ELSEVIER

Computational Geometry 7 (1997) 161–172

Computational Geometry Theory and Applications

Visibility graphs of towers [☆]

Paul Colley [a,1], Anna Lubiw [b,*], Jeremy Spinrad [c,2]

[a] *Queen's University, Kingston, Ontario, Canada*

[b] *University of Waterloo, Department of Computer Science, Waterloo, Ontario, N2L 3G1, Canada*

[c] *Vanderbilt University, Nashville, TN, USA*

Communicated by Jorge Urrutia; submitted 7 April 1993; accepted 15 August 1995

Abstract

A *tower* is a polygon consisting of two reflex chains sharing one common endpoint, together with one edge joining the other endpoints of the chains. A linear time algorithm is given to recognize the [vertex] visibility graphs of towers, and these graphs are characterized as bipartite permutation graphs with an added Hamiltonian cycle. Similar results have been obtained independently by Choi, Shin and Chwa (1992).

1. Introduction

Given a simple polygon P in the plane with vertex set V, the [*vertex*] *visibility graph* of P, denoted $G(P)$, is a graph on vertex set V with an edge between two vertices iff they are *visible* in P—i.e., the line segment joining them remains inside the closed region of the plane bounded by P. Note that by this definition the edges of P are edges of $G(P)$. See Fig. 1.

No one knows how to efficiently recognize visibility graphs—i.e., to decide given a graph, whether it is the visibility graph of some polygon. Nor is any alternative characterization of these graphs known. Everett and Corneil [11] solved these problems for the special case of *spiral* polygons, which consist of one reflex and one convex chain. They gave a polynomial time algorithm to recognize visibility graphs of spiral polygons, and they characterized these graphs as a subclass of *interval* graphs.

In this paper we will solve these problems for the special class of polygons that we call *towers*: We provide a linear time algorithm to recognize visibility graphs of *towers*, and we characterize them in terms of *bipartite permutation graphs*. Recent independent work of Choi, Shin and Chwa [4] also gives a linear time algorithm to recognize visibility graphs of towers (which they call "funnels"), and offers several different characterizations of them.

[☆] Work supported in part by NSERC.

[*] Corresponding author. E-mail: alubiw@uwaterloo.ca.

[1] E-mail: colley@qucis.queensu.ca.

[2] E-mail: spin@vuse.vanderbilt.edu.

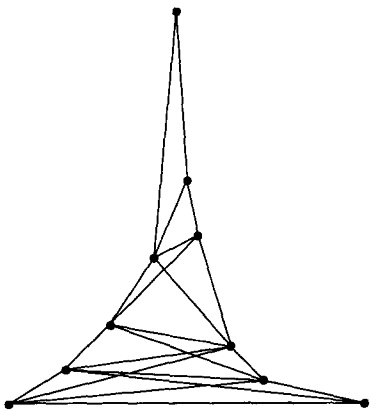
Fig 1.

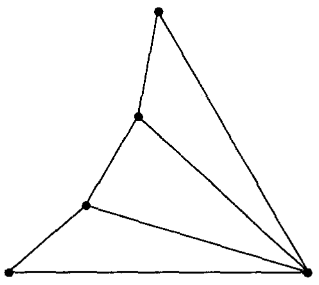
Fig. 2.

A *tower* is a simple polygon formed by two reflex chains plus one edge called the *base* of the tower. The two reflex chains must share one vertex, which is called the *tip* of the tower. See Fig. 1. (For basic definitions of *simple polygon*, *reflex chain*, etc., see the end of this section.) We will exclude the trivial case of a triangle. However, we will allow one of the reflex chains to be just an edge; in this case the designations of tip and base are not unique. See Fig. 2.

One natural unification of this work with that of Everett and Corneil would be to characterize visibility graphs of polygons which consist of one convex and two reflex chains. Spiral polygons are the special case when one of the reflex chains is trivial; towers are the special case when the convex chain is trivial.

Another natural extension of our work (this was in fact our route to the problem) would be to characterize the visibility graphs of *unimonotone* polygons. A polygon is *unimonotone* if its vertices are nondecreasing in the direction of some line. The edge of the polygon joining the last and first vertices is called the *long edge*. Note that the direction of monotonicity will not in general be parallel to the long edge. Towers are a simple special case—perhaps the simplest apart from convex polygons. In Colley's thesis [5,6] he showed that recognizing visibility graphs of unimonotone polygons *given* the Hamiltonian cycle that must form the polygon boundary is equivalent to recognizing visibility graphs of *staircase* polygons. This latter problem has recently been solved by Abello, Egeciolgu and Kumar [1,2].

Visibility graphs

O'Rourke's book [16] is a good source of information on visibility graphs, and [17] gives a recent update.

As mentioned above, no one knows a polynomial time algorithm to recognize visibility graphs. No one has a proof that the problem is NP-complete, or even in NP; the most obvious way to put the problem in NP requires bounding the number of digits needed to represent the vertex coordinates. The problem *is* in P-space, as shown by Everett [10].

