Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Good spanning trees in graph drawing

Md. Iqbal Hossain , Md. Saidur Rahman

Graph Drawing and Information Visualization Laboratory, Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka-1000, Bangladesh

ARTICLE INFO

Article history:
Received 15 November 2014
Received in revised form 1 September 2015
Accepted 2 September 2015
Available online xxxx

Keywords:
Good spanning tree
Graph drawing
Monotone drawing
Visibility representation

ABSTRACT

A plane graph is a planar graph with a fixed planar embedding. In this paper we define a special spanning tree of a plane graph which we call a good spanning tree. Not every plane graph has a good spanning tree. We show that every connected planar graph has a planar embedding with a good spanning tree. Using a good spanning tree, we show that every connected planar graph G of n vertices has a straight-line monotone grid drawing on an $O(n) \times O(n^2)$ grid, and such a drawing can be found in $O(n)$ time. Our results solve two open problems on monotone drawings of planar graphs posed by Angelini et al. Using good spanning trees, we also give simple linear-time algorithms for finding a 2-visibility representation of a connected planar graph G of n vertices on a $(2n-1) \times (2n-1)$ grid and for finding a spike-*VPG* representation of G on a $(2n-1) \times n$ grid.

1. Introduction

The field of graph drawing has been flourishing very much in the last two decades. Recent progress in computational geometry, topological graph theory, and order theory has considerably affected the evolution of this field, and has widened the range of issues being investigated. Drawing of planar graphs with various constraints imposed by application has been studied in recent years. Many algorithmic tools such as canonical ordering [1], regular edge labeling [2], Schnyder realizer [3], orderly spanning trees [4,5] have been developed to solve various types of graph drawing problems. In this paper we introduce a special spanning tree in a plane graph which we call a *good spanning tree*. A good spanning tree is an ordered rooted spanning tree of a plane graph where the tree edges and the non-tree edges incident to a vertex obey some properties. Fig. 1 illustrates a good spanning tree T where the tree edges are drawn as thick lines. Observe the ordering of the tree edges and non-tree edges incident to vertex v in T where the two sets of consecutive non-tree edges are separated by a set of consecutive tree edges and the tree edge (v, u). Also no non-tree edge incident to v has the other end on the path from r to v in T. A good spanning tree of a plane graph can be considered as a generalization of a Schnyder realizer of a triangulated plane graph. We give a formal definition of a good spanning tree in Section 2. Not every plane graph has a good spanning tree: for example, the plane graphs in Fig. 2 do not have good spanning trees. However, we show that every planar graph has a planar embedding with a good spanning tree. Furthermore, we show that a good spanning tree has useful applications in the field of graph drawing, namely, in monotone drawings, in 2-visibility representations and in *VPG* representations. We next discuss each of the representations briefly and present our result for each case.

* Corresponding author.
E-mail addresses: mdiqbalhossain@cse.buet.ac.bd (M.I. Hossain), saidurrahman@cse.buet.ac.bd (M.S. Rahman).

http://dx.doi.org/10.1016/j.tcs.2015.09.004
0304-3975/

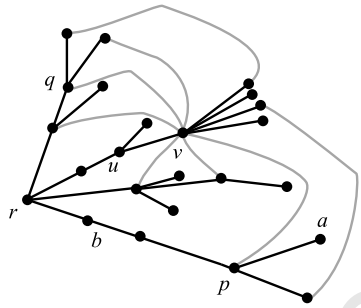

Fig. 1. Example of a good spanning tree T.

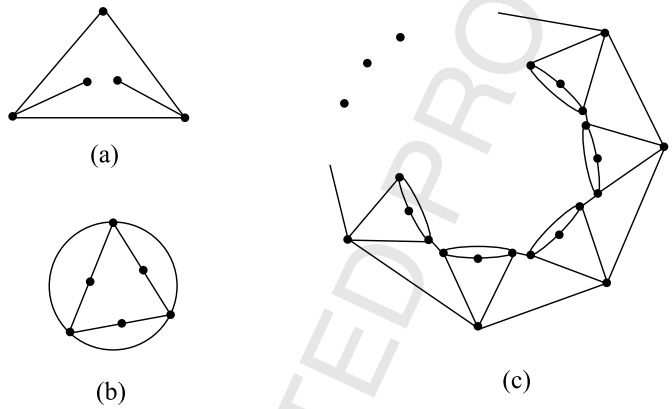

Fig. 2. Some examples of plane graphs that have no good spanning trees: (a) a connected plane graph, (b) a biconnected plane graph and (c) a plane graph with many vertices that has no good spanning tree.

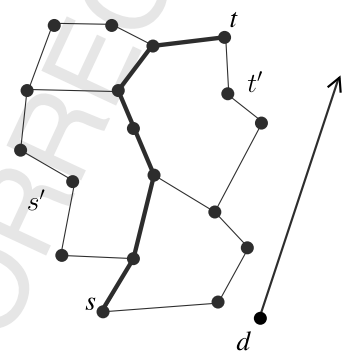

Fig. 3. The path between vertices s and t (as shown as a thick line) is monotone with respect to the direction d.

1.1. Monotone drawings

A *straight-line drawing* of a planar graph G is a drawing of G in which each vertex is drawn as a point and each edge is drawn as a straight-line segment without any edge crossing. A path P in a straight-line drawing of a planar graph is *monotone* if there exists a line l such that the orthogonal projections of the vertices of P on l appear along l in the order induced by P. A straight-line drawing Γ of a planar graph G is a *monotone drawing* of G if Γ contains at least one monotone path between every pair of vertices [6–8]. In the drawing of a graph in Fig. 3, the path between the vertices s and t drawn as a thick line is a monotone path with respect to the direction d, whereas no monotone path exists with respect to any direction between the vertices s' and t'. We call a monotone drawing of a planar graph a *monotone grid drawing* if every vertex is drawn on a grid point.

Monotone drawings of graphs are well motivated by human subject experiments by Huang et al. [9], who showed that the "geodesic tendency" (paths following a given direction) is important in comprehending the underlying graph. *Upward drawings* [10–13] are related to monotone drawings where every directed path is monotone with respect to the vertical line, while in a monotone drawing each monotone path, in general, is allowed to be monotone with respect to a different line.

Please cite this article in press as: M.I. Hossain, M.S. Rahman, Good spanning trees in graph drawing, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.09.004

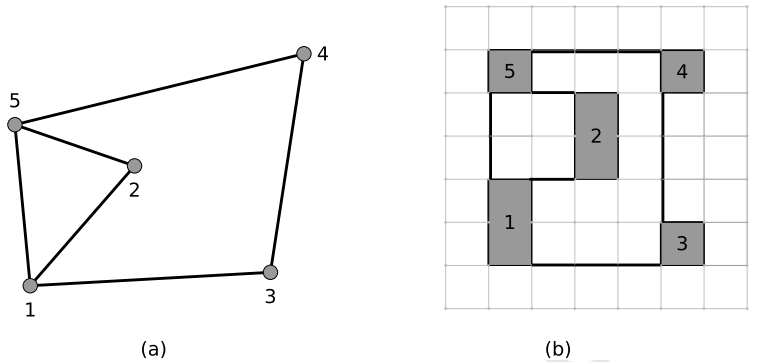

Fig. 4. (a) A planar graph G and (b) a 2-visibility representation of G.

Arkin et al. [14] showed that any strictly convex drawing[1] of a planar graph is monotone and they gave an $O(n \log n)$ time algorithm for finding such a path between a pair of vertices in a strictly convex drawing of a planar graph of n vertices. Angelini et al. [6] showed that every biconnected planar graph of n vertices has a monotone drawing in real coordinate space. They also showed that every tree of n vertices admits a monotone grid drawing on a grid of size $O(n) \times O(n^2)$, and posed an open problem "Is it possible to construct monotone grid drawings of biconnected planar graphs in polynomial area?" Addressing the problem, Hossain and Rahman [7] showed that every series-parallel graph of n vertices admits a monotone grid drawing on an $O(n) \times O(n^2)$ grid, and such a drawing can be found in $O(n \log n)$ time. However, the problem mentioned above remains open.

It is known that not every plane graph (with a fixed embedding) admits a monotone drawing [6]. However, every outerplane graph of n vertices admits a monotone grid drawing on a grid of area $O(n) \times O(n^2)$ [8]. Angelini et al. showed that every connected plane graph admits a "polyline" monotone grid drawing on an $O(n) \times O(n^2)$ grid using at most two bends per edge [8], and posed an open problem "Given a planar graph G and an integer $k \in \{0, 1\}$, what is the complexity of deciding whether there exists a planar embedding G_ϕ such that G_ϕ admits a monotone drawing with the curve complexity k?"

We show that every connected planar graph of n vertices has a monotone grid drawing on an $O(n) \times O(n^2)$ grid, and such a drawing can be computed in $O(n)$ time. We thus solve the pair of open problems mentioned above by showing that every planar graph has a planar embedding G_ϕ such that G_ϕ admits a monotone grid drawing with curve complexity 0 on an $O(n) \times O(n^2)$ grid, and such a planar embedding and also a monotone drawing can be found in linear time.

1.2. 2-Visibility representations

A *visibility representation* or a *1-visibility representation* (VR for short) of a planar graph G is a planar drawing of G on a two-dimensional grid such that the vertices of G are represented by non-overlapping horizontal line segments (called vertex segments) and each edge of G is represented by a vertical line segment touching the vertex segments of its end vertices (see Fig. 4). The problem of computing a compact VR has practical applications in VLSI layout [15,16]. Several research works have been done on 1-visibility representations [17–20].

A *2-visibility representation (2-VR)* of a planar graph G is a planar drawing of G on a two dimensional grid such that each vertex is drawn as a rectangle (rather than a small point) and each edge is drawn as either a horizontal or a vertical line segment that lies on a grid line. A 2-visibility representation is a straightforward generalization of a 1-visibility representation where both vertical and horizontal edges are allowed. The drawing model of a 2-visibility representation is directly associated with VLSI layout diagrams, database schema diagrams and entity-relationship diagrams. A very similar drawing is found in literature that is called a *box-rectangular drawing* where each vertex is drawn as a rectangle, called a *box*, each edge is drawn as either a horizontal line segment or a vertical line segment, and the contour of each face is drawn as a rectangle. Not every planar graph admits a box-rectangular drawing [21–23].

Every tree admits a 2-visibility representation [24]. Fößmeier and Kaufmann showed that every planar graph has a 2-visibility representation using rectangles of equal width with at most two bends per edge [25]. Later Fößmeier et al. improved the result and they showed that every planar graph has a 2-visibility drawing with at most 1 bend per edge [26].

In this paper using a good spanning tree we give a simple linear-time algorithm to find a 2-visibility representation of a planar graph on a grid of size $(2n - 1) \times (2n - 1)$, where each edge is drawn as a straight-line segment of length at least one unit.

[1] A *strictly convex drawing* of a planar graph is a straight-line drawing in which all faces, including the outer face, are strictly convex polygons.

