Straight-Line Drawings of 2-Outerplanar Graphs on Two Curves
(Extended Abstract) *

Emilio Di Giacomo and Walter Didimo

Università di Perugia
({digiacomo,didimo}@diei.unipg.it).

Abstract. We study how to draw a 2-outerplane graph on two concentric circles, so that the internal (resp. external) vertices lie on the internal (resp. external) circle.

1 Introduction

A 1-*outerplanar embedded graph* (or simpler an *outerplane graph*) is an embedded planar graph where all vertices are on the external face. A k-*outerplane graph* G ($k > 1$) is an embedded planar graph such that the embedded graph obtained by removing all vertices of the external face is a $(k-1)$-outerplane graph. In other words, iteratively removing k times from G the vertices of the external face, the graph becomes empty. Each embedded planar graph is a k-outerplane graph for some constant k. The properties of k-outerplane graphs have been studied in several papers (see, e.g., [1,2,4]).

In this paper we concentrate on the problem of drawing 2-outerplane graphs. A natural way to conceive a drawing of a 2-outerplane graph consists of mapping all its vertices on two concentric circles in such a way that all the external vertices lie on the external circle, and all the internal vertices lie on the internal circle. Again, it would be desirable to have straight-line edges with no crossings. We call such a drawing a *2-curve drawing* . The computation of a 2-curve drawing is a problem related to the automatic layout of graphs where vertices are constrained to be on a given set of lines or curves in the plane, and edges bend as little as possible. The main results of this paper are listed below:

– We show that every 2-outerplane graph whose internal vertices induce a biconnected subgraph admits a 2-curve drawing , which can be computed in linear time. Our result is in the middle of two opposite results known in the literature. Namely, in [5] it is shown that every planar graph can be drawn without crossings by mapping its vertices on a circular arc and by drawing every edge as a polyline with at most one bend. However, the class

* Research partially supported by "Progetto ALINWEB: Algoritmica per Internet e per il Web", MIUR Programmi di Ricerca Scientifica di Rilevante Interesse Nazionale.

G. Liotta (Ed.): GD 2003, LNCS 2912, pp. 419–424, 2004.

of graphs that have a planar straight-line drawing where vertices lie on a circular arc coincides with the class of outerplanar graphs, and this class is further reduced if the vertices must lie on two parallel straight lines [3]. Finally, note that if a plane graph admits a planar straight-line drawing where vertices lie on two circular arcs, it is necessarily a k-outerplanar graph with $k \in \{1, 2\}$. Therefore, k-outerplanar graphs with $k \in \{1, 2\}$ are the only graphs that one can hope to draw on two circular arcs with straight-line edges and without crossings.

– We extend our drawing technique to 2-outerplane graphs whose internal vertices induce a simply connected graph. In this case, the computed drawings may have a number of bends that is linear in the number of edges connecting external vertices to cut-vertices of the graph induced by the internal vertices. The problem of deciding whether a general 2-outerplane graph admits a 2-curve drawing remains open.

2 Preliminaries

Let $G = (V, E)$ be a 2-outerplane graph. $V_{ext} \subseteq V$ denotes the set of the vertices on the external face of G, and $V_{int} = V - V_{ext}$ is the set of its internal vertices. Also, $E_{int} = \{(u, v) \in E | u, v \in V_{int}\}$, $E_{ext} = \{(u, v) \in E | u, v \in V_{ext}\}$, and $E_{mix} = \{(u, v) \in E | u \in V_{int}, v \in V_{ext}\}$. Clearly, $E = E_{int} \cup E_{ext} \cup E_{mix}$. $G(V_{int})$ and $G(V_{ext})$ denote the subgraphs of G induced by the vertices of V_{int} and V_{ext}, respectively. Figure 1(a) shows an example of a 2-outerplane graph.