Other results on visibility graphs involve restricting the class of graphs, or adding extra information to the graph. El-Gindy [9] proved that any maximal outerplanar graph is a visibility graph (see [16] for an exposition). Coullard and Lubiw [7] showed that visibility graphs can be recognized given the additional information of the Euclidean distances along visibility edges. Ghosh [13] gave some necessary conditions for a graph to be a visibility graph when the Hamiltonian cycle forming the polygon boundary is specified. Everett, Lubiw and O'Rourke [12] considered the problem of recovering the convex hull of the polygon from the *outer* visibility graph. Lin and Skiena [15] show that many graph problems become no easier when restricted to visibility graphs, and consider the grid size needed to represent a graph as a visibility graph.

Bipartite permutation graphs

We will denote graphs as $G = (V, E)$ where V is the vertex set and E is the edge set. A *Hamiltonian cycle* in a graph is a simple cycle including every vertex. A graph is *bipartite* if its vertices can be partitioned into two sets, with no edge joining vertices from the same set.

A graph on vertex set $V = \{1, 2, \ldots, n\}$ is a *permutation graph* if there is a permutation π of V such that for $u < v$, (u, v) is an edge iff $\pi(u) > \pi(v)$. By a *bipartite permutation graph* we mean a graph that is both bipartite and a permutation graph.

Bipartite permutation graphs have been studied by Brandstädt, Spinrad and Stewart [3]. They present a linear-time recognition algorithm for graphs in the class and polynomial-time algorithms for the Hamilton circuit, crossing number, and minimum fill problems. Steiner and Stewart [20] showed that the jump number problem can be solved in polynomial time for the related class of *two dimensional bipartite partial orders*: these are permutation graphs in which edge (u, v), $u < v$, is directed from u to v.

We will make use of an alternate characterization of bipartite permutation graphs. A *strong ordering* of a bipartite graph, with vertex set partitioned into two independent sets U and V, is an ordering of U and an ordering of V such that if $u, u' \in U$ with $u < u'$, and $v, v' \in V$ with $v < v'$, and there are edges (u, v') and (u', v), then there are edges (u, v) and (u', v'). Intuitively, if the vertices of U and V are placed in order on opposite sides of a rectangle, then whenever two edges cross the four vertices induce a complete bipartite subgraph.

Theorem 1.1 (Spinrad, Brandstädt and Stewart [19]). *A bipartite graph is a permutation graph iff it has a strong ordering.*

If G is a connected bipartite graph, it follows immediately that a strong ordering of G must also have the *adjacency property*, which says neighbors of every vertex occur consecutively in the ordering of U or V.

We will have occasion to test if an ordering is a strong ordering. This can be done in linear time. First test that the adjacency property holds. If so, then each vertex i of U is adjacent to some range of vertices $[\min(i), \max(i)]$ of V. It suffices to verify that the sequence $\{\min(i)\}$ is increasing in i, and the sequence $\{\max(i)\}$ is increasing in i.

Basic polygon definitions

A polygon P in the plane is specified by a cyclically ordered sequence of distinct points $p_0, \ldots, p_{n-1}$, called the *vertices*. The *edges* of P are the closed line segments (p_i, p_{i+1}), $i = 0, \ldots, n-2$, and (p_{n-1}, p_0). P is *simple* if two edges intersect only at a common endpoint. We will use "polygon" to mean "simple polygon". The Jordan curve theorem implies that the plane minus the edges of P is divided into two open regions, the bounded interior of P and the unbounded exterior. (Courant and Robbins [8] give an elementary proof of the Jordan curve theorem for polygons.) Two vertices of P are *visible* or *see* each other if the line segment joining them contains no points exterior to P.

The line segment joining two visible vertices that are not joined by an edge of the polygon is a *chord*. Two chords *cross* if they intersect at some point other than a common endpoint.

A *reflex chain* of the polygon P is a subsequence of the vertices, $p_i, p_{i+1}, \dots, p_j$ (subscript addition modulo n), with the property that the interior angle of the polygon formed at any of the vertices $p_{i+1}, \dots, p_{j-1}$ is at least 180°. Two vertices p_a and p_b of a reflex chain see each other iff a and b differ by 1.

For present purposes a *subpolygon* of a polygon P is a polygon S whose interior is a subset of the interior of P, whose vertices are vertices of P, and whose edges are edges or chords of P. A *triangulation* of a polygon P is a set of triangular (i.e., 3-vertex) subpolygons of P such that their interiors are pairwise disjoint, and every point in the interior or boundary of P is a point in the interior or boundary of some triangle. It can be shown that every polygon has a triangulation (see [16]), and that a triangulation is coextensive with a maximal set of noncrossing chords—the chords form edges of the triangles. In particular, any chord can be used in some triangulation.

