Algorithmica
DOI 10.1007/s00453-013-9790-3

Monotone Drawings of Graphs with Fixed Embedding

Patrizio Angelini · Walter Didimo · Stephen Kobourov · Tamara Mchedlidze · Vincenzo Roselli · Antonios Symvonis · Stephen Wismath

Received: 7 February 2012 / Accepted: 18 April 2013

Abstract A drawing of a graph is a *monotone drawing* if for every pair of vertices u and v there is a path drawn from u to v that is monotone in some direction. In this paper we investigate planar monotone drawings in the *fixed embedding setting*, i.e.,

A shorter version of this work appeared in the Proceedings of the 18th International Symposium on Graph Drawing (GD 2011). Research supported in part by the MIUR project AlgoDEEP prot. 2008TFBWL4 and by the ESF project 10-EuroGIGA-OP-003 GraDR "Graph Drawings and Representations". Work on these results began at the 6th Bertinoro Workshop on Graph drawing. Discussion with other participants is gratefully acknowledged.

P. Angelini (✉) · V. Roselli
Università Roma Tre, Roma, Italy
e-mail: angelini@dia.uniroma3.it

V. Roselli
e-mail: roselli@dia.uniroma3.it

W. Didimo
Università degli Studi di Perugia, Perugia, Italy
e-mail: didimo@diei.unipg.it

S. Kobourov
University of Arizona, Tucson, AZ, USA
e-mail: kobourov@cs.arizona.edu

T. Mchedlidze
Faculty of Informatics, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
e-mail: mched@iti.uka.de

A. Symvonis
National Technical University of Athens, Athens, Greece
e-mail: symvonis@math.ntua.gr

S. Wismath
University of Lethbridge, Lethbridge, Canada
e-mail: wismath@uleth.ca

Published online: 03 May 2013

a planar embedding of the graph is given as part of the input that must be preserved by the drawing algorithm. In this setting we prove that every planar graph on n vertices admits a planar monotone drawing with at most two bends per edge and with at most $4n - 10$ bends in total; such a drawing can be computed in linear time and requires polynomial area. We also show that two bends per edge are sometimes necessary on a linear number of edges of the graph. Furthermore, we investigate subclasses of planar graphs that can be realized as embedding-preserving monotone drawings with straight-line edges. In fact, we prove that biconnected embedded planar graphs and outerplane graphs always admit such drawings, and describe linear-time drawing algorithms for these two graph classes.

Keywords Monotone drawings · Fixed embedding · Planar graph drawing · Polynomial area · Curve complexity

1 Introduction

A drawing of a graph is a *monotone drawing* if for every pair of vertices u and v there is a path drawn from u to v that is monotone in some direction. In other words, a drawing is monotone if, for any given direction d (e.g., from left to right) and for each pair of vertices u and v, there exists a suitable rotation of the drawing for which a path from u to v becomes monotone in the direction d.

Monotone drawings have been recently introduced [2] as a new visualization paradigm, which is well motivated by human subject experiments by Huang *et al.* [13], who showed that the "geodesic tendency" (paths following a given direction) is important in comprehending the underlying graph. Monotone drawings are related to well-studied drawing conventions, such as upward drawings [10, 11], greedy drawings [1, 14, 15], and the geometric problem of finding monotone trajectories between two given points in the plane avoiding convex obstacles [3].

Planar monotone drawings with straight-line edges form a natural setting and it is known that biconnected planar graphs and trees always admit such drawings, for some combinatorial embedding of the graph [2]. However, the question whether a simply connected planar graph always admits a planar straight-line monotone drawing or not is still open.

On the other hand, in the *fixed embedding setting* (i.e., the planar embedding of the graph is given as part of the input and the drawing algorithm is not allowed to alter it) it is known [2] that there exist simply connected planar embedded graphs that admit no straight-line monotone drawings.

In this paper we study planar monotone drawings of graphs in the fixed embedding setting, answering the natural question whether monotone drawings with a given constant number of bends per edge can always be computed, and identifying some subclasses of planar graphs that always admit planar monotone drawings with straight-line edges. Our contributions are summarized below:

- We prove that every n-vertex planar embedded graph has an embedding-preserving monotone drawing with *curve complexity* 2 and with at most $4n - 10$ bends in total. Such a drawing can be computed in linear time and requires polynomial area. We recall that the curve complexity is the maximum number of bends along an edge.

- We show that our bound on the curve complexity is tight, i.e., there exist infinitely many embedded planar graphs that do not admit any embedding-preserving monotone drawing with at most one bend per edge. More specifically, we prove that each of these graphs requires two bends on a linear number of edges.
- We investigate what subfamilies of embedded planar graphs can be realized as embedding-preserving monotone drawings with straight-line edges. We prove that biconnected embedded planar graphs and outerplane graphs always admit such a drawing, which can be computed in linear time.

The paper is structured as follows. Basic definitions and results are given in Sect. 2. An algorithm for computing embedding-preserving monotone drawings of general embedded planar graphs with curve complexity 2 and with at most $4n - 10$ bends in total is described in Sect. 3, together with a proof that such a bound is tight. Algorithms for computing straight-line monotone drawings of meaningful subfamilies of embedded planar graphs are given in Sect. 4. Concluding remarks and open questions are presented in Sect. 5.

2 Preliminaries

We recall some basic concepts of graph drawing (see [10] for more details).

Let G be a graph. A drawing Γ of G maps each vertex of G to a distinct point of the plane and each edge to a simple Jordan curve connecting the points that represent its end-vertices. Drawing Γ is *planar* if no two distinct edges intersect, except possibly at common end-vertices. Graph G is *planar* if it admits a planar drawing. A planar drawing Γ of G partitions the plane into topologically connected regions called the *faces* defined by Γ. The unbounded face is called the *outer face* (or *external face*); the remaining faces are the *internal faces*. A face f is identified by the circular list of vertices and edges that are encountered when walking on its boundary in the clockwise direction if f is internal and in the counterclockwise direction if f is external. An *embedding* ϕ of a planar graph G is an equivalence class of planar drawings that define the same set of faces for G. Hence, an embedding ϕ can be regarded as the description of a set of (internal and external) faces or, equivalently, as the description of the circular clockwise ordering of the edges incident to each vertex plus the choice of the outer face. A planar graph G along with an embedding ϕ is called an *embedded planar graph* and is denoted as G_ϕ. An *embedding-preserving drawing* (or just a *drawing*) Γ of G_ϕ is a planar drawing of G that defines the set of faces of ϕ.

An *ordered* tree is a rooted tree for which a linear ordering is specified for the children of each vertex. Let T be an ordered spanning tree of G_ϕ, rooted at some vertex r, obtained by removing edges from G_ϕ. We say that T *preserves* the planar embedding ϕ.

A *subdivision* of a graph G is obtained by replacing some edges of G with paths. A *k-subdivision* of G is such that any path replacing an edge of G has at most k internal vertices.

A graph G is *connected* if there exists a path between every pair of vertices. A graph G is *biconnected* (resp. *triconnected*) if removing any vertex (resp. any two

vertices) leaves G connected. If G is connected but not biconnected, a *cut-vertex* of G is a vertex whose removal disconnects G. A *biconnected component* of G is a maximal biconnected subgraph of G. If G is biconnected but not triconnected, a *separation pair* is a pair of vertices whose removal disconnects G. Informally speaking, the *triconnected components* of G are a system of smaller graphs that describe all of the separation pairs in G (a triconnected component of G is not necessarily a subgraph of G). The notion of triconnected component is not as intuitive as the notion of biconnected component. In order to handle the decomposition of a biconnected graph into its triconnected components, we use the well-known *SPQR-tree* data structure [5], briefly recalled in the following.

2.1 SPQR-trees

SPQR-trees have been introduced by Di Battista and Tamassia (see, e.g., [4, 5]). Here we recall some basic definitions about SPQR-trees. Further details can be found in [4, 5, 12].

A graph is *st-biconnectible* if adding edge (s, t) to it yields a biconnected graph. Let G be an st-biconnectible graph. A pair of vertices $\{u, v\}$ is a *split pair* if it is a separation pair or if there exists edge (u, v). A *maximal split component* of G with respect to a split pair $\{u, v\}$ (or, simply, a maximal split component of $\{u, v\}$) is either an edge (u, v) or a maximal subgraph G' of G such that G' contains u and v, and $\{u, v\}$ is not a split pair of G'. A vertex $w \neq u, v$ belongs to exactly one maximal split component of $\{u, v\}$. We call *split component* of $\{u, v\}$ the union of any number of maximal split components of $\{u, v\}$.

In the paper, we assume that any SPQR-tree of a graph G is rooted at one edge of G, called the *reference edge*. The rooted SPQR-tree $\mathcal{T}$ of a biconnected graph G, with respect to a reference edge e, describes a recursive decomposition of G induced by its split pairs, and implicitly represents all planar embeddings of G with edge e on the outer face. An example of an SPQR-tree is given in Fig. 1. The nodes of $\mathcal{T}$ are of four types: S, P, Q, and R. Their connections are called *arcs*, in order to distinguish them from the edges of G. Each node μ of $\mathcal{T}$ has an associated st-biconnectible multigraph, called the *skeleton* of μ and denoted by $skel(\mu)$, that shows how the children of μ, represented by "virtual edges", are arranged into μ. The virtual edge in $skel(\mu)$ associated with a child node ν is called the *virtual edge of* ν *in* $skel(\mu)$.

For each virtual edge e_i of $skel(\mu)$, recursively replace e_i with the skeleton $skel(\mu_i)$ of its corresponding child μ_i. The subgraph of G that is obtained in this way is the *pertinent graph* of μ and is denoted by $pert(\mu)$.

Given a biconnected graph G and a reference edge $e = (u', v')$, tree $\mathcal{T}$ is recursively defined as follows. At each step, a split component G^*, a pair of vertices $\{u, v\}$, and a node ν in $\mathcal{T}$ are given. A node μ corresponding to G^* is introduced in $\mathcal{T}$ and attached to its parent ν. Vertices u and v are the *poles* of μ and are denoted by $u(\mu)$ and $v(\mu)$, respectively. The decomposition possibly recurs on some split components of G^*. At the beginning of the decomposition $G^* = G - \{e\}$, $\{u, v\} = \{u', v'\}$, and ν is a Q-node corresponding to e.

Base Case: Graph G^* consists of exactly one edge between u and v. Then, μ is a Q-node whose skeleton is G^* itself.