Given any two distinct concentric circles and a horizontal straight line l through the center, denote by Λ_{int} and Λ_{ext} the two semicircles that lie on the upper half-plane defined by l, where Λ_{int} is the semicircle with smaller radius. A *2-curve drawing* of G on Λ_{int}, Λ_{ext} is a planar drawing of G such that: (i) The vertices of V_{int} and V_{ext} are mapped to points of Λ_{int} and Λ_{ext}, respectively; (ii) Each edge of E is drawn as a straight-line segment.

A *biconnected-core 2-outerplane* graph is a 2-outerplane graph G such that $G(V_{int})$ is biconnected. For the sake of brevity, a biconnected-core 2-outerplane graph is called a *BC2O-graph* from now on.

Without loss of generality, in the following sections we always assume that G is a *BC2O-graph* graph where all internal faces are triangles. We call such a graph *quasi-triangulated*. Indeed, if G is not quasi-triangulated we can recursively decompose an internal non-triangular face of G avoiding that a vertex of V_{int} becomes an internal vertex of $G(V_{int})$, and avoiding multiple edges. In this way we obtain a 2-outerplane graph including G and preserving the 2-outerplanar embedding of G. More specifically, each internal face having only vertices of V_{int} or only vertices of V_{ext} can be recursively triangulated with a classical approach. Also, for each internal face f that has both a vertex $u \in V_{int}$ and a vertex $v \in V_{ext}$, such that u and v are not adjacent in G, we can split f with a new edge (u, v) (which will be a new edge of E_{mix}). The obtained graph is still a 2-outerplane graph with the same sets V_{int}, V_{ext} as in G. There exists a simple linear time procedure that can be used to recursively decompose all faces that

have both internal and external vertices. Due to space limitation, we omit the detailed description of such a procedure.

3 The Drawing Algorithm for BC2O-Graphs

Let $G = (V, E)$ be a quasi-triangulated $BC2O$-graph, and let v be a vertex of $G(V_{int})$. Since $G(V_{int})$ is biconnected, v is encountered only once when traversing the external boundary of $G(V_{int})$. Let u and w be the two vertices that immediately precede and succeed v while walking counterclockwise the external boundary of $G(V_{int})$. Denote by $e_{in}(v)$ the edge (u, v) and by $e_{out}(v)$ the edge (v, w). Also, denote by $E(v)$ the circular list of edges incident on v in clockwise order. The following properties hold. Proofs are omitted for reasons of space.

Property 1. Let v be a vertex of $G(V_{int})$. There is no edge $e \in E_{mix} \cap E(v)$ between $e_{in}(v)$ and $e_{out}(v)$ in $E(v)$.

Property 2. Let v be a vertex of V_{ext}, and let $e_i = (v, v_i)$ $(i = 1, \ldots, k)$ be the edges in $E_{mix} \cap E(v)$ in the order they appear in $E(v)$. Vertices $v_1, v_2, \ldots, v_k$ are encountered in this order when traversing counterclockwise the external boundary of $G(V_{int})$.

Property 3. There exists a vertex $v \in V_{int}$ such that there are two consecutive edges of E_{mix} in $E(v)$.

Based on the above properties, a $BC2O$-graph G can be described in terms of a general structure (see also Figure 1(b)) that is useful to prove the correctness of the drawing algorithm. Let v be a vertex of V_{int} with two incident edges $e_w = (v, w')$ and $e_u = (v, u')$ of E_{mix}, that are consecutive in $E(v)$. This vertex exists by Property 3. If w' is adjacent to some vertex of V_{int} distinct from v, let $w = w'$, otherwise let w be the first vertex encountered in counterclockwise order in the adjacency list of v starting from w' such that it is adjacent to some vertex of V_{int} distinct from v. Analogously, if u' is adjacent to some vertex of V_{int} distinct from u, let $u = u'$, otherwise let u be the first vertex encountered in clockwise order in the adjacency list of v starting from u' such that it is adjacent to some vertex of V_{int} distinct from v. Note that u and w always exist and they are distinct since the graph is simple. Let h be the external boundary of $G(V_{int})$. Denote by $u_1, u_2, \ldots, u_r$ the vertices of V_{int} that are adjacent to u, in counterclockwise order walking on h. Also, denote by $w_1, w_2, \ldots, w_s$ the vertices of V_{int} that are adjacent to w, in counterclockwise order walking on h. In the general structure of G we distinguish among three subgraphs, G_1, G_2, and G_3, whose edges partition the set of edges of G (see Figure 1). The three subgraphs are outerplane and thus easy to embed. They are defined as follows: (i) G_1 is the subgraph induced by the following vertices: v, u, w, and all the vertices of V_{ext} that are between u and w while walking on the external boundary of G counterclockwise; (ii) G_2 coincides with $G(V_{int})$; (iii) G_3 is the subgraph whose edges are those of G that are not in G_1 and G_2, and whose vertices are exactly the end-vertices of these edges.

