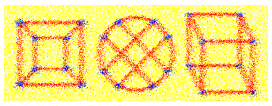

Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 15, no. 3, pp. 437–456 (2011)

Straight-Line Grid Drawings of Label-Constrained Outerplanar Graphs with $O(n \log n)$ Area

Md. Rezaul Karim [1,2] *Md. Jawaherul Alam* [1] *Md. Saidur Rahman* [1]

[1]Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka-1000, Bangladesh.
[2]Department of Computer Science and Engineering, University of Dhaka, Dhaka-1000, Bangladesh.

Abstract

A straight-line grid drawing of a planar graph G is a drawing of G on an integer grid such that each vertex is drawn as a grid point and each edge is drawn as a straight-line segment without edge crossings. Any outerplanar graph of n vertices with maximum degree d has a straight-line grid drawing with area $O(dn \log n)$. In this paper, we introduce a subclass of outerplanar graphs, which we call label-constrained outerplanar graphs, that admits straight-line grid drawings with $O(n \log n)$ area. We give a linear-time algorithm to find such a drawing. We also give a linear-time algorithm for the recognition of label-constrained outerplanar graphs.

Submitted:	Reviewed:	Revised:	Accepted:
May 2009	September 2009	July 2010	October 2010
	Final: December 2010	Published: July 2011	
Article type: Regular paper		Communicated by: S. Das and R. Uehara	

E-mail addresses: rkarim@univdhaka.edu (Md. Rezaul Karim) jawaherul@yahoo.com (Md. Jawaherul Alam) saidurrahman@cse.buet.ac.bd (Md. Saidur Rahman)

1 Introduction

Recently, automatic aesthetic drawings of graphs have created intense interest due to their broad applications in computer networks, VLSI layout, information visualization etc., and as a consequence a number of drawing styles have come out [4, 9, 14, 15]. A classical and widely studied drawing style is the "straight-line drawing" of a planar graph. A *straight-line drawing* of a planar graph G is a drawing of G such that each vertex is drawn as a point and each edge is drawn as a straight-line segment without edge crossings. A *straight-line grid drawing* of a planar graph G is a straight-line drawing of G on an integer grid such that each vertex is drawn as a grid point as illustrated in Figure 1(d). The *area* of a drawing is the area of the smallest rectangle that encloses the drawing. It is

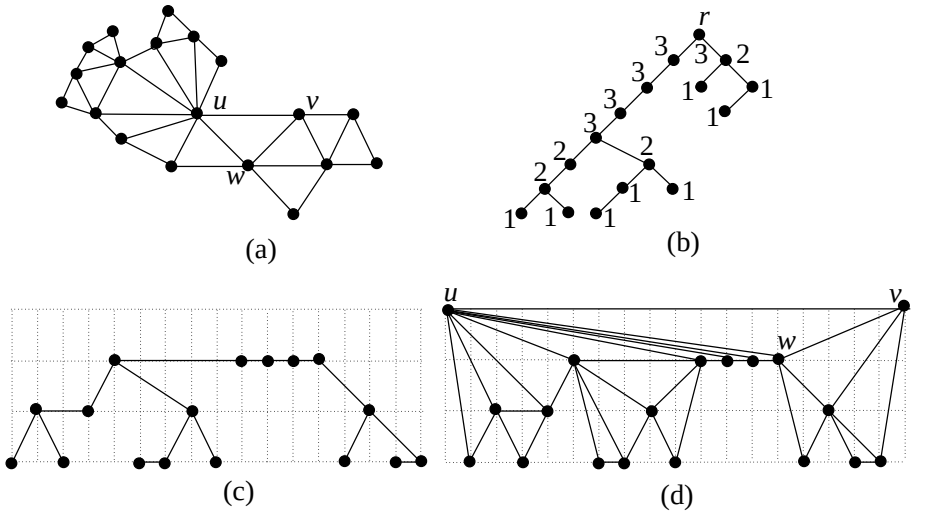

Figure 1: (a) A label-constrained outerplanar graph G, (b) the dual rooted ordered tree T_r of G, (c) a straight-line grid drawing of T_r, and (d) a straight-line grid drawing of G.

