Abstract

Let $C$ be the family of 2D curves described by concave functions, let $G$ be a planar graph, and let $L$ be a linear ordering of the vertices of $G$. $L$ is a curve embedding of $G$ if for any given curve $\Lambda \in C$ there exists a planar drawing of $G$ such that: (i) the vertices are constrained to be on $\Lambda$ with the same ordering as in $L$, and (ii) the edges are polylines with at most one bend. Informally speaking, a curve embedding can be regarded as a two-page book embedding in which the spine is bent. Although deciding whether a graph has a two-page book embedding is an NP-hard problem, in this paper it is proven that every planar graph has a curve embedding which can be computed in linear time. Applications of the concept of curve embedding to upward drawability and point-set embeddability problems are also presented.

1. Introduction

The problem of drawing a graph in such a way that all its vertices are constrained to be on a given straight line and its edges are polylines with at most one bend has been widely studied in the literature. The straight line is called the spine, and each edge is drawn on a half-plane determined by the spine. In a queue layout edges can cross, but there are no two disjoint edges that are nested. In a stack layout, also
called a book embedding, edge nesting is allowed but there are no edge crossings. Queue and stack layouts can also be defined in a purely combinatorial and equivalent way, i.e., in terms of a linear ordering of the vertices and a partition of the edges into queues or stacks so that they obey FIFO or LIFO disciplines (see e.g. [24,25] for details).

A natural problem related with computing either a queue layout or a book embedding of a graph $G$ is that of minimizing the number of half-planes needed to draw the edges. The queue number and the page number of $G$ are the minimum number of half-planes needed to compute a queue layout and a book embedding of $G$, respectively. While proving that planar graphs have constant queue number remains an outstanding open problem (see e.g. [16,24,25,33]), the counterpart of this question for stack layouts has been completely solved. Bernhart and Kainen [2] show that a graph has a book embedding in the plane if and only if it is sub-hamiltonian. Therefore, it is NP-hard to decide whether a graph has page number two [21]. A ground breaking result by Yannakakis [34] proves that four half-planes are always sufficient to compute a three-dimensional book embedding of a planar graph and that there exist planar graphs whose page number is four.

In this paper we study 2D drawings of graphs where the vertices are constrained to be on a given curve and the edges have at most one bend; informally speaking, these drawings can be seen as two-page book embeddings where the spine is “bent”. Besides being of theoretical interest in its own right, our research has applications to upward drawings [13] and to point-set embeddings [4,5,29]. In order to better outline the results in this paper, we first describe these two types of drawing conventions.

A drawing of a planar digraph is upward planar if the edges are monotonically increasing in a common direction and two distinct edges never cross. Every planar graph $G$ can be oriented in such a way that the resulting directed graph admits an upward planar drawing where the vertices are at integer grid points and the edges are straight lines [13]. The orientation is called an upward planar orientation of $G$; the set of upward planar orientations of plane graphs are characterized in [15]. Much less is known about orienting a graph in such a way that there exists an upward planar drawing where the vertices are constrained to be on a curve or on a straight line. Upward drawings of series-parallel digraphs where the vertices are all collinear and each edge has at most one bend are studied in [1,14].

Let $P$ be a set of $n$ distinct points in the plane and let $G$ be a planar graph with $n$ vertices. A point-set embedding of $G$ on $P$ is a planar drawing of $G$ such that each vertex is mapped to a distinct point of $P$ and each edge is drawn as a polygonal chain connecting its endpoints (note that the mapping of the vertices to the points is not specified as part of the input). If the elements of $P$ are in general position, then any outerplanar graph can be drawn on $P$ with no bends per edge [4,5]. Since outerplanar graphs are the largest class of graphs admitting a straight-line point-set embedding on any set of points [22], Kaufmann and Wiese [29] investigate the problem of computing a point-set embedding of a planar graph with a small number of bends per edge. They show that any planar graph admits a point set embedding with at most two bends per edge on any given set of points, and that two bends are required in some cases. Pach and Wenger show that if the mapping of the vertices of $G$ to the points of $P$ is given, then a planar drawing of $G$ exists with $O(n)$ bends per edge [30]. The problem of deciding if there exists a point set embedding with straight-line edges of a planar graph on a given set of points is, in general, NP-hard [6].

Upward point-set embeddings of directed planar graphs have also been recently studied [14].

The main contributions in this paper can be outlined as follows.

- We introduce and study the concept of a curve embedding of a planar graph. Let $C$ be the family of 2D curves described by concave functions, let $G$ be a planar graph, and let $L$ be a linear ordering of
the vertices of $G$. $L$ is a curve embedding of $G$ if for any given curve $A \in C$ there exists a planar drawing of $G$ such that: (i) the vertices are constrained to be on $A$ with the same ordering as in $L$, and (ii) the edges are polylines with at most one bend. Fig. 1 shows a planar graph $G$, a curve embedding $L$ of $G$, and a drawing of $G$ on a semicircle such that the ordering of the vertices is equal to $L$ and the edges have at most one bend.

- We prove that every planar graph admits a curve embedding and present a linear-time algorithm to compute one. This sharply contrasts with the NP-hardness result for the two-page book embeddability of a graph [2]. We also show that if the curve of $C$ is a semicircle then a drawing that corresponds to the curve embedding can also be computed in linear time.

- We study the interplay between curve embeddings and upward drawings. Namely we show that a curve embedding of a planar graph $G$ can be used to compute an upward planar orientation of $G$ such that the resulting digraph: (i) admits an upward planar drawing where all vertices are collinear and every edge has at most two bends, (ii) admits an upward planar drawing where all vertices lie on a concave curve and every edge has at most one bend.

- We use curve embeddings to simplify and extend the algorithm in [29] for point set embeddability. The algorithm in [29] first computes a hamiltonian augmentation of the input graph via results due to Chiba and Nishizeki [8,9] by four-connecting the graph and then uses the augmented ordering to construct the drawing. We exploit curve embeddings to compute a hamiltonian augmented graph without initially four-connecting the graph and to map the vertices to the points so that all edges of the drawing are monotone in one direction. Our augmentation technique favorably compares with similar techniques presented in the literature [30].

The proof that every planar graph has a curve embedding is constructive. We describe an algorithm that draws the vertices of the input graph on a curve by adding a new vertex and its incident edges at each step; the next vertex to be inserted in the drawing is chosen by using the well-known canonical ordering defined by de Fraysseix, Pach and Pollack [12]. Note however that the computed curve embedding is in general very different from the computed canonical ordering. In graph $G$ of Fig. 1(a) the vertices are numbered according to a canonical ordering of $G$, which is different from the curve embedding $L$ of Fig. 1(b).

Finally, we recall that an increasing number of papers about drawing graphs where vertices lie on given curves or surfaces have been published in recent years. A limited list of references includes [3,7,11,17,18,20,26,31,32]. The interested reader is also referred to [13,28] for additional references.

The remainder of this paper is organized as follows. Some basic definitions are given in Section 2. In Section 3 the algorithm for computing a curve embedding is presented and in Section 4 the applications of curve embeddings to the upward drawability problem and to the point-set embeddability problem are discussed. In Section 5 we give some open problems related to our results.

2. Preliminaries

We assume familiarity with basic graph theory and graph connectivity [23]. Let $G$ be a graph. A drawing $\Gamma$ of $G$ maps each vertex $v$ of $G$ to a distinct point $p(v)$ of the plane and each edge $e = (u, v)$ of $G$ to a simple Jordan curve connecting $p(u)$ and $p(v)$. Drawing $\Gamma$ is planar if no two distinct edges intersect except at common endvertices. Graph $G$ is planar if it admits a planar drawing. A planar
Fig. 1. (a) A planar graph $G$. The vertices are numbered according to a canonical ordering of $G$. (b) A curve embedding $L$ of $G$ and a curve drawing $\Gamma$ of $G$ on a semicircle (the dotted line). $\Gamma$ induces $L$ as curve embedding. Point $v_4$ is after $v_6$ and before $v_5$.

drawing $\Gamma$ of $G$ partitions the plane into topologically connected regions called the faces defined by $\Gamma$. The unbounded face is called the external face. The boundary of a face is its delimiting cycle described by the circular list of its edges and vertices. The boundary of the external face, also called the external boundary, is the circular list of edges and vertices delimiting the unbounded region.