Please cite this article in press as: M.I. Hossain, M.S. Rahman, Good spanning trees in graph drawing, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.09.004

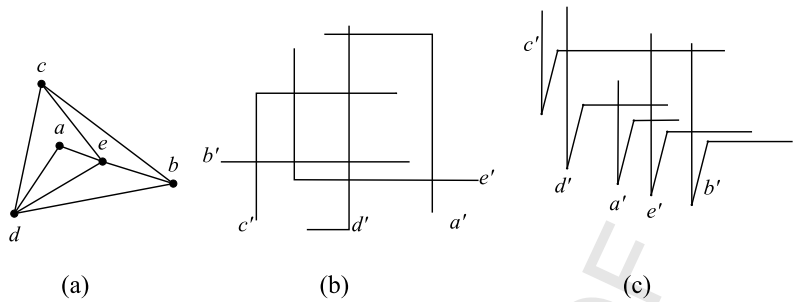

Fig. 5. (a) A planar graph G, (b) a B_1-*VPG* representation of G and (c) a spike-*VPG* representation of G.

1.3. VPG representations

Let $\mathcal{P}$ be a set of simple paths. The *vertex intersection graph* $VPG(P)$ of $\mathcal{P}$ has the vertex set V, where every vertex $v \in V$ corresponds to a path $P_v \in \mathcal{P}$, and the edge set E, where an edge $(u, v) \in E$ if and only if the corresponding paths P_u and P_v intersect, i.e., $E = \{(u, v) | u, v \in V, u \neq v, P_u \cap P_v \neq \emptyset\}$. We call a graph G a *VPG* graph if $G = VPG(\mathcal{P})$, for some $\mathcal{P}$. If $\mathcal{P}$ is a set of simple paths on a grid, where each path has at most k bends, then the graph G is called a B_k-*VPG*. Fig. 5(b) illustrates a B_1-*VPG* representation of the planar graph in Fig. 5(a).

Interval graphs, trees, bipartite graphs are B_0-*VPG* graphs [27]. Every circle graph is a B_1-*VPG*. Asinowski et al. [28] showed that all planar graphs are B_3-*VPGs* [28]. Recently, Chaplick and Ueckerdt [29] improved the result of Asinowski et al. and showed that a B_2-*VPG* of a planar graph can be constructed by using two types of paths (Z-shape and C-shape) in $O(n^{3/2})$ time, where n is the number of vertices. For both the B_3-*VPG* and B_2-*VPG* cases the drawing sizes are not known.

In this paper we consider a variation of a B_2-*VPG* representation, where each path is a spike shaped curve. A spike shaped curve consists of three straight-line segments, two of them lie on grid lines, as illustrated in Fig. 5(c). Since each path has two bends but one line segment of a spike shaped curve does not lie on a grid line, we call this representation a *spike-VPG* representation. A spike-*VPG* can be considered as a weak B_2-*VPG* representation of a graph. Fig. 5(c) illustrates a spike-*VPG* representation of the planar graph in Fig. 5(a). In this paper we show that every planar graph has a spike-*VPG* representation on a $(2n - 1) \times n$ grid and such a representation can be found in $O(n)$ time.

The rest of the paper is organized as follows. In Section 2, we define a good spanning tree and give an algorithm to compute a planar embedding of a planar graph having a good spanning tree. As applications of a good spanning tree we deal with monotone drawings, 2-visibility representations and *VPG* representations in Sections 3, 4 and 5, respectively. Finally, Section 6 concludes the paper with discussions. An early version containing some results on monotone drawings has been presented as [30].

2. Good spanning trees

In this section we give some graph theoretic terminologies, define a good spanning tree of a plane graph and show that every planar graph has a planar embedding containing a good spanning tree. For the graph theoretic terminologies not defined in this paper, we refer to [12].

Let $G = (V, E)$ be a graph with vertex set V and edge set E such that $|V| = n$ and $|E| = m$. The degree of a vertex v in G is denoted by $d(v)$. We denote an edge joining vertices u and v of G by (u, v). A *subgraph* of G is a graph $G' = (V', E')$ such that $V' \subseteq V$ and $E' \subseteq E$. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. The *union* of G_1 and G_2, denoted by $G_1 \cup G_2$, is the graph $G_3 = (V_3, E_3)$ such that $V_3 = V_1 \cup V_2$ and $E_3 = E_1 \cup E_2$. If v is a vertex in G, then $G - v$ is the subgraph of G obtained by deleting the vertex v and all the edges incident to v. Similarly, if e is an edge of G, then $G - e$ is a subgraph of G obtained by deleting the edge e. A vertex v of a connected graph G is a *cut vertex* of G if $G - v$ is disconnected.

Let $G = (V, E)$ be a connected graph and $T = (V, E')$ be a spanning tree of G. An edge $e \in E$ is called a *tree edge* of G for T if $e \in E'$ otherwise e is said to be a *non-tree edge* of G for T. Let e be a non-tree edge of G for T. Then by $T \cup e$ we denote the subgraph G' of G obtained by adding edge e to T. The graph $G' = T \cup e$ has exactly one cycle C and we call C the *cycle induced by the non-tree edge* e. If X is a set of edges of G and T is a spanning tree of G then $T \cup X$ denotes the graph obtained by adding the edges in X to T and removing multiple edges. Let T be a rooted tree and let u be a vertex of T. Then by T_u we denote the subtree of T rooted at u. By $T - T_u$ we denote the tree obtained from T by deleting the subtree T_u.

A graph is *planar* if it can be embedded in the plane without edge intersections except at endpoints. A *plane graph* is a planar graph with a fixed planar embedding. A plane graph divides the plane into some connected regions called *faces*. The unbounded region is called the *outer face* and each of the other faces is called an *inner face*. Let G be a plane graph. The contour of the outer face of G is called the *outer boundary* of G. For the plane graph in Fig. 6, the contour $a, b, c, e, f, k, j, i, h, f, g, f, e, a$ is the outer boundary. We call a vertex v of G an *outer vertex* of G if v is on the outer boundary of G, otherwise v is an *inner vertex* of G. We call a simple cycle induced by the outer boundary of G an *outer cycle* of G. Note that if G is not biconnected, G may have more than one outer cycles. The plane graph in Fig. 6 has two outer

Please cite this article in press as: M.I. Hossain, M.S. Rahman, Good spanning trees in graph drawing, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.09.004

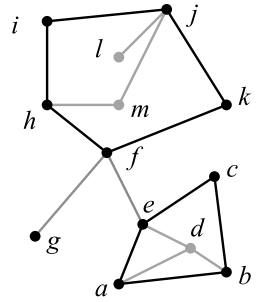

Fig. 6. Two outer cycles in the graph are shown as dark edges.

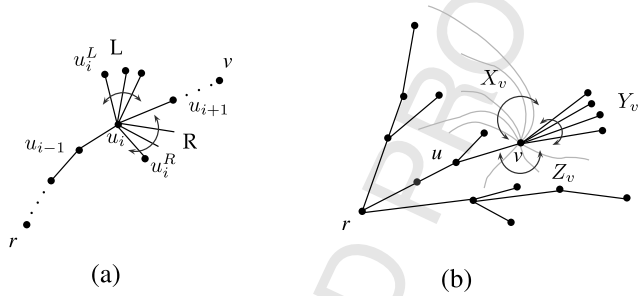

Fig. 7. (a) An illustration for $P(r, v)$, L and R groups, (b) an illustration for X_v, Y_v and Z_v sets of edges.

cycles $C_1 = a, b, c, e, a$ and $C_2 = f, k, j, i, h, f$. For a cycle C in a plane graph G, we denote by $G(C)$ the plane subgraph of G inside C (including C). Let T be a rooted tree. We call T an *ordered rooted tree* if the children of each vertex v in T are ordered in a fixed ordering, say, in a counterclockwise ordering around v.

Let G be a connected planar graph and let G_ϕ be a planar embedding of G. Let T be an ordered rooted spanning tree of G_ϕ such that the root r of T is an outer vertex of G_ϕ, and the ordering of the children of each vertex v in T is consistent with the ordering of the neighbors of v in G_ϕ. Let $P(r, v) = u_1(= r), u_2, \ldots, u_k(= v)$ be the path in T from the root r to a vertex $v \neq r$. The path $P(r, v)$ divides the children of u_i $(1 \leq i < k)$, except u_{i+1}, into two groups; the left group L and the right group R. A child x of u_i is in the group L and denoted by u_i^L if the edge (u_i, x) appears before the edge (u_i, u_{i+1}) in clockwise ordering of the edges incident to u_i when the ordering is started from the edge (u_i, u_{i-1}), as illustrated in the Fig. 7(a). Similarly, a child x of u_i is in the group R and denoted by u_i^R if the edge (u_i, x) appears after the edge (u_i, u_{i+1}) in clockwise order of the edges incident to u_i when the ordering is started from the edge (u_i, u_{i-1}). We call T a *good spanning* tree of G_ϕ if every vertex v $(v \neq r)$ of G satisfies the following two conditions with respect to $P(r, v)$.

(Cond1) G does not have a non-tree edge (v, u_i), $i < k$; and
(Cond2) the edges of G incident to the vertex v excluding (u_{k-1}, v) can be partitioned into three disjoint (possibly empty) sets X_v, Y_v and Z_v satisfying the following conditions (a)–(c) (see Fig. 7(b)):
- (a) Each of X_v and Z_v is a set of consecutive non-tree edges and Y_v is a set of consecutive tree edges.
- (b) Edges of set X_v, Y_v and Z_v appear clockwise in this order from the edge (u_{k-1}, v).
- (c) For each edge $(v, v') \in X_v$, v' is contained in $T_{u_i^L}$, $i < k$, and for each edge $(v, v') \in Z_v$, v' is contained in $T_{u_i^R}$, $i < k$.

Fig. 1 illustrates a good spanning tree T in a plane graph. In the rest of the section we prove that every connected planar graph G has a planar embedding G_ϕ such that G_ϕ contains a good spanning tree T, as in the following theorem.

Theorem 1. *Let G be a connected planar graph of n vertices. Then G has a planar embedding G_ϕ that contains a good spanning tree. Furthermore, G_ϕ and a good spanning tree T of G_ϕ can be found in $O(n)$ time.*

To prove Theorem 1, we need a few definitions. Let v be a cut-vertex in a connected graph G. We call a subgraph H of G a *v-component* of G if H consists of a maximal connected subgraph H' of $G - v$ and all edges joining v to the vertices of H'. A pair $\{u, v\}$ of vertices in G is a *split pair* if there exist two subgraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ satisfying the following two conditions: 1. $V = V_1 \cup V_2, V_1 \cap V_2 = \{u, v\}$; and 2. $E = E_1 \cup E_2, E_1 \cap E_2 = \emptyset, |E_1| \geq 1, |E_2| \geq 1$. Thus every pair of adjacent vertices is a split pair. A *$\{u, v\}$-split component* of a split pair $\{u, v\}$ in G is a maximal connected subgraph H of G such that $\{u, v\}$ is not a split pair of H.

