Discrete Mathematics, Algorithms and Applications
Vol. 7, No. 2 (2015) 1550007 (12 pages)

DOI: 10.1142/S179383091550007X

World Scientific
www.worldscientific.com

Straight-line monotone grid drawings of series–parallel graphs

Md. Iqbal Hossain* and Md. Saidur Rahman†

Graph Drawing and Information Visualization Laboratory
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka 1000, Bangladesh
**mdiqbalhossain@cse.buet.ac.bd*
†*saidurrahman@cse.buet.ac.bd*

Received 21 August 2013
Revised 17 September 2014
Accepted 21 January 2015
Published 26 February 2015

A monotone drawing of a planar graph G is a planar straight-line drawing of G where a monotone path exists between every pair of vertices of G in some direction. Recently monotone drawings of graphs have been discovered as a new standard for visualizing graphs. In this paper we study monotone drawings of series–parallel graphs in a variable embedding setting. We show that a series–parallel graph of n vertices has a straight-line planar monotone drawing on a grid of size $O(n) \times O(n^2)$ and such a drawing can be found in linear time.

Keywords: Monotone drawings; series–parallel graph; straight-line drawing; grid drawing.

Mathematics Subject Classification 2010: 68R10, 05C62, 05C85, 68Rxx, 68Wxx

1. Introduction

During the last two decades numerous results on various drawing styles of planar graphs were published [5, 11, 13]. 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 path* 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 of a planar graph is a *monotone drawing* if it contains at least one monotone path for each pair of vertices. In the drawing of a graph in Fig. 1(a), the path between the vertices s and t drawn as a thick line is a monotone path with respect to direction d, whereas no path between the vertices s' and t' is monotone with respect to any direction.

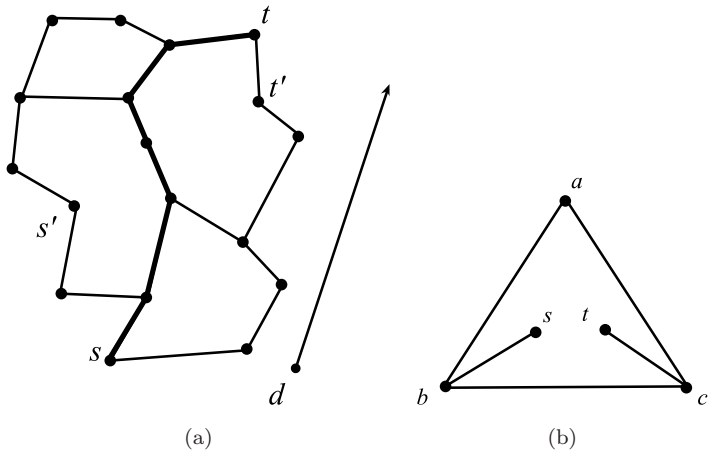


Fig. 1. (a) The path between vertices s and t (as shown using a thick line) is monotone with respect to direction d and (b) a plane graph G with no monotone drawing.

Upward drawings [6, 1] are related to monotone drawings where every directed path is monotone with respect to the vertical direction, while in a monotone drawing each monotone path, in general, is monotone with respect to some direction. Arkin *et al.* [4] showed that any strictly convex drawing of a planar graph is monotone. Angelini *et al.* [2] showed that every biconnected planar graph has a straight-line monotone drawing in real coordinate space. They also showed that every tree admits a straight-line planar monotone drawing in $O(n) \times O(n^2)$ or $O(n^{1.6}) \times O(n^{1.6})$ area. Every connected plane graph admits a monotone grid drawing on an $O(n) \times O(n^2)$ grid using at most two bends per edge and an outerplane graph of n vertices admits a straight-line monotone drawing on a grid of area $O(n) \times O(n^2)$ [3]. It is also known that not every plane graph (with fixed embedding) admits a straight-line monotone drawing [2].

Thus a natural question is whether every connected planar graph has a straight-line monotone drawing and what is the minimum area requirement for such a drawing on a grid. In this paper, we investigate this problem for a nontrivial subclass of planar graphs called "series–parallel graphs". We show that every series–parallel graph of n vertices admits a straight-line monotone drawing on an $O(n) \times O(n^2)$ grid and such a drawing can be computed in $O(n)$ time. Note that a series–parallel graph considered in this paper is a connected graph which becomes biconnected after adding at most one edge.

We now give an outline of our algorithm for constructing a monotone drawing of a series–parallel graph G. We construct an ordered "SPQ-tree" of G. We then assign a slope to each node of the SPQ-tree. We finally draw G on a grid taking into consideration the slope assigned to each node of the SPQ-tree. Not every plane series–parallel graph has a monotone drawing [2]. Figure 1(b) illustrates a plane series–parallel graph that has no monotone drawing. Hence it may be necessary

to change the embedding of a plane series–parallel graph for finding a monotone drawing. (An *embedding* of a graph is defined by the clockwise ordering of adjacent edges around each vertex of the graph in a drawing of the graph in two-dimensional plane.) The preliminary version of this research has been presented at [9].