An embedding of a planar graph $G$ is an equivalence class of planar drawings that define the same set of faces, that is, the same set of face boundaries. A planar graph $G$ together with the description of a set of faces $F$ is called an embedded planar graph. A maximal embedded planar graph is such that all faces are triangles, that is, the boundary of each face has three vertices and three edges. Given any embedded planar graph $G$, it is easy to add edges that split the faces of $G$ in order to obtain a maximal embedded planar graph that includes $G$.

**Definition 1.** Let $I = [\alpha, \beta] \subset \mathbb{R}$ be an interval and let $f : I \to \mathbb{R}$ be a function defined on $I$. The set of points $\{(x, f(x)) : x \in I\}$ is a concave curve iff:

- $f(x)$ has a second derivative on $I$;
- $f''(x) < 0$, $\forall x \in I$.

Let $x_1, x_2 \in I$ such that $x_1 < x_2$; we say that $(x_1, f(x_1))$ is before $(x_2, f(x_2))$ and $(x_2, f(x_2))$ is after $(x_1, f(x_1))$.

**Definition 2.** Let $G$ be a planar graph and let $A$ be a concave curve. A curve drawing of $G$ on $A$ is a planar drawing of $G$ such that:

- each vertex $v$ of $G$ is mapped to a unique point $p(v)$ of $A$;
each edge \( e = (u, v) \) of \( G \) is drawn as a polyline with endpoints \( p(u) \) and \( p(v) \); a bend of \( e \) is a common point of two consecutive segments of the polyline representing \( e \).

Fig. 1(b) shows an example of a curve drawing of a planar graph on a concave semicircle. In the drawing, \( v_4 \) is after \( v_6 \) and before \( v_5 \). Each edge has at most one bend.

**Definition 3.** A curve embedding of \( G \) is a linear ordering \( L \) of the vertices of \( G \) such that for each concave curve \( \Lambda \):

- \( G \) has a curve drawing \( \Gamma \) on \( \Lambda \);
- if \( u \) precedes \( v \) in \( L \) then \( p(u) \) is before \( p(v) \) in \( \Gamma \);
- each edge in \( \Gamma \) has at most one bend.

We also say that \( L \) is the curve embedding induced by \( \Gamma \).

Let \( p \) and \( q \) be two points in the plane. We denote by \( pq \) the straight-line segment connecting \( p \) and \( q \).

**Definition 4.** Let \( C = \{c_1, \ldots, c_s\} \) be a set of points on \( \Lambda \), such that \( c_{i+1} \) is after \( c_i \) \((1 \leq i < s - 1)\), and let \( \hat{c}_i \) denote the intersection point between \( \tau(\Lambda, c_i) \) and \( \tau(\Lambda, c_{i+1}) \). The tangent polyline of \( \Lambda \) with respect to \( C \) is the polyline \( \tau(\Lambda, C) = c_1\hat{c}_1 \cup \hat{c}_1c_2 \cup \hat{c}_2c_3 \cup \cdots \cup \hat{c}_{s-1}c_{s-1} \cup \hat{c}_{s-1}c_s \).

**Definition 5.** Let \( C = \{c_1, \ldots, c_s\} \) be a set of points on \( \Lambda \), such that \( c_{i+1} \) is after \( c_i \) \((1 \leq i < s - 1)\). The region \( P(\Lambda, c_i) \) \((c_i \in C)\) is the closed half-plane defined by \( \tau(\Lambda, c_i) \) containing \( \Lambda \). The region \( P(\Lambda, C) \) is the intersection of all \( P(\Lambda, c_i) \) \((1 \leq i \leq s)\).

Fig. 2 illustrates the above definitions. Since each \( P(\Lambda, c_i) \) contains \( \Lambda \), \( P(\Lambda, C) \) also contains \( \Lambda \). The following properties hold.

**Property 1.** \( \tau(\Lambda, C) \) is contained in the boundary of \( P(\Lambda, C) \).

**Property 2.** \( \hat{c}_i\hat{c}_{i+1} \) is contained in \( P(\Lambda, C) \).

Let \( \Gamma \) be a curve drawing of a planar graph \( G \) on a concave curve \( \Lambda \).

**Definition 6.** A point \( p \) on \( \Lambda \) is externally visible if the straight line orthogonal to \( \tau(\Lambda, p) \) and passing through \( p \) does not intersect any point of \( \Gamma \) in the region \( \mathbb{R}^2 - P(\Lambda, p) \). An interval \( I = [\alpha, \beta] \) on \( \Lambda \) is externally visible if all points of \( I \) are externally visible.

**Definition 7.** Let \( y_M \) be the maximal \( y \)-coordinate of any point of \( \Gamma \), and let \( e \) and \( e' \) be any two distinct edges of \( G \). We say that \( e \) is externally covered by \( e' \) in \( \Gamma \) if for each point \( p \) with \( y \)-coordinate greater than \( y_M \) and for each point \( q \) of \( e \), the segment \( pq \) intersects \( e' \).

Intuitively, if \( e \) is externally covered by \( e' \) no point of \( e \) can be reached by a straight-line segment from a point that has “infinite” \( y \)-coordinate without intersecting \( e' \). For example, in Fig. 1(b) edges \((v_4, v_3)\) and \((v_3, v_2)\) are externally covered by edge \((v_4, v_2)\).
In Section 3 we describe an algorithm that computes a curve embedding of a planar graph. This algorithm makes use of a particular ordering of the vertices introduced by de Fraysseix et al., which is known as a canonical ordering [12].

**Definition 8 (Canonical ordering).** [12] Let $G$ be a maximal embedded planar graph with external boundary $u, v, w$. A canonical ordering of $G$ with respect to $u, v$ is an ordering of the vertices $v_1 = u, v_2 = v, v_3, \ldots, v_n = w$ of $G$ with the following properties for every integer $k$ such that $4 \leq k \leq n$:

- The subgraph $G_{k-1}$ of $G$ induced by $v_1, v_2, \ldots, v_{k-1}$ is biconnected and the external boundary $C_{k-1}$ contains edge $(u, v)$;
- $v_k$ is in the external face of $G_{k-1}$, and its neighbors in $G_{k-1}$ form a subpath of the path $C_{k-1} - (u, v)$.

Fig. 3 illustrates the properties of a canonical ordering. Fig. 1(a) shows a canonical ordering of a maximal embedded planar graph.

The following lemma states a known result about the time complexity of computing a canonical ordering (see e.g. [10,27]).

**Lemma 1.** Let $G$ be a maximal embedded planar graph with external boundary $u, v, w$. A canonical ordering of $G$ with respect to $u, v$ can be computed in $O(n)$ time, where $n$ is the number of vertices of $G$.

We introduce some further notation that will be subsequently used. Let $p$ and $q$ be two distinct points in the plane. We denote by $l(p, q)$ the straight line passing through $p$ and $q$. The perpendicular bisector
of the segment $\overline{pq}$ is the straight line orthogonal to $\overline{pq}$ and passing through the middle point of $\overline{pq}$. Given a straight line or a segment $l$ that forms an angle $\theta$ with the $x$-axis, we denote by $\sigma(l) = \tan(\theta)$ the slope of $l$.

3. Computing curve embeddings

In this section we give an algorithm that finds a curve embedding of a planar graph $G$ in linear time. We first describe a strategy to compute a curve drawing of $G$ on a concave semicircle with at most one bend per edge, and then we show how the same strategy can be used to compute a curve drawing on any concave curve, maintaining the linear ordering of the vertices along the curve. This ordering is the desired curve embedding. Since each embedded planar graph $G$ can be easily augmented in linear time to a maximal embedded planar graph that includes $G$, in the following we concentrate on maximal planar graphs.