2. Structural properties

In this section we establish some properties of visibility graphs of towers, culminating in a characterization of these graphs in terms of bipartite permutation graphs.

Let P be a tower polygon, formed by the reflex chains $p_0, p_1, \dots, p_m$ and $q_0, q_1, \dots, q_n$, where $p_0 = q_0$ is the tip vertex, and (p_m, q_n) is the base edge.

Lemma 2.1. *Any vertex except the tip sees a non-empty interval subset of the vertices of the opposite chain (i.e., the adjacency property holds).*

Proof. Without loss of generality, consider vertex p_i, $i > 0$. We will first show that p_i sees some vertex of the opposite chain. Triangulate the polygon. The edge (p_{i-1}, p_i) must be in some triangle. Since no other vertex of the p chain sees both p_{i-1} and p_i, the third vertex of the triangle must be on the q chain (and not q_0). Thus p_i sees a vertex of the opposite chain.

Next we will show that p_i sees an interval subset of the opposite chain. Suppose p_i sees q_a and q_c with $a < b < c$. We will show that p_i sees q_b. The chords (p_i, q_a) and (p_i, q_c) do not cross, so triangulate using them. The only way that the subpolygon formed by $p_i, q_a, q_{a+1}, \dots, q_c$ can be triangulated is by using all chords of the form (p_i, q_j), for $a < j < c$. Thus p_i must see q_b. □

Lemma 2.2. *If $i < r$, $j < s$, p_i sees q_s and p_r sees q_j, then p_i sees q_j and p_r sees q_s (i.e., we have a strong ordering). See Fig. 3.*

Proof. We will show that p_i sees q_j; the other result is symmetrical. Triangulate using the chord (p_r, q_j). The edge (p_i, p_{i+1}) must be in some triangle, and this triangle must use a vertex of the q chain, say q_a. Furthermore, a must be less than or equal to j, because the chord (p_i, q_a) cannot

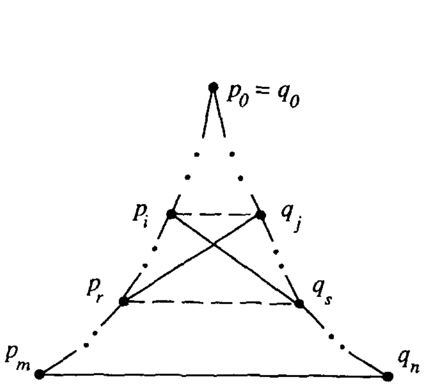

Fig 3.

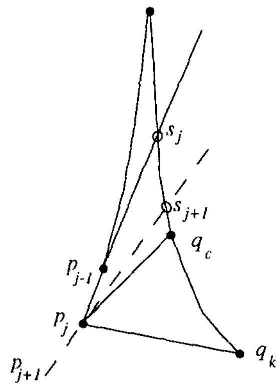

Fig. 4.

cross the chord (p_r, q_j). Now p_i sees q_a and q_s so by Lemma 2.1 it must see q_j, which lies between them. □

Lemma 2.3. *If p_i sees q_j, then p_i sees q_{j+1} and/or p_{i+1} sees q_j (or $i = m$ and $j = n$).*

Proof. Supposing that (p_i, q_j) is not the base, triangulate the polygon using chord (p_i, q_j). Some triangle must use this chord and a vertex later in the p or q chain. This vertex must see both p_i and q_j. Since only p_{i+1} and q_{j+1} can possibly see both p_i and q_j, therefore one of them must. □

Theorem 2.4 (Characterization of visibility graphs of towers). *Removing the edges of the reflex chains from a visibility graph of a tower gives an isolated vertex (the tip) plus a connected bipartite permutation graph, for which the ordering of the vertices in the chains provides a strong ordering. Conversely, any connected bipartite permutation graph arises in this manner from some tower, and furthermore, such a tower can be constructed in linear time on a real RAM.*

Theorem 2.4 implies that if the Hamiltonian cycle that must form the polygon boundary is specified, along with the tip vertex and the base edge, then recognizing visibility graphs of towers is simply a matter of testing whether the derived graph is bipartite, and the induced ordering is a strong ordering. These tests can be done in linear time. (If the Hamiltonian cycle is specified, but the tip and base are not, we can still test in polynomial time, at worst by trying all possibilities for the tip and base.) The more complicated problem that arises when we are not given the Hamiltonian cycle is solved in the next section.

Proof of Theorem 2.4. Removing the edges of the reflex chains of a tower polygon isolates the tip $p_0 = q_0$; the remaining graph G' is bipartite, since vertices on a common chain see each other iff they are consecutive on the chain. Lemma 2.1 implies that G' has no isolated vertices. Suppose $\{p_1, \ldots, p_i\} \cup \{q_1, \ldots, q_j\}$ is a maximal prefix set of vertices which induce a connected subgraph. This set will be non-empty since p_1 and q_1 are joined by an edge. The last vertices p_i and p_j will be joined by an edge by Lemma 2.2. But then we must have $i = m$ and $j = n$, otherwise Lemma 2.3 would allow us to add the next vertex of one of the chains. Thus the graph G' is connected.