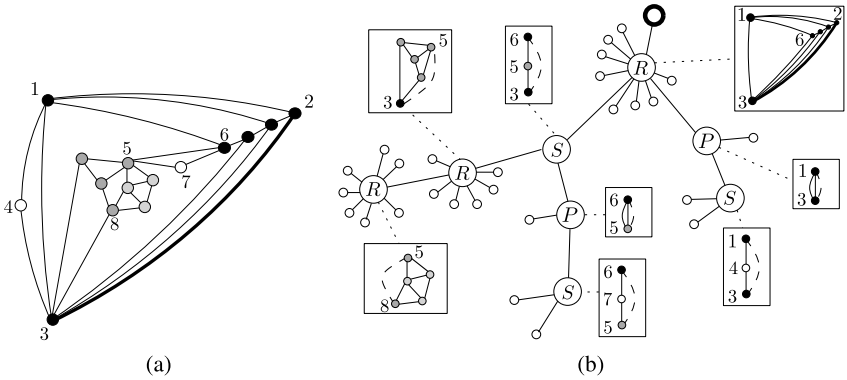

Fig. 1 (**a**) A planar biconnected graph and (**b**) its SPQR-tree $\mathcal{T}$, rooted at the Q-node corresponding to the reference edge $(2, 3)$. Edge $(2, 3)$ and the root of $\mathcal{T}$ are highlighted by using *thicker lines*. Skeletons are drawn inside *boxes*, and the virtual edge representing the rest of the graph, that is, the virtual edge containing the root, is drawn as a *dashed line*. S-, P-, and R-nodes of $\mathcal{T}$ are labeled; the leaves and the root of $\mathcal{T}$ are Q-nodes

Parallel Case: Graph G^* is composed of at least two maximal split components $G_1, \ldots, G_k$ $(k \geq 2)$ of G with respect to $\{u, v\}$. Then, μ is a P-node. Graph $skel(\mu)$ consists of k parallel virtual edges between u and v, denoted by $e_1, \ldots, e_k$ and corresponding to $G_1, \ldots, G_k$, respectively. The decomposition recurs on $G_1, \ldots, G_k$, with $\{u, v\}$ as pair of vertices for every graph, and with μ as parent node.

Series Case: Graph G^* is composed of exactly one maximal split component of G with respect to $\{u, v\}$ with cut-vertices $c_1, \ldots, c_{k-1}$ $(k \geq 2)$, appearing in this order on a path from u to v. Then, μ is an S-node. Graph $skel(\mu)$ is the path $e_1, \ldots, e_k$, where virtual edge e_i connects c_{i-1} with c_i $(i = 2, \ldots, k-1)$, e_1 connects u with c_1, and e_k connects c_{k-1} with v. The decomposition recurs on the split components corresponding to each of $e_1, e_2, \ldots, e_{k-1}, e_k$ with μ as parent node, and with $\{u, c_1\}, \{c_1, c_2\}, \ldots, \{c_{k-2}, c_{k-1}\}, \{c_{k-1}, v\}$ as pair of vertices, respectively.

Rigid Case: If none of the above cases applies, the purpose of the decomposition step is that of partitioning G^* into the minimum number of split components and recurring on each of them. Some further definitions are needed. Given a maximal split component G' of a split pair $\{s, t\}$ of G^*, a vertex $w \in G'$ *properly belongs* to G' if $w \neq s, t$. Given a split pair $\{s, t\}$ of G^*, a maximal split component G' of $\{s, t\}$ is *internal* if neither u nor v (the poles of G^*) properly belongs to G', and *external* otherwise. A *maximal split pair* $\{s, t\}$ of G^* is a split pair of G^* that is not contained into an internal maximal split component of any other split pair $\{s', t'\}$ of G^*. Let $\{u_1, v_1\}, \ldots, \{u_k, v_k\}$ be the maximal split pairs of G^* $(k \geq 1)$ and, for $i = 1, \ldots, k$, let G_i be the union of all the internal maximal split components of $\{u_i, v_i\}$. Observe that each vertex of G^* either properly belongs to exactly one G_i or belongs to some maximal split pair $\{u_i, v_i\}$. Node μ is an R-node. Graph $skel(\mu)$ is the graph obtained from G^* by replacing each subgraph G_i with the virtual edge e_i between u_i and v_i. The decomposition recurs on each G_i with μ as parent node and with $\{u_i, v_i\}$ as pair of vertices.

For each node μ of $\mathcal{T}$, the construction of $skel(\mu)$ is completed by adding a virtual edge (u, v) representing the rest of the graph.

The SPQR-tree $\mathcal{T}$ of a graph G with n vertices and m edges has m Q-nodes and $O(n)$ S-, P-, and R-nodes. Also, the total number of vertices of the skeletons stored at the nodes of $\mathcal{T}$ is $O(n)$. Finally, given a biconnected planar graph G, the SPQR-tree $\mathcal{T}$ of G can be computed in linear time [4, 5, 12].

2.2 Monotone Drawings

Let Γ be a drawing of a graph. A path $u = u_1, \ldots, u_k = v$ between vertices u and v in Γ is *monotone* with respect to a direction d if the orthogonal projections of vertices $u_1, \ldots, u_k$ on d appear in the same order as the vertices appear in the path. Drawing Γ is *monotone* if for every pair of vertices u and v there exists a path $p(u, v)$ and a direction d such that $p(u, v)$ is monotone with respect to d.

In [2] it has been shown that every tree admits a straight-line monotone drawing in polynomial area. Namely, the authors provide two algorithms, called *Algorithm BFS-based* and *Algorithm DFS-based*, that construct drawings requiring $O(n^{1.6}) \times O(n^{1.6})$ and $O(n) \times O(n^2)$ area, respectively. Both algorithms rely on the concept of the *Stern-Brocot tree* [6, 16] $\mathcal{SB}$, an infinite tree whose nodes are in bijective mapping with the irreducible positive rational numbers. The first four levels of the Stern-Brocot tree are depicted in Fig. 2(a).

In particular, *Algorithm DFS-based* assigns slopes $\frac{1}{1}, \frac{2}{1}, \ldots, \frac{n-1}{1}$ to the edges of the input tree T according to the order given by a DFS-visit of T. Note that the assigned slopes correspond to the first $n - 1$ elements of the rightmost path of $\mathcal{SB}$. The drawing of a tree obtained with Algorithm DFS-based is illustrated in Fig. 2(b).

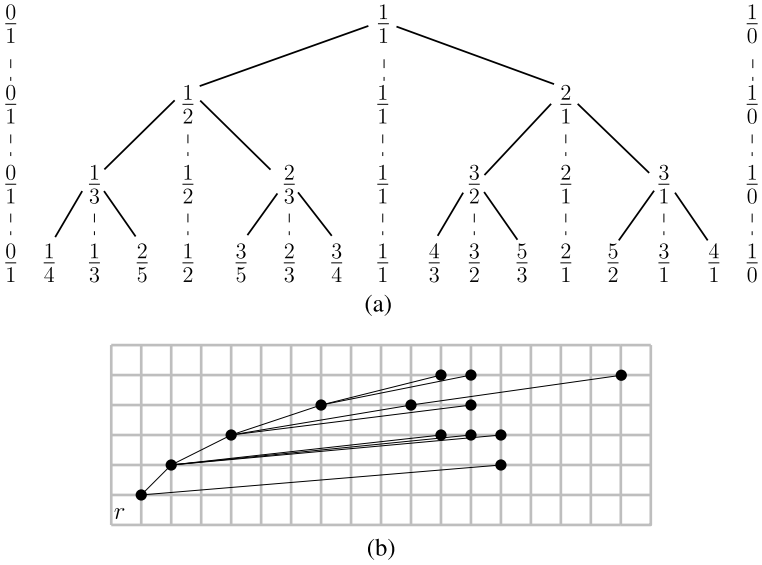

Fig. 2 (**a**) A drawing Γ of an embedded planar graph G_ϕ. (**b**) An upright spanning tree of G_ϕ. (**c**) A spanning tree of G_ϕ that is not upright

The $O(n) \times O(n^2)$ area bound is due to the fact that the sum of the denominators of the slopes assigned to the edges, that corresponds to the maximum height of a drawing constructed with this algorithm, is $\sum_{i=1}^{n-1} 1 = n - 1$, while the sum of the numerators, that corresponds to the maximum width, is $\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$.

In [2] it has also been proved that in any straight-line monotone drawing of a tree, the length of each edge can be arbitrarily modified without affecting planarity and monotonicity. This is formalized in the following property.

Property 1 [2] Let Γ be a straight-line monotone drawing of a tree T. Then, any drawing Γ' of T such that the slopes of each edge $e \in T$ in Γ' is the same as the slope of e in Γ is monotone. Also, the slopes of any two leaf-edges e' and e'' of T in Γ are such that e' and e'' diverge, which implies that the elongations of e' and e'' do not cross each other.

3 Poly-line Monotone Drawings of Embedded Planar Graphs

In this section we study monotone drawings of embedded planar graphs. We remark that it is still unknown whether every planar graph admits a straight-line monotone drawing in the variable embedding setting, while it is known that straight-line monotone drawings do not always exist if the embedding of the graph is fixed [2]. We therefore investigate monotone drawings with bends along some edges, and we show that two bends per edge are always sufficient and sometimes necessary for the existence of a monotone drawing in the fixed embedding setting.

We need some preliminary definitions. An *upright spanning tree* T of an embedded planar graph G_ϕ is a rooted ordered spanning tree of G_ϕ such that:

(i) T preserves the planar embedding of G_ϕ;
(ii) the root of T is a vertex r of the outer face of G_ϕ;
(iii) there exists a planar drawing of G_ϕ that contains an upward drawing of T such that no edge of $G_\phi \setminus T$ passes below r.

Figure 3(b) and (c) show two different ordered spanning trees of the embedded planar graph G_ϕ of Fig. 3(a) rooted at node 1. The first tree is an upright spanning tree, while the second one is not. Namely, even if the tree in Fig. 3(c) preserves the embedding ϕ, edge $(7, 3)$ passes below vertex 1.

Given an embedded planar graph G_ϕ, an upright spanning tree T of G_ϕ can be computed as follows. Construct any planar straight-line drawing Γ of G_ϕ. Orient the edges of G_ϕ in Γ according to the upward direction, by giving a random orientation to horizontal edges. Let r be a source on the outer face of G_ϕ with the smallest y-coordinate in Γ. Then, compute any spanning tree T of G_ϕ rooted at r such that the left-to-right order of the children of r in T is consistent with the left-to-right order of the neighbors of r in Γ and the left-to-right order of the children of each vertex w in T is consistent with the clockwise order of the neighbors of w in G_ϕ, computed starting from the edge connecting w to its parent in T.