We start by defining the invariant properties to be maintained by our algorithm. Let $G$ be a maximal embedded planar graph with external boundary $u, v, w$, and let $u = v_1, v = v_2, \ldots, v_n = w$ be a canonical ordering of $G$ with respect to $u, v$. Let $G_k$ be the subgraph of $G$ induced by $v_1, \ldots, v_k$ and let $C_k: u = c_1, \ldots, c_s = v$ be the external boundary of $G_k$. Denote by $\Gamma_k$ a curve drawing of $G_k$ on a concave curve $\Lambda$ such that the following properties hold:

Property P0: No vertex of $G_k$ is drawn on the endpoints of $\Lambda$ in $\Gamma_k$;
Property P1: For each $c_i \in C_k$, the edges of $G_k$ incident to $c_i$ are drawn in $\Gamma_k$ as polylines contained in the half-plane $P(\Lambda, c_i)$;
Property P2: The external boundary of $\Gamma_k$ is equal to $C_k$ and for each pair $c_i, c_{i+1} \in C_k$, $c_i$ is drawn before $c_{i+1}$ on $\Lambda$.
Property P3: Edge $(u, v)$ has 0 bends. Each edge $e = (a, b) \neq (u, v)$ of $\Gamma_k$ has either 0 or 1 bend, according to the following rules:

- if $e$ has 0 bends then $|e \cap \Lambda| = 2$, i.e., $e$ intersects $\Lambda$ only at its endpoints $a$ and $b$;
- if $e$ has 1 bend then $|e \cap \Lambda| = 4$ (see also Fig. 4). Denote by $a = z_0, z_1, z_2, z_3 = b$ the four intersection points between $\Lambda$ and $e$, with $z_{i+1}$ after $z_i$ ($i = 0, \ldots, 2$). We call $z_1$ and $z_2$ the first crossing and the second crossing of $e$, respectively.

The following basic results hold.

Fig. 4. Intersections between $\Lambda$ and an edge $e$ with one bend. Point $z_1$ is the first crossing of $e$ and point $z_2$ is the second crossing of $e$. 

Lemma 2. $\Gamma_k$ is contained in $P(\Lambda, C_k)$.

Proof. By Property P2, the vertices of the external face of $\Gamma_k$ are the vertices $c_1, \ldots, c_s$ of $C_k$; also each $c_i$ is drawn before $c_{i+1}$ on $\Lambda$ ($i = 1, \ldots, s - 1$). It follows that Definition 5 can be applied and $P(\Lambda, C_k)$ exists. By Property P2, the external boundary of $\Gamma_k$ is the same as the external boundary of $G_k$. Therefore, we only need to prove that every edge of $C_k$ in $\Gamma_k$ is contained in $P(\Lambda, C_k)$.

Denote the edges of $C_k$ as $e_1 = (c_1, c_2)$, $e_2 = (c_2, c_3)$, $e_{s-1} = (c_{s-1}, c_s)$, $e_s = (c_s, c_1)$.

By Property P1, $e_i$ ($1 \leq i < s - 1$) is contained in $P(\Lambda, c_i) \cap P(\Lambda, c_{i+1})$. Also, $e_i$ is in the closed half-plane $\Pi$ defined by $l(c_i, c_{i+1})$ and containing $\hat{c}_i$ because, by Property P3, it either coincides with $c_ic_{i+1}$, or it has two intersection points with $\Lambda$ in the sub-interval $(c_i, c_{i+1})$ of $\Lambda$. Therefore, $e_i$ is in the closed region $\Pi \cap P(\Lambda, c_i) \cap P(\Lambda, c_{i+1})$, i.e., the triangle $c_i, \hat{c}_i, c_{i+1}$ (see for example Fig. 5).

The polyline $c_i\hat{c}_i \cup \hat{c}_ic_{i+1}$ is a portion of $\tau(\Lambda, C_k)$ and by Property 1 part of the boundary of $P(\Lambda, C_k)$; it follows that the triangle $c_i, \hat{c}_i, c_{i+1}$ is contained in $P(\Lambda, C_k)$ and $e_i$ is inside $P(\Lambda, C_k)$. Finally, since edge $e_s$ is drawn as a straight-line segment, by Property 2 it is contained in $P(\Lambda, C_k)$.

Lemma 3. For each vertex $c_i$ on the external face of $\Gamma_k$, there exists on $\Lambda$ a neighborhood of $c_i$, denoted as $I_{c_i} = (\alpha_{c_i}, \beta_{c_i})$, that is externally visible.

Proof. By Property P2, the external boundary of the drawing $\Gamma_k$ is equal to $C_k$ and for each pair $c_i, c_{i+1} \in C_k, c_i$ is drawn before $c_{i+1}$ on $\Lambda$. By Property P3 each edge $e_i = (c_i, c_{i+1})$ either intersects $\Gamma$ in two distinct points in the sub-interval $(c_i, c_{i+1})$ of $\Lambda$, or it is a chord between $c_i$ and $c_{i+1}$. Therefore, the neighborhood $I_{c_i}$ can be defined as follows: $\alpha_{c_i}$ ($2 \leq i \leq s$) is equal to $c_{i-1}$ if $e_{i-1} = (c_{i-1}, c_i)$ is drawn as a straight-line segment, otherwise it is equal to the second crossing of $e_{i-1}$; $\beta_{c_i}$ ($1 \leq i \leq s - 1$) is equal to $c_{i+1}$ if $e_i = (c_i, c_{i+1})$ is a straight-line segment, otherwise it is equal to the first crossing of $e_i$. Finally $\alpha_{c_i}$ is any point before $c_1$ and $\beta_{c_i}$ is any point after $c_s$. The fact that such $\alpha_{c_i}$ and $\beta_{c_i}$ exist is a consequence of Property P0.

Fig. 6 shows an example of $I_{c_i} = (\alpha_{c_i}, \beta_{c_i})$ when both $e_{i-1}$ and $e_i$ cross $\Lambda$.

3.1. Drawing on a semicircle

In this section we concentrate on computing a curve drawing of a maximal planar graph on a semicircle. The next result shows that there always exists a curve drawing of a maximal planar graph
on a semicircle, with at most one bend per edge. The proof of this result also defines a constructive algorithm to compute such a drawing.

**Lemma 4.** Let $G$ be a maximal embedded planar graph and let $\Lambda$ be a concave semicircle. There exists a curve drawing of $G$ on $\Lambda$ with at most one bend per edge.

**Proof.** Let $f(x)$ be the function describing $\Lambda$, with $f(x)$ defined in an interval $I = [\alpha, \beta]$. Let $u$, $v$ and $w$ be the vertices on the external face of $G$, and let $v_1 = u$, $v_2 = v$, ..., $v_n = w$ be a canonical ordering of $G$ with respect to $u$ and $v$. We construct the drawing iteratively by adding one vertex per step. In step $k$ ($1 \leq k \leq n$) vertex $v_k$ is added, along with all edges connecting $v_k$ to vertices of $C_{k-1}$. The drawing obtained at the end of step $k$ is denoted by $\Gamma_k$. We prove by induction on $k$ ($1 \leq k \leq n$) that $v_k$ can be added to $\Gamma_{k-1}$ so that $\Gamma_k$ is a curve drawing that satisfies Properties $P_0$–$P_3$.

**Base case:** For $k = 1$, vertex $v_1$ is drawn on a point $(x_1, f(x_1))$ such that $\alpha < x_1 < \beta$. For $k = 2$, vertex $v_2$ is drawn on a point $(x_2, f(x_2))$ such that $\alpha < x_1 < x_2 < \beta$. Edge $(v_1, v_2)$ is drawn as a straight-line segment. For $k = 3$, vertex $v_3$ is drawn on a point $(x_3, f(x_3))$ such that $\alpha < x_1 < x_3 < x_2 < \beta$. Edges $(v_3, v_2)$ and $(v_3, v_1)$ are drawn as straight-line segments. Clearly, at this step the drawing is planar (it is a triangle with vertices $v_1$, $v_2$ and $v_3$) and Properties $P_0$–$P_3$ hold.

**Inductive case:** Suppose by induction that $\Gamma_{k-1}$ ($k > 3$) is a curve drawing that verifies Properties $P_0$–$P_3$. Let $w_1, \ldots, w_h$ be the vertices of $C_{k-1}$ that are connected to $v_k$, in the clockwise order they appear in $C_{k-1}$. We first show how to construct $\Gamma_k$. Then we show that $\Gamma_k$ is a curve drawing by proving its planarity, and finally we prove that $\Gamma_k$ satisfies Properties $P_0$–$P_3$.