well known that a planar graph of n vertices admits a straight-line grid drawing on a grid of area $O(n^2)$ [16, 3]. A lower bound of $\Omega(n^2)$ on the area-requirement for straight-line grid drawings of certain planar graphs is also known [3]. Garg and Rusu showed that an n-node binary tree has a planar straight-line grid drawing with area $O(n)$ [8]. Although trees admit straight-line grid drawings with linear area, triangulations may require a grid of quadratic size. Hence finding nontrivial classes of planar graphs of n vertices richer than trees that admit straight-line grid drawings with area $o(n^2)$ is posted as an open problem in [2], and several recent works have addressed this open problem (see, e.g., [7, 11, 12, 13]). The problem of finding straight-line grid drawings of outerplanar graphs with $o(n^2)$ area was first posed by Biedl in [1], and Garg and Rusu showed that an outerplanar graph with n vertices and maximum degree d has

a planar straight-line drawing with area $O(dn^{1.48})$ [7]. Di Battista and Frati showed that a "balanced" outerplanar graph of n vertices has a straight-line grid drawing with area $O(n)$ and a general outerplanar graph of n vertices has a straight-line grid drawing with area $O(n^{1.48})$ [5]. Recently Frati showed that a general outerplanar graph with n vertices admits a straight-line grid drawing with area $O(dn \log n)$, where d is the maximum degree of the graph [6].

In this paper, we introduce a subclass of outerplanar graphs which has a straight-line grid drawing on a grid of area $O(n \log n)$. We give a linear-time algorithm to find such a drawing. We call this class "label-constrained outerplanar graphs" since a "vertex labeling" of the dual tree of this graph satisfies certain constraints. Figure 1(a) illustrates a "label-constrained outerplanar graph" G, and a straight-line grid drawing of G with $O(n \log n)$ area is illustrated in Figure 1(d). The "label-constrained outerplanar graphs" are richer than "balanced" outerplanar graphs. We also give a linear-time algorithm for recognition of a "label-constrained outerplanar graph."

The remainder of the paper is organized as follows. In Section 2, we give some definitions. Section 3 provides the drawing algorithm. Section 4 presents a linear-time algorithm for recognition of a "label-constrained outer planar graph," and Section 5 concludes the paper. An early version of this paper is presented at [10].

2 Preliminaries

In this section we give some definitions, introduce a labeling of a binary tree and define a class of outerplanar graphs which we call "label-constrained outerplanar graphs."

Let $G = (V, E)$ be a connected simple graph. Throughout the paper, we denote by n the number of vertices in G, that is, $n = |V|$, and denote by m the number of edges in G, that is, $m{=}|E|$. We denote by $G - \{u, v\}$ a graph $G' = (V', E')$ where $V' = V(G) - \{u, v\}$ and E' is the set of edges induced by V' in G. A *path* in G is an ordered list of distinct vertices $v_1, v_2, ..., v_q \in V$ such that $(v_{i-1}, v_i) \in E$ for all $2 \leq i \leq q$. Vertices v_1 and v_q are the end vertices of the path $v_1, v_2, ..., v_q$. We call a path u-v *path* if u and v are the *end vertices* of the path.

A graph is *planar* if it can be embedded in the plane so that no two edges intersect geometrically except at a vertex to which they are both incident. A *plane graph* is a planar graph with a fixed embedding. A plane graph G divides the plane into connected regions called $faces$. A bounded region is called an *inner face* and the unbounded region is called the *outer face*. For a face f in G we denote by $V(f)$ the set of vertices of G on the boundary of f.

A plane graph is an *outerplanar graph* if all its vertices lie on the outer face. Let G be an outerplanar graph. We now define the *dual tree* of G. The dual tree T of G is a tree whose vertices correspond to the inner faces of G, and two vertices x and y of T are adjacent if the faces of G corresponding to x and y share an edge in G. A *maximal outerplanar graph* is an outerplanar graph to

which no edge can be added without losing outerplanarity. Clearly, each inner face of a maximal outerplanar graph has three edges. It is easy to see that any outerplanar graph can be augmented in linear time to a maximal outerplanar graph by adding only a linear number of extra edges. A vertex of the dual tree of a maximal outerplanar graph G has degree at most three, hence the dual tree T of G is a binary tree.

We now define a vertex labeling of a rooted binary tree. Let T be a binary tree and let r be the root of T. Then the *labeling of a vertex u of T with respect to r*, which we denote by $L_r(u)$, is defined as follows:

(a) if u is a leaf node then $L_r(u) = 1$;

(b) if u has only one child q and $L_r(q) = k$ then $L_r(u) = k$;

(c) if u has two children s and t, and $L_r(s) = k$ and $L_r(t)=k'$ where $k > k'$, then $L_r(u) = k$; and

(d) if u has two children s and t, and $L_r(s) = k$ and $L_r(t)=k$, then $L_r(u) = k + 1$.