The rest of the paper is organized as follows. Section 2 describes some of the definitions that we have used in our paper. Section 3 deals with straight-line monotone drawings of series–parallel graphs. Finally, Sec. 4 concludes our paper with discussions.

2. Preliminaries

In this section we give some definitions and preliminary known results.

A graph is *planar* if it can be drawn in the plane without edge crossings except at the vertices where the edges are incident. 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 the vertices on the outer face are called *outer vertices*. The edges lie on the outer face are called *outer edges*. A *cut vertex* in a connected graph G is a vertex whose removal disconnects G. A connected graph with no cut vertex is called a *biconnected graph*. A maximal biconnected subgraph of a connected graph G is called a *biconnected component* of G.

A graph $G = (V, E)$ is called a *series–parallel* graph (with source s and sink t) if either G consists of a pair of vertices connected by a single edge or there exist two series–parallel graphs $G_i = (V_i, E_i)$, $i = 1, 2$, with source s_i and sink t_i such that $V = V_1 \cup V_2, E = E_1 \cup E_2$, and either $s = s_1, t_1 = s_2$ and $t = t_2$ or $s = s_1 = s_2$ and $t = t_1 = t_2$ [12]. A biconnected component of a series–parallel graph is also a series–parallel graph. By definition, a series–parallel graph G is a connected planar graph and G has exactly one source s and exactly one sink t.[a] Thus the following fact holds.

Fact 1. Let $G = (V, E)$ be a series–parallel graph with the source vertex s and the sink vertex t. Assume that $(s, t) \notin E(G)$. Then $G' = (V, E \cup (s, t))$ is a planar biconnected series–parallel graph.

A pair u, v of vertices of a connected graph 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 *split component* of a split pair u, v is either an edge (u, v) or a maximal connected subgraph H of G such that u, v is not a split pair of H.

[a]Another definition of a series–parallel graph is found in literature: a graph G is a series–parallel graph if and only if G is K_4-minor free [10]. By this definition, tree and outerplanar graphs are subclasses of series–parallel graphs. However, we are not using this definition in this paper.

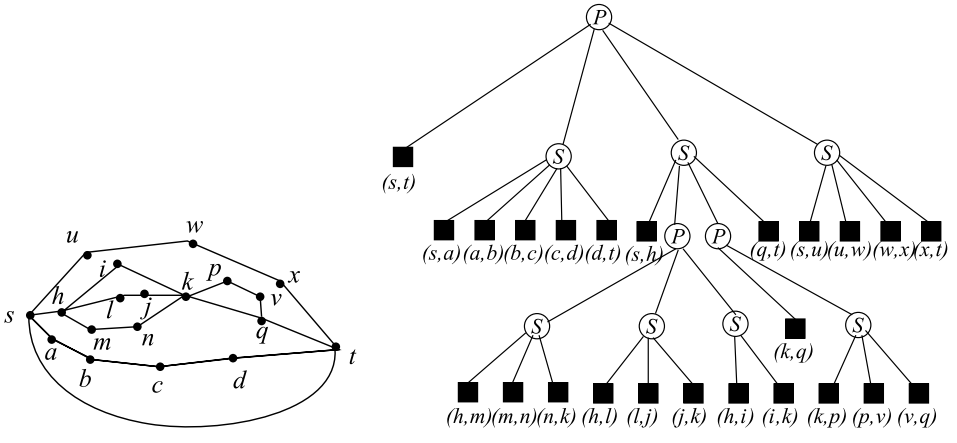
Let G be a biconnected series–parallel graph. Let (u, v) be an edge of G. The *SPQ-tree* [8, 7] $\mathcal{T}$ of G with respect to a *reference edge* $e = (u, v)$ describes a recursive decomposition of G induced by its split pairs. Tree $\mathcal{T}$ is a rooted ordered tree whose nodes are of three types: S, P and Q. Each node $\mathcal{X}$ of $\mathcal{T}$ corresponds to a subgraph of G, called its *pertinent graph* denoted by pert($\mathcal{X}$). Each node $\mathcal{X}$ of $\mathcal{T}$ has an associated biconnected multi-graph, called the *skeleton* of $\mathcal{X}$ and denoted by skeleton($\mathcal{X}$). Tree $\mathcal{T}$ is recursively defined as follows.

Trivial Case: In this case, G consists of exactly two parallel edges e and e' joining u and v. $\mathcal{T}$ consists of a single Q-node $\mathcal{X}$, and the skeleton of $\mathcal{X}$ is G itself. The pertinent graph pert($\mathcal{X}$) consists of only the edge e'.

Parallel Case: In this case, the split pair u, v has three or more split components $G_0, G_1, \ldots, G_k, k \geq 2$, and G_0 consists of only a reference edge $e = (u, v)$. The root of $\mathcal{T}$ is a P-node $\mathcal{X}$. The skeleton($\mathcal{X}$) consists of $k + 1$ parallel edges $e_0, e_1, \ldots, e_k$ joining s and t, where $e_0 = e = (u, v)$ and e_i, $1 \leq i \leq k$, corresponds to G_i. The pertinent graph pert($\mathcal{X}$) $= G_1 \cup G_2 \cup \cdots \cup G_k$ is the union of $G_1, G_2, \ldots, G_k$.