Let T be an upright spanning tree of G_ϕ. The *rgbb-coloring* $C(G_\phi, T)$ *of* G_ϕ *with respect to* T is a coloring of the edges of G_ϕ with four colors such that:

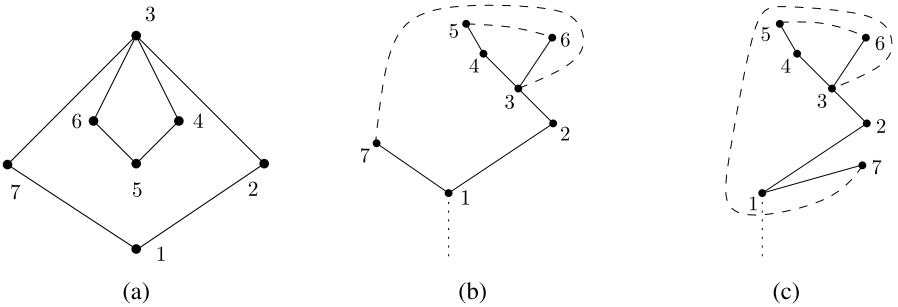

Fig. 3 (**a**) A drawing Γ of an embedded planar graph G_ϕ. (**b**) An upright spanning tree of G_ϕ. (**c**) A spanning tree of G_ϕ that is not upright

- An edge is colored *black* if it belongs to T;
- an edge is colored *green* if it connects two leaves of T;
- an edge is colored *red* if it connects a leaf to an internal vertex of T;
- an edge is colored *blue* if it connects two internal vertices of T.

We now describe an algorithm that, given an embedded planar graph G_ϕ with n vertices, an upright spanning tree T of G_ϕ, and the rgbb-coloring $C(G_\phi, T)$ of G_ϕ with respect to T, constructs a monotone drawing Γ of G_ϕ such that each black or green edge is drawn as a straight-line segment, each red edge has one bend, and each blue edge has two bends. We call this algorithm MONOTONE-FIXED-EMBEDDING; Lemma 1 will prove that it correctly computes a monotone drawing with curve complexity 2 and $O(n^3)$ area, in $O(n)$ time.

First, starting from G_ϕ and T, algorithm MONOTONE-FIXED-EMBEDDING constructs a graph G'_ϕ and an upright spanning tree T' of G'_ϕ such that:

(i) G'_ϕ is a 2-subdivision of G_ϕ;
(ii) T is a subtree of T';
(iii) all the edges of G'_ϕ that are not in T' connect two leaves of T'.

Graphs G'_ϕ and T' are constructed as follows. Initialize $G'_\phi = G_\phi$ and $T' = T$. Subdivide each red edge (s, t) of G'_ϕ with a dummy vertex k and add edge (t, k) to T', where t is the internal vertex of T'. Subdivide each blue edge (s, t) of G'_ϕ twice with two dummy vertices k and z, and add edges (s, k) and (t, z) to T'. Figures 4(a) and 4(b) show a graph G_ϕ with an upright spanning tree T and the corresponding graph G'_ϕ with its upright spanning tree T' satisfying (i)–(iii).

Then, a monotone drawing of G_ϕ with curve complexity 2 is constructed by first computing a straight-line monotone drawing of G'_ϕ and then replacing each subdivision vertex with a bend; see Fig. 4(c).

The straight-line monotone drawing of G'_ϕ is computed in two steps. First, with *Algorithm DFS-based* [2], construct a straight-line monotone drawing of T'. Second, add the remaining (non-tree) edges as straight-line segments by suitably elongating the leaf-edges of T', as described in the following. Observe that, this results in using

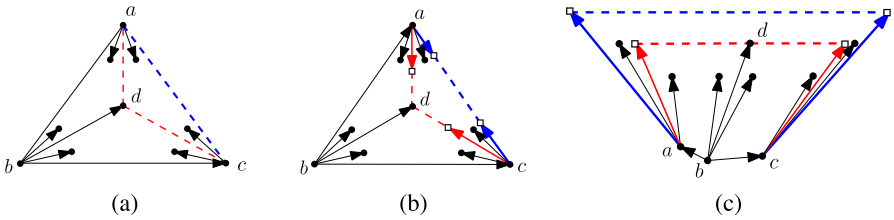

Fig. 4 (**a**) A graph G_ϕ with an upright spanning tree T rooted at vertex b. *Solid* edges belong to T, while *dashed* edges do not. *Blue* edges are thicker than *red* edges, which are thicker than *black* edges. (**b**) The corresponding graph G'_ϕ with its upright spanning tree T' constructed by algorithm MONOTONE–FIXED-EMBEDDING. *Solid* edges belong to T', while *dashed* edges do not. Subdivision dummy vertices are drawn as *squares*. (**c**) A *straight-line* monotone drawing of G'_ϕ that corresponds to a monotone drawing of G_ϕ with bent edges

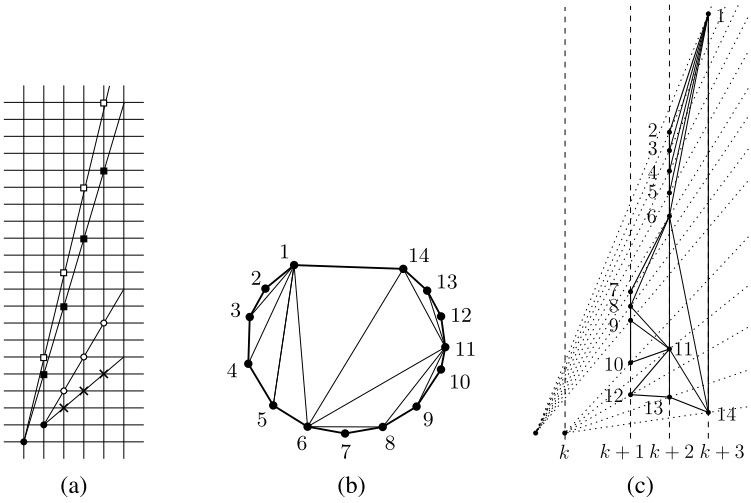

Fig. 5 (**a**) Leaf-edge elongations have integer intersections with all the *vertical lines* in the same order. (**b**) An augmented graph G_L. (**c**) The drawing of G_L, where the number l of levels is equal to 3

two segments for red edges and three segments for blue edges when dummy vertices are replaced by bends.

Consider any leaf-edge (u, v), where v is the leaf of T'. Observe that, as *Algorithm DFS-based* assigns slopes $\frac{1}{1}, \frac{2}{1}, \ldots, \frac{n-1}{1}$ to the edges of T', the elongation of (u, v) intersects at an integer grid point each vertical line $x = k$, where k is any integer value greater than the x-coordinate of u. Moreover, as the leaf-edge elongations do not cross, such intersections appear in the same order on each vertical line $x = k'$, where k' is any integer value greater than the x-coordinate of every internal vertex of T'; see Fig. 5(a).

Another key observation is that the graph induced by the leaves of T' is outerplanar and can be augmented, by adding dummy edges, to a biconnected outerplanar graph G_L such that every internal face of G_L is a 3-cycle and the order of the vertices on the outer face of G_L is the same as the left-to-right order of the leaves of T'; see Fig. 5(b).

The vertices of G_L are assigned to levels in such a way that the end-vertices of each edge of G_L are either on the same level or on adjacent levels, as follows. The first and the last vertex in the left-to-right order of the leaves of T' have level 1. Note that these two vertices are adjacent, as G_L is a biconnected outerplanar graph and the order of the vertices on its outer face is the same as the left-to-right order of the leaves of T'. Then, starting from this edge, consider any edge (u, v) on the outer face of the graph induced by the vertices whose level has been already assigned. Consider the unique vertex w that is connected to both u and v, and whose level has not been assigned yet, if any. Note that, either u and v have the same level i or one of them has level i and the other has level $i + 1$. In both cases, assign level $i + 1$ to w, as shown in Figs. 5(b) and 5(c). Let l be the number of levels of G_L. The drawing of G'_ϕ is completed by placing all the vertices at level i, with $i = 1, \ldots, l$, on a vertical line $x = k + l - i + 1$, where k is the x-coordinate of the rightmost internal vertex of T'; see Fig. 5(c).

Lemma 1 *Given an embedded planar graph G_ϕ with n vertices, an upright spanning tree T of G_ϕ, and the rgbb-coloring $C(G_\phi, T)$ of G_ϕ with respect to T, algorithm* MONOTONE-FIXED-EMBEDDING *constructs a planar monotone drawing Γ of G_ϕ with curve complexity 2 and $O(n) \times O(n^2)$ area in $O(n)$ time.*

Proof First, we prove that the drawing of the auxiliary graph G'_ϕ that is a subdivision of G_ϕ is planar and monotone. Namely, by Property 1, the slopes assigned to the edges of the spanning tree T' of G'_ϕ by *Algorithm DFS-based* [2] are such that for any elongation of the edges of T', the resulting drawing is planar and monotone. Further, since T' is an upright spanning tree of G'_ϕ, it preserves the planar embedding of G'_ϕ and there are no edges going below the root. Also, recall that algorithm MONOTONEFIXEDEMBEDDING places all the vertices at level i, with $i = 1, \ldots, l$, on a vertical line $x = k + l - i + 1$, where G_L is the biconnected outerplanar graph composed of the edges of $G'_\phi \setminus T'$ plus the dummy edges needed to make it biconnected, l is the number of levels of G_L, and k is the x-coordinate of the rightmost internal vertex of T'. This placement, together with the fact that each such vertical line intersects the elongations of all the leaf-edges in the same order, ensures the planarity of the straight-line drawing of G_L. Further, as the order of the vertices on the outer face of G_L is the same as the left-to-right order of the leaves of T', the edges of T' do not cross any edge of G_L. Hence the drawing of G'_ϕ is planar. Finally, since T' is a spanning tree of G'_ϕ, any two vertices of G'_ϕ are connected by a monotone path composed only of edges of T', and hence the drawing of G'_ϕ is also monotone.

The planarity of the drawing Γ of G_ϕ comes from the fact that Γ is obtained by just replacing the dummy vertices in the drawing of G'_ϕ with bends. To see that Γ is monotone, observe that any monotone path in the drawing of G'_ϕ traversing a leaf-edge of T' has the corresponding leaf as an end-vertex, and if such a leaf is a subdivision dummy vertex of any non-black edge, then it does not belong to G_ϕ. Hence, all the monotone paths in G_ϕ are composed only of edges of T, whose drawing is monotone since it is a subtree of T'. Also, Γ is obtained by replacing dummy vertices with bends in the drawing of G'_ϕ, which is a straight-line drawing; since every edge is subdivided at most twice (namely, each red edge is subdivided once and each blue edge is subdivided twice), drawing Γ has curve complexity 2.

The area of Γ can be derived from that required by the drawing of G'_ϕ. Namely, the elongation of the leaf-edges of T', computed by algorithm MONOTONEFIXEDEMBEDDING in order to reinsert the edges of $G'_\phi \setminus T'$ as straight-line segments, does not asymptotically increase the $O(n) \times O(n^2)$ area of the drawing of T' obtained with *Algorithm DFS-based*. Indeed, since the number of vertical lines added to host the drawing of G_L equals the number l of levels assigned to the vertices of G_L, and since l is bounded by the number of leaves (which is $O(n)$), the area of Γ remains $O(n) \times O(n^2)$.