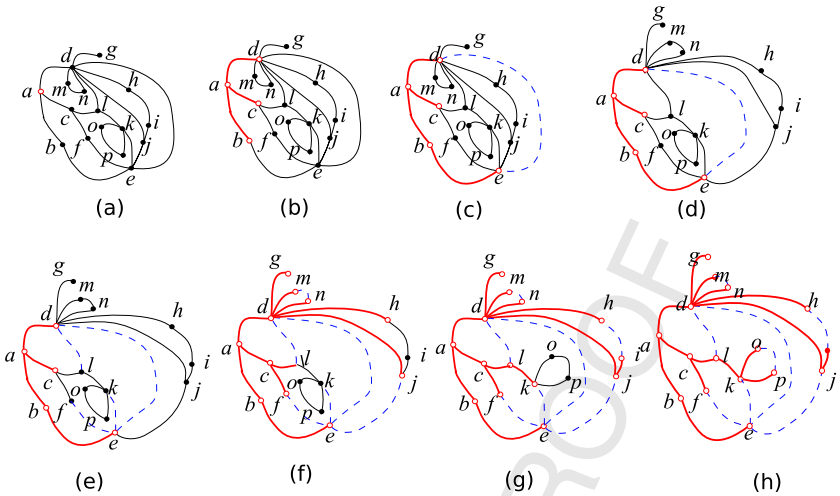

Fig. 8. Illustration for an outline of construction of a good spanning tree T. White vertices are visited vertices. Black vertices are not visited. Solid edges are tree edges. Dashed edges are non-tree edges. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

We give a constructive proof of Theorem 1. Before giving our formal proof we give an outline of our construction using an illustrative example in Fig. 8. We take an arbitrary planar embedding G_γ of G and start a breath-first-search (BFS) from an arbitrary outer vertex r of G_γ and regard r as the root of our desired spanning tree. In Fig. 8(a) the BFS is started from vertex a, and vertex b, c and d are visited from a in this order, as illustrated in Fig. 8(b). We next visit e from b, as illustrated in Fig. 8(c). When we visit a new vertex u, we check whether there is an edge (u, v) such that v is already visited and there is a (u, v)-split component or a u-component or a v-component inside the cycle induced by the edge (u, v) which does not contain the root r. The $\{e, d\}$-split component H_1 induced by the vertices $\{d, h, i, j, e\}$ is such a split component in Fig. 8(c) and the subgraph H_2 induced by vertices d, m, n is such a u-component for $u = d$ which are inside the cycle induced by the edge (e, d). We move the subgraphs H_1 and H_2 out of the cycle induced by the non-tree edge (e, d), as illustrated in Fig. 8(d). Since (b, e) is a tree edge and (e, d) is a non-tree edge, according to the definition of a good spanning tree, the edges (e, f) and (e, k) must be non-tree edges. Similarly, since (a, d) is a tree edge and (e, d) a non-tree edge, the edge (d, l) must be a non-tree edge. We mark (e, f), (e, k) and (d, l) as non-tree edges as shown in the Fig. 8(e). We then visit vertices f, l, m, n, g, h and j, as illustrated in Fig. 8(f). When we visit k, we find a k-component H induced by vertices $\{k, p, o\}$ and we move H out of the cycle induced by (e, k) as shown in Fig. 8(g). The BFS continues visit the remaining vertices i, k, p and o. At the end of the BFS, we find a planar embedding G_ϕ of G and a good spanning tree T as illustrated in Fig. 8(h), where the edges of the good spanning tree T are drawn as solid lines, and non-tree edges are drawn as dashed lines.

We now formally prove Theorem 1.

Proof of Theorem 1. Let G_γ be any arbitrary planar embedding of G. We first mark an arbitrary outer vertex r of G_γ visited, and start counterclockwise BFS from r. The vertex r will be the root of the BFS tree T.

Note that after visiting each vertex by BFS, the embedding of G may be changed by our algorithm. Let G_γ^i be the planar embedding of G after visiting the ith vertex by BFS. Then $G_\gamma^1 = G_\gamma$ since we do not change the embedding after visiting r. Let T^i be the BFS tree after visiting the ith vertex. Then T^1 contains the single vertex r. In the counterclockwise BFS, we first visit a neighbor s of r such that s is an outer vertex of G_γ and if s is on an outer cycle C of G_γ then s is the neighbor of r which is next to r in the counterclockwise ordering of the vertices on C. We call the edge (r, s) the *BFS-Start edge*.

We now assume that vertices $w_1(=r), w_2, \ldots, w_{j-1}$ $(j-1 < n)$ are visited by BFS and we are visiting w_j from w', that is, w' is the parent of w_j in T^j. We mark w_j as visited and mark (w', w_j) as a tree edge. If there is no edge $e = (w_j, q)$ such that $q \in V(T^{j-1})$ and $q \neq w'$, then we proceed for the next vertex w_{j+1}. Otherwise, an edge $e = (w_j, q)$ exists such that $q \in V(T^{j-1})$ and $q \neq w'$. In such a case we mark e as a non-tree edge, and change the embedding of G_γ^{j-1} to get G_γ^j, if necessary, as follows.

We set $u = w_j$ and $v = q$ if q comes earlier in the counterclockwise postorder traversal of T^j started from the BFS-Start edge; otherwise, we set $u = q$ and $v = w_j$. Let C be the cycle induced by the non-tree edge $e = (u, v)$. We check whether there is a (u, v)-split component or a u-component or a v-component in $G_\gamma^{j-1}(C)$. If there is a (u, v)-split component or a u-component or a v-component H in $G_\gamma^{j-1}(C)$ such that $r \notin V(H)$, we move all such components out of the cycle C and obtain the embedding G_γ^j. In Fig. 9(a) $I_1, \ldots, I_k$ are (u, v)-split components, $J_1, \ldots, J_l$ are u-components and $K_1, \ldots, K_m$ are v-components, those are moved out of C in Fig. 9(b).

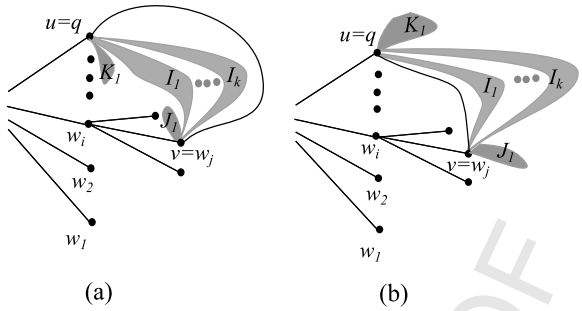

Fig. 9. Illustration for (u, v)-split components, u-components and v-components.

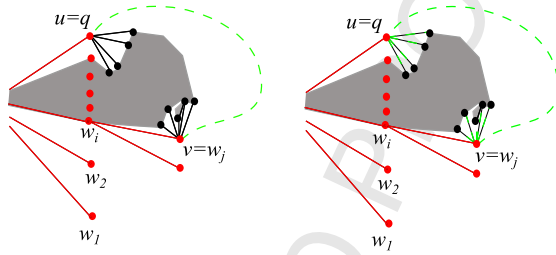

Fig. 10. Marking non-tree edges from u and v.

One can observe that unvisited vertices inside cycle C can be accessed from a visited vertex other than u and v in a later BFS step. We thus mark some edges incident to u and v as non-tree edges for maintaining the properties of a good spanning tree for T^j, as follows. Let u' and v' be the parent of u and v in T^j, respectively. Let $E_u = \{e_1, e_2, \ldots, e_k\}$ be the set of edges which are incident to the vertex u, and between the edges (u, u') and (u, v) in counterclockwise order starting from the edge (u, u') in G^j_γ, as shown in Fig. 10. We mark the edges in E_u as non-tree edges. Note that the edge-set $E_u \cup \{(u, v)\}$ will be the set Z_u with respect to the vertex u in the final good spanning tree. Let $E_v = \{e_1, e_2, \ldots, e_l\}$ be the edges which are incident to the vertex v between the edges (v, v') and (v, w) in clockwise order started from the edge (v, v') in G^j_γ as shown in Fig. 10. We also mark the edges in E_v as non-tree edges. The edge set $E_v \cup \{(v, u)\}$ will be the set X_v with respect to the vertex v in the final spanning tree.

Finally we get G^n_γ and T^n after visiting the nth vertex of G. We now show that T^n is a good spanning tree in G^n_γ.

Clearly T^n is a spanning tree in G^n_γ, since T^n consists of tree edges identified by BFS in the connected graph G^n_γ.

We now show that the embedded tree T^n is a good spanning tree in the embedded graph G^n_γ. We first show that each vertex v of T^n satisfies the condition (Cond1) in the definition of a good spanning tree. The tree edges are marked in BFS steps. For any edge $e = (u, v)$, if u lies on the path $P(r, v)$ in T^n then u is the parent of v. In this case e must be a tree edge. Similarly, v does not lie on the path $P(r, u)$ in T^n when $e = (u, v)$ is a non-tree edge. Hence, (Cond1) holds.

We next show that T^n satisfies the condition (Cond2). Consider the situation when we have dealt with the edge (u, v) while constructing G^j_γ and T^j. For the non-tree edge (u, v), the non-tree edges in Z_u are consecutive non-tree edges with respect to u in G^n_γ. Let Z be the set of vertices that contain other end vertices of the edges in Z_u edges. Clearly each vertex in Z must be inside the cycle induced by (u, v) in T^n. One can easily observe that if we traverse T^n as counterclockwise postorder traversal starting form the BFS-Start edge, the vertex u will be visited after visiting all vertices in Z. Thus each vertex in Z is contained in $T_{u_i^R}$ where u_i is a vertex that lies on the path $P(r, u)$ and $u \neq u_i$ in T^n. Similarly for the vertex v, it can be shown that the other end vertices of the edges in X_v are contained in $T_{u_i^L}$. Thus one can easily observe that all edges incident to a vertex v in T^n can be partitioned into three consecutive edge sets X_v, Y_v and Z_v. Hence, (Cond2) holds. Hence T^n is a good spanning tree T in $G_\phi = G^n_\gamma$.

We now prove the time complexity of finding G_ϕ and T. A v-component is introduced by a cut vertex. All cut vertices in a graph of n vertices and m edges can be found in $O(n+m)$ time using DFS. v-Components and $\{u, v\}$-split components can also be found in linear time [31]. We maintain a data structure to store each cut vertex or every pair of vertices with split components. We then use this record in the intermediate steps for finding G_ϕ. Let us assume we are traversing the non-tree edge (u, v) in an intermediate step j of our algorithm. We can check whether any u-components, v-components and $\{u, v\}$-split components of G exist in $G_j(C)$ by checking each edges incident to u and v. This checking takes $O(d(u) + d(v))$ time. Throughout the algorithm it needs $O(m)$ time. For moving a component outside of cycle C we need to change at most four pointers in the adjacency list of u and v, which takes $O(1)$ time. Hence the required time is $O(m)$. Since G is a planar graph, G_ϕ and its good spanning tree T can be found in $O(n)$ time. □

Please cite this article in press as: M.I. Hossain, M.S. Rahman, Good spanning trees in graph drawing, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.09.004

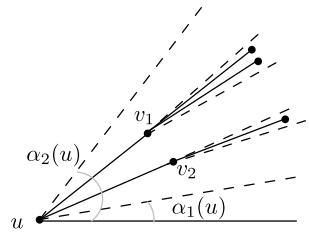

Fig. 11. A slope-disjoint drawing of a tree.