The ordering of the vertices in the chains provides a strong ordering of G' by Lemma 2.2. Thus by Theorem 1.1, the bipartite graph is a permutation graph.

It remains to prove that any connected bipartite permutation graph arises from a tower in this way.

Tower construction algorithm

Let G be a connected bipartite permutation graph, with vertices partitioned into independent sets $P = \{p_1, \ldots, p_m\}$ and $Q = \{q_1, \ldots, q_n\}$, ordered according to a strong ordering. Form a graph G' by adding an edge between any two consecutive vertices of P, adding an edge between any two consecutive vertices of Q, and adding one new vertex $p_0 = q_0$ adjacent to p_1 and q_1. Let $N = n+m+1$ be the number of vertices of G'. We claim that G' is a visibility graph of a tower polygon with chains $\{p_0\} \cup P$ and $\{q_0\} \cup Q$. We must show how to construct this tower.

As we construct the tower we will also explicitly construct the points s_i, $i = 1, \ldots, m$, and r_i, $i = 1, \ldots, n$, where s_i is the intersection of the line through p_i and p_{i+1} with the Q chain, and r_i is the intersection of the line through q_i and q_{i+1} with the P chain. We begin by placing $p_0 = q_0$, which will be the tip of the tower, at the point $(0, 0)$; q_1 at $(2^N, -2^N)$; and p_1 at $(-2^N, -2^N)$. Note that $s_1 = r_1 = p_0 = q_0$. Note that p_1 and q_1 must each be connected to something in G, and thus by the strong ordering property they must be joined to each other.

At a given stage we will have placed the points p_1 through p_j, s_1 through s_j, q_1 through q_k, and r_1 through r_k, and we will describe a method for placing either p_{j+1}, s_{j+1} or q_{k+1}, r_{k+1}. Note that the placement of p_j and q_k should be thought of as "extensible", since we will allow these vertices to be placed on extensions of their current segments, while the placements of the other points are completely fixed. The points will be placed so that the following conditions hold:

(1) the chain $p_0, \ldots, p_j$ is a reflex chain of line segments with slopes strictly between 180° and 270° (angle measured counter-clockwise from the positive x axis);
(2) the chain $q_0, \ldots, q_k$ is a reflex chain of line segments with slopes strictly between 270° and 360°;
(3) we have the correct set of visibilities among $p_0, \ldots, p_j, q_0, \ldots, q_k$;
(4) p_j is adjacent to q_k;
(5) no three vertices from $p_0, \ldots, p_j, q_0, \ldots, q_k$ are collinear.

Observe that these conditions hold for the initial placement, and that establishing these conditions for the final placement suffices to prove the lemma.

So assume that p_1 through p_j, s_1 through s_j, q_1 through q_k, and r_1 through r_k have been placed, and that the above conditions hold. We claim that if some vertex is unplaced, either p_{j+1} is adjacent in G to q_k, or q_{k+1} is adjacent in G to p_j: Since G is connected, there must be some edge e which connects $\{p_1, \ldots, p_j\} \cup \{q_1, \ldots, q_k\}$ to $\{p_{j+1}, \ldots, p_m\} \cup \{q_{k+1}, \ldots, q_n\}$. If e connects p_i, $i \leqslant j$, to q_l, $l > k+1$, then any edge out of q_{k+1} crosses e or (p_j, q_k), which implies that q_{k+1} is adjacent to p_i or p_j by the strong ordering property. If q_{k+1} is adjacent to p_i and $i < j$ then, since this edge crosses the edge (p_j, q_k), thus q_{k+1} is adjacent to p_j by the strong ordering property. On the other hand, if e connects q_i, $i \leqslant k$, to p_l, $l > j+1$, then a symmetric argument shows that p_{j+1} is adjacent to q_k.

We choose to place whichever vertex of $\{p_{j+1}, q_{k+1}\}$ is adjacent to the most recently placed vertex of the opposite chain. We will discuss placement of p_{j+1}; the placement of q_{k+1} can be done in an analogous fashion. Follow the construction in Fig. 4. Vertex p_{j+1} is adjacent to $q_c, \ldots, q_d$, $c \leqslant k$, $d \geqslant k$. Note that $c \geqslant 1$ since p_{j+1} is not adjacent to q_0. Note also that p_j is adjacent to q_c: apply the strong ordering property to (p_j, q_k) and (p_{j+1}, q_c).

We will first ensure that p_j is not above q_c by pulling p_j further along the line $p_{j-1}p_j$: to be concrete, replace p_j by $2p_j - p_{j-1}$ (the reflection of p_{j-1} in the point p_j) until p_j is not above q_c. Observe that the line through p_{j-1} and p_j is not changed by this procedure. The position of p_j becomes fixed at this point.