Note that for a rooted binary tree, the labeling of a vertex with respect to the root is unique. We denote by $L_r(T_r)$ the labeling of all the vertices of T_r with respect to r. We also denote by T_x^r the subtree of T_r rooted at x and by $L_r(T_x^r)$ the labeling of all the vertices of T_x^r with respect to r. Figure 2 illustrates the vertex labeling of a binary tree T rooted at r where an integer value represents the label of the associated vertex. The following lemma is immediate from the

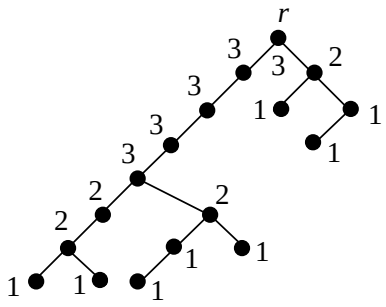

Figure 2: Vertex labeling of a binary tree T.

labeling defined above.

Lemma 1 *Let T be a binary tree and let r be the root of T. Let u and v be two vertices of T such that u is an ancestor of v. Then $L_r(u) \geq L_r(v)$.*

The following lemma gives an upper bound on the value of the vertex labeling.

Lemma 2 *Let T be a binary tree with n vertices and let r be the root of T. Assume that $L_r(r) = k$. Then $k = O(logn)$.*

Proof: We first show that T has at least $2^k - 1$ vertices using an induction on k. The claim is obvious for $k = 1$. Assume that $k \geq 2$ and the result is true for all the trees T' with the root of label k' such that $k' < k$. Let T be a tree with the root r of label k. Let us assume that s is the farthest vertex from r in T such that s has label k. Then each of the left and the right children of s is labeled with $k-1$, and hence by induction hypothesis each of the subtrees of s has at least $2^{k-1} - 1$ vertices. Therefore T has at least $2^k - 1$ vertices. Hence $n \geq 2^k - 1$, i.e., $k = O(logn)$. □

We also have the following lemma.

Lemma 3 *Let T be a binary tree and let r be the root of T. Assume that all the vertices of T are labeled with respect to r using vertex labeling. Let $V(k)$ be a set of vertices such that for all $v \in V(k)$, $L_r(v) = k$. Then any connected component of the subgraph of T induced by $V(k)$ is a path.*

Proof: Let a connected component of the subgraph induced by $V(k)$ in T be $T'(k)$. Assume for a contradiction that $T'(k)$ is not a path. Then a vertex $v \in T'(k)$ has degree three. In such a case, v and the two children of v have the same label in T which is a contradiction to the definition of vertex labeling of T, thus proving the lemma. □

A binary tree T is *ordered* if one child of each vertex v of T is designated as the *left child* and the other is designated as the *right child*. (Note that the left child and/or the right child of a vertex may be empty.) Let T be a rooted ordered binary tree. For any vertex $v \in T$, we call the subtree of T rooted at the left child (if any) of v the *left subtree* of v. Similarly we define the *right subtree* of v. We call an u-v path of T a *left-left path* if u is an ancestor of all the vertices of the path and each vertex of this path, except u, is the left child of its parent. Similarly we define a *right-right path* of T. We call a maximal left-left path of T the *leftmost* path of T if one of the end vertices of the path is the root of T. Similarly we define the *rightmost* path of T. For any vertex $x \in T$, we call a path $x, v_1, ..., v_m$ is the *left-right* path of x if v_1 is the left child of x, and v_{i+1} is the right child of v_i where $1 \leq i \leq m-1$. Similarly we define the *right-left* path of x.

We call $L_r(T_r)$ a *flat labeling* if every path induced by the vertices of T_r with the same label is either a left-left path or a right-right path.

We now have the following fact.

Fact 4 *Let T_r be an ordered binary tree and let r be the root of T_r. Suppose that $L_r(T_r)$ is a flat labeling. Then for any vertex $x \in T_r$, each of the vertices of the left-right (right-left) path of x except the left (right) child of x has a smaller label than the label of x.*

Let x be a vertex of T_r. A *cross path* $x, v_1, ..., v_m$ at x is either a left-right path or a right-left path of x induced by the same label vertices of $L_r(T_r)$. Note that $L_r(T_r)$ is a flat labeling if and only if there is no cross path at any vertex of T_r.

