1 illustrates an orthogonal polyhedron, how to subdivide to obtain rectangular faces, and the corresponding DOCP-graph, which is necessarily *4-regular* (all vertices have degree 4.)

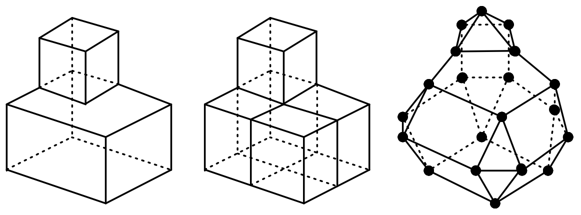

Figure 1: A polyhedron and its 4-regular DOCP-graph.

From now on, we will study how to recognize whether a given graph is a DOCP-graph (which is equivalent to recognizing an OCP-graph.) So let G be a 4-regular graph with a fixed combinatorial embedding. A *straight cycle* of G is a set of vertices and edges $e_1 v_1 e_2 \dots v_k e_k v_{k+1} e_{k+1}$ such that $e_{k+1} = e_1$ and for all $1 \leq i \leq k$, edge e_i and e_{i+1} are *not* consecutive at v_i in the combinatorial embedding. Since G is 4-regular, every vertex is on two straight cycles (or maybe on one straight cycle twice, but then G for sure is not a DOCP-graph.) We say that two straight cycles *cross* if they have a vertex in common. We will not consider any other cycles, and often drop "straight" from now on.

An X*-stripe* of an orthogonal polyhedron with rectangular faces is a connected set of faces whose center points have the same X-coordinate. An X-stripe contains only Y-faces and Z-faces; we will therefore also call it a YZ*-stripe*. Similarly define XY*-stripes* and XZ*-stripes*. If G is the DOP-Graph of a polyhedron P, then each straight cycle of G corresponds to a stripe of P. We call a cycle an XY*-cycle* if it corresponds to an XY-stripe, and say that the cycle has *cycle-type* XY. Likewise, we define XZ*-cycles* and YZ*-cycles*. The following observations are straightforward.

Lemma 1 *Let G be a DOP-graph.*

1. *No two cycles that cross can have the same type.*
2. *If three straight cycles mutually cross, then the types of two of them determine the type of the third.*
3. *A vertex corresponds to an X-face iff the two cycles containing it are an XY-cycle and an XZ-cycle.*
4. *For any straight cycle C, all dual edges of edges in C have the same edge length.*

3.1 Identifying face types

Lemma 1(3) implies that to identify face types, it suffices to determine cycle types. The idea of our algorithm to do so is very simple. First, arbitrarily fix the types of two straight cycles that cross, since we can rotate the polyhedron suitably. Assume that we know the type of a cycle C, say it is an XY-cycle. Then any cycle C' that crosses C cannot be an XY-cycle. So we exclude the type "XY" from the list $T(C')$ of possible types of C'. If now $|T(C')| = 1$ (initially $|T(C')| = 3$ for all cycles), then we know the type of C'. We repeat until all cycles are identified; if this is not possible then (as we will see) the graph was not a DOCP-graph. See Figure 2 for pseudo-code.

```
(1) Pick two straight cycles $C_1$ and $C_2$ that cross.
(2) Set $T(C_1) = \{XY\}$ and $T(C_2) = \{XZ\}$.
(3) For all $C_i \neq C_1, C_2$; set $T(C_i)$=$\{XY, YZ, XZ\}$
(4) For $k = 1, \dots, 3$, create the empty sets DOF[k] .
(5) foreach $C_i$, put $C_i$ into DOF[|$T(C_i)$|]
(6) while DOF[2]$\neq \emptyset$ or DOF[3]$\neq \emptyset$
(7)     if DOF[1] is empty, output an error message.
(8)     else remove a cycle $C$ from DOF[1].
(9)     foreach cycle $C'$ that crosses $C$
(10)        Set $T(C') = T(C') - T(C)$
(11)        Move $C'$ to DOF[|$T(C')$|]
```

Figure 2: Algorithm IDENTIFYCYCLETYPES.