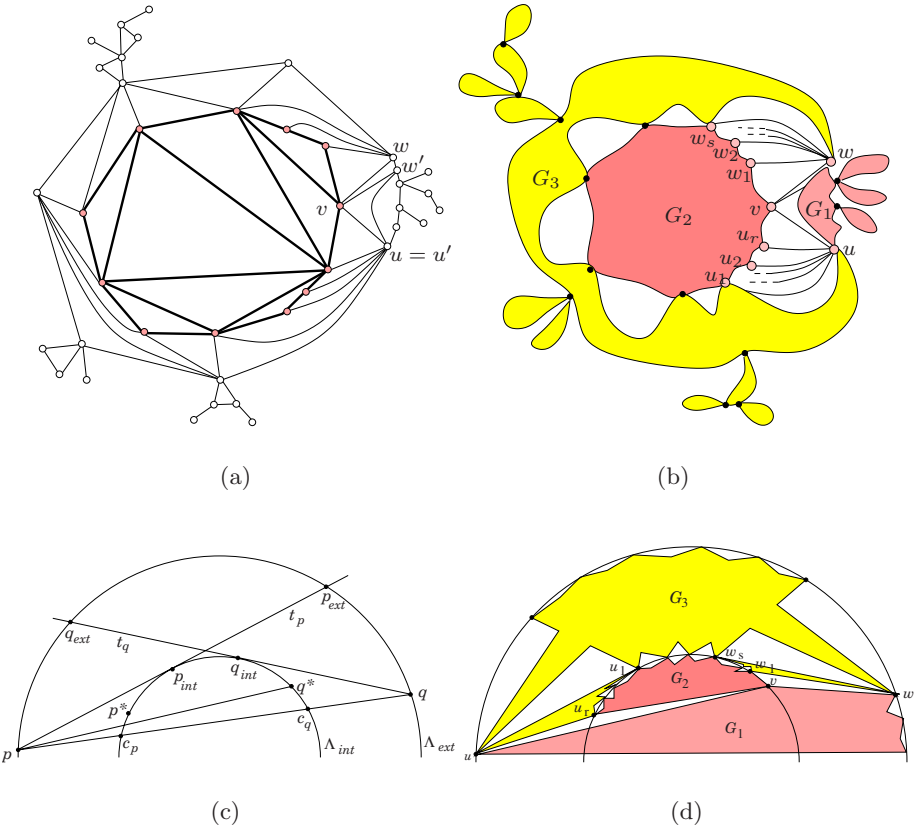

Fig. 1. (a) A 2-outerplane graph G where $G(V_{int})$ is biconnected, that is, a $BC2O$-graph. The edges of $G(V_{int})$ are thick. (b) Structure of G. (c) Notation used in the description of the drawing algorithm. (d) Structure of a drawing computed by the drawing algorithm.