Concerning the time complexity of the algorithm, we analyze the time required by each individual step. The computation of the graphs T, G'_ϕ, and T' can be easily performed in $O(n)$ time. Also, the slopes of the edges of T' can be computed in linear time with *Algorithm DFS-based* [2]. The biconnected planar graph G_L can be constructed in $O(n)$ time by augmenting the outerplanar graph induced by the leaves of T' [7]. Finally, the assignment of levels to the vertices of G_L is performed in $O(n)$ time, as each vertex is considered just once and its level is assigned only based on the levels of its two neighbors. Hence, algorithm MONOTONEFIXEDEMBEDDING runs in linear time. □

Note that, by Lemma 1, there always exists a monotone drawing Γ of G_ϕ with curve complexity 2 and at most $4n - 10$ bends in total. Namely, since G_ϕ has at most $3n - 6$ edges and every spanning tree of G_ϕ has $n - 1$ edges, and since the edges of the spanning tree are drawn as straight-line segments, drawing Γ has at most $2(3n - 6 - n + 1) = 4n - 10$ bends in total.

In the following we prove that our bound on the maximum number of bends per edge is tight, i.e., we show infinitely many embedded planar graphs that do not admit any embedding-preserving monotone drawing with at most one bend per edge. More interestingly, we prove that every monotone drawing of these graphs contains $\Omega(n)$ edges with two bends, hence its total number of bends is linear in the number of vertices. This implies that in general it is not possible to improve the drawing algorithm of Lemma 1 to achieve a sublinear number of bends in total; therefore, the $4n - 10$ bound is asymptotically optimal. We first prove in Lemma 2 that there exist embedded planar graphs requiring at least one bend on some edges. Then, based on this lemma, we prove in Lemma 3 that there exist infinitely many embedded planar graphs whose monotone drawings require two bends on a linear number of edges.

We first introduce a definition that will be useful to prove the claimed lemmata. The *turn angle* from the edge (u, v) to the edge (v, w) is the smallest angle between the half-line from u through v and the edge (v, w); see Fig. 6. The turn angle is *positive* if the rotation defined by this angle is clockwise and *negative* if it is counterclockwise.

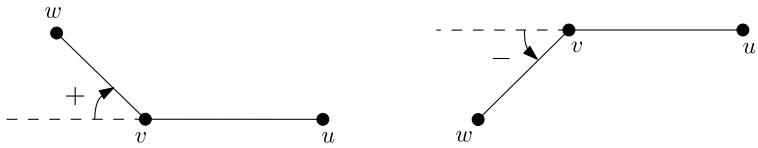

Fig. 6 A positive and a negative turn angle from edge (u, v) to edge (v, w)

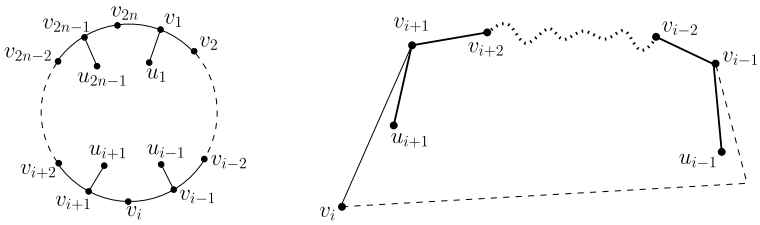

Fig. 7 (**a**) A graph G_ϕ with $3n$ vertices that does not admit any embedding-preserving *straight-line* monotone drawing. (**b**) Path P_i^2 cannot be monotone

Lemma 2 *For every $n \geq 3$ there exists an embedded planar graph G_ϕ with $3n$ vertices and $3n$ edges that does not admit any straight-line monotone drawing.*

Proof Graph G_ϕ consists of a simple cycle $C = v_1, \ldots, v_{2n}$ of length $2n$ and of n vertices $u_1, u_3, \ldots, u_{2n-1}$ of degree 1, called *legs*, adjacent to the vertices $v_1, v_3, \ldots, v_{2n-1}$ of C with odd indices, respectively. The embedding of G_ϕ is such that all the legs are inside C, that is, they are incident to the unique internal face of C; see Fig. 7(a).

We prove that there exists no straight-line monotone drawing of G_ϕ. Assume, for a contradiction, that such a straight-line monotone drawing exists. Consider two arbitrary consecutive legs u_{i-1} and u_{i+1} of G_ϕ. Note that there exist only two paths $P_i^1 = u_{i-1}, v_{i-1}, v_i, v_{i+1}, u_{i+1}$ and $P_i^2 = u_{i-1}, v_{i-1}, v_{i-2}, \ldots, v_1, v_{2n}, \ldots, v_{i+2}, v_{i+1}, u_{i+1}$ connecting u_{i-1} and u_{i+1}.

We show that path P_i^2 cannot be monotone. Refer to Fig. 7(b). Namely, if P_i^2 is monotone then, by Property 1, edges (v_{i-1}, u_{i-1}) and (v_{i+1}, u_{i+1}) diverge, as u_{i-1} and u_{i+1} are leaves of P_i^2. Hence, in order to connect v_{i-1} to v_{i+1} with a polyline while keeping u_{i-1} and u_{i+1} inside the polygon representing C, at least three straight-line segments are necessary. However, only two edges, namely (v_{i-1}, v_i) and (v_i, v_{i+1}), lie between v_{i-1} and v_{i+1} in $C \setminus P_i^2$, a contradiction.

Thus, for any pair of consecutive legs u_{i-1} and u_{i+1} the monotonicity between them is provided by path P_i^1. We show that this leads to a contradiction.

Let α_i, $i = 1, \ldots, 2n$, be the turn angle from edge (v_{i-1}, v_i) to edge (v_i, v_{i+1}), where the indices are taken modulo $2n$. Let also β_i and γ_i, $i = 1, 3, \ldots, 2n-1$ be the turn angles from (v_{i-1}, v_i) to (v_i, u_i), and from (u_i, v_i) to (v_i, v_{i+1}), respectively. Note that:

$$\sum_{i=1}^{2n} \alpha_i = 2\pi. \tag{1}$$

Also, as shown in Figs. 8(a–d), for each $i = 1, 3, \ldots, 2n-1$, regardless of whether the internal angle at v_i is convex or reflex and of the position of the leg u_i, the following equation holds:

$$\beta_i + \gamma_i = \alpha_i + \pi \tag{2}$$

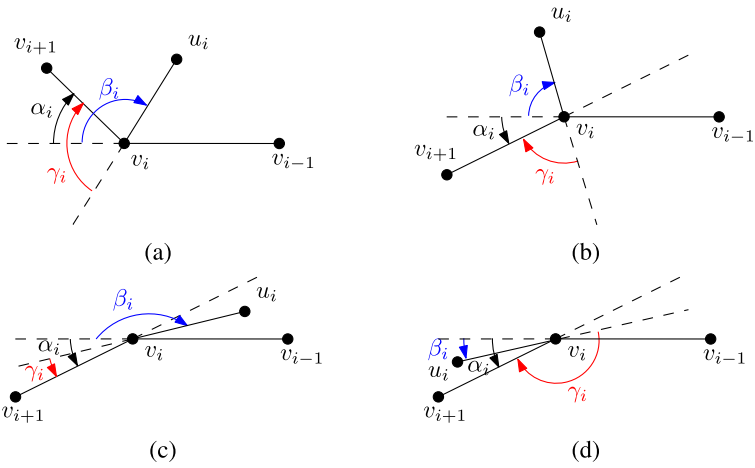

Fig. 8 The proof that equality $\beta_i + \gamma_i = \alpha_i + 180$ holds for each $i = 1, 3, \ldots, 2n-1$. (**a**) The internal angle at v_i is convex. Equality $|\beta| + |\gamma| = 180 + |\alpha|$ holds and $\alpha, \beta, \gamma \geq 0$. (**b**) The internal angle at v_i is reflex and the leg u_i is between the half-line from v_{i-1} through v_i and the half-line from v_{i+1} through v_i in the circular ordering around v_i. Equality $|\beta| + |\gamma| = 180 - |\alpha|$ holds, and $\alpha \leq 0$; $\beta, \gamma \geq 0$. (**c**) The internal angle at v_i is reflex and the leg u_i is between the half-line from v_{i+1} through v_i and edge (v_{i-1}, v_i) in the circular ordering around v_i. Equality $|\beta| - |\gamma| = 180 - |\alpha|$ holds, and $\alpha, \gamma \leq 0$; $\beta \geq 0$. (**d**) The internal angle at v_i is reflex and the leg u_i is between the half-line from v_{i-1} through v_i and edge (v_{i+1}, v_i) in the circular ordering around v_i. Equality $|\gamma| - |\beta| = 180 - |\alpha|$ holds, and $\alpha, \beta \leq 0$; $\gamma \geq 0$

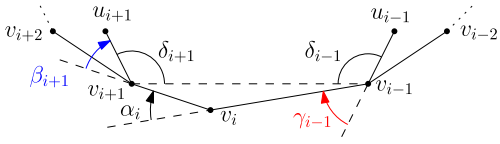

Fig. 9 $\delta_{i-1} + \delta_{i+1} + \gamma_{i-1} + \alpha_i + \beta_{i+1} = 2\pi$. As legs u_{i-1} and u_{i+1} diverge, $\delta_{i-1} + \delta_{i+1} \geq \pi$. Hence, $\gamma_{i-1} + \alpha_i + \beta_{i+1} < \pi$

Further, as path P_i^1 from u_{i-1} to u_{i+1} through v_i is monotone, we have that, for each $i = 1, 3, \ldots, 2n-1$, the following inequality holds; see Fig. 9.

$$\gamma_{i-1} + \alpha_i + \beta_{i+1} < \pi \tag{3}$$

By summing up inequality (3) over $i = 1, 3, \ldots, 2n-1$, we get:

$$\begin{aligned} &\gamma_1 + \alpha_2 + \beta_3 + \gamma_3 + \alpha_4 + \beta_5 + \cdots + \gamma_{2n-1} + \alpha_n + \beta_1 \\ &\quad = (\beta_1 + \gamma_1) + \alpha_2 + (\beta_3 + \gamma_3) + \alpha_4 + \cdots + (\beta_{2n-1} + \gamma_{2n-1}) + \alpha_n \\ &\quad < n\pi \end{aligned}$$

Applying (2), we get $(\alpha_1 + \pi) + \alpha_2 + (\alpha_3 + \pi) + \alpha_4 + \cdots + (\alpha_{2n-1} + \pi) + \alpha_n = \sum_{i=1}^{2n} \alpha_i + n\pi < n\pi$. By (1), we get $(n+2)\pi < n\pi$, a contradiction. □

