Recognizing Visibility Graphs of Uni-monotone Polygons

Paul A. Colley*
colley@qucis.queensu.ca
Queen's University, Kingston, Ontario, Canada

June 17, 1992

Abstract

The *visibility graph* of a simple polygon P is defined as the graph with a node for each vertex of P, and an edge connecting pairs of nodes if and only if the corresponding vertices can *see* each other inside P.

In 1985, ElGindy characterized a class of graphs, *maximal outer-planar* graphs, as being the visibility graphs of polygons. We extend ElGindy's result to a larger class of graphs, which we term *tree of cliques* graphs. We show that tree of cliques are a strict subset of visibility graphs of *uniform uni-monotone* polygons.

Then, using linear programming, we develop a recognition algorithm for the visibility graphs of uniform uni-monotone polygons (but the algorithm requires that the outside face of the polygon be fixed).

A strong relationship is discovered between the visibility graphs of uni-monotone polygons and the visibility graphs of *staircase* polygons (also called *orthogonal convex fans*).

The problem of recognizing visibility graphs of general uni-monotone polygons where the Hamilton cycle is not given remains open.

*These results were part of my Master's Thesis done at University of Waterloo under the supervision of Anna Lubiw. Funding was provided by NSERC

1 Introduction

Given a simple polygon in the plane, two vertices u and v of the polygon are mutually *vertex-visible* (*see* each other) if the straight line connecting them does not intersect the exterior of the polygon. We will use "visible" and "visibility" to mean vertex-visibility.

The *visibility graph* of a polygon is a graph with a node for each vertex of the polygon, and an edge joining two nodes if and only if the two corresponding vertices can see each other.

The problem of visibility graph recognition, "Given an arbitrary graph G, does there exist a polygon P such that G is the visibility graph of P?" has known results for only a handful of special cases. Characterizing visibility graphs, and finding algorithms to recognize them, remains an open question [8, pp. 165, 171] [5, pp. 14–34].

Any complete graph is a visibility graph. A convex polygon has the property that all vertices can see each other, so a complete graph on n nodes corresponds to the visibility graph of a convex polygon with n vertices. For a fixed number of vertices, complete graphs have the most edges of any visibility graph.

Any *maximal outer-planar* graph is a visibility graph (ElGindy [8, pp. 169–171] gives a linear time recognition and construction algorithm). A graph is *outer-planar* if it has a pla-

nar representation with all the vertices on the outer face. A graph is *maximal* outer-planar if, in addition, no more edges can be added.

One notable aspect of ElGindy's algorithm is that the constructed polygon is *uni-monotone*. A polygon is *monotone* if the outside face of the polygon can be broken into two chains of edges, such that the vertices of each chain are monotone increasing with respect to some direction line, which we call the *direction of monotonicity*. There is a pair of vertices which is common to both chains. A *uni-monotone* polygon is a monotone polygon where one chain consists of a single edge, the *long edge*. A *uniform* uni-monotone polygon has the vertices evenly spaced with respect to the direction of monotonicity.

Every visibility graph contains a maximal outer-planar graph. Therefore, for a fixed number of vertices, maximal outer-planar graphs have the fewest edges of any visibility graph.

A *staircase* polygon consists of only horizontal and vertical line segments from the positive Y axis down and right to the positive X axis, plus the parts of the axes needed to complete the polygon. If the vertices are equally spaced with respect to the X axis, the polygon is called a *uniform step length* staircase polygon. James Abello and Ömer Eğecioğlu give a polynomial time recognition algorithm using linear programming [1] to recognize visibility graphs of uniform step length staircase polygons

2 Tree of Cliques

We define a new class of graphs, *tree of cliques* graphs as follows: Let H be any two-connected outer-planar graph. We construct a tree of cliques H' by replacing each face of H (except the outer face) with a clique on the same vertices. Both maximal outer-planar graphs and complete graphs are subsets of this new class.

The following theorem gives a full characterization of tree of cliques graphs. This characterization gives both useful properties of tree of cliques graphs, and a practical test to determine if a given graph is a tree of cliques.