One can show that all steps of this algorithm, as well as the preprocessing steps to compute the dual graph and the straight cycles, can be done in linear time. (This assumes that the given graph has faces of degree 4; the preprocessing step to achieve faces of degree 4 may take quadratic time.)

The correctness of this algorithm will be proved in a sequence of lemmas, but most proofs have to be sketched for space reasons. The first two lemmas follow easily from planarity.

Lemma 2 *For a planar 4-regular graph, two straight cycles cross an even number of times.*

Lemma 3 *Let G be a 4-regular planar graph and let C be a straight cycle in G. For $i = 1, 2$, let C_i be a straight cycle that crosses C at u_i and v_i. Assume $\{u_1, v_1\}$ interleaves $\{u_2, v_2\}$ on C; i.e., the order is either u_1, u_2, v_1, v_2 or u_1, v_2, v_1, u_2 while going along C. Then C_1 and C_2 cross each other.*

The following two lemmas are crucial to show that cycle types can be propagated from the first two to all straight cycles.

Lemma 4 *Let G be a DOCP-graph with a straight cycle C. Define a graph H with a vertex for each cycle C_i that crosses C and an edge (C_i, C_j) iff C_i and C_j cross each other. Then H is connected.*

Proof. The stripe represented by cycle C projects to an orthogonally convex polygon p. Each C_i crosses C at two vertices which correspond to parallel aligned edges of p and we represent C_i by the rectangle between these edges as in Figure 3. If two such rectangles intersect, then their endpoints interleave, and by Lemma 3 the corresponding straight cycles cross each other and are connected by an edge in H. We can get from any point on p to any other point on p via a sequence of rectangles such that consecutive rectangles intersect (see also Figure 3). These rectangles form a path in H, thus any two vertices in H are connected by a path. $\square$

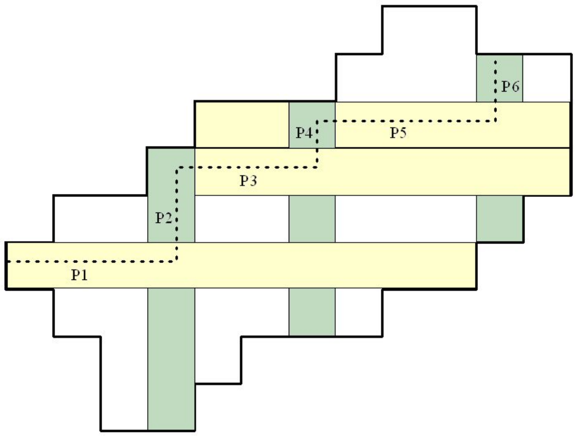

Figure 3: An orthogonal path between two points on p and the corresponding rectangles (which represent straight cycles.)

Lemma 5 *Let G be a DOCP-graph and let C and C' be two straight cycles that cross each other and for which the types are known. Then we can identify the type of all cycles that cross C.*

Proof. Let C'' be a cycle that crosses C. We can find a path from C' to C'' in the graph H defined in Lemma 4. Let C_1 be the first cycle after C' on this path. C_1 crosses C by definition of H, so applying Lemma 1(2) to C, C' and C_1 gives the type of C_1. Iterating this, we can identify the types of all cycles on the path, and in particular the type of C''. $\square$

Theorem 6 *For a DOCP-graph, algorithm* IDENTIFYCYCLETYPES *correctly identifies all cycle-types.*

Proof. We start knowing the types of two cycles that cross each other. From Lemma 5, we can identify all cycles that cross either one of these cycles. Then in turn we can identify the types of all cycles that cross these cycles. This process eventually must reach all cycles, because a DOCP-graph is connected, and we can get from any cycle to any other cycle via a path of cycles that cross each other. $\square$

3.2 Identifying coordinates

Now we show how to compute coordinates for the center points of all faces. This happens by ordering cycles of each type suitably and adding up the edge lengths to get the coordinate. See Figure 4 for the algorithm.