Lemma 3 *For every odd $n \geq 9$ there exists an embedded planar graph G_ϕ with n vertices and $\frac{3}{2}(n-1)$ edges such that every monotone drawing of G_ϕ has at least $\frac{n-3}{6}$ edges with at least two bends and thus at least $\frac{n-3}{3}$ bends in total.*

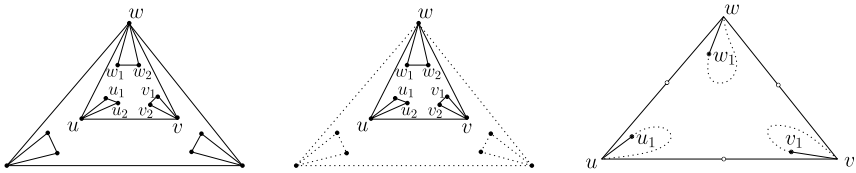

Fig. 10 (**a**) A graph G_ϕ with $n = 15$ vertices, which coincides with a graph G^3_ϕ constructed from G^2_ϕ by adding vertices $u_1, u_2, v_1, v_2, w_1, w_2$ inside the triangular face u, v, w. (**b**) A subgraph G^t_ϕ of G_ϕ induced by a triangle (u, v, w) and all the vertices inside it. (**c**) A subdivision (*white circles*) of the subgraph G^h_ϕ (*solid* edges) of G^t_ϕ induced by u, v, w, u_1, v_1, w_1. By Lemma 2, it does not admit a *straight-line* monotone drawing

Proof Consider an odd integer $n \geq 9$. We construct G_ϕ iteratively. Let G^1_ϕ be a triangle graph. Graph G^i_ϕ is constructed from G^{i-1}_ϕ as follows. Initialize $G^i_\phi = G^{i-1}_\phi$. Let (u, v, w) be a triangular internal face of G^i_ϕ. Add 6 new vertices $u_1, u_2, v_1, v_2, w_1, w_2$ and 9 new edges (u, u_1), (u, u_2), (u_1, u_2), (v, v_1), (v, v_2), (v_1, v_2), (w, w_1), (w, w_2), (w_1, w_2) to G^i_ϕ in such a way that all the new vertices are inside (u, v, w). Note that the n-vertex graph G^i_ϕ is planar and has $\frac{3}{2}(n-1)$ edges; see Fig. 10(a).

We prove that any monotone drawing of G_ϕ has at least $\frac{n-3}{6}$ edges with at least two bends. Let G^t_ϕ be a subgraph of G_ϕ induced by a triangle (u, v, z) of G_ϕ and by all the vertices drawn inside it (see Fig. 10(b)). Let Γ be a drawing of G_ϕ and let Γ^t be a drawing of G^t_ϕ that coincides with Γ restricted to the edges of G^t_ϕ. We have the following.

Claim 1 *Γ is a monotone drawing only if Γ^t is a monotone drawing.*

Proof If Γ^t is not monotone there are two vertices a and b of G^t_ϕ that are not connected by any monotone path in Γ^t. Assume for contradiction that Γ is monotone. Then, a and b are connected by a monotone path P in Γ. As P does not lie entirely in G^t_ϕ, it passes twice through a cut vertex c which is a common vertex of G^t_ϕ and G_ϕ. Thus, the path obtained from P by removing the part between the two occurrences of c is a monotone path between a and b only composed of edges of G^t_ϕ, a contradiction. □

Claim 2 *In any monotone drawing of G^t_ϕ, at least one of the edges (u, v), (v, w), (w, u) has two bends.*

Proof Consider the subgraph G^h_ϕ of G^t_ϕ induced by vertices u, v, w, u_1, v_1, and w_1; see Fig. 10(c). Every monotone drawing of G^t_ϕ restricted to the edges of G^h_ϕ is a monotone drawing of G^h_ϕ. Thus, if there exists a monotone drawing of G^t_ϕ such that none of the edges (u, v), (v, w), (w, u) has two bends, then G^h_ϕ also has such a monotone drawing. However, we show that G^h_ϕ does not admit any such drawing. Let G^s_ϕ be a 1-subdivision of G^h_ϕ. Note that, G^s_ϕ satisfies the preconditions of Lemma 2, that is, it is composed of a cycle plus a set of internal legs connected to every second

vertex of the cycle, and hence it does not admit a straight-line monotone drawing. Thus, G_ϕ^h does not admit a monotone drawing with curve complexity 1, as bends on the edges of G_ϕ^h correspond to the subdivision vertices in G_ϕ^s. □

Claim 1 implies that the drawing of every subgraph defined as G_ϕ^t is monotone, and Claim 2 implies that in any monotone drawing of such a subgraph one of the edges of the outer triangle of G_ϕ^t has two bends. As the number of different triangles that contain a subgraph G_ϕ^t in their interior is $\frac{n-3}{6}$, any monotone drawing of G_ϕ has at least $\frac{n-3}{6}$ edges with two bends, and thus $\frac{n-3}{3}$ bends in total. □

Lemmas 1 and 3 together provide a tight bound on the curve complexity of monotone drawings in the fixed embedding setting. The following theorem summarizes the main contribution of this section.

Theorem 1 *Every embedded planar graph with n vertices admits a monotone drawing with curve complexity 2, at most $4n - 10$ bends in total, and $O(n) \times O(n^2)$ area; such a drawing can be computed in $O(n)$ time. Also, there exist infinitely many embedded planar graphs any monotone drawing of which requires two bends on $\Omega(n)$ edges.*

4 Straight-line Monotone Drawings of Embedded Planar Graphs

In this section we prove that there exist meaningful subfamilies of embedded planar graphs that can be realized as straight-line monotone drawings. In particular, we prove that the class of outerplane graphs and the class of embedded planar biconnected graphs have this property.

4.1 Outerplane Graphs

An embedded planar graph G_ϕ is an *outerplane graph* if all its vertices are incident to the outer face. We prove the following result.

Theorem 2 *Every n-vertex outerplane graph admits a straight-line monotone drawing, which can be computed in $O(n)$ time and requires $O(n) \times O(n^2)$ area.*

Proof Let T be an upright spanning tree of G_ϕ obtained by performing a "rightmost DFS" visit of G_ϕ, that is, a DFS visit in which the neighbors of each vertex are considered in counterclockwise order; see Fig. 11(a). Consider a decomposition of G_ϕ into its maximal biconnected components. Observe that, for each maximal biconnected component B that is connected to the root of T through a cut-vertex v, T contains all the edges of B except for its internal chords (dashed edges in Fig. 11(a)) and for its leftmost edge incident to v (dotted edges in Fig. 11(a)).

A straight-line monotone drawing of G_ϕ is constructed by first computing a straight-line monotone drawing of T, with *Algorithm DFS-based* [2], and then reinserting the edges not in T as straight-line segments.

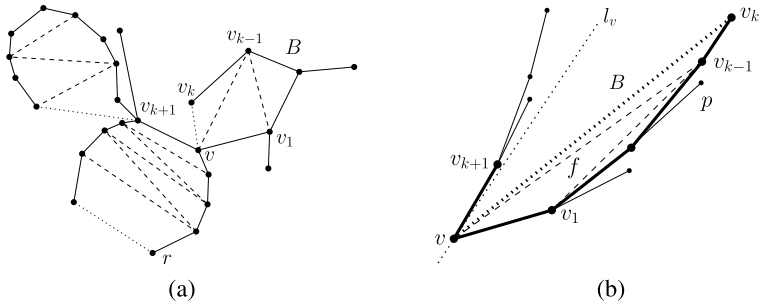

Fig. 11 (**a**) An outerplane graph G_ϕ and its upright spanning tree T obtained by performing a "rightmost DFS" visit. Edges of T are *solid* segments. (**b**) A strictly convex drawing of a maximal biconnected component B of G_ϕ

In order to prove that such edges can be reinserted as straight-line segments without creating crossings, for each maximal biconnected component B consider the path $p = (v, v_1, \ldots, v_k)$ that is composed of the edges belonging both to B and to T. According to *Algorithm DFS-based* [2], the slopes of the edges of p are all positive and are increasing with respect to the distance from v in p. Hence, path p is drawn in T as a polygonal line "convex on the left side", that is, the straight-line segment connecting any two non-consecutive vertices of p completely lies to the left of p; see Fig. 11(b). Thus, when reinserting edge (v, v_k) as the straight-line segment between v and v_k, the boundary of B, namely the cycle composed of the edges of p plus (v, v_k), delimits a strictly-convex region f.

We show that f does not contain any other vertex of T. In particular, it is sufficient to show that the vertex v_{k+1} such that edge (v, v_{k+1}) follows (v, v_1) in the counterclockwise order of the edges around v in T lies outside f.

First, consider a straight line l_v through v that is parallel to edge (v_{k-1}, v_k). According to *Algorithm DFS-based*, the slope of (v_{k-1}, v_k) is the greatest among the slopes of the edges of p. Hence, vertex v_k is to the right of l_v. Thus, the slope of (v_{k-1}, v_k) is greater than or equal to the slope of (v, v_k), with the equality being possible only if $v = v_{k-1}$. Since the slope of (v, v_{k+1}) is greater than the slope of (v_{k-1}, v_k), it follows that v_{k+1} lies outside f; see Fig. 11(b).

From the discussion above, it follows that f is an empty strictly-convex region, and the chords of B can be reinserted as straight-line segments while maintaining planarity.

The obtained drawing is monotone since every pair of vertices is connected by a monotone path composed only of edges of T.

The area of the drawing is the same as the area of T computed by *Algorithm DFS-based*, namely $O(n) \times O(n^2)$. The drawing can be computed in $O(n)$ time. Namely, drawing T by using *Algorithm DFS-based* takes $O(n)$ time [2], and the same holds for reinserting missing edges. □

4.2 Biconnected Graphs

It is known [2] that straight-line monotone drawings of biconnected planar graphs in the variable embedding setting can always be computed by means of an algorithm

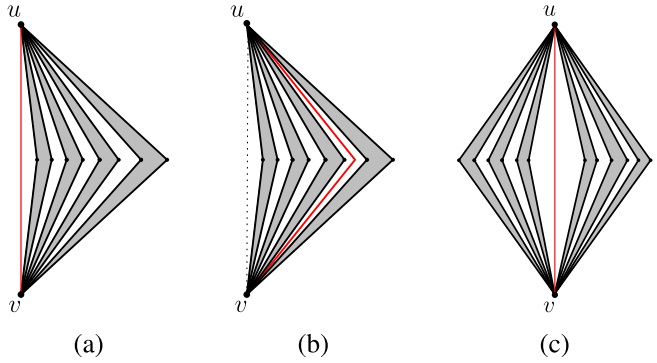

Fig. 12 (**a**) The algorithm in [2] places an edge between the poles of a parallel component μ_P either as the first or as the last element in the order of the children of μ_P. (**b**) An embedding-preserving monotone drawing of μ_P with curve complexity 1. (**c**) An embedding-preserving *straight-line* monotone drawing of μ_P produced with the algorithm described in this section

that, for each component μ of the SPQR-tree of the input graph, draws the pertinent graph of μ inside a shape, called *boomerang*, while respecting some geometrical properties concerning planarity and monotonicity. In Fig. 12(a) the drawing of a parallel component inside a boomerang is depicted.