Series Case: In this case the split pair u, v has exactly two split components, and one of them consists of the reference edge e. One may assume that the other split component has cut-vertices $c_1, c_2, \ldots, c_{k-1}$, $k \geq 2$, that partition the component into its blocks $G_1, G_2, \ldots, G_k$ in this order from t to s. Then the root of $\mathcal{T}$ is an S-node $\mathcal{X}$. The skeleton of $\mathcal{X}$ is a cycle $e_0, e_1, \ldots, e_k$ where $e_0 = e, c_0 = u, c_k = v$, and e_i joins c_{i-1} and $c_i, 1 \leq i \leq k$. The pertinent graph pert($\mathcal{X}$) of node $\mathcal{X}$ is the union of $G_1, G_2, \ldots, G_k$. Figure 2 shows a series–parallel graph and its *SPQ*-tree decomposition.

Let $\mathcal{T}$ be the *SPQ*-tree of a series–parallel graph G and let $\mathcal{X}$ be a node of $\mathcal{T}$. For an S-node $\mathcal{X}$, we denote by $n(\mathcal{X})$ the number of vertices in pert($\mathcal{X}$) excluding



(a) (b)

Fig. 2. (a) A series–parallel graph G and (b) the *SPQ*-tree $\mathcal{T}$ of G.

u and v. For a P-node $\mathcal{X}$, we denote by $n(\mathcal{X})$ the number of vertices in pert($\mathcal{X}$) including u and v. According to the SPQ-tree decomposition, a P-node cannot be the parent of another P-node and an S-node cannot be the parent of another S-node. Throughout the paper, by drawing of a node $\mathcal{X}$ in $\mathcal{T}$ we mean the drawing of pert($\mathcal{X}$).

Monotone Drawings

Let p be a point in the plane and l be a half-line starting at p. The *slope* of l, denoted by slope(l), is the angle spanned by a counter-clockwise rotation that brings a horizontal half-line starting at p and directed toward increasing x-coordinates to coincide with l.

Let Γ be a drawing of a graph G and let (u, v) be an edge of G. The half-line starting at u and passing through v, denoted by $d(u, v)$, is the *direction of* (u, v). The *direction of an edge* e is denoted by $d(e)$ and *slope* of $d(e)$ is denoted by slope(e).

Let $P(u_1, u_q) = (u_1, u_2, \ldots, u_q)$ be a path in a straight-line drawing of a graph. Let e_i be the edge from u_i to u_{i+1} $(1 \leq i \leq q-1)$. Let e_j and e_k be two edges of the path $P(u_1, u_q)$ such that slope(e_j) $\geq$ slope(e_i) and slope(e_k) $\leq$ slope(e_i) for $i = 1, \ldots, q-1$. We call e_j and e_k *extremal edges* of the path $P(u_1, u_q)$. The path $P(u_1, u_q)$ is a *monotone path* with respect to a direction d if the orthogonal projections of vertices $u_1, \ldots, u_q$ on a line along the direction d appear in the same order as the vertices appear in the path.

Let $P(u_1, u_q) = (u_1, u_2, \ldots, u_q)$ be a monotone path. Let e_j and e_k be the extremal edges of $P(u_1, u_q)$. If we draw each edge $e_i(u_i, u_{i+1})$ such that u_i is placed on the origin of the axis, then e_j and e_k create two closed wedges at the

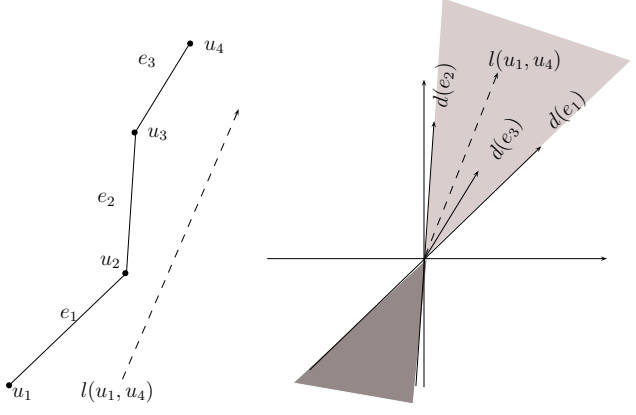


Fig. 3. (a) A monotone path $P(u_1, u_4)$ with extremal edges e_1 and e_2 and (b) the light-gray shaded region represents the range of $P(u_1, u_4)$ defined by $d(e_1)$ and $d(e_2)$ and the dark-gray shaded region represents the opposite range of $P(u_1, u_4)$. Observe that range($P(u_1, u_4)$) contains the half-line $l(u_1, u_4)$ from u_1 through u_4.

origin of the axis. One closed wedge delimited by $d(e_j)$ and $d(e_k)$ containing all the half-lines $d(u_i, u_{i+1})$, for $i = 1, \ldots, q-1$, is the *range* of $P(u_1, u_q)$ and is denoted by range$(P(u_1, u_q))$, while the closed wedge delimited by $d(e_j) - \pi$ and $d(e_k) - \pi$, and not containing $d(e_j)$ and $d(e_k)$, is the *opposite range* of $P(u_1, u_q)$ and is denoted by opp$(P(u_1, u_q))$. Figure 3 illustrates an example of range$(P(u_1, u_4))$ and opp$(P(u_1, u_4))$.