Lemma 7 *For any DOCP-graph G, the YZ-cycles can be ordered as $C_1, \ldots, C_t$ such that for any $i < j < k$, any path from C_i to C_k intersects C_j.*

```
(1) Let $C_1, \ldots, C_t$ be the $YZ$-cycles,
        ordered such that for any $i < j < k$
        any path from $C_i$ to $C_k$ intersects $C_j$.
(2) For $i = 1, \ldots, t$, set $\ell(C_i)$ to be the edge length
        of the dual edges of $C_i$. (Recall Lemma 1(4)).
(3) Set $x(C_1) = 0$.
(4) For $i = 1, \ldots, t-1$
(5)     Set $x(C_{i+1}) = x(C_i) + \frac{1}{2}(\ell(C_i) + \ell(C_{i+1}))$.
(6) For any face $F$ in a $YZ$-cycle $C$,
        the center point of $F$ has $x$-coordinate $x(C)$.
```

Figure 4: Algorithm IDENTIFYCOORDINATES.

Proof. Let P be the orthogonally convex polyhedron with graph G. A YZ-cycle of G corresponds to a set of faces of P whose center points all have the same X-coordinate. For any $x \in \mathcal{R}$ the $(X = x)$-plane intersects P in one connected region, so only one YZ-cycle can exist for which a face has coordinate x. Hence, sort the cycles by the x-coordinates of the corresponding stripes. To get from stripe C_i to stripe C_k, we then necessarily must go through a face that has an x-coordinate of stripe C_j, and hence belongs to C_j. $\square$

We visualize this lemma by thinking of the stripes $C_1, \ldots, C_l$ as forming “onion rings” (two halves of an onion) in the planar embedding. See Figure 5. Since we know all YZ-cycles, testing whether they form onion rings can be done in linear time by doing a modified breadth-first search. The rest of algorithm IDENTIFYCOORDINATES also can be implemented in linear time.

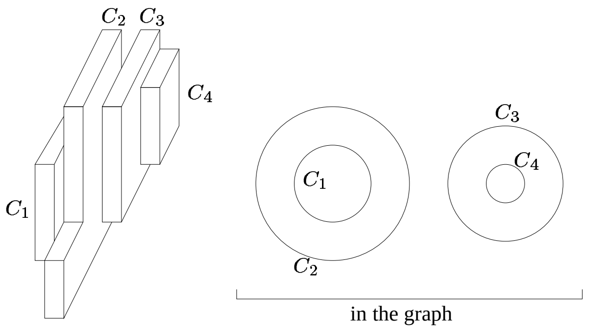

Figure 5: The stripes of an orthogonally convex polyhedron form an onion-ring structure.

From the onion-ring structure, algorithm IDENTIFYCOORDINATES correctly retrieves the x-coordinates of the center points of all vertices on a YZ-cycle, since these are determined by the width of each stripe, which corresponds to the edge lengths in the original graph.

From this, we can get x-coordinates of all vertices of the polyhedron in linear time. Note that not all vertices are on a YZ-stripe, however, the coordinate information for such vertices can be retrieved from their neighbor vertices in linear time. Similarly we can determine the y-coordinates and the z-coordinates of all vertices using XZ-cycles and XY-cycles respectively.

Theorem 8 *If G is an OCP-graph, then our algorithm reconstructs an orthogonally convex polyhedron for which G is the graph.*

Note that our algorithm can be used to test whether a graph is an OCP-graph. If it succeeds in constructing a polyhedron, then we simply verify whether its graph is G. If this is not the case, or if the algorithm fails, then G cannot have been an OCP-graph.

4 Recognizing OP-graphs

In this section, we show that recognizing OP-graphs is NP-hard. Our construction uses a polyhedron of genus 1, but can be generalized to polyhedra of genus 0.