Let G be a 3-connected plane graph. Then G has neither a cut vertex nor a $\{u, v\}$-split component, and hence to find a good spanning tree from a 3-connected plane graph we do not need to change the embedding. Therefore, the following corollary holds.

Corollary 1. *Let G be a 3-connected plane graph. Then G contains a good spanning tree.*

3. Monotone grid drawings

In this section we show that every connected planar graph of n vertices has a monotone grid drawing on an $O(n) \times O(n^2)$ grid. In Section 3.1 we give some definitions and some known results on monotone drawings. We give our algorithm for finding a monotone drawing of a planar graph in Section 3.2.

3.1. Monotone drawings preliminaries

Let p be a point in the plane and l be a half-line with an end at p. The slope of l, denoted by *slope*(l), is the tangent of the angle spanned by a counterclockwise rotation that brings a horizontal half-line started at p and directed towards increasing x-coordinates to coincide with l. Let Γ be a straight-line drawing of a graph G and let (u, v) be an edge of G. We denote the direction of a half-line which starts at u and goes through v by $d(u, v)$. The direction of a drawing of an edge e is denoted by $d(e)$ and the slope of the drawing of e is denoted by *slope*(e).

Let T be a tree rooted at a vertex r. A *slope-disjoint* drawing of T satisfies the following conditions [6] (see Fig. 11):

- for every vertex $u \in T$, there exist two angles $\alpha_1(u)$ and $\alpha_2(u)$, with $0 < \alpha_1(u) < \alpha_2(u) < \pi$, such that $\alpha_1(u) <$ *slope*(e) $< \alpha_2(u)$ for every edge e that is either in T_u or that connects u with its parent;
- for every two vertices $u, v \in T$ where v is a child of u, it holds that $\alpha_1(u) < \alpha_1(v) < \alpha_2(v) < \alpha_2(u)$; and
- for every two vertices v_1, v_2 with the same parent, it holds that either $\alpha_1(v_1) < \alpha_2(v_1) < \alpha_1(v_2) < \alpha_2(v_2)$ or $\alpha_1(v_2) < \alpha_2(v_2) < \alpha_1(v_1) < \alpha_2(v_1)$.

Let G be a planar graph and Γ be a straight-line drawing of G. A path $u_1(= u), \ldots, u_k(= v)$ between vertices u and v in G is denoted by $P(u, v)$. The drawing of the path $P(u, v)$ 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 on the path. The drawing Γ is a *monotone drawing* of G if there exists a direction d for every pair of vertices u and v such that at least a path $P(u, v)$ is monotone with respect to d. A monotone drawing is a *monotone grid drawing* if every vertex is drawn on a grid point. The following lemma is known from [6].

Lemma 1. *Let T be a rooted tree of n vertices. Then T admits a monotone grid drawing on a grid of area $O(n) \times O(n^2)$, and such a drawing can be found in $O(n)$ time.*

In this paper we use a modified version of the algorithm for monotone grid drawing of a tree in [6], which we call Algorithm **Draw-Monotone-Tree** throughout this paper. Algorithm **Draw-Monotone-Tree** first assigns a slope to each vertex of a planar embedded rooted tree then obtains a slope-disjoint drawing of the tree which is monotone. The algorithm draws each edge (u, u') as a straight-line segment by using the assigned slope to u where u' is the parent of u. A brief description of the algorithm is given below. Let T be an embedded rooted tree of n vertices. (Note that in [6] T is not embedded, but here we use T as an embedded tree for the sake of our algorithm.) Let $S = \{s_1, s_2, \ldots, s_{n-1}\} = \{1/1, 2/1, 3/1, \ldots, (n-1)/1\}$ be the ordered set of $n-1$ slopes in increasing order, where each slope is represented by the ratio $y/1$. Let $v_1, v_2, \ldots, v_n$ be an ordering of the vertices in T in a counterclockwise postorder traversal. (In a counterclockwise postorder traversal of a rooted ordered tree, subtrees rooted at the children of the root are recursively traversed in counterclockwise order and then the root is visited.) Then we assign the slope s_i to vertex v_i $(i \neq n)$. Let $u_1, u_2, \ldots, u_k$ be the children of v in T. Then the subtree T_{u_i} gets $|T_{u_i}|$ consecutive elements of S from the $(1 + \sum_{j=1}^{i-1} |T_{u_j}|)$-th to the $(\sum_{j=1}^{i} |T_{u_j}|)$-th. Let v' be the parent of v. If v is not the root of T then the drawing of the edge $e = (v', v)$ will be a straight-line with slope s_i.

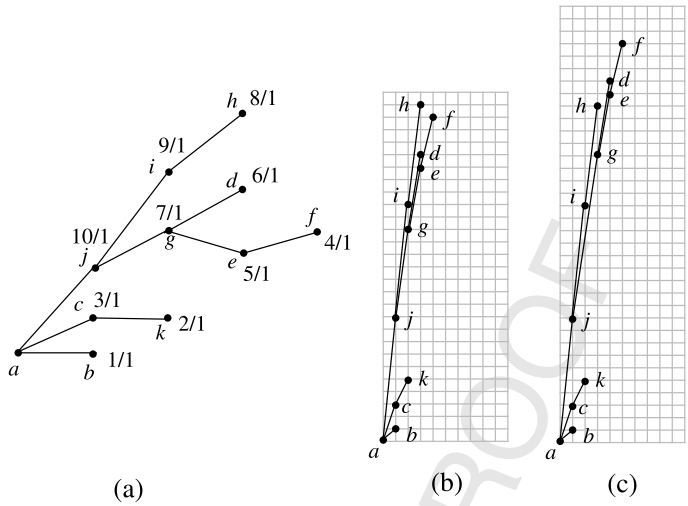

Fig. 12. (a) A tree T with assigned slope to each vertex, (b) a monotone drawing Γ of T and (c) a monotone drawing Γ' of T with elongation of the edge (j, g).

We now describe how to find a monotone grid drawing of T using the slope assigned to each vertex of T. We first draw the root vertex r at $(0, 0)$, and then use a counterclockwise preorder traversal for drawing each vertex of T. (In a counterclockwise preorder traversal of a rooted ordered tree, first the root is visited and then the subtrees rooted at the children of the root are visited recursively in counterclockwise order.) We fix the position of a vertex u when we traverse u. Note that when we traverse u, the position of the parent u' of u has already been fixed. Let $(x(u'), y(u'))$ be the position of u'. Then we place u at the grid point $(x(u')+1, y(u')+y_b)$, where $y_b/1$ is the slope of u. Fig. 12(b) illustrates a monotone grid drawing of the tree as shown in the Fig. 12(a). Algorithm **Draw-Monotone-Tree** computes a slope-disjoint monotone drawing of a tree on an $O(n) \times O(n^2)$ grid in linear time [6]. The following lemma is from [6].

Lemma 2. *Let Γ be a slope-disjoint monotone drawing of a rooted tree T. Let Γ' be a straight-line drawing of T obtained from Γ by elongation of the drawing of an edge e of T preserving the slope of e. Then Γ' is also a slope-disjoint monotone drawing. (See Fig. 12(c), where the edge (j, g) is elongated.)*

We now have the following lemma based on the properties of a good spanning tree and Lemma 2.

Lemma 3. *Let T be a good spanning tree of G_ϕ. Let v be a vertex in G, and let u be the parent of v in T. Let $X \subseteq X_v$ and $Z \subseteq Z_v$. Assume that Γ is the monotone drawing of $T \cup X \cup Z$ where a monotone path exists between every pair of vertices in the drawing of T in Γ. If a straight-line drawing Γ' of $T \cup X \cup Z$ is obtained from Γ by elongation of the drawing of the edge (u, v) preserving the slope of (u, v) and by shifting the drawing of T_v upward, then Γ' is a monotone drawing of $T \cup X \cup Z$ where a monotone path exists between every pair of vertices in the drawing of T in Γ'.*

Proof. Let r be the root of T. Let m be the slope assigned to the vertex v in T.

Let M_X and M_Z be the sets of slopes assigned to $T_{u_i^L}$ and $T_{u_i^R}$, respectively. According to the assignment of slopes, for any $m_x \in M_X$ and $m_z \in M_Z$ the relation $m_x > m > m_z$ holds.

Since each vertex in X and Z is visible from the vertex v in Γ and $m_x < m < m_z$, v must be visible from each vertex in X and Z even after elongation of edge (u, v) without changing the slope of (u, v). Note that the elongation only changes the slopes of the drawings of non-tree edges in X and Z. The drawing of T_v is shifted outwards preserving the slopes of the edges in T_v and the drawing of $T - T_v$ remains the same. Let Γ' be the new drawing of $T \cup X \cup Z$. Then obviously the edges in X and in Z do not produce any edge crossing in Γ'. By Lemma 2, the elongation of the edge (u, v) does not break the monotone property in the drawing of T in Γ'. Thus a monotone path exists between every pair of vertices in the drawing of T in Γ'. □

3.2. Monotone grid drawings of planar graphs

In this section we give our algorithm for monotone grid drawings of planar graphs. We first give an outline of our algorithm. We first construct a good spanning tree T of G and find a monotone drawing of T by Algorithm **Draw-Monotone-Tree**. We then draw each non-tree edge as a straight-line segment by shifting the drawing of some subtree of T, if necessary. Fig. 13 illustrates the steps of our algorithm. The input planar graph G is shown in Fig. 13(a). We first find a planar embedding of G containing a good spanning tree as illustrated in Fig. 13(b), where the edges of the spanning tree are drawn as

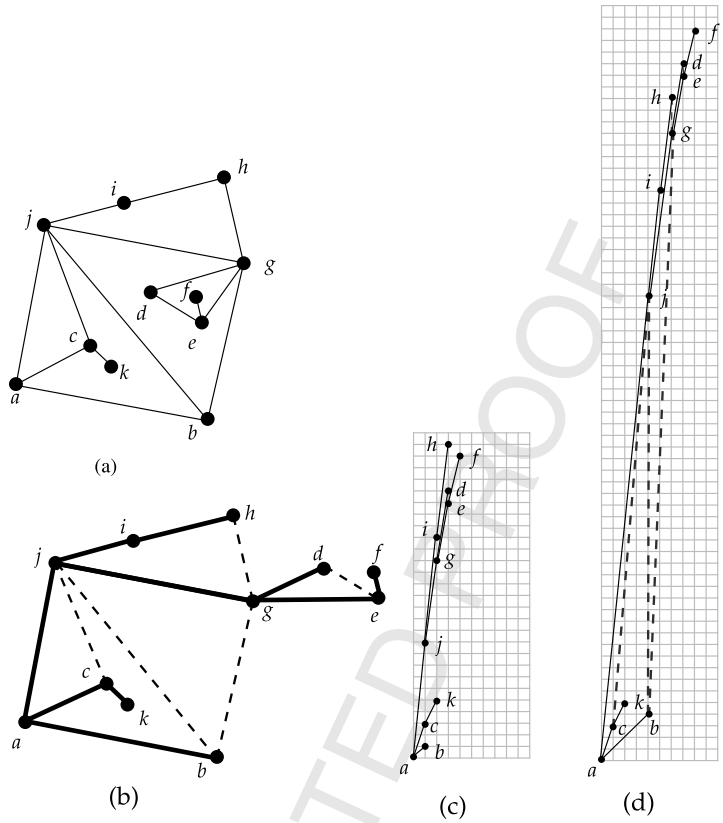

Fig. 13. Illustration for an outline of our algorithm.

solid lines. We then find a monotone drawing of T on an $O(n) \times O(n^2)$ grid using by Algorithm **Draw-Monotone-Tree** as illustrated in Fig. 13(c). Finally we elongate the drawing of some edges and draw the non-tree edges of G using straight-line segments as illustrated in Fig. 13(d).