Consider s_j. Note that s_j is strictly above q_c since p_j sees q_c. Construct s_{j+1} as follows: if s_j is above q_{c-1} then let s_{j+1} be the midpoint between q_{c-1} and q_c; otherwise s_j is in the edge (q_{c-1}, q_c)—let s_{j+1} be the midpoint between s_j and q_c. Observe that s_{j+1} is strictly above p_j. Now let p_{j+1} be $2p_j - s_{j+1}$—this is the reflection of s_{j+1} in the point p_j. Finally, we want to ensure that p_{j+1} is lower in the plane than q_k, so that it sees all the vertices q_c through q_k. To do this, replace p_{j+1} by $2p_{j+1} - p_j$ (the reflection of the point p_j in the point p_{j+1}) until p_{j+1} is below q_k.

Observe that p_{j+1}, p_j, s_{j+1} are collinear, that the slope of the new segment (p_j, p_{j+1}) is between 180 and 270 degrees, and that the two segments touching p_j form an angle greater than 180 degrees. Thus condition (1) holds. We have added nothing to the Q chain, so condition (2) still holds.

To show that condition (3) holds observe first that moving p_j further along the segment (p_{j-1}, p_j) does not alter the visibilities. Next observe that we placed p_{j+1} in such a way that it does not see vertices $q_0, \ldots, q_{c-1}$ and does see vertices $q_c, \ldots, q_k$—exactly the edges we need. Observe further that placing p_{j+1} does not alter any previous visibilities. Condition (4) holds by our choice of p_{j+1}. Condition (5) holds by construction.

Observe that this algorithm can be implemented in linear time on a real RAM. □

3. Recognition algorithm

In this section we turn to the problem of recognizing visibility graphs of towers. Given a graph $G = (V, E)$, we wish to test if it is a visibility graph of a tower, and to produce such a tower if so. We can use the algorithm of the previous section to construct a tower, but only after we have identified a Hamiltonian cycle whose removal leaves a bipartite permutation graph. That is the problem to be solved in this section.

We need to assign each vertex to one chain or the other, and to order each chain. Actually, we will do these two things in reverse. We will first identify the tip vertex of the polygon, and then place the vertices by pairs into successive levels corresponding roughly to height, starting with the two neighbours of the tip. Each level will have one vertex from each of the two chains, but we will not attempt to decide which vertex belongs to which chain until after all the levels have been assigned. This division of the algorithm into two steps is natural in the sense that the levelling is (almost) unique—if a graph is a visibility graph of more than one tower then the non-uniqueness comes about in the final assignment of vertices to chains.

More formally, the algorithm will place the vertices V into *levels* $l_1, \ldots, l_k$. Levels other than the first and last are called *internal levels*. Levels will have the following properties:

- Each internal level has two vertices; the first level has one vertex, and the last level may have one or two vertices. The first and last levels are disjoint from other levels; internal levels need not be disjoint, though they must be distinct.
- The vertices of level l_i, $1 \leqslant i < k$, are in a unique maximal clique in the graph induced on $(V - \{l_1 \cup l_2 \cup \cdots \cup l_{i-1}\}) \cup l_i$, and that clique is $l_i \cup l_{i+1}$.

For example in Fig. 5 (either picture) the levels are $\{6\}$, $\{5,7\}$, $\{4,7\}$, $\{3,8\}$, $\{2,8\}$, $\{1\}$.

The remainder of this section consists of three parts: in Section 3.1 we consider the algorithmic problem of finding a levelling of a general (or almost general) graph; in Section 3.2 we prove that any visibility graph of a tower has a levelling—thus the algorithm from the first subsection will succeed in finding a levelling; finally, in Section 3.3 we show how to complete the test of whether a graph is a visibility graph of a tower after having found a levelling. Uniqueness will be explored in each subsection.

3.1. Finding a levelling

First consider how to begin a levelling.

Claim 3.1. *If the graph G can be levelled then there are at most two choices for the first level vertex.*

Proof. The vertex in level l_1 must have degree two, since it is in a unique clique with level l_2, which must have two vertices and be disjoint from level l_1. We claim that there can be at most two vertices of degree two in G: any vertex in an internal level must be in at least two maximal cliques, each of size at least 3; if the last level has two vertices then they have degree at least three; and if the last level has a single vertex then it may have degree two. □

Rather than give an algorithm to find a levelling of a completely general graph, we will make use of one simple property possessed by visibility graphs of towers. Our algorithm will thus either discard a graph because it does not possess this property, or find a levelling of the graph, or declare that no levelling exists. Here is the property we will use.