Theorem 1 *A graph G is a tree of cliques if and only if:*

1. *G is two-connected.*
2. *Each separation pair (two vertices whose removal disconnects G) is joined by an edge, and breaks G into exactly two pieces.*
3. *Each three-connected component is a clique.*
4. *Within a three-connected component, the edges common with other three-connected components form a subset of a Hamilton cycle on the vertices of the component.*

Proof of Sufficiency:

Suppose we have a graph G that satisfies the conditions listed above. We will show that G is a tree of cliques.

First, G has a Hamilton cycle. This follows from the hypothesis that each three-connected component has a Hamilton cycle which uses the edges that are common to other components (condition 4). Two cycles which have exactly one edge in common, and no other vertices in common, can be joined into a larger cycle which uses all the vertices of both cycles by removing the common edge. In our case, the individual cycles are the Hamilton cycles on each three-connected component. Doing a merge once for each separation pair will combine these cycles to form a Hamilton cycle for G.

We then embed this Hamilton cycle into the plane, with the vertices along the circumference of a circle. Adding the edges corresponding to separation pairs never causes crossing

edges, because if the edges crossed, the vertices of each of the crossing edges would not be a separation pair.

This gives an outer-planar graph in the plane: Outer, because all the vertices are along the outside of a circle; planar, because no edges cross.

Each face of this outer-planar graph corresponds to a three-connected component of G, because the borders of the faces are the separation pairs. But, by hypothesis, each three-connected component is a clique.

We conclude that G is formed from a two-connected outer-planar graph with faces replaced by cliques, and thus G is a tree of cliques graph.

Proof of Necessity:

Suppose G is a tree of cliques graph. We know G is two-connected because of the underlying two-connected outer-planar graph.

Since each face of the outer-planar graph corresponds to a clique on at least three vertices, the components corresponding to the faces are three-connected, and separation pairs can only occur where the components corresponding to two faces join. The removal of the edge joining two faces (and the two vertices of that edge) separates the graph into two components, since the faces of the outer-planar graph form a tree structure. The separation pairs of G are then precisely the joins between faces, and the three-connected components of G are determined by the faces of the underlying outer-planar graph, with the vertices corresponding to each face forming a clique.

Finally, condition 4 is trivially satisfied by using the face itself as the Hamilton cycle in each component. ■

Given the characterization developed above, a recognition algorithm follows directly by verifying the individual properties. This can be done in $O(n+m)$ time, giving a linear time algorithm.

Note that two-connectedness can be checked in linear time [3, pp. 185–187], and there is an $O(n+m)$ time algorithm to find three-connected components [6] which also gives the separation pairs of the graph.

Checking of the remaining conditions in linear time is straight forward.

We prove that all tree of cliques graphs are visibility graphs by giving an extension to ElGindy's algorithm which creates a uniform uni-monotone polygon with the required visibility[1].

However, not every uniform uni-monotone polygon has a visibility graph which is a tree of cliques. Take a six-vertex convex uni-monotone polygon, and move one of the vertices in just enough to cut one line of sight. The resulting polygon is still uni-monotone, but has a visibility graph of K_6 with one edge deleted. This graph is not a tree of cliques.

For the embedding algorithm, we assume that the recognition algorithm has already been used, so that the three-connected components, common edges, and a particular Hamilton cycle have been identified.

Let G be a tree of cliques graph, with m three-connected components (cliques) $G_1, G_2, \ldots, G_m$, and a particular Hamilton cycle as output by the recognition algorithm.

Without loss of generality, let G_1 be a clique containing an edge of the Hamilton cycle and let e be an edge of G_1 belonging to the Hamilton cycle. Label the vertices with integers from 1 to n around the Hamilton cycle so that $e = (v_1, v_n)$.

The algorithm will start with clique G_1, and with vertices v_1 and v_n embedded in the plane at $(1,0)$ and $(n,0)$ respectively, with an edge connecting them.

Since e is on a Hamilton cycle, v_1 and v_n

[1] Parts of these results for tree of cliques graphs may have been discovered independently, as reported by V. Madhaven at the 1990 Canadian Computational Geometry conference.

are not a separation pair of G, and the edge e is not in common with any other clique of G.

The following recursive algorithm POLYGONIZE takes a three-connected component (clique) and an already embedded edge of the clique, and embeds the remaining vertices so that they can see each other and the previously embedded edge of the clique, but are positioned high enough so that they can not see any other embedded vertices of the polygon. The new vertices are embedded in the correct left-to-right order as determined by their labelling, by embedding each vertex v_i along $x = i$.