The reduction is from PARTITION, which is known to be NP-hard [6]. In this problem, we are given numbers $a_1, \ldots, a_n$, and we want to find a set $S \subset \{1, \ldots, n\}$ such that $\sum_{i \in S} a_i = \frac{1}{2} \sum_{i=1}^{n} a_i$. Given such an instance, we describe how to construct an orthogonal polyhedron (or really the graph of this polyhedron); the polyhedron can be realized if and only if PARTITION has a solution. The crucial idea is taken from [3]. Let L be a suitably large number, and consider an orthogonal polygon with the following edge-lengths: $L, 1, a_1, 1, a_2, 1, \ldots, a_n, 1, L, n+1$. See the left half of Figure 6. It was shown in [3] that such a polygon exists if and only if the PARTITION instance has a solution.

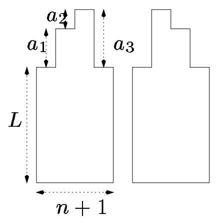

Figure 6: The cross-section.

For space reasons, we can only sketch the construction. We convert this polygon into a polyhedron by applying an "orthogonalized revolution". More precisely, we construct a polyhedron that consists of "stacked cubes" of height $L, a_1, \ldots, a_n, L$. Each cube can either be set atop the previous one or be sunk inside it. The top surface of the last cube is actually a hole inside the bottom surface of the first cube. See Figure 7 for an illustration. The cross-section of this polyhedron with the $(Z = 0)$-plane is then the two polygons in Figure 6. Hence this polyhedron exists if and only if PARTITION has a solution. Testing whether the graph of our construction is an OP-graph is therefore NP-hard.

5 Conclusion

In this paper, we studied how to recognize whether a given graph (with edge lengths and facial angles) is the graph of an orthogonal polyhedron. This can be decided in polynomial time for orthogonally convex polyhedra, but is NP-hard for general orthogonal polyhedra.

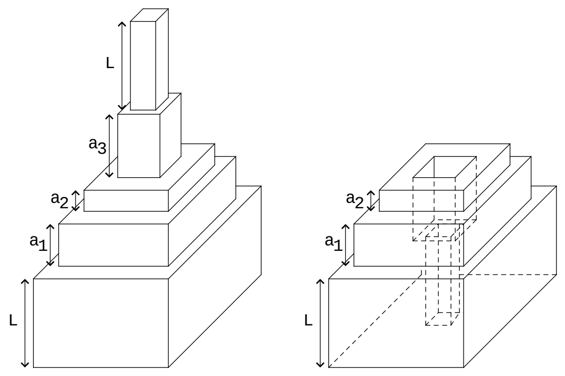

Figure 7: Stacked cubes, and the polyhedron if PARTITION has a solution.

It remains open whether we can at least identify the face types (Section 3.1) for a given OP-graph. We conjecture that the crucial Lemma 4 holds for all orthogonal polyhedra, but our proof depends on convexity.

Another interesting case occurs if edge lengths are not part of the input. Our algorithm for orthogonally convex polyhedra with rectangular faces still works. The NP-hardness proof requires given edge lengths. Is it NP-hard to test whether a given graph without edge lengths is the graph of some orthogonal polyhedron?

References

[1] A. Aguilera and D. Ayala. Orthogonal polyhedra as geometric bounds in constructive solid geometry. In *ACM Symp. on Solid Modeling and Appl.*, pp. 56–67, 1997.

[2] M. Aigner and G. Ziegler. *Proofs from THE BOOK*. Springer Verlag, 1999.

[3] T. Biedl, A. Lubiw, and J. Sun. When can a net fold to a polygon? In *CCCG'99*, pages 1–5, 1999.

[4] O. Bournez, O. Maler, and A. Pnueli. Orthogonal polyhedra: Representation and computation. In *Hybrid Systems: Computation and Control*, volume 1569, pages 46–60, Nijmegen, Pays-Bas, 29–31 Mar. 1999.

[5] E. Demaine and J. O'Rourke. Folding and unfolding in computational geometry, 2004. To appear.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, 1979.