We now have the following lemma.

Lemma 4. *Let G be a connected planar graph of n vertices and let G_ϕ be a planar embedding of G. Assume that G_ϕ has a good spanning tree T. Then G_ϕ admits a monotone grid drawing on an $O(n) \times O(n^2)$ grid.*

Proof. Let T be a good spanning tree of G_ϕ. We prove the claim by induction on the number k of non-tree edges of G for T. Algorithm **Draw-Monotone-Tree** uses a counterclockwise postorder traversal for finding a vertex ordering in the tree and assigns a slope to each vertex of the tree using that ordering. Note that the ordering of the vertices is fixed once the child of r that has to be visited first is fixed. Let T be a good spanning tree of G_ϕ and let r be the root of T. We take a child s of r as the first child to be visited in counterclockwise postorder traversal such that s is an outer vertex of G_ϕ and if s is on an outer cycle C of G_ϕ then s is the neighbor of r which is next to r in the counterclockwise ordering of the vertices on C. We call the edge (r, s) the *reference edge* of T. Such a reference edge always exists since the BFS-Start edge mentioned in the proof of Theorem 1 while constructing T can serve as the reference edge. By induction on the number k of non-tree edges of G_ϕ, we now prove the claim that G_ϕ admits a monotone grid drawing on an $O(n) \times O(n^2)$ grid and a monotone path exists between every pair of vertices of G_ϕ through the edges of T in the drawing.

We first assume that $k = 0$. In this case $G_\phi = T$. We then find a monotone drawing Γ of T on an $O(n) \times O(n^2)$ grid using Algorithm **Draw-Monotone-Tree** taking the reference edge (r, s) as the starting edge in the traversal (counterclockwise postorder traversal for slope assignment and counterclockwise preorder traversal for drawing vertices). Since $G_\phi = T$, Γ is a monotone drawing of G_ϕ. That is, a monotone path exists between every pair of vertices of T in Γ.

We thus assume that $k > 0$ and the claim holds for any plane graph G_ϕ with less than k non-tree edges.

Let G_ϕ have k non-tree edges for T and let $e = (u, v)$ be a non-tree edge of the outer boundary of G_ϕ.

Let m_u and m_v be the slopes assigned to the vertices u and v, respectively in T by Algorithm **Draw-Monotone-Tree**. Without loss of generality let us assume $m_u > m_v$. Let w be the lowest common ancestor of u and v in T and let u' and v' be the parents of u an v in T, respectively. According to (Cond1) u does not lie on the path $P(v, r)$ and v does not lie on the path $P(u, r)$. Let $C = \{P(u, w) \cup P(v, w) \cup (u, v)\}$ and $G'_\phi = P(r, w) \cup G_\phi(C)$. Clearly the number of non-tree edges in $G_\phi - (u, v)$ is less than k. Hence by the induction hypothesis, $G_\phi - (u, v)$ has a straight-line monotone drawing on an

$O(n) \times O(n^2)$ grid where the edges in T are drawn with the slope assigned to them and a monotone path exists between every pair of vertices through the edges in T. Let Γ' be the drawing of $G'_\phi - (u, v)$ in Γ. Let $x_l(\Gamma')$ be the largest x-coordinate used for the drawing of Γ'. We now shift the drawing of T_u and T_v such that u and v lie on the line $x = x_l(\Gamma') + 1$ by preserving the slopes of the drawings of the edges (u', u) and (v', v). Since the slopes are integer numbers, it guarantees that all vertices remain on grid points after the shifting operation. According to Lemma 3 elongations of (u, u') and (v, v') do not produce any edge crossing in the drawing. According to (Cond2), e belongs to the set Z_u and the set X_v. Then no tree edge incident to u exists between the edge (u, u') and (u, v) in counterclockwise from the edge (u, u'), and no tree edge incident to v exists between the edge (v, v') and (v, u) in clockwise from the edge (v, v'). (Remember that we have used counterclockwise postorder traversal starting from a reference edge for ordering the vertices in algorithm **Draw-Monotone-Tree**.) Hence we can draw the edge e on the line $x = x_l(\Gamma') + 1$ as a straight-line segment without any edge crossings. We observe that at least one vertex lie on every x-coordinate which gives width of the drawing $O(n)$. Since slope of a tree edge can be at most n, height of the drawing $O(n^2)$. Hence in the worst case the drawing takes a grid of size $O(n) \times O(n^2)$. □

The following theorem is our main result on monotone drawings of planar graphs.

Theorem 2. *Every connected planar graph of n vertices admits a monotone grid drawing on a grid of area $O(n) \times O(n^2)$, and such a drawing can be found in $O(n)$ time.*

Proof. Let G be a connected planar graph. By Theorem 1 G has a planar embedding G_ϕ such that G_ϕ contains a good spanning tree T, and G_ϕ and T can be found in $O(n)$ time.

After constructing G_ϕ and T we can construct a monotone drawing of G_ϕ using a recursive algorithm based on the inductive proof of Lemma 4. We now describe a technique to implement the recursive algorithm in $O(n)$ time. In the inductive step the challenging tasks are to find the value of $x_l(\Gamma')$ and shift the drawing of T_u and T_v upward. Since we have assigned a slope to each vertex, fixing only the x-coordinates of the vertices is sufficient to get the drawing.

Let (u, v) be a non-tree edge in G_ϕ with respect to T. Let w be the lowest common ancestor of u and v in T. Let $C = \{P(u, w) \cup P(v, w) \cup (u, v)\}$. We denote the number of vertices in G by $n(G)$. One can easily observe that $x(w) + n(G_\phi(C)) \geq x_l(\Gamma')$, where $x(w)$ is the x-coordinate of w. Hence $x(w) + n(G_\phi(C))$ can be taken safely as $x_l(\Gamma')$ for the drawing Γ' of $G'_\phi - (u, v)$ in the proof of Lemma 4.

It is known that the common ancestors for all pairs of vertices in T can be found in linear time [32]. The value of $n(G_\phi(C))$ for a non-tree edge (u, v) can be found in constant time using counterclockwise DFS labeling $1, 2, \ldots n$ of the vertices as follows. Let $L(u)$ and $D(u)$ be the level of u in the rooted tree T and the DFS number of u in T, respectively. Then one can observe that $n(G_\phi(C)) = L(u) - L(w) + 1 + D(v) - D(u) - n(T_u) + 1$. The vertices of graphs in Fig. 14 are labeled by counterclockwise DFS and the DFS labels are written inside the small circles corresponding to the vertices. For the edge (7, 16) in the graph in Fig. 14(a), $w = 4$ and $n(G_\phi(C)) = 5 - 3 + 1 + 16 - 7 - 3 + 1 = 10$.

To keep the running time $O(n)$ for shifting in total, instead of calculating of exact x-coordinates we only calculate the amount to be shifted for each vertex in the recursive steps which we call *offset* of that vertex for the non-tree edge considered. In the final step we calculate the exact x-coordinates of the vertices from this offsets. The calculation of offsets and x-coordinates are given bellow. We initialize the offset of each vertex by 1 (except the root), as illustrate in Fig. 14(a) where offsets are written beside the vertices. In each recursive step we update the offsets of the two end vertices of the non-tree edge (u, v) by the value of $n(G_\phi(C))$, if previous offset is less than $n(G_\phi(C))$, as shown in Fig. 14(b)-(d).

Finally we calculate the x-coordinates using a BFS starting from the root of T. If no non-tree edge is incident to a vertex u then the x-coordinate of u is the x-coordinate of its parent plus the offset of u, otherwise a non-tree edge (u, v) is incident to u, and we calculate x-coordinate of u as the x-coordinate of w plus the offset of u as illustrated in Fig. 14(e) where x-coordinates are written beside the vertices. □

4. 2-Visibility representations

In this section we give a simple linear-time algorithm to find a straight-line 2-*VR* of a planar graph on a grid of size $(2n - 1) \times (2n - 1)$. In our drawing model, we draw each corner of a rectangle corresponding to a vertex on an integer grid point and each edge on a grid line.

Let G be a connected planar graph of n vertices and let T be a good spanning tree in a planar embedding G_ϕ of G. We first construct a 2-visibility representation of T on a $(2n - 1) \times (2n - 1)$ grid and then obtain a 2-visibility representation of G by modifying the 2-visibility representation of T and adding the drawings of all non-tree edges.

Let T be a good spanning tree of a connected plane graph G and let v be a vertex of T. We denote by $C(v)$ the number of vertices in the subtree of T rooted at v. Let Γ_T be a 2-VR of T. We denote the rectangle in Γ_T corresponding to a vertex v of T by R_v. Let h_v be the height of R_v and w_v be the width of R_v in Γ_T. We represent the position of a rectangle in the drawing by the coordinates of its left-bottom corner.