POLYGONIZE is initialized with the long edge (v_1, v_n) already embedded, and initial parameters (v_1, v_n, G_1).

POLYGONIZE(a, b, H)

Let the coordinates of a vertex v be denoted (v_x, v_y).

The parameters a and b are two previously embedded vertices connected by an edge, with a to the left of b. H is a k-clique of G containing the edge (a, b). POLYGONIZE also uses global information about the input graph G.

1. Remove the edge (a, b) (unless this is the edge (v_1, v_n), which occurs only on the first call to POLYGONIZE).

2. Let u be the previously embedded vertex to the left of a ($u_x < a_x$), if any, that maximizes the slope $r_a = (a_y - u_y)/(a_x - u_x)$. Because the base vertex v_1 lies below and to the left of all other vertices, r_a is always non-negative.

   Similarly let v be the previously embedded vertex to the right of b ($v_x > b_x$), if any, that *minimizes* the slope $r_b = (v_y - b_y)/(v_x - b_x)$. The vertex v_n guarantees that r_b is never positive, since v_n lies below and to the right of all other vertices.

3. We choose the height of the new vertices to be placed to be at least y, the maximum height that a previously placed vertex can see in the horizontal range from a to b. In precise terms, let $y = \max\{a_y, b_y, a_y + r_a(b_x - a_x), b_y - r_b(b_x - a_x)\}$. If either or both of u or v are missing, then remove the corresponding undefined terms from the maximum.

   Vertices placed horizontally between a and b with Y coordinates greater than y will be not be visible to any previously placed vertices except a and b.

4. The next objective is to place the vertices of H vertically above y and in order horizontally between a_x and b_x, so that they can see a, b, and each other.

   We place the new vertices along a semi-circle above the horizontal line at height y, facing downward. Let $v_{i_x} = i$ and $v_{i_y} = y + \sin(\pi(v_{i_x} - a_x)/(b_x - a_x))$. The diameter of this semi-circle extends horizontally the width between a and b, and above both a and b, so a, b, and all the vertices on the semi-circle are mutually visible.

5. Add $k - 1$ edges $(a, v_1), (v_1, v_2), \ldots, (v_{k-2}, b)$.

   For each of these edges (u, v) which is common with another clique G_j, recursively call POLYGONIZE(u, v, G_j).

Proof of correctness omitted.

3 Linear Programming

We develop a different approach using linear programming, to recognize all visibility graphs

of uniform uni-monotone polygons. This linear programming approach was discovered independently by Abello and Eğecioğlu for dealing with uniform step length staircase polygons [1]. We will latter show the strong relationship that exists between uni-monotone polygons and staircase polygons.

Unfortunately, constructing the linear program requires that the Hamilton cycle corresponding to the outside face of the polygon be known. Finding a Hamilton cycle is an NP-complete problem in general; it is not known if this problem is simpler for visibility graphs.

Given a graph G, a Hamilton cycle of G, and an edge of the Hamilton cycle as the long edge (if the edge isn't specified, try each of the n choices), we fix the X coordinates of the vertices (using the Hamilton cycle and long edge information), and represent the Y coordinates as variables. Linear inequality constraints are then determined, based on the visibility graph, and the result is an n variable, $O(n^3)$ constraint linear program which captures all the conditions necessary for a uniform uni-monotone polygon to have the given graph as its visibility graph. Constructing the linear program is done in $O(n^3)$ time, followed by a single call to solve the linear program. Since linear programming problems can be solved in polynomial time [7, p. 170], this gives a polynomial time recognition and embedding algorithm.

For each vertex v_i, we have a variable y_i representing the vertical position of the vertex in the plane. The horizontal position is $x_i = (i-1)/(n-1)$.

We now show how to construct the inequality constraints.

The vertices $v_2, \ldots, v_{n-1}$ of the polygon must lie strictly above the line from v_1 to v_n:

$$y_i > y_1 + \left(\frac{i-1}{n-1}\right)(y_n - y_1), 1 < i < n. \quad (1)$$

In addition, we need up to n more constraints for each pair of vertices v_α and v_β. Without loss of generality, assume v_α lies to the left of v_β, so α is less than β.