**Construction of $\Gamma_k$:** By Lemma 3 vertex $w_1$ has an externally visible neighborhood $I_{w_1} = (\alpha_{w_1}, \beta_{w_1})$ on $\Lambda$, where $\beta_{w_1}$ either coincides with $w_2$ if the edge $(w_1, w_2)$ is straight-line or coincides with the first crossing of $(w_1, w_2)$, otherwise. Vertex $v_k$ is drawn as a point of $I_{w_1}$ as follows (see for example Fig. 7(a)). Denote with $t$ the point of $\Lambda$ between $w_1$ and $\beta_{w_1}$ such that the tangent to $\Lambda$ at $t$ is parallel to $w_1 \beta_{w_1}$; point $t$ is the intersection between $\Lambda$ and the perpendicular bisector of segment $w_1 \beta_{w_1}$. Draw $v_k$ anywhere on $\Lambda$ between $w_1$ and $t$. Draw edges $e_1 = (v_k, w_1)$, $e_2 = (v_k, w_2)$, ..., $e_h = (v_k, w_h)$, as follows (see for example Fig. 7(b)). Edge $e_1$ is drawn as a straight-line segment connecting $w_1$ and $v_k$. Choose $h - 1$ points, denoted as $p_2, p_3, \ldots, p_h$, between $v_k$ and $t$ on $\Lambda$, such that $p_i$ is after $p_i$ ($3 \leq i \leq h$). Choose other $h - 1$ points, denoted as $q_2, q_3, \ldots, q_h$, as follows. By Lemma 3, each vertex $w_i$ ($2 \leq i \leq h$)
has an externally visible neighborhood \( I_{w_i} = (\alpha_{w_i}, \beta_{w_i}) \) on \( \Lambda \), where \( \alpha_{w_i} \) either coincides with \( w_{i-1} \) if edge \((w_{i-1}, w_i)\) is straight-line or coincides with the second crossing of \((w_{i-1}, w_i)\), otherwise. For each \( w_i \) \((2 \leq i \leq h)\), let \( q_i \) be any point on \( \Lambda \) between \( \alpha_{w_i} \) and \( w_i \). Let \( b_i \) be the intersection point between the straight lines \( l(v_k, p_i) \) and \( l(w_i, q_i) \) \((2 \leq i \leq h)\). Edge \( e_i \) \((2 \leq i \leq h)\) is drawn as the polyline \( v_kb_i \cup b_iw_i \); point \( b_i \) is the bend of \( e_i \).

**Proof of the planarity of \( \Gamma_k \):** We prove that no edge of \( e_1, \ldots, e_h \) crosses any edge of \( \Gamma_{k-1} \) and that no two edges of \( e_1, \ldots, e_h \) cross each other.

Let \( P(\Gamma_{k-1}) \) be the drawing of the external boundary of \( \Gamma_{k-1} \). By inductive hypothesis, \( \Gamma_{k-1} \) satisfies Properties \( \mathbb{P}0-\mathbb{P}3 \). By Lemma 2 and Property \( \mathbb{P}3 \), it follows that \( P(\Gamma_{k-1}) \) is a convex polygon. Each edge \( e_i \) \((2 \leq i \leq h)\) consists of two segments \( v_kb_i \) and \( b_iw_i \). Since \( \sigma(v_kb_i) > \sigma(\tau(\Lambda, i)) \), it follows that \( \sigma(v_kb_i) > \sigma(\beta_{w_i}) \) (see for example Fig. 8(a) where \( i = 3 \)). The straight line \( l(w_1, \beta_{w_1}) \) contains a side of \( P(\Gamma_{k-1}) \). Since \( P(\Gamma_{k-1}) \) is convex, it follows that \( v_kb_i \) is outside \( P(\Gamma_{k-1}) \). Also, \( \sigma(b_iw_i) < \sigma(\alpha_{w_i}) \) because point \( \alpha_{w_i} \) is before \( q_i \). The straight line \( l(\alpha_{w_i}, w_i) \) contains a side of \( P(\Gamma_{k-1}) \). Since \( P(\Gamma_{k-1}) \) is convex, it follows that \( b_iw_i \) is outside \( P(\Gamma_{k-1}) \), except for the point \( w_i \). Finally, edge \( e_1 \) is a chord between \( w_1 \) and \( v_k \), and \( v_k \) is before \( \beta_{w_1} \). It follows that \( \sigma(\alpha_{w_i}) > \sigma(l(w_1, \beta_{w_1})) \), and hence \( e_1 \) is outside \( P(\Gamma_{k-1}) \), except for \( w_1 \). Therefore each edge \( e_i \) \((i = 1, \ldots, h)\) does not intersect any edge of \( \Gamma_{k-1} \) except at common end-vertices.

All edges \( e_1, \ldots, e_h \) are internally disjoint. Indeed, for each pair \( e_i, e_{i+1} \) \((i = 1, \ldots, h)\) we have that:

(i) \( \sigma(v_kb_i) < \sigma(v_kb_{i+1}) \); (ii) \( \sigma(q_iw_i) = \sigma(b_iw_i) > \sigma(b_{i+1}w_{i+1}) = \sigma(q_{i+1}w_{i+1}) \); (iii) walking on \( \Lambda \) from left to right, points \( q_i, w_i, q_{i+1}, w_{i+1} \) are encountered in this order. These conditions imply that edge \( e_i \) is externally covered by \( e_{i+1} \) (see for example Fig. 8(b)).

**Proof of Properties \( \mathbb{P}0-\mathbb{P}3 \) for \( \Gamma_k \):** We certify Properties \( \mathbb{P}0-\mathbb{P}3 \) for \( \Gamma_k \) in this order.

- **Property \( \mathbb{P}0 \)** is guaranteed by the fact that vertices \( v_1 \) and \( v_2 \) are drawn on two points distinct from the endpoints of \( \Lambda \), and each other vertex is drawn after \( v_1 \) and before \( v_2 \).
- Denoting \( C_{k-1} : c_1, c_2, \ldots, c_i = w_1, c_{i+1} = w_2, \ldots, c_r = w_h, \ldots, c_s \), we observe that \( C_k : c_1, c_2, \ldots, c_l = w_1, v_k, c_r = w_h, \ldots, c_s \). Since none of the edges added in step \( k \) is incident to vertices \( c_1, \ldots, c_{i-1} \) and \( c_{r+1}, \ldots, c_s \), Property \( \mathbb{P}1 \) holds for these vertices by the inductive hypothesis. We prove Property \( \mathbb{P}1 \) for vertices \( c_l = w_1, v_k \) and \( c_r = w_h \). Given any two points \( z \) and \( z_1 \) on \( \Lambda \), any segment containing the chord \( \overline{z_1z_1} \) and having \( z \) as one of its endpoints lies in \( P(\Lambda, z) \). All edge-segments that are incident to \( z \in \{w_1, v_k, w_h\} \) have \( z \) as an endpoint and contain a chord \( \overline{z_1z_1} \). Thus Property \( \mathbb{P}1 \) holds for vertices \( w_1, v_k \) and \( w_h \).
About Property P2, we observe that the portion of the external boundary of $\Gamma_k$ from vertex $c_1$ to vertex $c_1 = w_1$ in clockwise order is the same as in $\Gamma_{k-1}$. The next edge of $C_k$ encountered in clockwise order is $(w_1, v_k)$. This edge is on the external face of $\Gamma_k$. Edge $(w_1, v_k)$ is outside the polygon $P(\Gamma_{k-1})$ and it is not externally covered by any other edge $e_i$ ($2 \leq i \leq h$) because all the first crossings of these edges are after $v_k$. The next edge of $C_k$ encountered in clockwise order is $(v_k, w_h)$. Also this edge is on the external face of $\Gamma_k$ since it is outside $P(\Gamma_{k-1})$ and all the edges $e_i$, $2 \leq i \leq h - 1$, are externally covered by it. The portion of the external boundary of $\Gamma_k$ from vertex $c_r = w_h$ to vertex $c_s$ in clockwise order is the same as the one of $\Gamma_{k-1}$. Therefore, the external boundary of $\Gamma_k$ is equal to $C_k$. Finally, by construction vertex $v_k$ is drawn on $\Lambda$ after $c_1$ and before $\beta_{c_1}$, which is before $c_r$. It follows that Property P2 holds.

Property P3 holds as an immediate consequence of the construction.