We now recall some important properties of monotone paths from [2] as in the following two lemmas.

Lemma 2.1. *A path $P(u_1, u_q)$ is monotone if and only if* range$(P(u_1, u_q)) < \pi$.

Lemma 2.2. *Let $P(u_1, u_q) = (u_1, \ldots, u_q)$ and $P(u_q, u_{q+k}) = (u_q, \ldots, u_{q+k})$ be monotone paths. Then the path $P(u_1, u_{q+k}) = (u_1, \ldots, u_q, u_{q+1}, \ldots, u_{q+k})$ is monotone if and only if* range$(P(u_1, u_q)) \cap$ opp$(P(u_q, u_{q+k})) = \emptyset$. *Furthermore, if $P(u_1, u_{q+k})$ is monotone, then* range$(P(u_1, u_q)) \cup$ range$(P(u_q, u_{q+k})) \subseteq$ range$(P(u_1, u_{q+k}))$.

3. Monotone Grid Drawing of Series–Parallel Graphs

In this section we give a linear-time algorithm to find a straight-line planar monotone grid drawing of a connected series–parallel graph G of n vertices on an $O(n) \times O(n^2)$ grid. If G is a single edge then it can be drawn on a line with slope 1 and that is a monotone drawing. We now assume that G is a simple series–parallel graph with $n > 2$. To get a monotone drawing of G we first construct an *SPQ*-tree. We then assign a slope to each node. We finally draw the graph on a grid taking into consideration the slopes assigned to each node of the tree. The details of our algorithm are as follows.

We assume that an edge exists between the source s and the sink t of the input series–parallel graph G: otherwise, we add a dummy edge between s and t of G. (Note that the graph remains planar after adding the dummy edge (s, t) by Fact 1.) Later we will show that the drawing of G obtained by our algorithm remains monotone even after removing the dummy edge (s, t) from the drawing. Clearly, G is a biconnected series–parallel graph (with the edge (s, t)). Let $\mathcal{T}$ be the *SPQ*-tree of G with respect to the edge (s, t).

According to decomposition of G, the root of $\mathcal{T}$ is a Q-node $\mathcal{R}$ and $\mathcal{R}$ has only child $\mathcal{X}$ which is either a P-node or an S-node. We now re-root $\mathcal{T}$ at $\mathcal{X}$ so that $\mathcal{R}$ is placed as the left most child of $\mathcal{X}$. Note that this operation does not break the ordering of other nodes.

Let $\mathcal{T}'$ be a *SPQ*-tree obtained from $\mathcal{T}$ as follows. We traverse each P-node of $\mathcal{T}$; if any Q-node exists as a child of a P-node then we put the Q-node as the leftmost child of the P-node.

We now assign a slope to each node of $\mathcal{T}'$. Let $1/1, 2/1, \ldots, (n-1)/1$ be $n-1$ slopes in increasing order. (See Fig. 4.) Initially we assign the slope $1/1$ to the root of $\mathcal{T}'$. We then traverse $\mathcal{T}'$ to assign a slope to each node $\mathcal{X}$ of $\mathcal{T}'$. Let $\mu/1$ be the

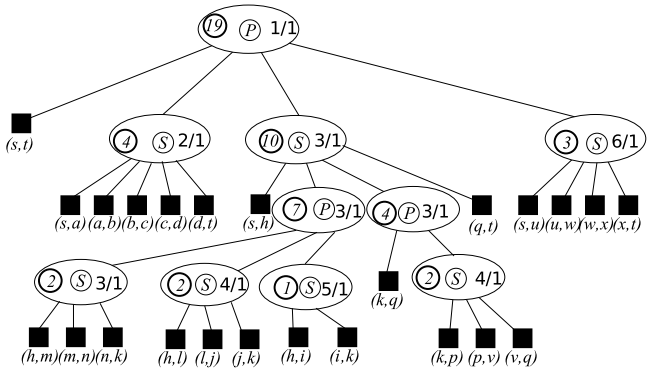


Fig. 4. Illustration for slope assignment in $\mathcal{T}'$. Number of vertices, node type and assigned slope of each node are written inside the node.

slope assigned to $\mathcal{X}$, and let $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k$ $(k < n)$ be the children of $\mathcal{X}$ in left to right order. We first consider the case where $\mathcal{X}$ is a P-node. We assign the slope $\mu/1$ to the leftmost child $\mathcal{X}_1$. Then, for each $i = 2, \ldots, k$ we assign the slope $(\mu_{i_{\max}}+1)/1$ to $\mathcal{X}_{i+1}$ where $\mu_{i_{\max}}/1$ is the largest slope assigned in the subtree rooted at $\mathcal{X}_i$. We now consider the case where $\mathcal{X}$ is an S-node. In this case we assign the same slope $\mu/1$ to $\mathcal{X}_i$ (for each $i \leq k$). Thus the largest slope assigned to a vertex can be at most $(n-1)/1$. Figure 4 illustrates the slope assignment to the nodes of the SPQ-tree for the graph G shown in Fig. 2(a).