Let G be a maximal outerplanar graph and let T be the dual tree of G. We convert T as a rooted ordered binary tree T_r by fixing its root r and the ordering of the children of each vertex of T, as follows. Let r be a vertex of T such that r corresponds to an inner face f_r of G containing an edge (u, v) on the outerface. (Note that the degree of r is either one or two in T.) We regard r as the root of T. Let w be the vertex of f_r other than u, v such that u, v and w appear in the clockwise order on f_r. We call u and v the *poles of* f_r and w the *central vertex of* f_r. We also call u the *left vertex of* f_r and v the *right vertex of* f_r. The vertex of T corresponding to the face (if any) sharing the vertices v and w with f_r is the *right child of* r and the vertex of T corresponding to the face sharing the vertices u and w (if any) of f_r is the *left child of* r. Let p and q be two vertices of T such that p is the parent of q, and let f_p and f_q be the two faces of G corresponding to p and q in T. Let v_1, v_2 and v_3 be the vertices of f_q in the clockwise order such that v_1 and v_2 are also in f_p. Then v_1 and v_2 are poles of f_q, and v_3 is the central vertex of f_q. The vertex v_1 is the left vertex of f_q and the vertex v_2 is the right vertex of f_q. The vertex of T corresponding to the face sharing the vertices v_2 and v_3 (if any) of f_q is the right child of q and the vertex of T corresponding to the face sharing the vertices v_1 and v_3 (if any) of f_q is the left child of q. Thus we have converted the dual tree T of a maximal outerplanar graph G to a rooted ordered dual tree T_r.

We are now ready to give the definition of "label-constrained outerplanar graphs." Let G be a maximal outerplanar graph and let T be the dual tree of G. We call G a *label-constrained outerplanar graph* if T can be converted to a rooted ordered binary dual tree T_r such that $L_r(T_r)$ is a flat labeling. Figure 3(a) illustrates a label-constrained outerplanar graph G since G is a maximal outerplanar graph and $L_r(T_r)$ is a flat labeling as illustrated in Figure 3(b). $L_p(T_p)$ is not a flat labeling since $L_p(T_p)$ has a cross path at q induced by the vertices with label 2 as illustrated in Figure 3(c).

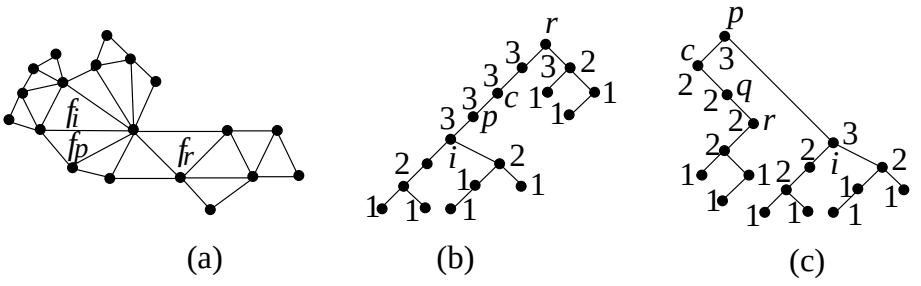

Figure 3: (a) A maximal outerplanar graph G, (b) T_r is a rooted ordered binary dual tree of G where the vertex r of T_r corresponds to the face f_r of G, and (c) T_p is a rooted ordered binary dual tree of G where the vertex p of T_p corresponds to the face f_p of G.

3 Drawing Algorithm

In this section we give a linear-time algorithm for finding a straight-line grid drawing of a label-constrained outerplanar graph with $O(n \log n)$ area.

Let G be a maximal outerplanar graph and let T_r be the rooted ordered binary dual tree of G where r is the root of T_r and f_r is the face of G corresponding to r. In [5], Di Battista and Frati defined a bijection function γ between the vertices of T_r and vertices of G except for the poles of f_r, where each of the vertices of T_r is mapped to the central vertex of the corresponding face of G. We immediately have the following lemma from [5].

Lemma 5 *Let G be a maximal outerplanar graph and let T_r be the rooted ordered binary dual tree of G. Then G contains a copy of T_r which is a spanning tree T' of $G - \{u, v\}$, where u and v are the poles of the face f_r corresponding to the root of T_r.*

Figure 4(b) illustrates the dual tree of G in Figure 4(a) where f_r contains vertices u, v and w in clockwise order. G contains a copy of T_r, which is a spanning tree T' of $G - \{u, v\}$, such that each of the vertices of T_r is mapped to the central vertex of the corresponding face of G as illustrated in Figure 4(c) where the edges of T' are drawn by solid lines.