Claim 3.2. *In a visibility graph of a tower a maximal clique of size three contains an edge that is in no other maximal clique.*

Proof. A clique of size three must contain two vertices of one chain, and it is easy to see that the edge joining them is in no other maximal clique. □

Theorem 3.3. *If G is a graph such that any maximal clique of size three contains an edge in no other maximal clique, then G can be levelled in at most one way given the vertex for the first level. There is a linear time algorithm to find a levelling of a graph, or declare that none exists, or show that the graph violates the condition on maximal 3-cliques.*

Proof. We will describe the algorithm, and point out that in case a levelling is found, it is unique given the level 1 vertex.

Look for degree two vertices in G. If there are none, or are more than two, then by Claim 3.1 G has no levelling. Otherwise, try each vertex of degree two to be the level l_1 vertex.

Given the level l_1 vertex, v, level l_2 must consist of its neighbours. There must be two such neighbours and they must be adjacent, otherwise G has no levelling starting from v. In general, once level l_i, $i \geqslant 2$, is decided, look at the vertices not yet assigned to any level that are adjacent to both vertices of level l_i. They must form a clique, and there must be one or two of them; if not then the graph has no levelling starting from v. In case there are two such vertices they must form level l_{i+1}. Consider the case when there is only one such vertex, u. If it exhausts the vertices then it (alone) forms the last level l_k. Otherwise, we must decide which of the two vertices of l_i to include together with u in level l_{i+1}. The vertices $l_i \cup u$ form a maximal 3-clique. The edge in l_i is in at least two maximal cliques. If both the edges incident to u are in more than one maximal clique then G fails the required property. If exactly one of the edges incident to u is in more than one maximal clique then it must form the next level l_{i+1}. Otherwise neither of the edges incident with u is in more than one maximal clique, and G has no levelling starting from v because we cannot proceed beyond this level.

Observe that the algorithm, if it succeeds in finding a levelling starting from v, has no choices to make. Thus the levelling is unique given the level l_1 vertex.

Note also that the algorithm can be implemented in linear time. □

3.2. A visibility graph of a tower has a levelling

Lemma 3.4. *Any visibility graph of a tower can be levelled. The possibilities for the first level vertex are the tip of the tower and possibly one vertex of the base of the tower. In either case, any size two level will consist of a vertex from each chain.*

Proof. Let P be a tower polygon, formed by the reflex chains $p_0, p_1, \ldots, p_m$ and $q_0, q_1, \ldots, q_n$, where $p_0 = q_0$ is the tip vertex, and (p_m, q_n) is the base edge.

We will prove that the visibility graph of P can be levelled by showing that the algorithm to find a levelling of a graph (given in the proof of Theorem 3.3) succeeds. First consider the choice of a level l_1 vertex. Only the tip, and possibly one of the base vertices of P has degree two, so these are the only possibilities for the level l_1 vertex. Next we will prove by induction on h, that the levelling algorithm succeeds up to level l_h, and furthermore, that the vertices of levels $l_1 \cup \cdots \cup l_h$ consist of $\{p_0, p_1, \ldots p_i\} \cup \{q_0, q_1, \ldots, q_j\}$—in case the tip forms level l_1—or $\{p_i, \ldots p_m\} \cup \{q_j, \ldots q_n\}$—in case the level l_1 vertex is a base vertex.

The basis case is $h = 2$. It is clear that, whether the level l_1 vertex is the tip or a base vertex, level l_2 is successfully found by the algorithm, and the above property holds. Consider the general case of level l_h, supposing that the induction hypothesis holds for l_{h-1}.

Case 1. Level l_1 is the tip. Consider $V' = (V - \{l_1, \ldots, l_{i-1}\}) \cup l_i = \{p_i, \ldots, p_m\} \cup \{q_j, \ldots, q_n\}$. The subgraph induced on these vertices is the visibility graph of a tower with the top chopped off. The only vertices of this set that can possibly be in a clique with p_i and q_j are p_{i+1} and q_{j+1}. By Lemma 2.3, p_i sees q_{j+1} and/or p_{i+1} sees q_j. If both then by Lemma 2.2, p_{i+1} sees q_{j+1} and $\{p_i, q_j, p_{i+1}, q_{j+1}\}$ forms a unique maximal clique in V' containing p_i and q_j. The algorithm thus sets $l_{i+1} = \{p_{i+1}, q_{j+1}\}$, and the induction step holds. Otherwise, suppose that p_i does not see q_{j+1}. Then $\{p_i, q_j, p_{i+1}\}$ forms a unique maximal clique in V' containing p_i and q_j. Furthermore, edge (p_{i+1}, q_j) is in another clique, and edge (p_i, p_{i+1}) is not, so the algorithm sets $l_{i+1} = \{p_{i+1}, q_j\}$, and the induction step holds.