We next draw G on a grid using the slope assigned to each node of $\mathcal{T}'$. Our algorithm uses a post-order traversal on the ordered SPQ-tree. For describing the algorithm we need to define some terms.

Let $\mathcal{X}$ be a node of $\mathcal{T}'$. We denote by $\mathcal{X}'$ the parent of $\mathcal{X}$ and by $\mathcal{X}''$ the parent of $\mathcal{X}'$ in $\mathcal{T}'$. (Then $\mathcal{X}''$ is the grandparent of $\mathcal{X}$.) For a node $\mathcal{X}$ in $\mathcal{T}'$ we denote the source and the sink of $\mathcal{X}$ by $s_{\mathcal{X}}$ and $t_{\mathcal{X}}$, respectively. We denote the slope assigned to $\mathcal{X}$ by $\mu_{\mathcal{X}}$.

Assume that $\mathcal{X}$ is a Q-node of $\mathcal{T}'$. Then $\text{pert}(\mathcal{X})$ is the edge $(s_{\mathcal{X}}, t_{\mathcal{X}})$. The parent $\mathcal{X}'$ can be either an S-node or a P-node. If $\mathcal{X}'$ is a P-node then $\mathcal{X}$ is the only Q child of $\mathcal{X}'$ and all others children of $\mathcal{X}'$ are S-nodes. In this case $s_{\mathcal{X}} = s_{\mathcal{X}'}$ and $t_{\mathcal{X}} = t_{\mathcal{X}'}$ are the source and sink of $\mathcal{X}'$, respectively. Otherwise $\mathcal{X}'$ is an S-node. In this case, let $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k$ be the sequence of Q-nodes found by using a post-order traversal in the tree rooted at $\mathcal{X}'$. If (a, b) and (w, v) are $\text{pert}(\mathcal{X}_1)$ and $\text{pert}(\mathcal{X}_k)$, respectively, then $s_{\mathcal{X}''} = a$ and $t_{\mathcal{X}''} = v$ are the source and the sink of $\mathcal{X}'$, respectively. We call $\mathcal{X}_1$ and $\mathcal{X}_k$ the *first node* and the *last node* of $\mathcal{X}'$, respectively. Similarly, we call b and w the *first vertex* and *last vertex* of $\text{pert}(\mathcal{X}')$, respectively.

We are now ready to describe the drawing algorithm. At the beginning of the drawing we put the source and the sink of G on $(0, 0)$ and (n, n) points, respectively. For completing the drawing we use a post-order traversal in $\mathcal{T}'$ to draw G on grid as follows. For each node $\mathcal{X}$ in $\mathcal{T}'$ we have following three cases to consider.

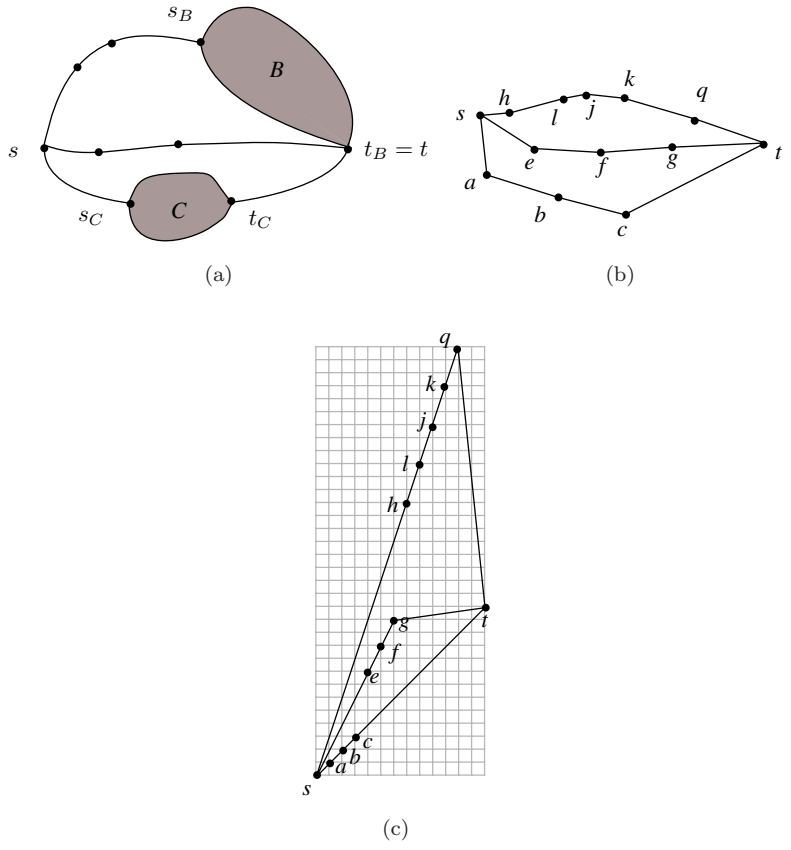


Fig. 5. (a) An example of a P-node for Cases 1 and 2, (b) a P-node $\mathcal{X}$, and (c) a drawing of $\mathcal{X}$ on a grid.