We now give our algorithm for constructing a 2-VR Γ_T of T. Let r be the root of T. We first place R_r at $(0, 0)$ point on the grid such that $w_r = 2n - 1$ and $h_r = 1$. In our drawing R_v always has $w_v = 2C(v) - 1$ and $h_v = 1$. Let v be any vertex

Please cite this article in press as: M.I. Hossain, M.S. Rahman, Good spanning trees in graph drawing, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.09.004

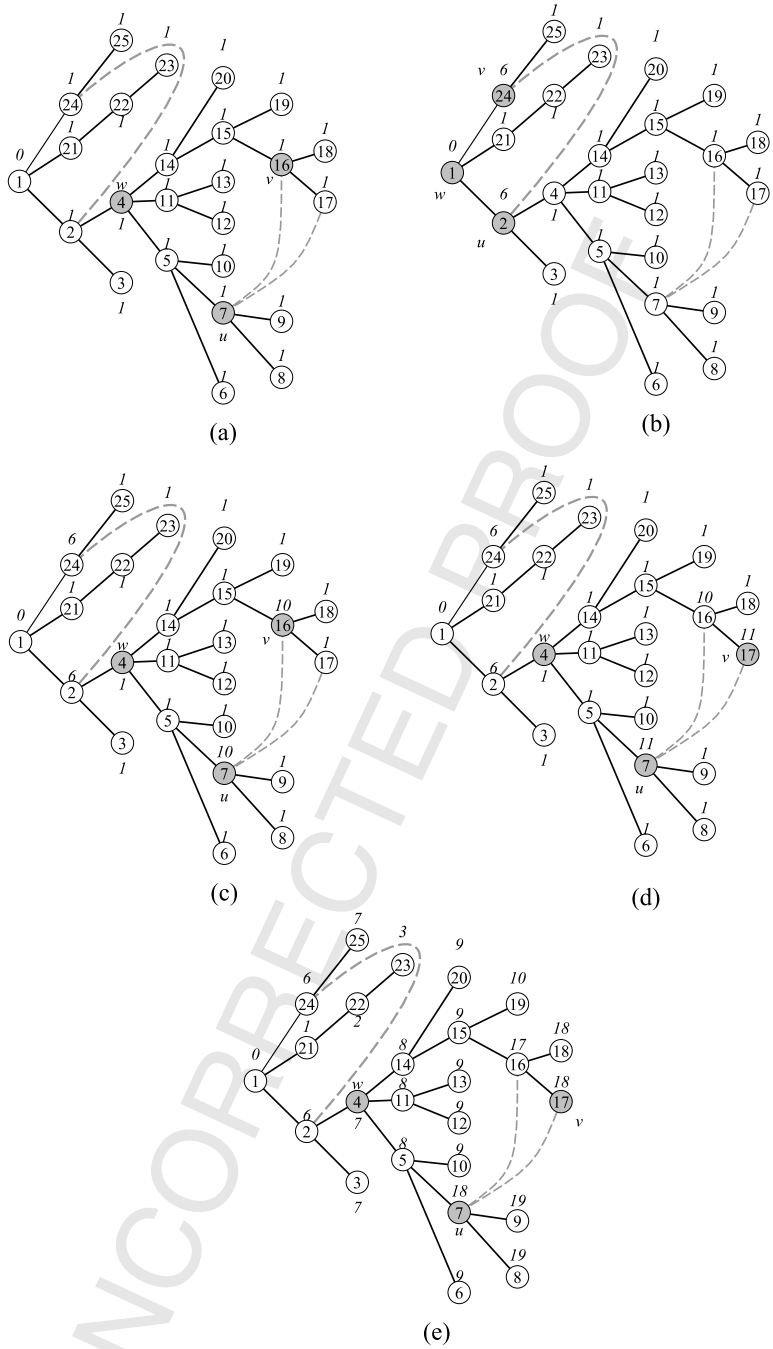

Fig. 14. (a) Initial offset for each vertex, (b)–(d) updating offset value as $n(G(C))$ for non-tree edges and (e) calculating the x-coordinate for each vertex by a BFS.

of T and let $u_1, u_2, \ldots, u_k$ be the children of v. Assume that we have drawn the box corresponding to v on the grid at point (x, y). We now draw R_{u_i} $(1 \leq i \leq k)$ on the grid by fixing the position of R_{u_i} on $(x', y+2)$ where $x' = x + 2\sum_{j=1}^{j=i-1} C(u_j)$. We draw the edge (v, u_i) by a vertical line segment between $(x', y+2)$ and $(x', y+1)$ so that it connects R_v and R_{u_i}. (See Fig. 15(d).) We call the algorithm above Algorithm **Draw-2-visibility-Tree**.

We now have the following lemma.

Lemma 5. *Let T be a good spanning tree of a connected plane graph of n vertices. Algorithm* **Draw-2-visibility-Tree** *gives a 2-visibility representation Γ_T of T on a $(2n-1) \times 2H$ grid, where H is the depth of T. Furthermore, the algorithm computes Γ_T in linear time.*

Please cite this article in press as: M.I. Hossain, M.S. Rahman, Good spanning trees in graph drawing, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.09.004

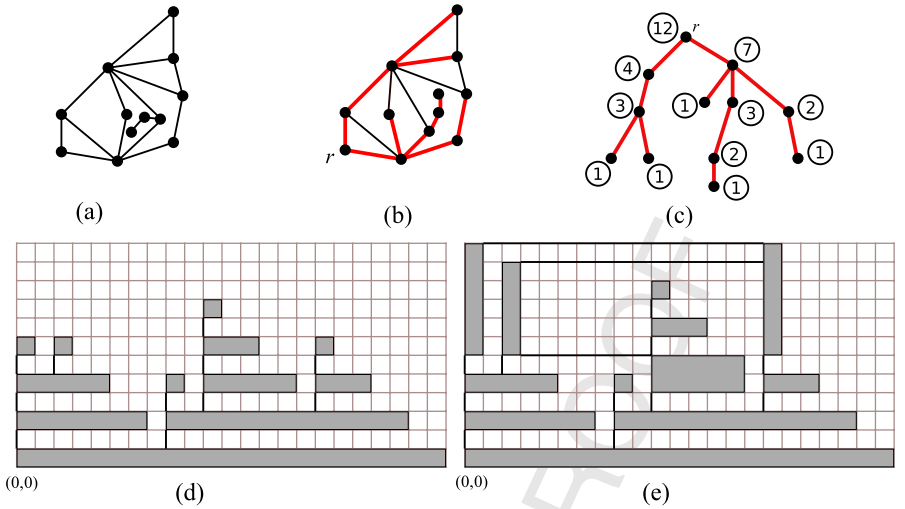

Fig. 15. (a) A planar graph G, (b) a planar embedding G_ϕ of G, red (thick) edges represent the tree edges of the good spanning tree T of G_ϕ, (c) vertex count of each subtree rooted at each vextex of T, (d) a 2-VR of T and e) a 2-VR of G_ϕ. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Proof. We first need to show that the rectangles in Γ_T do not overlap each other. Since $x + 2\sum_{j=1}^{j=i-1} C(u_j)$ is smaller than $x + 2\sum_{j=1}^{j=i} C(u_j)$, R_{u_i} does not overlap with $R_{u_{i+1}}$ and R_{u_i} does not overlap with R_v because h_v is 1. We now prove the grid size of Γ_T. The rectangle R_r corresponding to the root has the largest width which is $2C(r) - 1 = 2n - 1$. Since $h_v = 1$ for any vertex v of T and y-coordinate is increased by 2 for each unit depth, the height of the drawing is $2H$. Note that H is at most $n - 1$.

The algorithm traverses T and updates some trivial data structures. Hence, it takes linear time. □

We now have the following observation.

Observation 1. *Let T be a good spanning tree of a connected plane graph of n vertices. Let Γ_T be a 2-visibility representation of T produced by Algorithm* **Draw-2-visibility-Tree**. *Let v be a vertex of T. Let Γ'_T be a drawing obtained by increasing the height of R_v and shifting the drawing of T_v in Γ_T upward to any extent. Then Γ'_T is also a 2-visibility representation of T.*

We next obtain a 2-visibility representation of G_ϕ by drawing each non-tree edge as a horizontal line segment in the 2-visibility representation Γ_T of T. Let (u, v) be a non-tree edge in G_ϕ with respect to T. Let (x_u, y_u) and (x_v, y_v) be the coordinates of R_u and R_v, respectively. We have the following theorem.

Theorem 3. *Let G be a connected planar graph of n vertices and let G_ϕ be a planar embedding of G such that G_ϕ has a good spanning tree T. Then G_ϕ admits a 2-visibility representation on a $(2n - 1) \times (2n - 1)$ grid.*

Proof. Let T be a good spanning tree of G_ϕ. We prove the claim by induction on the number k of non-tree edges of G for T. Let r be the root of T. We first assume that $k = 0$. In this case $G_\phi = T$. We then find a 2-visibility representation 2-$VR(T)$ of T on a $(2n - 1) \times (2n - 1)$ grid using Algorithm **Draw-2-visibility-Tree**. Since $G_\phi = T$, 2-$VR(T)$ is a 2-visibility representation of G_ϕ.

We thus assume that $k > 0$ and the claim holds for any plane graph G_ϕ with less than k non-tree edges.

Let G_ϕ have k non-tree edges for T and let $e = (u, v)$ be a non-tree edge on the outer boundary of G_ϕ.

Let R_u and R_v be the rectangles of the vertices u and v, respectively in a 2-visibility drawing of T by Algorithm **Draw-2-visibility-Tree**. Let w be the lowest common ancestor of u and v in T and let u' and v' be the parents of u and v in T, respectively. According to (Cond1) u does not lie on the path $P(v, r)$ and v does not lie on the path $P(u, r)$. Let $C = \{P(u, w) \cup P(v, w) \cup (u, v)\}$ and $G'_\phi = P(r, w) \cup G_\phi(C)$. Clearly the number of non-tree edges in $G_\phi - (u, v)$ is less than k. Hence by induction hypothesis, $G_\phi - (u, v)$ has a 2-visibility representation on a $(2n - 1) \times (2n - 1)$ grid. Let $VR(G)$ be the drawing of $G'_\phi - (u, v)$ in $VR(G)$. Let $y_l(\Gamma')$ be the largest y-coordinate used for the drawing of 2-$VR(G'_\phi - (u, v))$. We now increase height of R_u and R_v up to $y_l(\Gamma') + 2$. We then add a horizontal straight line on $y = y_l(\Gamma') + 2$ to connect R_u and R_v.

According to Observation 1 increasing the height of R_u and R_v does not produce any edge crossing or rectangle overlapping in the drawing. Since e is an outer edge, increasing the height of R_u and R_v does not create problem for other non-tree edges that have already been drawn. (See Fig. 15(e).) According to (Cond2), e belongs to set Z_u and set X_v. Then

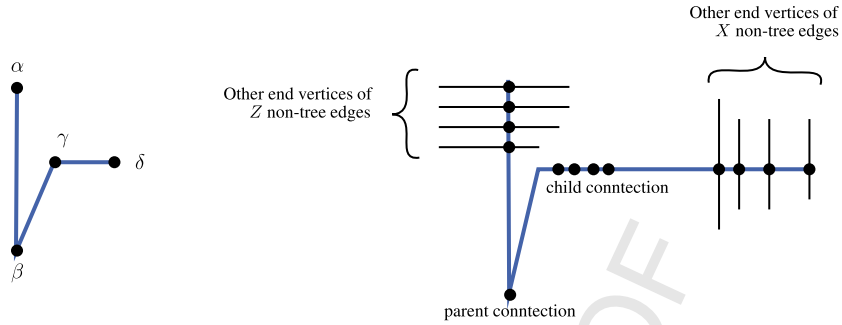

Fig. 16. Illustration for a spike shaped curve.