Note that, this algorithm preserves any given embedding of the input graph, except for the case in which the graph contains a parallel component μ_P whose poles are connected by an edge. In fact, in this case, such an edge is always drawn either as the first or as the last element in the order of the children of μ_P, as in Fig. 12(a), while this might not happen in the given embedding.

On the other hand, when this type of edge exists, an embedding-preserving monotone drawing could be obtained by adding one bend along each such edge in order to place it in its correct position, as in Fig. 12(b), hence obtaining a monotone drawing with curve complexity 1.

In this section we prove that in fact it is possible to compute an embedding-preserving monotone drawing of every embedded biconnected planar graph with no bends at all. Intuitively, when dealing with a parallel component μ_P, we use a shape called *diamond* instead of a boomerang to draw the pertinent graph of μ_P. A diamond is composed of two mirrored copies of a boomerang that are separated by the line through the poles. Then, the edge between the poles, if it exists, is drawn as a straight-line segment, the components preceding it in the order of the children of μ_P are drawn in one of the copies, and the components following it are drawn in the other copy. See Fig. 12(c).

In the following we first give a more detailed description of the algorithm for the variable embedding setting [2] and then we describe our algorithm, that we call BiCoFixedEmbedding, to compute an embedding-preserving straight-line monotone drawing of every embedded biconnected planar graph G.

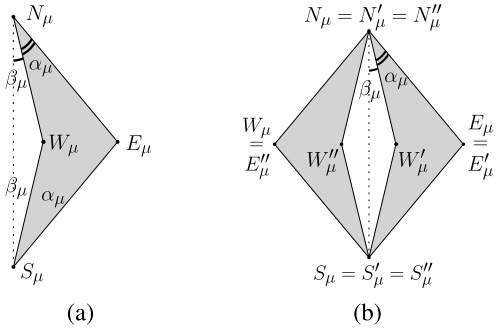

Fig. 13 (**a**) A boomerang. (**b**) A diamond

We start by giving some definitions. A boomerang $boom(\mu)$ is a quadrilateral[1] $(N_\mu, E_\mu, S_\mu, W_\mu)$ such that W_μ is inside triangle $\triangle(N_\mu, S_\mu, E_\mu)$ and $2\alpha_\mu + \beta_\mu < \frac{\pi}{2}$, where $\alpha_\mu = \widehat{W_\mu S_\mu E_\mu} = \widehat{W_\mu N_\mu E_\mu}$ and $\beta_\mu = \widehat{W_\mu S_\mu N_\mu} = \widehat{W_\mu N_\mu S_\mu}$; see Fig. 13(a).

A path monotone with respect to a direction d is (α, d)-*monotone* (with $\alpha < \frac{\pi}{2}$) if for each edge e it holds that $d - \alpha < slope(e) < d + \alpha$. A path from a vertex u to a vertex v is an (α, d_1, d_2)-*path* if it is a composition of an (α, d_1)-monotone path from u to a vertex w and of an (α, d_2)-monotone path from w to v.

The algorithm described in [2] inductively constructs a drawing of any biconnected planar graph G by means of a bottom-up visit of the SPQR-tree of G, as follows. When a component μ with child components $\mu_1, \ldots, \mu_k$ is visited, a drawing Γ_μ of the pertinent graph of μ satisfying the properties (A), (B), (C) hereunder is constructed by composing the drawings of $\mu_1, \ldots, \mu_k$ which, by induction, satisfy the same properties. The composition is based on whether μ is an S-, a P-, a Q-, or an R-node.

(A) Γ_μ is monotone;
(B) Γ_μ is planar and, with the possible exception of edge (u, v), it is contained inside $boom(\mu)$, with u drawn on N_μ and v on S_μ;
(C) each vertex $w \in pert(\mu)$ belongs to a $(\alpha_\mu, -d_N(\mu), d_S(\mu))$-path from u to v, where $d_N(\mu)$ (resp., $d_S(\mu)$) is the half-line from E_μ through N_μ (resp., S_μ).

Algorithm BICOFIXEDEMBEDDING for the fixed-embedding case also relies on a bottom-up visit of the SPQR-tree of G. However, in order to cope with the possible existence of edges connecting the poles of some P-nodes, we defined a new shape, called *diamond* and denoted by $diam(\mu)$, as a convex quadrilateral $(N_\mu, E_\mu, S_\mu, W_\mu)$ composed of two boomerangs $boom'(\mu) = (N'_\mu, E'_\mu, S'_\mu, W'_\mu)$ and $boom''(\mu) = (N''_\mu, E''_\mu, S''_\mu, W''_\mu)$ such that $N_\mu = N'_\mu = N''_\mu$, $S_\mu = S'_\mu = S''_\mu$, $E_\mu = E'_\mu$, and $W_\mu = E''_\mu$. A diamond $diam(\mu)$ is depicted in Fig. 13(b).

Then, when considering a P-node μ having an edge e between its poles, one of the two boomerangs composing the diamond contains the child components of μ that come before e in the ordering of the components around the poles, while the other boomerang contains the other components. In this case, the drawing Γ_μ of $pert(\mu)$ must still satisfy Properties (A), (B), and (C), but in Property (B) the whole

[1] In [2] points $N_\mu, E_\mu, S_\mu, W_\mu$ are denoted as $p_N(\mu), p_E(\mu), p_S(\mu), p_W(\mu)$, respectively.

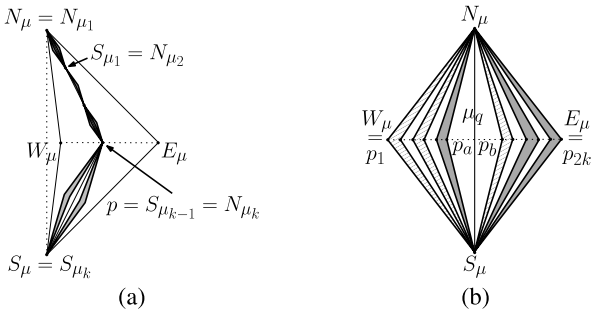

Fig. 14 Construction of a drawing of $pert(\mu)$ satisfying the inductive hypothesis. (**a**) μ is an S-node. (**b**) μ is a P-node. *Grey shaded* boomerangs contain child components, while *grey* tiled boomerangs do not

drawing (including edge (u, v)) is drawn inside the diamond. On the other hand, S- and R-nodes, and P-nodes without an edge between the poles are still drawn inside boomerangs; however, slight modifications are required in their drawing algorithm, as they could now have child P-nodes drawn inside diamonds to handle.

We describe how algorithm BICOFIXEDEMBEDDING computes the drawing Γ_μ for each type of node μ.

Q-node: If μ is a Q-node, $pert(\mu)$ consists only of one edge between the poles. Draw it between points N_μ and S_μ of $boom(\mu)$.

S-node: If μ is an S-node, $pert(\mu)$ must be drawn inside a boomerang $boom(\mu)$. Recall that each child component $\mu_1, \ldots, \mu_k$ might be inductively drawn either inside a boomerang or inside a diamond. Apply the same algorithm as for the variable embedding case. Namely, for each child component μ_i, with $i = 1, \ldots, k-1$, place points N_{μ_i} and S_{μ_i} on the bisector line of $\widehat{W_\mu N_\mu E_\mu}$, and place N_{μ_k} and S_{μ_k} on the bisector line of $\widehat{W_\mu S_\mu E_\mu}$. Hence, even if μ_i $(1 \leq i \leq k)$ is a parallel node drawn inside a diamond, it is still possible to choose angles α_{μ_i} and β_{μ_i} small enough to fit $diam(\mu_i)$ inside $boom(\mu)$, hence satisfying Property (B); see Fig. 14(a). Properties (A) and (C) are easily satisfied by induction, as in the variable embedding case.

More formally, let p be the intersection point between segment $\overline{W_\mu E_\mu}$ and the bisector line of $\widehat{W_\mu N_\mu E_\mu}$. Consider k equidistant points $p_1, \ldots, p_k$ on segment $\overline{N_\mu p}$ such that $p_1 = N_\mu$ and $p_k = p$. For each μ_i, with $i = 1, \ldots, k-1$, consider a boomerang $boom(\mu_i) = (N_{\mu_i}, E_{\mu_i}, S_{\mu_i}, W_{\mu_i})$ such that $N_{\mu_i} = p_i$, $S_{\mu_i} = p_{i+1}$, and such that E_{μ_i} and W_{μ_i} determine $\beta_{\mu_i} + 2\alpha_{\mu_i} < \frac{\alpha_\mu}{2}$. Also, consider a boomerang $boom(\mu_k) = (N_{\mu_k}, E_{\mu_k}, S_{\mu_k}, W_{\mu_k})$ such that $N_{\mu_k} = p$, $S_{\mu_k} = S_\mu$, and such that E_{μ_k} and W_{μ_k} determine $\beta_{\mu_k} + 2\alpha_{\mu_k} < \frac{\alpha_\mu}{2}$. Further, for each μ_i $(1 \leq i \leq k)$ that is a P-node, consider a diamond $diam(\mu_i)$ composed of two mirroring copies of $boom(\mu_i)$. Then, for each $i = 1, \ldots, k$ apply the inductive algorithm to μ_i and either $boom(\mu_i)$ or $diam(\mu_i)$.

P-node: If μ is a P-node, μ must be drawn inside a diamond $diam(\mu)$, while all its child components μ_i, with $i = 1, \ldots, k$, are inductively drawn inside boomerangs $boom(\mu_i)$, as none of them can be a P-node.

First, note that at most one child component μ_q of μ can be a Q-node representing an edge between the poles of μ. Draw such an edge, if any, as a straight-line segment

between N_μ and S_μ. Then, in order to respect the given embedding around the poles of μ, child components $\mu_1, \ldots, \mu_{q-1}$ are drawn inside boomerangs that are contained in the left-hand side of $diam(\mu)$, while the child components $\mu_{q+1}, \ldots, \mu_k$ are drawn inside boomerangs that are contained in the right-hand side of $diam(\mu)$. In order to ensure that the constructed drawings are monotone, we need to fix a total ordering of the components with respect to the angles they form with the line through N_μ and S_μ. This ordering is implicitly guaranteed among components that are on the same side of such a line. In order to obtain it among all the components, we place components $\mu_1, \ldots, \mu_{q-1}$ inside the $q-1$ boomerangs that are the closest to the line through N_μ and S_μ to the left of it, while components $\mu_{q+1}, \ldots, \mu_k$ inside the $k-q$ boomerangs that are the farthest from the line through N_μ and S_μ to the right of it. Refer to Fig. 14(b).