3.2. Curve embedding

The drawing procedure described in the proof of Lemma 4 can be extended to general concave curves. Indeed, the proof of Lemma 4 does not rely upon the fact that $\Lambda$ is a semicircle, except when point $t$ is computed in order to establish where $v_k$ must be drawn. However, a point with the same property as $t$ exists on any concave curve. Namely, let $f(x)$ be the function representing $\Lambda$ and consider a chord $(a, f(a))(b, f(b))$. By Lagrange’s Theorem there exists a point $c$ in $[a, b]$ such that $f’(c) = (f(b) - f(a))/(b - a)$; that is, the tangent at $(c, f(c))$ on $\Lambda$ is parallel to $(a, f(a))(b, f(b))$. Therefore, we can compute $t$ by assuming $(a, f(a)) = w_1$ and $(b, f(b)) = \beta_{w_1}$, and by observing that $f’(x)$ is invertible since it is a monotone decreasing function. Hence, Lemma 4 can be extended as follows.

Lemma 5. Let $G$ be a maximal planar graph and let $\Lambda$ be a concave curve. There exists a curve drawing of $G$ on $\Lambda$ with at most one bend per edge.

Also, since the linear ordering of the vertices along the curve computed by the procedure described in Lemma 4 (and extended by Lemma 5) does not depend on the choice of the curve, we can conclude that such an ordering is a curve embedding of $G$. We obtain the following.

Theorem 1. Let $G$ be a planar graph with $n$ vertices. There exists an $O(n)$-time algorithm that computes a curve embedding of $G$. Also, let $I = [\alpha, \beta] \subset \mathbb{R}$ and let $\Lambda$ be a concave curve represented by a function $f: I \to \mathbb{R}$. A curve drawing of $G$ on $\Lambda$ can be computed in $O(n)$ time if $\Lambda$ is a semicircle or, more in general, if $(f’(x))^{-1}$ is explicitly known $\forall x \in I$. 
Proof. We first prove that a curve drawing of $G$ on a semicircle can be computed in $O(n)$ time by using the algorithm described in the proof of Lemma 4. Since a curve embedding of $G$ is induced by a curve drawing of $G$ on a semicircle, this also implies that a curve embedding of $G$ can be computed in linear time.

By Lemma 1, the canonical ordering can be computed in $O(n)$ time. The addition of a vertex $v_k$ to the drawing requires the execution of two steps: (i) the choice of the position of $v_k$ and (ii) the choice of the points $p_i$ and $q_i$ where edges $(v_k, w_i)$ ($2 \leq i \leq h$) intersect $A$. We first observe that for each vertex $w_i$, the extreme points $\alpha_{w_i}, \beta_{w_i}$ of the neighborhood $I = (\alpha_{w_i}, \beta_{w_i})$, can be stored and updated in constant time throughout the algorithm. The position of $v_k$ can be chosen in constant time once $t$ is computed. On the other hand, the coordinates of $t$ can be computed in $O(1)$ time from the coordinates of $w_1$ and $\beta_{w_1}$ by simple trigonometry. It follows that step (i) requires $O(1)$ time.

Each $p_i$ is chosen in $O(1)$ time from the coordinates of $t$ and $v_k$. Each $q_i$ is chosen in $O(1)$ time from the coordinates of $\alpha_{w_i}$ and $w_i$. It follows that step (ii) requires $O(1)$ time for each edge incident to $v_k$. Therefore the overall time complexity is $O(n)$.

To complete the proof, it remains to analyze the time complexity of computing a curve drawing of $G$ on a general concave curve $\Lambda$ represented by a function $f(x)$. As pointed out at the beginning of this section, such a drawing can be computed using the same algorithm as for the case in which $\Lambda$ is a semicircle, except for the computation of point $t$. In this case, $t$ can be computed by inverting function $f'(x)$. As a consequence, if $(f'(x))^{-1}$ is known, we can compute $t$ in $O(1)$ time also in this case.

4. Upward drawings and point-set embeddings

In this section we study the interplay between a curve drawing and a drawing where all vertices are collinear and each edge has at most two bends. The relationship between the two types of drawings is used to prove new results on two well-studied graph drawing topics.

The section is organized as follows. In Section 4.1 we introduce the notion of a monotone spine embedding and use it for computing upward drawings where all vertices are collinear and each edge has at most two bends. In Section 4.2 we show how to apply curve embeddings to simplify and extend a previous result by Kaufmann and Wiese [29] for drawing a planar graph with the vertices constrained to be mapped to a given set of points and with each edge having at most two bends.

In the next subsections we call CURVEDRAWER the algorithm described in the proof of Lemma 4 to compute a curve drawing of $G$ on a semicircle.

4.1. Spine embeddings and upward drawability

Let $G$ be a planar graph and let $L$ be a linear ordering of its vertices. Ordering $L$ is a spine embedding of $G$ if there exists a crossing-free drawing $\Sigma$ of $G$ in the plane with the following properties: (i) The vertices of $G$ are represented in $\Sigma$ as points that lie on a straight line and that respect the ordering $L$; (ii) each edge of $G$ is represented in $\Sigma$ as a polyline with at most two bends. The concept of spine embedding is equivalent to that of two-page topological book embedding used in [19] with the additional constraint that each edge crosses the spine at most once (i.e., the interior of each edge can share at most one point with the spine). Drawing $\Sigma$ is a spine drawing of $G$ and the straight line on which the vertices of $\Sigma$ lie is the spine, which defines two half-planes called pages. A spine drawing is monotone if all
its edges are monotone in a common direction (for example the $x$-direction). A spine embedding is said to be monotone if it gives rise to a monotone spine drawing. We also say that the linear ordering of the vertices along the spine of a monotone spine drawing $\Sigma$ is the monotone spine embedding induced by $\Sigma$. Fig. 9 shows a monotone spine drawing of the graph of Fig. 1(a); the induced monotone spine embedding is $L = \{v_1, v_6, v_4, v_5, v_3, v_2\}$. In Fig. 9 edge $(v_4, v_2)$ crosses the spine and the crossing is highlighted with a small vertical segment.

In the following we study spine embeddings and their relationships with curve embeddings. Namely, we prove that a monotone spine embedding of a planar graph $G$ can be chosen as a particular curve embedding of $G$, and that a monotone spine drawing of $G$ inducing such a spine embedding can be computed in linear time. To this aim, we first describe an algorithm that computes a monotone spine drawing $\Sigma$ of $G$ by starting from a curve drawing $\Gamma$ of $G$, in such a way that the spine embedding induced by $\Sigma$ is equal to the curve embedding induced by $\Gamma$. Then, we prove that the algorithm is correct and can be performed in linear time. Before giving the description of the algorithm, we recall and introduce some notation.

Let $\Lambda$ be a semicircle and let $\Gamma$ be the curve drawing of $G$ on $\Lambda$ constructed by Algorithm CURVE DRA WER. We recall that each edge $e = (v, w)$ with one bend in $\Gamma$ has two crossings with $\Lambda$ distinct from its end-vertices, called the first and second crossings and denoted by $z_1$ and $z_2$, respectively. We say that $z_1$ is a spine crossing if there is some vertex between $v$ and $z_1$ and another vertex between $z_1$ and $w$ along $\Lambda$. Intuitively, a spine crossing will correspond to an intersection point between the monotone spine drawing and the spine. Without loss of generality, we assume that the spine is horizontal and that its $y$-coordinate is zero. Also, we denote by $x(p)$ and $y(p)$ the $x$-coordinate and the $y$-coordinate of a point $p$ in the plane, respectively.

A high-level description of the algorithm is as follows. A curve drawing $\Gamma$ of $G$ is first computed (step (1)) and then it is transformed to a spine drawing. All vertices and spine crossings of $\Gamma$ are mapped to points of the spine (step (2)). Every edge of $\Gamma$ is drawn as a polyline with at most two bends and forming angles of $\frac{\pi}{4} + \frac{k\pi}{2}$ ($k = 0, 1, 2, 3$) with the spine (step (3)). Finally, if any two edges overlap, one of them is rotated (step (4)). A more detailed description of the algorithm is the following.

**Algorithm SPINE DRA WER**

**Input:** An embedded planar graph $G$.

**Output:** A monotone spine drawing $\Sigma$ of $G$. 

Fig. 9. A monotone spine drawing of the graph in Fig. 1(a). Edge $(v_4, v_2)$ crosses the spine and the crossing is highlighted with a small vertical segment.
(1) Compute a curve drawing \( \Gamma \) of \( G \) on a semicircle \( \Lambda \) by applying Algorithm \textsc{CurveDrawer}. Denote by \( p_1, p_2, \ldots, p_m \) the sequence of vertices and spine crossings on \( \Lambda \) such that \( p_i \) is drawn before \( p_{i+1} \) (\( 1 \leq i < m \)).