Case 2. Level l_1 is a base vertex. Consider $V' = (V - \{l_1, \ldots, l_{i-1}\}) \cup l_i = \{p_0, \ldots, p_i\} \cup \{q_0, \ldots, q_j\}$. The subgraph induced on these vertices is the visibility graph of a tower with the bottom chopped off. The argument is similar to that in Case 1, just working upward rather than downward. □

3.3. The final recognition algorithm

Our goal is to recognize if a graph G is a visibility graph of a tower. So far, Sections 3.1 and 3.2 show how we can, in linear time, either declare that the graph is not the visibility graph of a tower, or find a levelling of the graph. Furthermore that levelling is unique up to the choice of the level 1 vertex. What do we do next? Lemma 3.4 provides the idea: if the graph G is the visibility graph of a

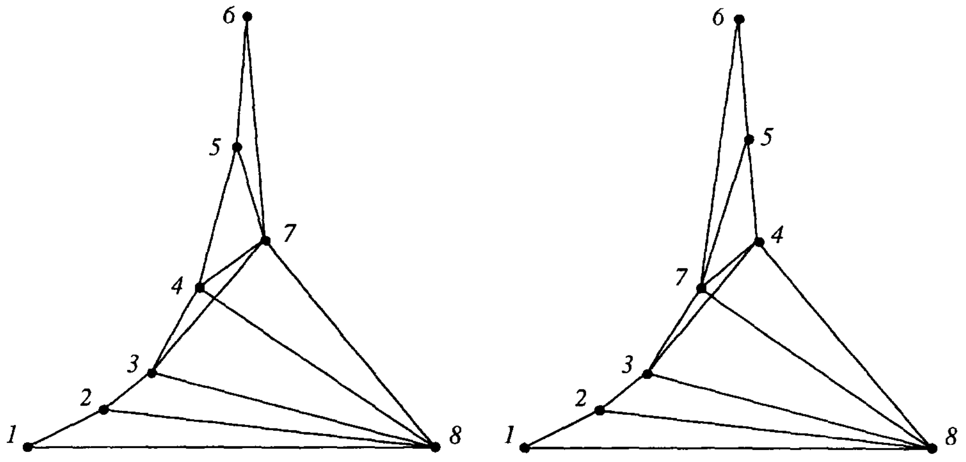

Fig. 5.

tower then it can be levelled and the vertices can be partitioned into two sets A and B (the two reflex chains) such that the following three conditions hold.

(1) Each size two level contains one vertex of A and one vertex of B.

Let G' be the graph formed from G by removing the edges joining vertices of A consecutive in the levelling order, and edges joining vertices of B consecutive in the levelling order.

(2) G' is bipartite.

(3) The level ordering is a strong ordering of G', and G' is connected save for the isolated vertex of level 1.

We will search for a bipartition satisfying conditions (1) and (2), and then test if (3) holds. (As pointed out in the Introduction, such a test can be performed in linear time.) If condition (3) holds then we have a visibility graph of a tower by Theorem 2.4. If condition (3) fails then, by the following claim, the graph is not a visibility graph of a tower.

Claim 3.5. *If G is a visibility graph of a tower, then for any levelling of G, and any bipartition satisfying* (1) *and* (2), *condition* (3) *holds.*

Note the non-uniqueness: if G is a visibility graph of a tower then it has one or two levellings, and for each of these levellings there may be many bipartitions satisfying (1) and (2). See Fig. 5 for an example of two different bipartitions corresponding to one levelling.

Before proving Claim 3.5, we will describe how to find a bipartition satisfying conditions (1) and (2). Given a graph G and a levelling of it, construct a graph G'' by removing from G any edge going from $u \in l_i - l_{i-1}$ to $v \in l_{i+1} - l_i$. Note that G'' still contains an edge joining any two vertices in a common level. Bipartitions of G'' are coextensive with bipartitions of G satisfying (1) and (2). Thus it is simple to find a bipartition of G satisfying (1) and (2) in linear time.

Proof of Claim 3.5. Given a visibility graph G of a tower and a levelling of its vertices, let A, B be a bipartition of its vertices satisfying (1) and (2). Suppose that condition (3) does not hold. The connectivity condition holds simply because we had a levelling. Thus it must be the strong ordering property that fails. So there are vertices $a, b \in A$ and $c, d \in B$ with $a < b$, $c < d$ in the levelling order, with edges (a,d) and (b,c), but no edge (a,c) or no edge (b,d). Suppose the former—the argument for the latter is symmetric. We will show that no bipartition satisfies (1), (2) and (3).

Since there is no edge between them, vertices a and c are not in a common level, nor in adjacent levels. Suppose that a is in earlier level(s) than c. Vertices c and d are not in a common level since they are both in B. Thus vertices a, c, d are in distinct levels, in that order. In particular, a and d are in different levels and, since they are joined by an edge, cannot be on the same side of any bipartition satisfying (2).

Consider the topmost vertex in a common level with vertex c. Call it c'. Note that there is an edge (c,c'), and that a is not in a level adjacent to the level $\{c,c'\}$. Thus there is no edge (a,c'). In any bipartition c and c' must be on opposite sides by condition (1). Since the level ordering is respected, edges (c,c') and (a,d) will cross in any bipartition satisfying (1) and (2). But then, regardless of whether it is c or c' on the opposite side from a, condition (3) is violated. □

To summarize.