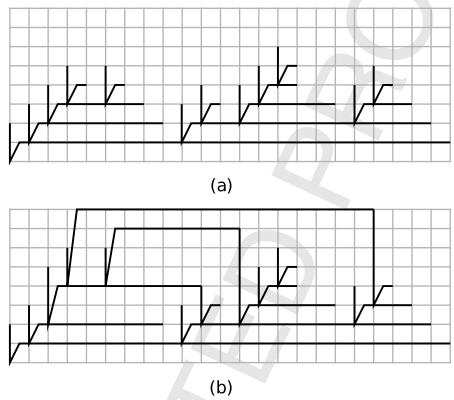

Fig. 17. (a) A spike-*VPG* representation of T in Fig. 15(b) and (b) a spike-*VPG* representation of G in Fig. 15(a).

no tree edge incident to u exists between the edge (u, u') and (u, v) that is counterclockwise from the edge (u, u'), and no tree edge incident to v exists between the edge (v, v') and (v, u) that is clockwise from the edge (v, v'). Hence we can draw the edge e on the line $y = y_l(\Gamma') + 2$ by a straight-line segment without any edge crossing. This operation does not change the width of the drawing, and the height can increase exactly one unit. Thus the 2-*VR*(G) takes a grid of size $(2n-1) \times (2n-1)$. □

5. Spike-*VPG* representations of planar graphs

In this section we give an algorithm to find a spike-*VPG* representation of a planar graph G on a grid of size $(2n-1) \times n$.

Let G be a connected planar graph of n vertices and let T be a good spanning tree in a planar embedding G_ϕ of G. Let r be the root of T. We first construct a spike-*VPG* representation of T on a $n \times n$ grid and then obtain a spike-*VPG* representation of G by modifying the spike-*VPG* representation of T.

In our drawing model we represent a path as a spike shaped curve, which is defined as follows. Let $\mathcal{C}(\alpha, \beta, \gamma, \delta)$ be a curve where $\mathcal{C}$ starts from α and ends at δ, and β, γ are two adjusting points. A typical $\mathcal{C}$ is found by placing α, β, γ, and δ on $(x, y+2)$, (x, y), $(x+0.5, y+1)$, $(x+1, y+1)$, respectively where (x, y) is any integer coordinate in the grid. Fig. 16(a) illustrates an example of $\mathcal{C}$. In our drawing model α, β, δ lie on integer points and γ lies on half integer point of the gird. Let $\mathcal{C}_v$ denotes a curve corresponding to a vertex v. We denote $\alpha, \beta, \gamma, \delta$ points of $\mathcal{C}_v$ by $\mathcal{C}_v(\alpha)$, $\mathcal{C}_v(\beta)$, $\mathcal{C}_v(\gamma)$, and $\mathcal{C}_v(\delta)$, respectively. Change of line segment $l(\mathcal{C}_v(\beta), \mathcal{C}_v(\gamma))$ is used for adjusting the height of $\mathcal{C}_v$. The line segment $l(\mathcal{C}_v(\alpha), \mathcal{C}_v(\beta))$ is used for receiving crossing from the vertices those are other end vertex of each of the edge in Z_v. The line segment $l(\mathcal{C}_v(\gamma), \mathcal{C}_v(\delta))$ is used for maintaining the connection of the children of v in T, and the connection for the vertices those are other end vertex of each of the edge in X_v (see Fig. 16(b)). We denote a spike-*VPG* representation of G by spike-*VPG*(G).

We first show that how we obtain a spike-*VPG* representation of a tree as follows. In the spike-*VPG* representation of tree $l(\mathcal{C}_v(\alpha), \mathcal{C}_v(\beta))$ line segment of each $\mathcal{C}_v$ remains unused.

We first place $\mathcal{C}_r$ on the grid so that $\mathcal{C}_r(\beta)$ lies on $(0, 2)$, $(0, 0)$, $(0.5, 1)$, $(1, 2n-1)$ points in the grid. Let v be any vertex of T and let $u_1, u_2, \ldots, u_k$ are the children of v. Let (x, y) be the coordinate of $\mathcal{C}_v(\beta)$. We now show the placement of each $\mathcal{C}_{u_i}$ $(1 \le i \le k)$. We place $\mathcal{C}_{u_i}$ such that $\mathcal{C}_{u_i}(\alpha)$, $\mathcal{C}_{u_i}(\beta)$, $\mathcal{C}_{u_i}(\gamma)$ and $\mathcal{C}_{u_i}(\delta)$ lie on $(x'+1, y+1+2)$, $(x'+1, y+1)$, $(x'+1+0.5, y+1+1)$, $(x'+1+2C(u_i), y+1+1)$, respectively where $(x' = x + 2\sum_{j=1}^{j=i-1} C(u_j))$.

Note that the curves $\mathcal{C}_v$ and $\mathcal{C}_{u_i}$ cross at $\mathcal{C}_{u_i}(\beta)$ which represents the edge (v, u_i). Fig. 17(a) illustrates a spike-*VPG* representation of the good spanning tree T in Fig. 15(b). We call the algorithm described above Algorithm **Draw-spike-VPG-Tree**.

Please cite this article in press as: M.I. Hossain, M.S. Rahman, Good spanning trees in graph drawing, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.09.004

Lemma 6. *Let T be a good spanning tree of a connected plane graph of n vertices. Algorithm* **Draw-spike-VPG-Tree** *gives a spike-VPG representation of T on a $(2n-1)\times H$ grid, where H is the depth of T. Furthermore the algorithm computes a spike-VPG representation of T in linear time.*

Proof. We first need to show that the curve $\mathcal{C}_{u_i}$ only crosses $\mathcal{C}_v$. Clearly, $\mathcal{C}_{u_i}(\beta)$ point lies on $l(\mathcal{C}_v(\gamma),\mathcal{C}_v(\delta))$. Let us analyze the positions of $\mathcal{C}_{u_i}$ and $\mathcal{C}_{u_{i+1}}$. If (x,y) is the coordinate of $\mathcal{C}_v(\beta)$ then $\mathcal{C}_{u_i}(\beta)$ and $\mathcal{C}_{u_{i+1}}(\beta)$ lie on $(x+2\sum_{j=1}^{j=i} C(u_j), y+1)$ and $(x+2\sum_{j=1}^{j=i+1} C(u_j), y+1)$. This means length of $l(\mathcal{C}_{u_i}(\gamma),\mathcal{C}_{u_i}(\delta))$ is $2C(u_i)$. Note that, in the drawing the y-coordinate is increased by 1 for each unit depth, this implies that the height of the drawing can be at most H. The algorithm traverses T and updates some trivial data structure. Hence, it takes linear time. □

We now have the following observation.

Observation 2. *Let T be a good spanning tree of a connected plane graph of n vertices. Let Γ_T be a spike-VPG representation of T produced by Algorithm* **Draw-2-visibility-Tree**. *Let v be a vertex of T. Let $\mathcal{C}_v((x,y+2),(x,y),(x+0.5,y+1),(x+k_1,y+1))$ be the curve that represents v in Γ_T. Let Γ'_T be a drawing obtained by changing of $l(\mathcal{C}_v(\gamma),\mathcal{C}_v(\delta))$ such that $\mathcal{C}_v(\gamma)$ and $\mathcal{C}_v(\delta)$ lie on $(x+0.5,y+1+k_2)$ and $(x+k_1,y+1+k_2)$, and shifting the drawing of T_v in Γ_T upward to any extent. Then Γ'_T is also a spike-VPG representation of T.*

We next obtain a spike-*VPG* drawing of G_ϕ by modifying the spike-*VPG* drawing of T. We have the following theorem.

Theorem 4. *Let G be a connected planar graph of n vertices and let G_ϕ be a planar embedding of G such that G_ϕ has a good spanning tree T. Then G_ϕ admits a spike-VPG representation on a $(2n-1)\times n$ grid.*

Proof. Let T be a good spanning tree of G_ϕ. We prove the claim by induction on the number k of non-tree edges of G for T. Let r be the root of T. We first assume that $z=0$. In this case $G_\phi = T$. We then find spike-*VPG*(T) of T on a $(2n-1)\times n$ grid using Algorithm **Draw-spike-VPG-Tree**. Since $G_\phi = T$, spike-*VPG*(T) is a visibility representation of G_ϕ.

We thus assume that $k>0$ and the claim holds for any plane graph G_ϕ with less than k non-tree edges.

Let G_ϕ have k non-tree edges for T and let $e=(u,v)$ be a non-tree edge of the outer boundary of G_ϕ.

Let $\mathcal{C}_u$ and $\mathcal{C}_v$ be the curves in a drawing obtained by Algorithm **Draw-spike-VPG-Tree** corresponding to the vertices u and v in T, respectively in T by Algorithm **Draw-spike-VPG-Tree**. Let w be the lowest common ancestor of u and v in T and let u' and v' be the parents of u and v in T, respectively. According to (Cond1) u does not lie on the path $P(v,r)$ and v does not lie on the path $P(u,r)$. Let $C=\{P(u,w)\cup P(v,w)\cup(u,v)\}$ and $G'_\phi = P(r,w)\cup G_\phi(C)$. Clearly the number of non-tree edges in $G_\phi-(u,v)$ is less than k. Hence by induction hypothesis, $G_\phi-(u,v)$ has a spike-*VPG* representation on a $(2n-1)\times n$ grid. Let *VR*(G) be the drawing of $G'_\phi-(u,v)$ in spike-*VPG*(G). Let $y_l(\Gamma')$ be the largest y-coordinate used for the drawing of spike-*VPG*($G'_\phi-(u,v)$). To represent the edge u,v we shall change $\mathcal{C}_u$ and $\mathcal{C}_v$ so that they cross each other. To do it we follow the following four steps.

- Modify $\mathcal{C}_u$: We shift $l(\mathcal{C}_u(\gamma),\mathcal{C}_u(\delta))$ on line $y=y_l(\Gamma')+1$ by keeping x-coordinate of $\mathcal{C}_u(\gamma)$ unchanged. Then shift the drawing of T_u upward.
- Modify $\mathcal{C}_v$: We enlarge $l(\mathcal{C}_v(\alpha),\mathcal{C}_v(\beta))$ so that $\mathcal{C}_v(\alpha)$ lies on $y=y_l(\Gamma')+1$.
- Extend $l(\mathcal{C}_u(\gamma),\mathcal{C}_u(\delta))$ so that it crosses $l(\mathcal{C}_v(\alpha),\mathcal{C}_v(\beta))$ that represent e. Note that $l(\mathcal{C}_u(\gamma),\mathcal{C}_u(\delta))$ is a horizontal line and $l(\mathcal{C}_v(\alpha),\mathcal{C}_v(\beta))$ is a vertical line.
- Let S be the set of vertices that are neighbors of u in G'_ϕ. For any vertex $s\in U$, the edge $e'(u,s)$ there was a crossing between $\mathcal{C}_u$ and $\mathcal{C}_s$ in induction step. We need to enlarge $l(\mathcal{C}_s(\alpha),\mathcal{C}_s(\beta))$ so that $\mathcal{C}_s(\alpha)$ lies on $y=y_l(\Gamma')+1$.