More formally, consider two internal points p_a and p_b of segment $\overline{W_\mu E_\mu}$ such that p_a is to the left of segment $\overline{N_\mu S_\mu}$ and p_b is to the right of $\overline{N_\mu S_\mu}$.

Further, consider $2(k-1)$ points $p_1, \ldots, p_{2(k-1)}$ on segment $\overline{W_\mu p_a}$ such that $p_1 = W_\mu$, $p_{2(k-1)} = p_a$, and $\widehat{p_i N_\mu p_{i+1}} = \frac{\alpha_\mu}{2(k-1)-1}$, for each $i = 1, \ldots, 2(k-1)-1$. For each μ_i, with $i = 1, \ldots, q-1$, consider a boomerang $boom(\mu_j) = (N_{\mu_j}, E_{\mu_j}, S_{\mu_j}, W_{\mu_j})$, with $j = i+k-q$, such that $N_{\mu_j} = N_\mu$, $S_{\mu_j} = S_\mu$, $E_{\mu_j} = p_{2i-1}$, and $W_{\mu_j} = p_{2i}$. Then, apply the inductive algorithm to μ_i and $boom(\mu_j)$.

Finally, consider $2(k-1)$ points $p'_1, \ldots, p'_{2(k-1)}$ on segment $p_b\overline{E_\mu}$ such that $p'_1 = p_b$, $p'_{2(k-1)} = E_\mu$, and $\widehat{p'_i N_\mu p'_{i+1}} = \frac{\alpha_\mu}{2(k-1)-1}$, for each $i = 1, \ldots, 2(k-1)-1$. For each μ_i, with $i = q+1, \ldots, k$, consider a boomerang $boom(\mu_i) = (N_{\mu_i}, E_{\mu_i}, S_{\mu_i}, W_{\mu_i})$ such that $N_{\mu_i} = N_\mu$, $S_{\mu_i} = S_\mu$, $W_{\mu_i} = p_{2i-1}$, and $E_{\mu_i} = p_{2i}$. Apply the inductive algorithm to μ_i and $boom(\mu_i)$.

Claim 3 *If μ is a P-node, the drawing Γ_μ constructed by algorithm* BICOFIXEDEMBEDDING *satisfies Properties* (A)–(C).

Proof Property (B) is satisfied by construction. Property (C) is satisfied by induction. We prove that Γ_μ satisfies Property (A). Consider any two vertices $w_a, w_b \in pert(\mu)$. If they belong to the same child component, then there exists a monotone path between them by induction. If they belong to different components μ_a and μ_b, consider the $(\alpha_{\mu_a}, -d_N(\mu_a), d_S(\mu_a))$-path $P_a(u, v)$ from u to v through w_a and the $(\alpha_{\mu_b}, d_N(\mu_b), -d_S(\mu_b))$-path $P_b(v, u)$ from v to u through w_b, which exist by induction (Property C). Suppose that μ_a lies inside the left boomerang while μ_b lies inside the right boomerang, the other cases being analogous. Note that, by construction, this implies $\beta_{\mu_a} + 2\beta_{\mu_a} < \beta_{\mu_b}$. Also, suppose that w_b lies on the $(\alpha_{\mu_b}, d_N(\mu_b))$-monotone subpath of $P_b(v, u)$, the other case being analogous. Refer to Fig. 15(a).

Consider the $(\alpha_{\mu_b}, d_N(\mu_b))$-monotone path $P(w_b, u)$ from w_b to u and consider the $(\alpha_{\mu_a}, -d_N(\mu_a), d_S(\mu_a))$-path $P(u, w_a)$ from u to w_a that is a subpath of $P_a(u, v)$. We show that the path $P(w_b, w_a)$ composed of $P(w_b, u)$ and $P(u, w_a)$ is monotone. Refer to Fig. 15(b). Rotate the coordinate axes in such a way that u and v lie on the y-axis. Then, when translated to the origin of the axes, $d_N(\mu_b)$ is in the second quadrant, $-d_N(\mu_a)$ in the third quadrant, and $d_S(\mu_a)$ in the fourth quadrant. Since $\beta_{\mu_a} + 2\beta_{\mu_a} < \beta_{\mu_b}$, the wedge delimited by $d_N(\mu_b)$ and $d_S(\mu_a)$ and containing

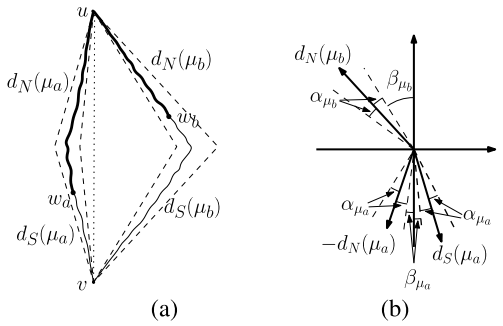

Fig. 15 If μ is a P-node, the constructed drawing satisfies Property (A), that is, it is monotone. (**a**) Component μ_a lies inside the *left* boomerang, μ_b lies inside the *right* boomerang, and w_b lies on the $(\alpha_{\mu_b}, d_N(\mu_b))$-monotone subpath of $P_b(v, u)$. (**b**) Since $\beta_{\mu_a} + 2\beta_{\mu_a} < \beta_{\mu_b}$, the directions of all the edges of the path between u and v are inside a wedge whose angle is smaller than π

the third quadrant has an angle smaller than $\pi - 2\frac{\alpha_\mu}{2k-1}$. Since, by definition, every edge of $P(w_b, u)$ creates an angle with $d_N(\mu_b)$ that is smaller than $\alpha_{\mu_b} = \frac{\alpha_\mu}{2k-1}$ and every edge of $P(u, w_a)$ creates an angle with $d_S(\mu_a)$ that is smaller than $\alpha_{\mu_a} = \frac{\alpha_\mu}{2k-1}$, it follows that the slopes of all the edges of $P(w_b, w_a)$ lie inside a wedge having an angle smaller than π. Hence, $P(w_b, w_a)$ is monotone. □

R-node: If μ is an R-node, $pert(\mu)$ must be drawn inside a boomerang $boom(\mu)$. Also, each child component $\mu_1, \ldots, \mu_k$ might be inductively drawn either inside a boomerang or inside a diamond. As in the S-node case, the drawing algorithm is almost the same as for the variable embedding; see Fig. 14(c). We briefly recall this algorithm and highlight the main differences with our variant for the fixed-embedding case. The algorithm consists of two steps. In the first step, a monotone drawing of $skel(\mu)$ is constructed satisfying some properties concerning monotonicity and the slope of the edges, while in the second step angles α_{μ_i} and β_{μ_i} are chosen in order to fit $boom(\mu_i)$ or $diam(\mu_i)$, for each i, inside $boom(\mu)$.

The monotone drawing of $skel(\mu)$ is constructed as follows. First, consider the graph G^* obtained by removing pole v from $skel(\mu)$. Note that, since $skel(\mu)$ is triconnected, G^* admits a convex drawing whose outer face is represented by any strictly convex polygon, as it satisfies all the conditions of Chiba and Nishizeki [8, 9]. Construct a convex drawing Γ^* of G^*, whose outer face is a convex polygon entirely lying in the interior of $boom(\mu)$, except for pole u which is on N_μ, such that the neighbors of v in $skel(\mu)$ are visible from S_μ. See Fig. 16(a). As Γ^* is convex, it is also monotone [3]. We remark that in [2] one of the neighbors w of u incident to the outer face of $skel(\mu)$ was placed on point E_μ, while in our algorithm this does not happen. Indeed, placing u on N_μ and w on E_μ makes virtual edge (u, w) be on the boundary of $boom(\mu)$. See Fig. 16(b). However, this implies that, if the node ν corresponding to (u, w) is a P-node whose pertinent graph has to be drawn inside a diamond $diam(\nu)$, then $diam(\nu)$ cannot be drawn completely inside $boom(\mu)$, hence not satisfying Property (B). See Fig. 16(c). This problem does not occur in [2], since $pert(\nu)$ is always drawn inside a boomerang, which can be turned in such a way that it is completely inside $boom(\mu)$.

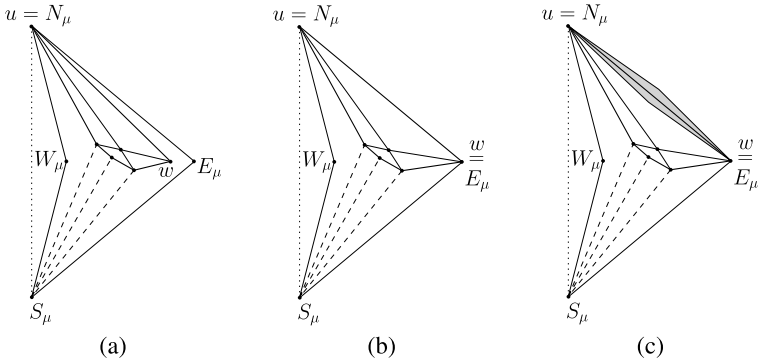

Fig. 16 (**a**) A convex polygon entirely lying in the interior of $boom(\mu)$, except for pole u which is on N_μ, such that the neighbors of v in $skel(\mu)$ are visible from S_μ. (**b**) The corresponding polygon in [2], in which w is on E_μ. (**c**) If the node v corresponding to (u, w) has to be drawn inside a diamond, then it is drawn outside $boom(\mu)$

Second, apply an affine transformation to Γ^*, called *directional-scale* in [2], in order to make the slopes of all the edges of G^* close enough to $-d_N(\mu)$. See Fig 17(a). Namely, the directional-scale applies a scaling to the drawing in a direction that is perpendicular to $-d_N(\mu)$. As proved in [2] the resulting drawing is still monotone and, for each edge $e \in G^*$, it holds $slope(-d_N(\mu)) - \frac{\alpha_\mu}{2} < slope(e) < slope(-d_N(\mu)) + \frac{\alpha_\mu}{2}$.