Case 1: *$\mathcal{X}$ is a P-node.* In this case $s_{\mathcal{X}}$ has already been placed in grid since we are doing post-order traversal. If $t_{\mathcal{X}} = t_{\mathcal{X}''}$ then $t_{\mathcal{X}}$ has also been placed on the grid (see node B in Fig. 5). Otherwise, we place $t_{\mathcal{X}}$ on $p_x(s_{\mathcal{X}}) + (n(\mathcal{X}), p_y(s_{\mathcal{X}}) + n(\mathcal{X}) \times \mu_{\mathcal{X}})$. That means, $t_{\mathcal{X}}$ is placed on the rightmost position of all vertices of pert($\mathcal{X}$). The edges of pert($\mathcal{X}$) will be drawn when we draw Q-node.

Case 2: *$\mathcal{X}$ is a S-node.* In this case we just proceed.

Case 3: *$\mathcal{X}$ is a Q-node.* In this case if $\mathcal{X}'$ is a P-node then s and t have already been placed on the grid (sec Case 1). Otherwise, $\mathcal{X}'$ is an S-node, and we have the following three subcases according to the position of $\mathcal{X}$ in the children list of $\mathcal{X}'$.

Subcase 3a: *$\mathcal{X}$ is the first child of $\mathcal{X}'$.* In this case, $s_{\mathcal{X}} = s_{\mathcal{X}''}$ where $s_{\mathcal{X}''}$ is the source of $\mathcal{X}''$, which is already placed in the grid. Let z be the number of vertices of part($\mathcal{X}''$) that are already drawn in the gird. We place $t_{\mathcal{X}}$ on $(p_x(s_{\mathcal{X}''}) + z$, $p_y(s_{\mathcal{X}''}) + z \times \mu_{\mathcal{X}})$, then add an edge between $s_{\mathcal{X}}$ and $t_{\mathcal{X}}$. Note that $\mu_{\mathcal{X}}$ is the largest slope among the part that are already drawn in the grid. That means we

are placing $t_{\mathcal{X}}$ at the highest y-coordinate and $p_x(t_{\mathcal{X}}) < p_x(t_{\mathcal{X}''})$. Clearly, $e(s_{\mathcal{X}}, t_{\mathcal{X}})$ does not cross any other edges. (See the edge (s, h) in the drawing of Fig. 5.)

Subcase 3b: *$\mathcal{X}$ is neither first child nor last child of $\mathcal{X}'$.* In this case $s_{\mathcal{X}}$ has already been placed on the grid, and we place $t_{\mathcal{X}}$ on $(p_x(s_{\mathcal{X}})+1, p_y(s_{\mathcal{X}})+\mu_{\mathcal{X}})$. Since $p_x(s_{\mathcal{X}})$ is the largest y-coordinate and $\mu_{\mathcal{X}}$ is positive, adding an edge between $s_{\mathcal{X}}$ and $t_{\mathcal{X}}$ does not produce any edge crossing in the drawing. Let $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k$ be Q-nodes with the same parent $\mathcal{X}'$. Let $(v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k)$ be the pertinent graph of $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k$, respectively. Then the path $P(v_1, v_k)$ lies on a straight line with slope $\mu_{\mathcal{X}'}$. (See the path $P(h, q) = \{h, l, j, k, q\}$ in the drawing of Fig. 5).

Subcase 3c: *$\mathcal{X}$ is the last child of $\mathcal{X}'$.* In this case $t_{\mathcal{X}} = t_{\mathcal{X}''}$, and both the vertices have already been placed on the grid. Since $p(t_{\mathcal{X}''})$ is the right most point and $p(s_{\mathcal{X}})$ is the top most point in the drawing, adding an edge between $s_{\mathcal{X}}$ and $t_{\mathcal{X}}$ does not produce any edge crossing in the drawing. Note that the slope of $e(s_{\mathcal{X}}, t_{\mathcal{X}})$ lies between $\pi/2$ and $-\pi/2$. (See the edges $(q, t), (g, t)$ and (c, t) in the drawing of Fig. 5). If $\mathcal{X}'$ is not the first child of $\mathcal{X}''$ then we call the edge $(s_{\mathcal{X}}, t_{\mathcal{X}})$ a *P-node closing edge*. In Fig. 5 edges (g, t) and (q, t) are P-node closing edges.

For the series–parallel graph in Fig. 2(a), a *SPQ*-tree, slope assignment and a straight-line monotone grid drawing are illustrated in Figs. 2(b), 4 and 6, respectively.

We call the algorithm described above ***Algorithm*** *Monotone-Draw*. We now have the following theorem.

Theorem 3.1. *Let G be a simple connected series–parallel graph of n vertices.* ***Algorithm*** *Monotone-Draw finds a monotone drawing of G on a grid of size $O(n) \times O(n^2)$ in $O(n)$ time.*

Proof. Let Γ be a drawing of G constructed by **Algorithm** Monotone-Draw. In the drawing step it has been ensured that there is no edge crossing in Γ. We now show that Γ is a monotone drawing of G. To prove the claim, we show that, a monotone path exists between every pair of vertices of G in Γ.