Call subedge the portion of edge connecting \( p_i \) to \( p_j \) in \( \Gamma \) (\( 1 \leq i \neq j \leq m \)) and denote it as \( \delta_{ij} \). If neither \( p_i \) nor \( p_j \) is a spine crossing, subedge \( \delta_{ij} \) is an edge of \( \Gamma \).

(2) Draw \( p_1, p_2, \ldots, p_m \) as points on the spine in this order, such that \( x(p_{i+1}) - x(p_i) = 1 \) (\( 1 \leq i < m \)).

(3) Draw each subedge \( \delta_{ij} \) (\( 1 \leq i < j \leq m \)) as the polyline \( \overline{p_i p_j} \cup \overline{pp_j} \), where \( x(p) = (x(p_i) + x(p_j))/2 \) and:
- If \( \delta_{ij} \) is a straight-line segment in \( \Gamma \) (\( \delta_{ij} \) is either an edge or the subedge connecting vertex \( p_i \) to the spine crossing \( p_j \)), then let \( y(p) = -(x(p_j) - x(p_i))/2 \);
- If \( \delta_{ij} \) is a polyline with one bend in \( \Gamma \) and \( p_i \) is a spine crossing, then let \( y(p) = (x(p_j) - x(p_i))/2 \);
- If \( \delta_{ij} \) is a polyline with one bend in \( \Gamma \) and \( p_i \) is not a spine crossing (\( \delta_{ij} \) is an edge), then:
  - If there is no vertex of \( \Gamma \) between \( p_i \) and the first crossing of edge \( (p_i, p_j) \) on \( \Lambda \), then let \( y(p) = -(x(p_j) - x(p_i))/2 \).
  - Else let \( y(p) = -(x(p_j) - x(p_i))/2 \).

At this point some edges incident to the same vertex may overlap (see for example Fig. 10(a)).

(4) Remove the overlapping, by using a technique similar to that described by Kaufmann and Wiese [29]. The technique for a vertex \( p_i \) with incident overlapping edges is as follows (see also Fig. 10(b)). The distance between any two non-overlapping parallel segments in the drawing computed by steps (1)–(3) is at least \( \varepsilon = 1/\sqrt{2} \). Indeed, let \( p_i, p_j \) (\( 1 \leq i < j \leq m \)) be two vertices or spine crossings, \( x(p_j) - x(p_i) \geq 1 \) and each segment forms an angle \( \alpha = \frac{\pi}{4} + k \frac{\pi}{2} \) (\( k = 0, 1, 2, 3 \)) with the spine. Let \( E_k \) (\( k = 0, 1, 2, 3 \)) be the set of overlapping segments incident to \( p_i \) which form an angle \( \frac{\pi}{4} + k \frac{\pi}{2} \) with the spine. For each \( E_k \), sort the segments in \( E_k \) according to their length in decreasing order and enumerate them starting from 0. Let \( \lambda_k \) be the maximum length of a segment in \( E_k \) and let \( \Delta_k = |E_k| \). Let \( \overline{pp'} \) be the \( h \)th segment in \( E_k \). \( \overline{pp'} \) is a portion of a subedge drawn by steps (1)–(3) as polyline \( \overline{pp'} \cup \overline{pp_j} \). Rotate the straight line \( l \) containing \( \overline{pp'} \) by an angle \( \alpha = h \varepsilon / \lambda_k \Delta_k \) towards the spine. Replace segment \( \overline{pp'} \) with segment \( pp' \) where \( p' \) is the intersection point of the rotated line \( l \) and \( \overline{pp_j} \). Fig. 10(c) shows the drawing obtained by applying step (4) to the drawing of Fig. 10(a).

We now give some results that will be used to prove the correctness of Algorithm \textsc{SpineDrawer}.

**Lemma 6.** Let \( G \) be an embedded planar graph and let \( \Sigma \) be the drawing computed by Algorithm \textsc{SpineDrawer}. Each edge of \( G \) in \( \Sigma \) crosses the spine at most once, has at most two bends and is monotone in the x-direction.

**Proof.** At the end of step (3) of Algorithm \textsc{SpineDrawer}, each subedge \( \delta_{ij} \) has exactly one bend. Also, each segment of \( \Sigma \) forms an angle of \( \frac{\pi}{4} + k \frac{\pi}{2} \) (\( k = 0, 1, 2, 3 \)) with the spine by construction. In step (4) the algorithm does not change the slope of segments incident to points representing spine crossings, because a spine crossing has only two incident segments that lie on different pages. Each edge \( e \) of \( G \) in \( \Sigma \) is the union of at most two subedges \( \delta_{ij} \) and \( \delta_{jh} \) such that: (i) \( p_j \) is a spine crossing; (ii) \( p_j \) is before \( p_i \) and after \( p_i \); (iii) \( \delta_{ij} \) and \( \delta_{jh} \) lie on different pages. It follows that \( e \) crosses the spine at most once, has at most two bends and is monotone in the x-direction. \( \square \)
Fig. 10. Illustration of Algorithm SPINE D RAWER applied to the graph of Fig. 1(a). (a) The drawing with overlapping edges at the end of step (3). (b) Illustration of the rotation technique of step (4). (c) The final drawing.

The following property is an immediate consequence of the construction performed by Algorithm CURVE D RAWER, described in the proof of Lemma 4. It will be extensively used in the proof of Lemma 7.

**Property 3.** Let $G$ be an embedded planar graph and let $\Gamma$ be the curve drawing computed by Algorithm CURVE D RAWER on a concave curve $\Lambda$. Let $(u,v)$ be any edge of $G$ that is drawn with one bend in $\Gamma$, and denote by $z_2$ the second crossing of $(u,v)$. No vertices and first crossings can lie in the interval $(z_2,v)$ of $\Lambda$.

**Lemma 7.** Let $G$ be an embedded planar graph and let $\Sigma$ be the drawing computed by Algorithm SPINE D RAWER. Drawing $\Sigma$ is planar.

**Proof.** We prove that, at the end of step (3), any pair of subedges $\delta_{ij}$, $\delta_{hk}$ on the same page do not cross each other (except for the possible overlapping) and that the removal of the overlapping in step (4) does not introduce any new crossing. If $\delta_{ij}$, $\delta_{hk}$ have one end-point in common, then they either overlap or do not cross, since all parallel segments have the same slope. If $\delta_{ij}$, $\delta_{hk}$ have no end-points in common there are three cases to consider: (a) $x(p_i) < x(p_h) < x(p_k) < x(p_j)$. In this case $\delta_{hk}$ is below $\delta_{ij}$ by construction. (b) $x(p_i) < x(p_j) < x(p_h) < x(p_k)$. In this case $\delta_{hk}$ and $\delta_{ij}$ are disjoint.
x(p_i) < x(p_h) < x(p_j) < x(p_k). We prove that this case is not possible, because it would imply a crossing in the curve drawing \( \Gamma \). The proof is by contradiction, analyzing an exhaustive set of cases. Namely, suppose by contradiction that \( x(p_i) < x(p_h) < x(p_j) < x(p_k) \) in \( \Sigma \). The following two cases are possible.