Theorem 3.6. *There is a linear time algorithm to test if a graph is a visibility graph of a tower, and—if it is—to construct* (*on a real RAM*) *such a tower.*

4. Conclusions

This paper provides an efficient recognition algorithm for the class of graphs arising as visibility graphs of tower polygons, and proves that, in internal structure, these graphs are precisely bipartite permutation graphs. We thus tie together a simple class of polygons and a known class of graphs.

When a graph is a visibility graph of a tower, we provide an algorithm to draw the tower. A main question we leave open is what size grid is required for this drawing. Our construction is quite bad: it uses doubly exponential area, or in other words, we need exponentially many bits to express the answer. The paper by Lin and Skiena [15] explores this issue of grid size for representing graphs as visibility graphs. They give a family of examples to show that exponential area can be necessary. Each polygon in their family is in fact a tower polygon with one extra vertex. The extra vertex forces the polygon to be a tower. (Note that in general a graph may be the visibility graph of a tower and also the visibility graph of some other kind of polygon.) Eliminating the extra vertex in the Lin–Skiena example yields a family of graphs that can be realized as the visibility graphs of towers only if the towers use exponential area. Thus there is an exponential lower bound and a doubly exponential upper bound on the area required to realize the visibility graphs of towers.

As mentioned in the introduction, towers are contained in the class of *unimonotone* polygons—see [5,6]. Characterizing their visibility graphs seems challenging.

It would also be of interest to unify our results with those of Everett and Corneil [11], by characterizing the visibility graphs of a class of polyons encompassing spiral and tower polygons, for example polygons consisting of one convex and two reflex chains.

References

[1] J. Abello, O. Egecioglu and K. Kumar, Visibility graphs of staircase polygons and the weak Bruhat order I: from visibility graphs to maximal chains, Manuscript (1992).

[2] J. Abello, O. Egecioglu and K. Kumar, Visibility graphs of staircase polygons and the weak Bruhat order II: from maximal chains to polygons, Manuscript (1992).

[3] A. Brandstädt, J. Spinrad and L. Stewart, Bipartite permutation graphs are bipartite tolerance graphs, Congressus Numerantium 58 (1987) 165–174.

[4] S.H. Choi, S.Y. Shin and K.-Y. Chwa, Characterizing and recognizing the visibility graphs of a funnel-shaped polygon, 3rd Internat. Symp. Algorithms Comput. (ISAAC), Japan, 1992. Also, Technical Report CS-TR-92-71, Computer Science Department, Korea Advanced Institute of Science and Technology, Seoul, Republic of Korea, Submitted for publication.

[5] P. Colley, Visibility graphs of uni-monotone polygons, M.Math. Thesis, Dept. of Computer Science, University of Waterloo, Waterloo, Canada, 1991.

[6] P. Colley, Recognizing visibility graphs of unimonotone polygons, in: Proc. 4th Canad. Conf. Comput. Geom., St. John's, Newfoundland (1992) 29–34.

[7] C. Coullard and A. Lubiw, Distance visibility graphs, Internat. J. Comput. Geom. Appl. 2 (1992) 349–362.

[8] R. Courant and H. Robbins, What is Mathematics? An Elementary Approach to Ideas and Methods (Oxford University, London, 1951).

[9] H. El-Gindy, Hierarchical decomposition of polygons with applications, Ph.D. Thesis, McGill University, Montreal, Canada, 1985.

[10] H. Everett, Visibility graph recognition, Ph.D. Thesis, University of Toronto, Toronto, Canada, 1990.

[11] H. Everett and D. Corneil, Recognizing visibility graphs of spiral polygons, J. Algorithms 11 (1990) 1–26.

[12] H. Everett, A. Lubiw and J. O'Rourke, Recovery of convexity from external visibility graphs, Manuscript (1991).

[13] S.K. Ghosh, On recognizing and characterizing visibility graphs of simple polygons, Lecture Notes in Computer Science 318 (Springer, Berlin, 1988) 96–104.

[14] M.C. Golumbic, Algorithmic Graph Theory and Perfect Graphs (Academic Press, New York, 1980).

[15] Lin and Skiena, Complexity aspects of visibility graphs, Manuscript (1992).

[16] J. O'Rourke, Art Gallery Theorems and Algorithms (Oxford University Press, Oxford, 1987).

[17] J. O'Rourke, Computational geometry column 18, SIGACT News 24 (1993) 20–25.

[18] F.P. Preparata and M.I. Shamos, Computational Geometry: an Introduction (Springer, New York, 1985).

[19] J. Spinrad, A. Brandstädt and L. Stewart, Bipartite permutation graphs, Discrete Appl. Math. 18 (1987) 279–292.

[20] G. Steiner and L.K. Stewart, A linear time algorithm to find the jump number of two dimensional partial orders, Order 3 (1987) 359–367.