How the constraints corresponding to an edge (v_α, v_β) are constructed varies depending on whether the edge is in G.

Case 1: Edge (v_α, v_β) exists in G

Clearly, if v_α and v_β are mutually visible, the top edge of the polygon does not intersect the line of sight between them. This is true when the vertices between v_α and v_β lie above the line connecting them.

For all values of i between α and β, we need that the vertex v_i lies above the line $\overline{v_\alpha v_\beta}$, giving the inequality

$$y_i \geq y_\alpha + \frac{i-\alpha}{\beta-\alpha}(y_\beta - y_\alpha) \quad (2)$$

for each such value of i.

Case 2: Edge (v_α, v_β) does not exist in G

Let v_i be the "rightmost" vertex between v_α and v_β such that v_i is visible from v_α. We can compute i as $\max\{j : \alpha < j < \beta, (v_\alpha, v_\beta) \in E(G)\}$.

The vertex v_i blocks the line of sight from v_α to v_β, leading to the inequality

$$y_i < y_\alpha + \frac{i-\alpha}{\beta-\alpha}(y_\beta - y_\alpha). \quad (3)$$

There are $O(n^3)$ constraints from case 1; each constraint is constructed in constant time. Constructing each of the constraints from case 2 takes time $O(n)$, but there are only $O(n^2)$ such constraints. The total number of constraints is $O(n^3)$, and constructing them takes time $O(n^3)$.

The constructed linear program has a solution if and only if G is a visibility graph of some uniform uni-monotone polygon (proof omitted).

This does not recognize all uni-monotone polygons, even if the Hamilton cycle corresponding to the outside face is specified. There exists a seven vertex uni-monotone polygon whose visibility graph (with the outer face specified) cannot be embedded uniformly. Example (derived from a staircase polygon in [1]) and proof omitted.

4 Staircase Polygons and Monotone Polygonal Chains

If we remove the long edge from a uni-monotone polygon, we get a monotone polygonal chain. We define the visibility of such a chain to be along a distinguished side of the chain (analogous to the interior of a polygon). The linear programming approach presented above works equally well for recogonizing visibility graphs of monotone polygonal chains; a uni-monotone polygon is just the special case where the ends of the chain are mutually visible.

The strong relationship between the visibility graphs of the two classes of polygons turns up in the recognition problems. For staircase polygons, the recognition problem is "Given a graph G, is there a staircase polygon with visibility graph G?" An instance of this recognition problem can be reduced in linear time to a specialized recognition problem for monotone polygonal chains, "Given a graph G and a vertex ordering π, is there a monotone polygonal chain with visibility graph G, and ordering π?"

The reverse reduction is also linear time. For uni-monotone polygons (instead of chains) the Hamilton cycle and long edge take the place of the ordering π, and the corresponding staircase polygon will have the axis vertices mutually visible. The reductions work the same.

In the visibility graph of a staircase polygon, the Hamilton cycle is unique, and can be determined in linear time. The corresponding feature to the long edge of a uni-monotone polygon, namely the axis vertices, can also be found in linear time.

Details and proofs of the reductions omitted.

References

[1] James Abello, Ömer Eğecioğlu, *Visibility Graphs of Staircase Polygons with Uniform Step Length*, manuscript 1991.

[2] James Abello, Ömer Eğecioğlu, Krishna Kumar, *Recognizing Visibility Graphs of Staircase Polygons*, manuscript 1991.

[3] Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley Publishing Company (1974)

[4] Paul Colley, *Visibility Graphs of Unimontone Polygons*, Master's thesis, University of Waterloo, 1991.

[5] Hazel Everett, *Visibility Graph Recognition*, Ph.D. dissertation, University of Toronto; also University of Toronto technical report 231/90 (January 1990)

[6] J.E. Hopcroft, R.E. Tarjan, *Dividing a Graph into Triconnected Components*, SIAM J. Computing 2:3, pp. 135–158, 1973.

[7] Christos H. Papadimitriou, Kenneth Steiglitz, *Combinatorial Optimization*, Prentice Hall, 1982.

[8] Joseph O'Rourke, *Art Gallery Theorems and Algorithms*, Oxford University Press (1987)