Let s and t be the source and the sink of G. In fact we will prove that a monotone path exists between every pair of vertices of G in the drawing of $G\backslash(s, t)$ in Γ. Let v be a vertex in G such that $v \notin \{s, t\}$. We first show that there exist a monotone path between v and s, and a monotone path between v and t. One can easily observe that a path $P(s, v)$ exists such that no P-node closing edge is contained in $P(s, v)$ and for each edge $e \in P(v, s)$, $\pi/2 > \text{slope}(e) \geq \pi/4$ holds. Then the path $P(s, v)$ is monotone since the range$(P(s, v))$ is smaller than π. On the other hand a path $P(v, t)$ exists such that $s \notin P(v, t), (s, t) \notin P(v, t)$ and $P(v, t)$ may contain some P-node closing edges. The path $P(v, t)$ is monotone since for each edge $e \in P(v, t)$ $\pi/2 < \text{slope}(e) < -\pi/2$ holds. Similarly, any path $P(s, t)$ in $\Gamma\backslash(s, t)$ is monotone since for each edge $e \in P(s, t)$, the relation $\pi/2 < \text{slope}(e) < -\pi/2$ holds.

We now show that for every pair of vertices $u, v \in G$ ($v \notin \{s, t\}$, $u \notin \{s, t\}$) there is a monotone path between u and v in Γ. Let $P(u, s)$ and $P(v, s)$ be two paths such

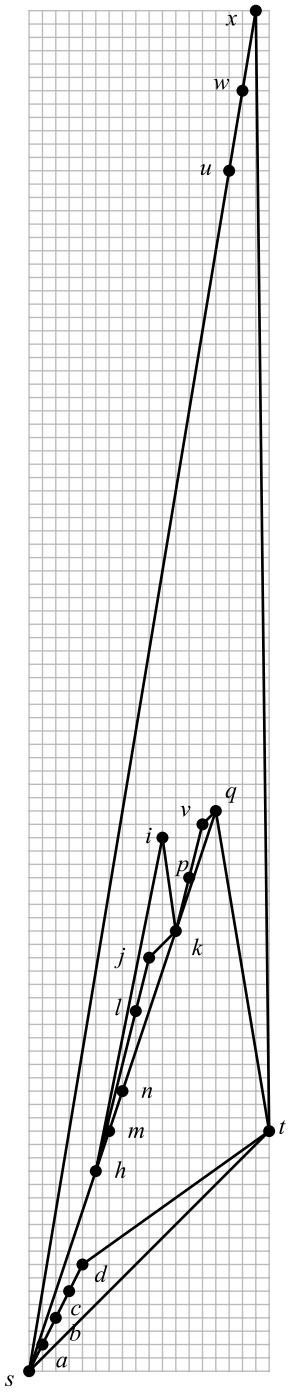


Fig. 6. Straight-line monotone drawing of G.

that none of them contains a P-node closing edge and assume that $e_1 = (u, u')$ lies on $P(u, s)$ and $e_2 = (v, v')$ lies on $P(v, s)$.

Let $\mathcal{M}$ and $\mathcal{N}$ be the two Q-nodes in $\mathcal{T}'$ such that $(u, u') \in \text{pert}(\mathcal{M})$ and $(v, v') \in \text{pert}(\mathcal{N})$, and let $\mathcal{X}$ be the lowest common ancestor of $\mathcal{M}$ and $\mathcal{N}$ in $\mathcal{T}'$. Let $\mathcal{U}$ and $\mathcal{V}$ be the children of $\mathcal{X}$ and ancestors of $\mathcal{M}$ and $\mathcal{N}$, respectively. Since e_1 and e_2 are not P-node closing edges, the slopes of $d(e_1)$ and $d(e_2)$ are $-\mu_{\mathcal{M}}$ and $-\mu_{\mathcal{N}}$, respectively.

We now have the following two cases to consider.

Case 1: *$\mathcal{X}$ is a P-node.* Without loss of generality we may consider $\mu_{\mathcal{M}} > \mu_{\mathcal{N}}$. So according to the slope assignment $\mu_{\mathcal{M}} \geq \mu_{\mathcal{U}} > \mu_{\mathcal{V}} \geq \mu_{\mathcal{N}}$. Let w and w' be the source and sink vertices of $\mathcal{X}$ in Γ. The path $P(u, v)$ $(w' \notin P(u, v))$ is composed of path $P(u, w)$ and of path $P(w, v)$. Clearly, for each edge $e \in P(u, w)$, and $e' \in P(w, v)$ it holds $\mu_{\mathcal{M}} \geq \textit{slope}(e) \geq \mu_{\mathcal{U}}$ and $\mu_{\mathcal{N}} \geq \textit{slope}(e') \geq \mu_{\mathcal{V}}$, respectively. So we have $\text{range}(P(u, w)) \cap \text{opp}(P(w, v)) = \emptyset$. Then by Lemma 2.2, $P(u, v)$ is a monotone path.