Third, construct a drawing $\Gamma(skel(\mu))$ of $skel(\mu)$ by placing v on S_μ in Γ^* and connecting it to its neighbors. Note that v is connected to each vertex w of $skel(\mu)$ by an $(\alpha_\mu, -d_N(\mu), d_S(\mu))$-path that is a composition of the $(\alpha_\mu, d_S(\mu))$-monotone path composed only of edge (v, w'), for some vertex w' adjacent to v, and of the $(\alpha_\mu, -d_N(\mu))$-monotone path connecting w' to w, which exists due to the fact that, for each edge $e \in G^*$, $slope(-d_N(\mu)) - \frac{\alpha_\mu}{2} < slope(e) < slope(-d_N(\mu)) + \frac{\alpha_\mu}{2}$. This, together with the fact that every pair of vertices different from v is connected in $\Gamma(skel(\mu))$ by the same monotone path as in Γ^*, implies that $\Gamma(skel(\mu))$ is monotone.

Finally, consider a drawing $\Gamma'(skel(\mu))$ of a subdivision of $skel(\mu)$ obtained from $\Gamma(skel(\mu))$ by placing the two edges incident to each subdivision vertex on the same straight-line segment. As proved in Lemma 3 of [2], $\Gamma'(skel(\mu))$ is still a monotone drawing.

Then, in order to obtain Γ_μ, each virtual edge of $skel(\mu)$ is replaced in $\Gamma'(skel(\mu))$ with either a boomerang or a diamond, depending on the type of node it represents, as follows. Consider the pair of vertices x, y belonging to the subdivision of $skel(\mu)$ such that the range $range(P(x, y))$ of the monotone path $P(x, y)$ between them in $\Gamma'(skel(\mu))$ creates the largest angle $\angle(x, y)$ among all the pairs of vertices. Let $\gamma = \pi - \angle(x, y)$. Further, let δ be the smallest angle between two adjacent edges in $\Gamma(skel(\mu))$. Finally, let ϵ be the smallest angle between an edge incident to u and segment $\overline{N_\mu E_\mu}$. For each node μ_i, with $i = 1, \ldots, k$, represented by virtual edge (u_i, v_i), let N_{μ_i} and S_{μ_i} be the points where u_i and v_i have been drawn in $\Gamma(skel(\mu))$, respectively. Then, consider a boomerang $boom(\mu_i) = (N_{\mu_i}, E_{\mu_i}, S_{\mu_i}, W_{\mu_i})$ such that E_{μ_i} and W_{μ_i} determine $\beta_{\mu_i} + 2\alpha_{\mu_i} < \min\{\frac{\delta}{2}, \frac{\gamma}{2}, \frac{\epsilon}{2}\}$. If μ_i is a P-node with an edge

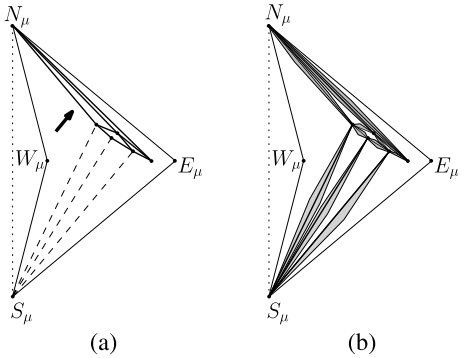

Fig. 17 (**a**) A directional-scale applied to Γ^* in the direction perpendicular to $-d_N(\mu)$. (**b**) The drawing of $pert(\mu)$ obtained with the algorithm described in this section

between its poles, consider a diamond $diam(\mu_i)$ composed of two mirrored copies of $boom(\mu_i)$. Then, apply the inductive algorithm to μ_i, with poles u_i and v_i, and to either $boom(\mu_i)$ or $diam(\mu_i)$. See Fig. 17(b).

Claim 4 *If μ is an R-node, the drawing Γ_μ constructed by algorithm* BICOFIXEDEMBEDDING *satisfies Properties (A)–(C).*

Proof The fact that Γ_μ satisfies Property (A) depends on the monotonicity of $\Gamma'(skel(\mu))$ and on the fact that $\beta_{\mu_i} + 2\alpha_{\mu_i} < \frac{\gamma}{2}$, where $\gamma = \pi - \angle(x, y)$ and $\angle(x, y)$ is the largest angle created by the range $range(P(x, y))$ of the monotone path connecting any two vertices x and y. This implies that, for every monotone path between two vertices w_1 and w_2 in $\Gamma'(skel(\mu))$ (whose range creates an angle smaller than $\angle(x, y)$, by definition), it is possible to construct a monotone path in Γ_μ having range (strictly) smaller than $\pi - \angle(x, y) + \angle(w_1, w_2) \leq \pi$.

Property (B) follows from the planarity of the drawings of the child nodes, which is guaranteed by induction, and on the fact that the boomerangs and the diamonds containing such drawings do not overlap, due to the fact that $\beta_{\mu_i} + 2\alpha_{\mu_i} < \frac{\delta}{2}$. Also, as remarked before, all such boomerangs and diamonds are contained inside $boom(\mu)$, as no vertex has been placed on point E_μ.

Since for each edge $e \in \Gamma(skel(\mu))$ it holds $slope(-d_N(\mu)) - \frac{\alpha_\mu}{2} < slope(e) < slope(-d_N(\mu)) + \frac{\alpha_\mu}{2}$, if e is not incident to v, and $slope(d_S(\mu)) - \frac{\alpha_\mu}{2} < slope(e) < slope(d_S(\mu)) + \frac{\alpha_\mu}{2}$, if it is incident to e, and since $\beta_{\mu_i} + 2\alpha_{\mu_i} < \frac{\alpha_\mu}{2}$, Property (C) is also satisfied by Γ_μ. □

We state the main theorem of this section.

Theorem 3 *Algorithm* BICOFIXEDEMBEDDING *computes a straight-line embedding-preserving monotone drawing of every n-vertex biconnected embedded planar graph G in $O(n)$ time.*

Proof The fact that algorithm BICOFIXEDEMBEDDING computes the required drawing follows from the fact that, as proved in the above discussion, Properties (A), (B), and (C) hold for each node of the SPQR-tree of G.

Concerning the linear bound on the computational complexity, first note that SPQR-trees can be constructed and handled in linear time [4, 5, 12]. Also, the computation of angles α_{μ_i} and β_{μ_i} at each step of the computation and, in the case of R-nodes, the construction of a convex drawing of $skel(\mu)$ and the operation directional-scale, can be performed in linear time in the size of the considered node, and hence in total linear time in the size of the graph. □

5 Conclusions and Open Problems

In this paper we studied monotone drawings of graphs in the fixed embedding setting. Since not all embedded planar graphs admit an embedding-preserving monotone drawing with straight-line edges, we focused on computing embedding-preserving monotone drawings with low curve complexity. We proved that curve complexity 2 always suffices and that this bound is worst-case optimal. Furthermore, we described algorithms for computing straight-line monotone drawings for meaningful subfamilies of embedded planar graphs. All the algorithms presented in this paper can be performed in linear time and most of them produce drawings which require polynomial area.

The results in this paper naturally give rise to several interesting open problems; some of them are listed below.

Existential Questions

Problem 1 Find meaningful subfamilies of embedded planar graphs (other than outerplane graphs and embedded biconnected graphs) that admit monotone drawings with curve complexity smaller than 2.

Problem 2 Is it possible to characterize the embedded planar graphs that admit monotone drawings with curve complexity smaller than 2?

Complexity Questions

Problem 3 Given an embedded planar graph G_ϕ and an integer $k \in \{0, 1\}$, what is the complexity of deciding whether G_ϕ admits a monotone drawing with curve complexity k?

Problem 4 Given a graph G and an integer $k \in \{0, 1\}$, what is the complexity of deciding whether there exists an embedding ϕ such that G_ϕ admits a monotone drawing with curve complexity k?

Problem 5 Given a graph G and an integer $k \in \{0, 1\}$, what is the complexity of deciding whether there exists an embedding ϕ such that G_ϕ does not admit any monotone drawing with curve complexity k?

Notice that, although Problems 3–5 are related, there is no evidence that answering one of them implies an answer for any other.

Algorithmic Questions

Observe that the length of the edges and the angles β_{μ_i} and α_{μ_i} in a drawing produced by the algorithm described in Theorem 3 are reduced at each step, which implies that the area of the drawing is not polynomially bounded.

Problem 6 Is there any algorithm that computes monotone drawings of embedded biconnected planar graphs in polynomial area?

Problem 7 Is there any algorithm that computes monotone drawings of outerplane graphs in subcubic area?

Acknowledgements We thank the anonymous reviewers for their valuable comments.

References

1. Angelini, P., Frati, F., Grilli, L.: An algorithm to construct greedy drawings of triangulations. J. Graph Algorithms Appl. **14**(1), 19–51 (2010)
2. Angelini, P., Colasante, E., Battista, G.D., Frati, F., Patrignani, M.: Monotone drawings of graphs. J. Graph Algorithms Appl. **16**(1), 5–35 (2012). Special Issue on Selected Papers from GD '10
3. Arkin, E.M., Connelly, R., Mitchell, J.S.B.: On monotone paths among obstacles with applications to planning assemblies. In: Symposium on Computational Geometry, pp. 334–343 (1989)
4. Battista, G.D.: On-line maintenance of triconnected components with SPQR-trees. Algorithmica **15**(4), 302–318 (1996)
5. Battista, G.D.: On-line planarity testing. SIAM J. Comput. **25**, 956–997 (1996)
6. Brocot, A.: Calcul des rouages par approximation, nouvelle methode. Rev. Chronom. **6**, 186–194 (1860)
7. Chazelle, B.: Triangulating a simple polygon in linear time. Discrete Comput. Geom. **6**, 485–524 (1991)
8. Chiba, N., Nishizeki, T.: Planar Graphs: Theory and Algorithms. Annals of Discrete Mathematics, vol. 32. North-Holland, Amsterdam (1988)
9. Chiba, N., Onoguchi, K., Nishizeki, T.: Drawing plane graphs nicely. Acta Inform. **22**, 187–201 (1985)
10. Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G.: Graph Drawing. Prentice Hall, Upper Saddle River (1999)
11. Garg, A., Tamassia, R.: Upward planarity testing. Order **12**(2), 109–133 (1995)
12. Gutwenger, C., Mutzel, P.: A linear time implementation of spqr-trees. In: Marks, J. (ed.) Graph Drawing. Lecture Notes in Computer Science, vol. 1984, pp. 77–90. Springer, Berlin (2001)
13. Huang, W., Eades, P., Hong, S.-H.: A graph reading behavior: geodesic-path tendency. In: PacificVis, pp. 137–144 (2009)
14. Leighton, T., Moitra, A.: Some results on greedy embeddings in metric spaces. Discrete Comput. Geom. **44**(3), 686–705 (2010)
15. Papadimitriou, C.H., Ratajczak, D.: On a conjecture related to geometric routing. Theor. Comput. Sci. **344**(1), 3–14 (2005)
16. Stern, M.A.: Ueber eine zahlentheoretische Funktion. J. Reine Angew. Math. **55**, 193–220 (1858)