We call 2-CURVE-DRAWER the algorithm that computes a 2-curve drawing of a $BC2O$-graph. The algorithm works as follows (see also Figure 1(c)). First it chooses two distinct points, p and q, of Λ_{ext} such that: (i) the x- and y-coordinate of p is less than the x- and the y-coordinate of q, respectively; (ii) segment $\overline{pq}$ is a chord of Λ_{ext} that has two intersection points c_p, c_q with Λ_{int}, where c_p is the first point encountered while walking on $\overline{pq}$ from p to q; (iii) there are two lines t_p and t_q passing for p and q, respectively, that are tangent to Λ_{int}, and intersecting in a point lying in the portion of the annulus delimited by Λ_{int} and Λ_{ext}. Denote by $p_{ext} \neq p$ and p_{int} the points where t_p intersects Λ_{ext} and Λ_{int}, respectively. Similarly, let $q_{ext} \neq q$ and q_{int} be the points where t_q intersects

Λ_{ext} and Λ_{int}, respectively. Also denote by q^* any point of Λ_{int} between q_{int} and c_q, and by p^* any point of Λ_{int} between p_{int} and the point $\overline{pq^*} \cap \Lambda_{int}$.

Then the algorithm proceeds as follows (see also Figure 1(d)): It maps all vertices of V_{int} to points of Λ_{int}, according to the clockwise order they appear on the external boundary of $G(V_{int})$, in such a way that: (i) u_r and u_1 are mapped to p^* and p_{int}, respectively; (ii) w_s is mapped to q_{int}; (iii) v is mapped to q^*.

Also, it maps all vertices of V_{ext} to points of Λ_{ext}, according to the clockwise order they appear on the external boundary of G, in such a way that: (i) u and w are mapped to p and q, respectively; (ii) all vertices from u to w are mapped to points between q_{ext} and p_{ext} (iii) all vertices from w to u are mapped to points below q.

Lemma 1. *Algorithm* 2-CURVE-DRAWER*(G) computes a 2-curve drawing of G.*

Proof. Denote by Γ_1, Γ_2, and Γ_3 the drawings of G_1, G_2, and G_3 in the drawing Γ of G. We first prove that these drawings lie in three distinct connected regions R_1, R_2, and R_3 of the plane whose interiors do not intersect (see Figure 1(d)). This guarantees that there is no crossing between edges of Γ_1, Γ_2, and Γ_3. Then, we prove that each Γ_i $(i = 1, 2, 3)$ is planar, thus proving that Γ is planar. Given two points a and b on a circular arc, we denote by $\widehat{ab}$ the sub-arc from a to b. All vertices of G_1 distinct from u, v, and w are mapped to points that are below q on Λ_{ext}. Let z be the lowest of these points. The boundary of R_1 consists of the segments $\overline{zp}$, $\overline{pq^*}$, $\overline{q^*q}$, and the arc $\widehat{qz}$. Since q^* is always above c_q, R_1 is a convex region. Hence, since all vertices of G_1 lie on the boundary of R_1, all edges of G_1 are inside R_1. The vertices of G_2 lie on the boundary of R_2, which is the region bounded by segment $\overline{p^*q^*}$ and arc $\widehat{q^*p^*}$. Since R_2 is convex, the edges of G_2 are inside R_2. The vertices of G_3 lie on the boundary of region R_3, which consists of segment $\overline{pp^*}$, arc $\widehat{p^*q^*}$ on Λ_{int}, segment $\overline{q^*q}$, and arc $\widehat{qp}$ on Λ_{ext}. Although R_3 is not convex, the edges of G_3 are still inside R_3. Indeed, all the vertices of G_3 that are on Λ_{ext}, except u and w, lie on arc $\widehat{q_{ext}p_{ext}}$, and hence edges among these vertices are inside R_3. Also, there is no edge of G_3 among vertices on Λ_{int}. The vertices connected to u are u_i $(i = 1, \ldots, r)$ plus, possibly, some vertices on Λ_{ext} lying on arc $\widehat{q_{ext}p_{ext}}$. On the other hand, by construction, u is mapped on p, p_{ext} is the intersection point between t_p and Λ_{ext}, u_r is mapped to p^*, and u_1 is mapped to p_{int} (which is the intersection point between t_p and Λ_{int}). Therefore, all vertices connected to u are visible from u within R_3, that is, each segment from u to any of its adjacent vertices is inside R_3. The same reasoning can be applied to prove that all the edges incident on w are inside R_3. Finally, the edges of E_{mix} that belong to G_3 are represented by segments that have one end-point on arc $\widehat{q_{ext}p_{ext}}$ of Λ_{ext} and the other on arc $\widehat{p_{int}q_{int}}$ of Λ_{int}. Hence they are inside R_3. The planarity of each Γ_i $(i = 1, 2, 3)$ is guaranteed by the fact that: (i) we maintain the planar embedding of G_i, (ii) all the vertices of G_i are on the boundary of the region R_i, and (iii) any two adjacent vertices of G_i can be connected with a segment inside R_i, as proved above.