Case 2: *$\mathcal{X}$ is an S-node.* If $\mathcal{M}$ and $\mathcal{N}$ are children of the same S-node then the case is straight-forward, the path $P(u, v)$ lies on a straight-line. Otherwise the path $P(u, v)$ could have some P-node closing edge. Let $\mathcal{X}'$ be the parent of $\mathcal{X}$. Clearly $\mathcal{X}'$ is a P-node. Let w' be the source vertex of $\mathcal{X}'$ in Γ. Then the path $P(u, v)$ $(w' \notin P(u, v))$ is monotone with respect to a horizontal half-line (see Fig. 7).

Thus we have proved that Γ is a monotone drawing of G.

We are placing the sink of each P-node on the $x = n(\mathcal{X})$ line. The drawings of all child nodes are inside the drawing of its parent P-node. Hence the largest x-coordinate of the drawing can be at most n.

We now analyze the time complexity of our algorithm. We construct SPQ-tree in linear time. We can compute slopes of each vertex of $\mathcal{T}'$ in linear time. Coordinates

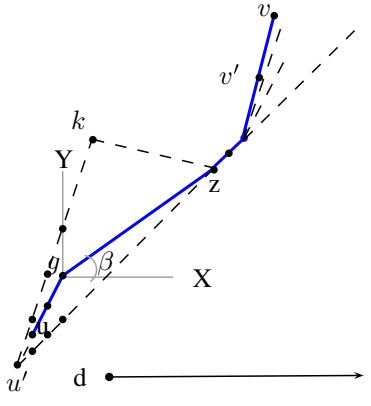


Fig. 7. Monotone path in Γ for series case.

of each vertex can be computed in constant time. Thus the overall time complexity of the algorithm is $O(n)$. □

4. Conclusion

In this paper we have studied monotone grid drawings of series–parallel graphs. We have shown that a series–parallel graph of n vertices has a straight-line planar monotone drawing on an $O(n) \times O(n^2)$ grid and such a drawing can be found in $O(n)$ time. Finding straight-line monotone grid drawings of larger classes of planar graphs is our future work.

Acknowledgments

The authors would like to thank the anonymous reviewer for his valuable comments and suggestions to improve the quality of the paper.

References

[1] Md. Abul Hassan Samee and Md. Saidur Rahman, Upward planar drawings of series–parallel digraphs with maximum degree three, in *Proc. Workshop on Algorithms and Computation*, *WALCOM'07* (2007), pp. 28–45.

[2] P. Angelini, E. Colasante, G. Di Battista, F. Frati and M. Patrignani, Monotone drawings of graphs, *J. Graph Algorithms Appl.* **16**(1) (2012) 5–35.

[3] P. Angelini, W. Didimo, S. Kobourov, T. Mchedlidze, V. Roselli, A. Symvonis and S. Wismath, Monotone drawings of graphs with fixed embedding, in *Proc. 19th Int. Conf. on Graph Drawing*, *GD*'11 (Springer-Verlag, Berlin, 2012), pp. 379–390.

[4] E. M. Arkin, R. Connelly and J. S. B. Mitchell, On monotone paths among obstacles with applications to planning assemblies, in *Proc. 5th Annual Symp. on Computational Geometry*, *SoCG* '89 (ACM, New York, 1989), pp. 334–343.

[5] G. Di Battista, P. Eades, R. Tamassia and I. G. Tollis, *Graph Drawing*: *Algorithms for the Visualization of Graphs* (Prentice-Hall, New York, 1999).

[6] G. Di Battista and R. Tamassia, Algorithms for plane representations of acyclic digraphs, *Theor. Comput. Sci.* **61**(2–3) (1988) 175–198.

[7] G. Di Battista and R. Tamassia, On-line maintenance of triconnected components with SPQR-trees, *Algorithmica* **15**(4) (1996) 302–318.

[8] G. Di Battista, R. Tamassia, R. Tamassia, L. Vismara and L. Vismara, Output-sensitive reporting of disjoint paths, *Algorithmica* **23** (1999) 302–340.

[9] Md. Iqbal Hossain and Md. Saidur Rahman, Straight-line monotone grid drawings of series–parallel graphs, in *Proc. 19th Annual Int. Computing and Combinatorics Conf.* (*COCOON* 2013), eds. D.-Z. Du and G. Zhang, Lecture Notes in Computer Science, Vol. 7936 (Springer, Berlin, 2013), pp. 672–679.

[10] K.-W. Lih, W.-F. Wang and X. Zhu, Coloring the square of a k_4-minor free graph, *Discrete Math.* **269**(13) (2003) 303–309.

[11] T. Nishizeki and Md. Saidur Rahman, *Planar Graph Drawing*, Lecture Notes Series on Computing (World Scientific, Singapore, 2004).

[12] Md. Saidur Rahman, N. Egi and T. Nishizeki, No-bend orthogonal drawings of series-parallel graphs, in *Proc. 13th Int. Conf. Graph Drawing, GD*'05, Lecture Notes in Computer Science, Vol. 3843 (Springer-Verlag, Berlin, 2006), pp. 409–420.

[13] R. Tamassia, *Handbook of Graph Drawing and Visualization*, Discrete Mathematics and Its Applications (Chapman & Hall/CRC, Boca Raton, FL, 2013).