**Case 1.** \( \delta_{ij} \) and \( \delta_{hk} \) lie on the top page in \( \Sigma \) (i.e., above the spine).

In this case both \( p_j \) and \( p_k \) are vertices, and both \( \delta_{ij} \) and \( \delta_{hk} \) have one bend in \( \Gamma \). We distinguish among the following sub-cases:

- **both \( p_i \) and \( p_h \) are spine crossings.** By Property 3 the second crossing of the edge containing \( \delta_{hk} \) is after \( p_j \), and the second crossing of the edge containing \( \delta_{ij} \) is after \( p_h \). This implies a crossing in \( \Gamma \) (see Fig. 11(a)), a contradiction.
- **\( p_i \) is a spine crossing and \( p_h \) is a vertex.** By Property 3 the second crossing of edge \( (p_h, p_k) \) is after \( p_j \), and the second crossing of the edge containing \( \delta_{ij} \) is after \( p_h \). Also, there cannot exist a vertex on \( \Lambda \) between \( p_i \) and the first crossing of \( (p_h, p_k) \), because it would imply that \( \delta_{hk} \) is drawn below the spine in \( \Sigma \). Hence, vertex \( p_j \) is after the first crossing of \( (p_h, p_k) \). This leads to a crossing in \( \Gamma \) (see Fig. 11(b)), a contradiction.
- **\( p_i \) is a vertex and \( p_h \) is a spine crossing.** Let \( (u, p_k) \) be the edge containing subedge \( \delta_{hk} \). By Property 3, the second crossing of \( (u, p_k) \) is after \( p_j \), and the second crossing of edge \( (p_i, p_j) \) is after \( p_h \). Vertex \( u \) can lie either before or after \( p_i \). If it is before \( p_i \) then there would be a crossing in \( \Gamma \) because \( p_h \) is after \( p_i \) by hypothesis. If it is after \( p_i \) then it must be after the first crossing of \( (p_i, p_j) \), because otherwise \( \delta_{ij} \) would be drawn below the spine; this implies a crossing in \( \Gamma \) (see Fig. 11(c)), a contradiction.
- **both \( p_i \) and \( p_j \) are vertices.** By Property 3, the second crossing of \( (p_h, p_k) \) is after \( p_j \), and the second crossing of \( (p_i, p_j) \) is after \( p_h \). Also, the first crossing of \( (p_i, p_j) \) is before \( p_j \), because otherwise \( \delta_{hk} \) would be drawn below the spine. For the same reason, the first crossing of \( (p_i, p_j) \) is before \( p_h \). Therefore, there is a crossing in \( \Gamma \) (see Fig. 11(d)), a contradiction.

**Case 2.** \( \delta_{ij} \) and \( \delta_{hk} \) lie on the bottom page in \( \Sigma \) (i.e., below the spine).

We distinguish among the following sub-cases. We recall that, in all the sub-cases, if a subedge has one bend in \( \Gamma \) then it is an edge by construction.

- **both \( \delta_{ij} \) and \( \delta_{hk} \) are straight-line segments in \( \Gamma \).** It is immediate to see that there would be a crossing in \( \Gamma \) (see Fig. 11(e)), a contradiction.
• \(\delta_{ij}\) is a straight-line segment and \(\delta_{hk}\) has one bend in \(\Gamma\). Since edge \(\delta_{hk}\) lies in the bottom page and it is not a straight-line segment in \(\Gamma\), then there must be a vertex between \(p_h\) and the first crossing of \((p_h, p_k)\). Also, vertex \(p_j\) is between \(p_h\) and \(p_k\) and it cannot be between the first crossing of \((p_h, p_k)\) and \(p_k\), otherwise this first crossing of \((p_h, p_k)\) would be a spine crossing. Hence, \(p_j\) must be between \(p_h\) and the first crossing of \((p_h, p_k)\). This causes a crossing in \(\Gamma\) (see Fig. 11(f)), a contradiction.

• \(\delta_{ij}\) has one bend and \(\delta_{hk}\) is a straight-line segment in \(\Gamma\). Since edge \(\delta_{ij}\) lies in the bottom page and it is not a straight-line segment in \(\Gamma\), then there must be a vertex between \(p_i\) and the first crossing of \((p_i, p_j)\) or there is no vertex between \(p_i\) and \(p_j\). Vertex \(p_h\) is between \(p_i\) and \(p_j\) and hence there must be a vertex between \(p_i\) and the first crossing of \((p_i, p_j)\). Since the first crossing of \((p_i, p_j)\) is not a spine crossing there cannot be vertices between it and \(p_j\). It follows that \(p_h\) must be between \(p_i\) and the first crossing of \((p_i, p_j)\). This causes a crossing in \(\Gamma\) (see Fig. 11(g)), a contradiction.

• both \(\delta_{ij}\) and \(\delta_{hk}\) have one bend in \(\Gamma\). By arguments analogous to those of the previous two cases, we have that \(p_h\) must be between \(p_i\) and the first crossing of \((p_i, p_j)\), and \(p_j\) must be between \(p_h\) and the first crossing of \((p_h, p_k)\). This causes a crossing in \(\Gamma\) (see Fig. 11(h)), a contradiction.

The fact that the removal of overlapping in step (4) does not introduce any new crossing was already shown in the work by Kaufmann and Wiese [29]; we recall their proof here for completeness. We use the same notation as in step (4). In order to avoid crossings, the straight line \(l\) defined in step (4) can be rotated towards the spine until it touches the highest bend \(b\) among those of the subedges nested inside \(\delta_{ij}\). The highest possible bend that can be encountered is the one of subedge \(\delta_{(i+1)(j-1)}\), if this subedge exists. Let \(\theta\) be the angle defined by line \(l\) and line \(l' = l(p_i, b)\). It follows that rotating \(l\) by an angle smaller than \(\theta\) does not introduce any crossings. Since \(0 \leq h \leq \Delta_k\), the maximum rotation angle for a segment in step (4) is \(\varepsilon/\lambda_k\). We need to prove that \(\varepsilon/\lambda_k < \theta\). Let \(p_ip^*\) be the longest segment in \(E_k\). The arc \(a\) covered by the rotation of segment \(p_ip^*\) of an angle \(\theta\) has length \(\lambda_k \theta\). Let \(b^*\) be the intersection point between \(a\) and \(l(p_{i+1}, b)\). We have (see also Fig. 12) that the length of segment \(p^*b^*\) is greater than \(\varepsilon\) and that the subarc of \(a\) from \(p^*\) to \(b^*\) is longer than \(p^*b^*\). Therefore, \(\varepsilon < \lambda_k \theta\), i.e., \(\varepsilon/\lambda_k < \theta\).

**Lemma 8.** Algorithm SPINEDRAWER computes a monotone spine drawing of a planar graph \(G\) with \(n\) vertices in \(O(n)\) time.
Proof. By Lemmas 6 and 7 it follows that the drawing $\Sigma$ computed by Algorithm \textsc{spine\textsc{drawer}} is a monotone spine drawing.

Concerning the time complexity of the algorithm, we show that each step can be performed in $O(n)$ time. In step (1), Algorithm \textsc{curve\textsc{drawer}} is executed in $O(n)$ time as claimed in Theorem 1. Also, the detection of the spine crossings can be performed in $O(n)$ time with the following simple strategy: (i) scan all vertices and first crossings of $\Gamma$ in the ordering they occur along $\Lambda$, and associate with each first crossing the vertex that is immediately before it in $\Gamma$. This can be done by updating the last vertex visited during the scanning; (ii) analogously, scan all vertices and first crossings in the inverse ordering they occur along $\Lambda$, and associate with each first crossing the vertex that is immediately after it in $\Gamma$. (iii) for each first crossing decide if it is a spine crossing, using the information previously stored. Since the number of first crossings is $O(n)$, we conclude that step (1) can be performed in linear time.

Steps (2) and (3) can be easily performed in $O(n)$ time, using the information stored in step (1).

About step (4), $\lambda_k$ and $\Delta_k$ can be computed in $O(\deg(p_i))$ time. Also, the rotation of the segments incident to a vertex $p_i$ can be performed in time $O(\deg(p_i))$, by observing that the enumeration of a group of overlapping segments, according to their lengths, follows the circular ordering of the segments around $p_i$. Since the sum of $\deg(p_i)$ over all vertices $p_i$ is $O(n)$, we conclude that step (4) can be performed in linear time.

An upward planar drawing of a directed planar graph is such that each edge is monotonically increasing in a common direction, for example the $x$-direction. An orientation of the edges of a planar graph is upward planar if the resulting digraph admits an upward planar drawing. By Lemma 8, a curve embedding can be used to compute a monotone spine drawing. Therefore, one can orient the edges of the monotone spine drawing from left to right so to obtain an upward drawing of the graph where all vertices are on a straight line and each edge has at most two bends. We also observe that in a curve drawing computed by algorithm \textsc{curve\textsc{drawer}} each edge $(u, v)$ can be oriented from $u$ to $v$, where $u$ is before $v$. This implies that the directed edge $(u, v)$ is monotone in the $x$-direction, since $x(u) < x(v)$ and since the first and the second crossings of $(u, v)$ with the curve (if any) are between $u$ and $v$. Further, this orientation for the edges of the curve drawing is the same as that defined above for the edges of the monotone spine drawing. Therefore, the following result holds.