Theorem 1. *Let $G = (V, E)$ be a biconnected-core 2-outerplane graph. There exists an $O(|V|)$-time algorithm that computes a 2-curve drawing of G.*

4 Extending the Algorithm

Let $G = (V, E)$ be a 2-outerplane graph whose internal vertices induces a simply connected graph. A *1-bend-2-curve drawing* of G on Λ_{int}, Λ_{ext} is a planar drawing of G such that: (i) The vertices of V_{int} and V_{ext} are mapped to points of Λ_{int} and Λ_{ext}, respectively; (ii) Each edge of E is drawn as a polyline with at most one bend.

Algorithm 2-CURVE-DRAWER can be extended in order to compute a *1-bend-2-curve drawing* of G with a "small" number of bends. Namely, if $G(V_{int})$ is not biconnected, we apply on G a preprocessing step in order to make $G(V_{int})$ biconnected and still outerplane. We sketch an iterative procedure that can be used to do that. Let v be a cut-vertex of $G(V_{int})$ and let u and w be two vertices adjacent to v and such that: (a) u and w belong to distinct biconnected components of $G(V_{int})$, (b) u and w are consecutive in the counterclockwise list of vertices adjacent to v. Add a dummy edge (u, w), splitting the external face of $G(V_{int})$. The new dummy edge may cross a certain number of edges of E_{mix} that are incident on v and that are between u and v in the counterclockwise list of edges incident on v. No other kind of edges are crossed. Replace each crossing with a dummy vertex that will be considered as a new vertex of V_{int}. Note that, dummy vertices are not cut-vertices of the new $G(V_{int})$.

It can be proven that the new $G(V_{int})$ is biconnected and still outerplanar and that each edge of E_{mix} in the original graph is split at most by one dummy vertex. Also, it can be proven that the procedure above takes linear time. To the new embedded graph we apply the algorithm presented in the previous section, so to compute a planar drawing on two circles. At the end, dummy vertices are removed. The removal of each dummy vertex gives rise to one bend on an edge of E_{mix}, so that the resulting drawing is a 1-bend-2-curve drawing of G.

Corollary 1. *Let $G = (V, E)$ be a 2-outerplane graph whose internal vertices induces a simply connected graph. There exists an $O(n)$-time algorithm that computes a 1-bend-2-curve drawing of G with $O(E^{(c)}_{mix})$ bends, where $E^{(c)}_{mix}$ are those edges of E_{mix} incident on cut-vertices of $G(V_{int})$.*

References

1. B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *JACM*, 41(1):153–180, 1994.
2. H. L. Bodlander. Some classes of graphs with bounded tree width. *Bulletin of the European Association for Theoretical Computer Science*, pages 116–126, 1988.
3. S. Cornelsen, T. Schank, and D. Wagner. Drawing graphs on two and three lines. In *Graph Drawing (Proc. GD '02)*, volume 2528 of *LNCS*, pages 31–41, 2002.
4. C. M. D. Bienstock. On the complexity of embedding planar graphs to minimize certain distance measures. *Algorithmica*, 5(1):93–109, 1990.
5. E. Di Giacomo, W. Didimo, G. Liotta, and S. K. Wismath. Drawing planar graphs on a curve. In *Proc. WG '03*, LNCS, to appear.