According to Observation 2, placing $l(\mathcal{C}_u(\gamma),\mathcal{C}_u(\delta))$ on line $y=y_l(\Gamma')+1$ does not produce any edge crossing in the drawing. It is trivial to see that elongation of $l(\mathcal{C}_v(\alpha),\mathcal{C}_v(\beta))$ do not produce any unwanted crossing. According to (Cond2), e belongs to set Z_u and set X_v. Then no tree edge incident to u exists between the edge (u,u') and (u,v) in counterclockwise from the edge (u,u'), and no tree edge incident to v exists between the edge (v,v') and (v,u) in clockwise from the edge (v,v'). Fig. 17(b) illustrates a spike-*VPG* representation of the planar graph G in Fig. 15(a). Hence we can draw the edge e on the line $y=y_l(\Gamma')+1$ as a straight-line segment without any edge crossings. This operation does not change width of the drawing and height can be increased exactly one unit. Thus spike-*VPG*(G) takes a grid of size $(2n-1)\times n$. □

Asinowski et al. [28] showed a technique where a T-shaped representation is converted into a B_3-*VPG* representation. A T-shaped representation have applications in floorplanning in VLSI chip design [33] where a vertex corresponds to a 2-bend module in a floorplanning. We show that our spike-*VPG* can be transformed into T-shaped (sometimes into L-shaped) representation as follows. Let G be a planar graph of n vertices. Let Γ be a spike-*VPG* representation of G. Let $\mathcal{C}_v$ be the curve corresponding to v in G. We analyze the α,β,γ and δ points of $\mathcal{C}_v$. If the point β lies on (x,y) then α,γ and δ lie on $(x,y+k_1)$, $(x+0.5,y+k_2)$ and $(x+0.5,y+k_2)$, respectively where k_1 and k_2 are integer. One can easily observe

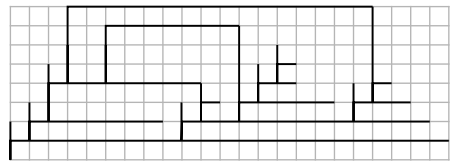

Fig. 18. T-shaped and L-shaped objects representation of G.

that there is an empty space in $\triangle(\beta, \gamma, (x, y + k_2))$ for each $\mathcal{C}_v$. Hence we move the point γ to $(x, y + k_2)$ for each $\mathcal{C}_v$. The operation turns $\mathcal{C}_v$ into a T-shaped object. If $k_1 < k_2$ then $\mathcal{C}_v$ turns into an L-shaped object. Hence each object lies on grid lines fully as illustrated in Fig. 18. Thus we have proved the following theorem.

Theorem 5. *Every planar graph has a representation using T-shaped and L-shaped objects.*

6. Conclusions

In this paper we have defined a special spanning tree of a plane graph and showed that it has many applications in planar graph drawings. We have solved two open problems on monotone drawings by showing that every connected planar graph of n vertices has a planar embedding that admits a straight-line monotone grid drawing on a grid of size $O(n) \times O(n^2)$. We have also showed that good spanning trees have applications in 2-visibility representations and in *VPG* representations of planar graphs.

Starting from an outer vertex of an arbitrary planar embedding of a connected planar graph G, we can find a planar embedding of G which has a good spanning tree. However, we have no algorithm to determine whether a given plane graph has a good spanning tree or not. We thus have the following open problems on good spanning trees.

- **Problem 1** Characterize plane graphs that contain good spanning trees and find an efficient algorithm to determine whether a given plane graph has a good spanning tree or not.
- **Problem 2** Compute a good spanning tree of a planar graph so that the depth of the tree becomes minimum. Note that such a good spanning tree may find applications for reducing grid sizes of some drawings of planar graphs.

Acknowledgements

We thank anonymous reviewers for their valuable suggestions for improving the presentation of the paper.

References

[1] H. de Fraysseix, J. Pach, R. Pollack, Small sets supporting fary embeddings of planar graphs, in: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing '88, STOC '88, ACM, New York, NY, USA, 1988, pp. 426–433.
[2] G. Kant, X. He, Regular edge labeling of 4-connected plane graphs and its applications in graph drawing problems, Theoret. Comput. Sci. 172 (1–2) (1997) 175–193.
[3] W. Schnyder, Embedding planar graphs on the grid, in: Proceedings of the First Annual ACM–SIAM Symposium on Discrete Algorithms '90, SODA '90, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1990, pp. 138–148.
[4] K. Miura, M. Azuma, T. Nishizeki, Canonical decomposition, realizer, Schnyder labeling and orderly spanning trees of plane graphs, Internat. J. Found. Comput. Sci. (2005) 117–141.
[5] Y.-T. Chiang, C.-C. Lin, H.-I. Lu, Orderly spanning trees with applications, SIAM J. Comput. 34 (4) (2005) 924–945, http://dx.doi.org/10.1137/S0097539702411381.
[6] P. Angelini, E. Colasante, G. Di Battista, F. Frati, M. Patrignani, Monotone drawings of graphs, J. Graph Algorithms Appl. 16 (1) (2012) 5–35.
[7] M.I. Hossain, M.S. Rahman, Straight-line monotone grid drawings of series-parallel graphs, in: Proceedings of Annual International Computing and Combinatorics Conference '13, in: Lecture Notes in Computer Science, vol. 7936, Springer, Berlin/Heidelberg, 2013, pp. 672–679.
[8] P. Angelini, W. Didimo, S. Kobourov, T. Mchedlidze, V. Roselli, A. Symvonis, S. Wismath, Monotone drawings of graphs with fixed embedding, Algorithmica (2013) 1–25, http://dx.doi.org/10.1007/s00453-013-9790-3.
[9] W. Huang, P. Eades, S.-H. Hong, A graph reading behavior: geodesic-path tendency, in: Proceedings of the 2009 IEEE Pacific Visualization Symposium, PACIFICVIS '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 137–144.
[10] G. Di Battista, P. Eades, R. Tamassia, I.G. Tollis, Graph Drawing: Algorithms for the Visualization of Graphs, Prentice-Hall, 1999.
[11] G. Di Battista, R. Tamassia, Algorithms for plane representations of acyclic digraphs, Theoret. Comput. Sci. 61 (2–3) (1988) 175–198.
[12] T. Nishizeki, M.S. Rahman, Planar Graph Drawing, Lecture Notes Series on Computing, vol. 12, World Scientific, Singapore, 2004.
[13] M.H. Samee, M. Rahman, Upward planar drawings of series-parallel digraphs with maximum degree three, in: Proceedings of International Workshop on Algorithms and Computation '07, 2007, pp. 28–45.
[14] E.M. Arkin, R. Connelly, J.S.B. Mitchell, On monotone paths among obstacles with applications to planning assemblies, in: Proceedings of the Fifth Annual Symposium on Computational Geometry, SCG '89, ACM, New York, NY, USA, 1989, pp. 334–343.
[15] P. Rosenstiehl, R. Tarjan, Rectilinear planar layouts and bipolar orientations of planar graphs, Discrete Comput. Geom. 1 (1) (1986) 343–353, http://dx.doi.org/10.1007/BF02187706.
[16] R. Tamassia, I.G. Tollis, A unified approach to visibility representations of planar graphs, Discrete Comput. Geom. 1 (1986) 321–341.
[17] H. Zhang, X. He, Improved visibility representation of plane graphs, Comput. Geom. 30 (1) (2005) 29–39, http://dx.doi.org/10.1016/j.comgeo.2004.08.002.

Please cite this article in press as: M.I. Hossain, M.S. Rahman, Good spanning trees in graph drawing, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.09.004

[18] H. Zhang, X. He, Compact visibility representation and straight-line grid embedding of plane graphs, in: Proceedings of Workshop on Algorithms and Data Structures, 2003, pp. 493–504.
[19] H. Zhang, X. He, Visibility representation of plane graphs via canonical ordering tree, Inform. Process. Lett. 96 (2) (2005) 41–48, http://dx.doi.org/10.1016/j.ipl.2005.05.024.
[20] H. Zhang, X. He, An application of well-orderly trees in graph drawing, Internat. J. Found. Comput. Sci. (2006) 1129–1142.
[21] M.S. Rahman, S.-I. Nakano, T. Nishizeki, Box-rectangular drawings of plane graphs, J. Algorithms (2000) 363–398.
[22] M.M. Hasan, M.S. Rahman, M.R. Karim, Box-rectangular drawings of planar graphs, J. Graph Algorithms Appl. 17 (6) (2013) 629–646, http://dx.doi.org/10.7155/jgaa.00309.
[23] X. He, A simple linear time algorithm for proper box rectangular drawings of plane graphs, J. Algorithms 40 (1) (2001) 82–101, http://dx.doi.org/10.1006/jagm.2001.1161.
[24] M. Hasan, M.S. Rahman, T. Nishizeki, A linear algorithm for compact box-drawings of trees, Networks 42 (3) (2003) 160–164.
[25] U. Fößmeier, M. Kaufmann, Drawing high degree graphs with low bend numbers, in: F. Brandenburg (Ed.), Proceedings of Graph Drawing '95, in: Lecture Notes in Computer Science, vol. 1027, Springer, Berlin, Heidelberg, 1996, pp. 254–266.
[26] U. Fößmeier, G. Kant, M. Kaufmann, 2-Visibility drawings of planar graphs, in: S. North (Ed.), Proceedings of Graph Drawing '96, in: Lecture Notes in Computer Science, vol. 1190, Springer, Berlin, Heidelberg, 1997, pp. 155–168.
[27] I.-A. Hartman, I. Newman, R. Ziv, On grid intersection graphs, Discrete Math. 87 (1) (1991) 41–52, http://dx.doi.org/10.1016/0012-365X(91)90069-E.
[28] A. Asinowski, E. Cohen, M. Golumbic, V. Limouzy, M. Lipshteyn, M. Stern, Vertex intersection graphs of paths on a grid, J. Graph Algorithms Appl. 16 (2) (2012) 129–150, http://dx.doi.org/10.7155/jgaa.00253.
[29] S. Chaplick, T. Ueckerdt, Planar graphs as vpg-graphs, J. Graph Algorithms Appl. (2013) 475–494.
[30] M.I. Hossain, M.S. Rahman, Monotone grid drawings of planar graphs, in: J. Chen, J. Hopcroft, J. Wang (Eds.), Proceedings of Frontiers in Algorithmics '14, in: Lecture Notes in Computer Science, vol. 8497, Springer International Publishing, 2014, pp. 105–116.
[31] J.E. Hopcroft, R.E. Tarjan, Dividing a graph into triconnected components, SIAM J. Comput. 2 (3) (1973) 135–158, http://dx.doi.org/10.1137/0202012.
[32] H.N. Gabow, R.E. Tarjan, A linear-time algorithm for a special case of disjoint set union, J. Comput. System Sci. 30 (2) (1985) 209–221, http://dx.doi.org/10.1016/0022-0000(85)90014-5.
[33] K. Yeap, M. Sarrafzadeh, Floor-planning by graph dualization: 2-concave rectilinear modules, SIAM J. Comput. 22 (3) (1993) 500–526, http://dx.doi.org/10.1137/0222035.

Please cite this article in press as: M.I. Hossain, M.S. Rahman, Good spanning trees in graph drawing, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.09.004

Highlights

- Introduces a good spanning tree of a plane graph.
- Solves two open problems on monotone drawings using good spanning trees.
- Finds a 2-visibility drawing of a planar graph using a good spanning tree.
- Gives algorithms to find an spike-VPG and T-shaped drawings of a planar graph.

Please cite this article in press as: M.I. Hossain, M.S. Rahman, Good spanning trees in graph drawing, Theoret. Comput. Sci. (2015), http://dx.doi.org/10.1016/j.tcs.2015.09.004