Theorem 2. Let $G$ be a planar graph with $n$ vertices. There exists an $O(n)$-time algorithm that finds an upward planar orientation of $G$ such that the resulting digraph:

- Admits an upward planar drawing where all vertices are collinear and each edge has at most two bends; such an upward planar drawing can be computed in $O(n)$ time.
- Admits an upward planar drawing where all vertices are on any concave curve and each edge has at most one bend; such an upward planar drawing can be computed in $O(n)$ time if the curve is described by a function $f : I \rightarrow \mathbb{R}$ such that $I = [\alpha, \beta] \subset \mathbb{R}$ and $(f'(x))^{-1}$ is explicitly known for all $x \in I$.

4.2. Point-set embeddings

In [29] Kaufmann and Wiese present an elegant $O(n \log n)$-time algorithm to draw any planar graph $G$ with $n$ vertices, by mapping the vertices to a given set $S$ of $n$ points, and with at most two bends per edge. We call this algorithm \textsc{point\textsc{set}\textsc{embedder}}, and we briefly outline its main steps in the following.
**Algorithm** \texttt{POINTSETEMBEDDER}

**Input:** An embedded planar graph $G$ and a set of points $S$.

**Output:** A point set embedding of $G$ on $S$ with at most two bends per edge.

1. Order the points of $S$ according to their $x$-coordinates;
2. Augment $G$ to a hamiltonian graph $G'$ by splitting edges with dummy vertices and by adding extra edges. Find a hamiltonian cycle $C$ of $G'$.
3. Map the vertices of $G$ to the ordered points of $S$, according to the order they have in $C$. For each dummy vertex $w$ of $G'$ that splits an edge $(u,v)$ of $G$, map $w$ to a dummy point $p$ such that the $x$-coordinate of $p$ is in between the $x$-coordinates of the two points of $S$ associated with $u$ and $v$.
4. Draw each edge of $G'$ as a polyline with at most one bend.
5. Rotate edges incident to dummy vertices in order to have the same slope for the two edges that are incident to the same dummy vertex.
6. Remove dummy vertices and edges.
7. Rotate edges in order to remove possible overlapping.

Since deciding whether a planar graph is hamiltonian is an NP-hard problem, in [29] step (2) of Algorithm \texttt{POINTSETEMBEDDER} is performed by augmenting $G$ to a four-connected planar graph, which always admits a hamiltonian cycle. Namely, step (2) in [29] consists of the following sub-steps:

(i) The separation triplets are found using an algorithm by Chiba and Nishizeki [8].
(ii) Then vertices and edges are added to the graph, to create an augmented graph which is four-connected. Every edge is split at most once by inserting a dummy vertex.
(iii) After the graph has been augmented, another result by Chiba and Nishizeki [9] is used to determine a hamiltonian cycle.

The time complexity of step (2) is linear. Also, the overall time complexity of Algorithm \texttt{POINTSETEMBEDDER} is dominated by the cost of step (1), which takes $O(n \log n)$ time.

We use the notion of spine embedding to define a variant of Algorithm \texttt{POINTSETEMBEDDER}. This variant provides an alternative strategy to perform step (2), and guarantees the monotonicity of all edges in the final drawing.

The spine embedding computed by Algorithm \texttt{SPINEDRAWER} can be used to define an augmentation of $G$ to a hamiltonian graph $G'$ by adding at most one bend per edge and such that $G'$ may not be four-connected. We recall that also Pach and Wenger provide an alternative linear time algorithm to augment $G$ to a hamiltonian graph without four-connecting it [30]. The algorithm by Pach and Wenger splits an edge at most twice and produces an augmented graph with at most $5n - 10$ vertices. Since the technique of Kaufmann and Wiese to compute a point-set embedding with at most two bends per edge strongly relies on the fact that each edge of $G$ is split at most once by a dummy vertex, it is not immediately clear how to use the result by Pach and Wenger within Algorithm \texttt{POINTSETEMBEDDER}.

**Lemma 9.** Let $G$ be a planar non-hamiltonian graph with $n$ vertices. There exists an $O(n)$-time algorithm that splits each edge of $G$ with at most one dummy vertex and adds extra edges to $G$ to create a hamiltonian augmented planar graph $G'$. The hamiltonian graph $G'$ has at most $4n - 6$ vertices and is not necessarily four-connected.

**Proof.** Graph $G'$ is computed as follows. Algorithm \texttt{SPINEDRAWER} is applied to $G$ to compute a monotone spine drawing $\Sigma$. Let $p_1, \ldots, p_m$ be the left-right sequence of vertices and crossings between
the edges and the spine in $\Sigma$. Replace each crossing with a dummy vertex. For each pair of consecutive vertices (dummy or not) of the sequence connect them by a dummy edge if they are not already connected. The hamiltonian cycle consists of all the edges connecting consecutive vertices of the sequence plus the edge $(p_m, p_1)$. Edge $(p_m, p_1)$ always exists because Algorithm CURVEDRAWER, and hence Algorithm SPINEDRAWER, defines $p_1$ and $p_m$ as the first and the second vertex in the canonical ordering, which are adjacent by definition.

Since by Lemma 6 each edge crosses the spine at most once, the number of dummy vertices in $G'$ is at most equal to the number of edges of $G$, i.e., $3n - 6$ (recall that $G$ is triangulated). Therefore the number of vertices of $G'$ is at most $4n - 6$.

To prove that $G'$ may not be four-connected it is sufficient to consider the example depicted in Fig. 13. This example shows a non-hamiltonian graph $G$, a monotone spine embedding of $G$, and the resulting augmented graph $G'$, which is not four-connected since it has some vertices of degree 3 (for example vertex $v_8$).
After the technique described in the proof of Lemma 9 has been used to replace step (2) of Algorithm \textsc{PointSetEmbedder}, we can apply unchanged steps (4), (6) and (7) of Algorithm \textsc{PointSetEmbedder}, while we can skip step (5). This leads to a drawing with at most two bends per edge and such that all edges are monotone in the $x$-direction. To prove that step (5) can be skipped and that the monotonicity of the edges is guaranteed, we must recall some further details about step (5). After step (4) of Algorithm \textsc{PointSetEmbedder} has been executed, all the edge-segments incident to a vertex have the same slope in absolute value (see [29]). If $z$ is a dummy vertex that splits edge $(u, v)$, step (5) rotates the edge-segments incident to $z$ in such a way that the removal of $z$ does not give rise to a new bend on $(u, v)$. Actually, the rotation is necessary only if the $x$-coordinate of the point to which $z$ is mapped is not in between those of the points to which $u$ and $v$ are mapped (see Fig. 14); in this case, the removal of $z$ produces an edge that is not monotone in the $x$-direction. Since our mapping follows the ordering of a monotone spine embedding, a dummy vertex (that is, a spine crossing) is always in between the end-vertices of its associated edge. Therefore, we do not need to perform step (5), and each edge will be monotone in the $x$-direction in the final drawing.

From the above reasoning, we obtain the following theorem, which extends the result in [29].

**Theorem 3.** Let $G$ be a planar graph with $n$ vertices and let $P$ be an arbitrary set of $n$ points in the plane. There exists an $O(n \log n)$-time algorithm that computes a drawing of $G$ by mapping its vertices to the elements of $P$ and such that each edge is a polyline with at most two bends that is monotone in the $x$-direction.

5. Conclusions and open problems

This paper has defined the notion of a curve embedding of a planar graph. We have shown that all planar graphs have a curve embedding that can be computed in linear time. Also, we have used the notion of curve embedding to obtain new results on upward drawings and point-set embeddings of planar graphs.

There are many open problems related to the notion of curve embedding. We mention here two of those that are among the most interesting.
• Given a linear ordering $L$ of the vertices of a planar graph $G$, recognize whether $L$ is a curve embedding of $G$.
• Given a planar directed acyclic graph $G$, determine whether $G$ has an upward planar drawing with all vertices drawn collinearly and with at most two bends per edge.

Acknowledgements

The research in this paper was partially supported by “Progetto ALINWEB: Algoritmica per Internet e per il Web”, MIUR Programmi di Ricerca Scientifica di Rilevante Interesse Nazionale, and by NSERC. The authors wish to thank the anonymous referees for their useful comments and suggestions.

References