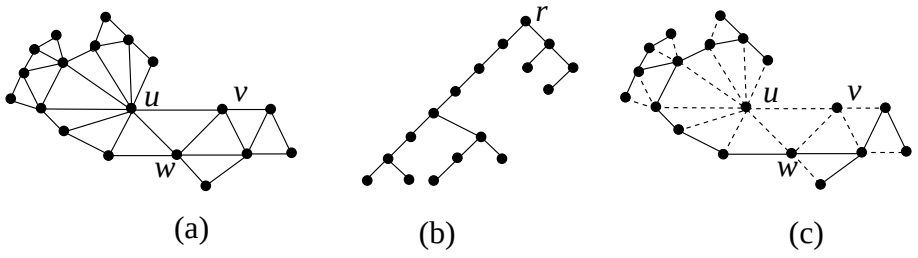

Figure 4: (a) A maximal outerplanar graph G, (b) the rooted ordered dual tree T_r of G, and (c) a spanning-tree T' of $G - \{u, v\}$ is drawn by the solid lines.

Our idea is as follows. We first draw the rooted ordered binary dual tree T_r of a label-constrained outerplanar graph G, and then we put the poles of the face f_r corresponding to the root r of T_r, and add each of the edges of G which are not in the drawing of T_r.

The x-coordinates of the vertices of T_r are assigned in the order of the inorder traversal of T_r in increasing order starting from 1. The y-coordinate of a vertex of T_r is the label of the vertex minus one. We now add the required edges to complete the drawing of T_r. Figure 1(c) illustrates a straight-line grid drawing of T_r in Figure 1(b).

We now put the poles of the face of G corresponding to the root of T_r. The left vertex and the right vertex of the face corresponding to the root of T_r are put at $(0, k)$ and at $(n - 1, k)$, respectively, where k is the label of the root of T_r. We now add each of the edges of G which are not in the drawing of

T_r using straight-line segments, and thus we complete the drawing of G. We call the algorithm described above for drawing an outerplanar graph Algorithm **Draw-Graph**. We now have the following theorem.

Theorem 1 *Let G be a label-constrained outerplanar graph. Then Algorithm* **Draw-Graph** *finds a straight-line grid drawing of G with $O(n \log n)$ area in linear time.*

In the rest of this section, we give a proof of Theorem 1. We first show that Algorithm **Draw-Graph** produces a straight-line drawing of G. The following lemmas are immediate from the assignment of x-coordinates and y-coordinates of the vertices of T_r.

Lemma 6 *Let G be a label-constrained outerplanar graph and let T_r be a rooted ordered binary dual tree of G. Let u be a vertex of T_r, and let s and t be the left and the right child of u, respectively. Then the x-coordinate of any vertex of the subtree rooted at s is less than the x-coordinate of any vertex of the subtree rooted at t.*

Lemma 7 *Let G be a label-constrained outerplanar graph and let T_r be a rooted ordered binary dual tree of G. Let u and v be vertices of T_r where u is an ancestor of v. Then the y-coordinate of u is greater than or equal to the y-coordinate of v.*

We also have the following lemma.

Lemma 8 *Let G be a maximal outerplanar graph and let T_r be a rooted ordered binary dual tree of G. Let q be any vertex of T_r, and let x and y be the left and the right child of q, respectively. Let f_q, f_x and f_y be the faces of G corresponds to the vertices q, x and y in T_r. Then the left vertex of f_q is the left vertex of f_x and the right vertex of f_q is the right vertex of f_y.*

Proof: Immediate from the definition of the left and the right vertex of a face in G. □

By Lemmas 6 and 7, the drawing of T_r is a straight-line grid drawing. We now show that each of the edges of G that are not in T_r can be drawn using straight-line segments without any edge crossings. An edge between the two pole vertices of the face corresponding to the root of T_r can be drawn using a straight-line segment without any crossings with the existing drawing of T_r since each of the pole vertices is placed above all the vertices of T_r. By Lemma 8, the left vertex of the face corresponding to the root of T_r is the left vertex of the faces corresponding to the vertices of the leftmost path of T_r. Therefore the left vertex of the face corresponding to the root of T_r is adjacent to all the vertices of the leftmost path of T_r in G. These edges can be drawn using straight-line segments without any edge crossings since the left vertex of the face corresponding to the root of T_r is placed strictly to the left and above of all the vertices on the leftmost path of T_r. Similarly, we can draw the edges between the

right vertex of the face corresponding to the root of T_r and the vertices on the rightmost path of T_r using straight-line segments without any edge crossings. In a maximal outerplanar graph, the rest of the edges are between any vertex $v \in T_r$ and a vertex on the left-right or the right-left path of v. We now show that we can draw these edges with straight-line segments without any edge crossings. From the x-coordinate of any vertex $v \in T_r$ and by Fact 4, one can see that a vertex $v \in T_r$ is placed strictly to the left (right) and above of all the vertices of the right-left (left-right) path of v except the right (left) child of v. We first consider that we are adding the edges from v to the vertices on the right-left path of v in T_r other than the right child of v. (Note that the edge between v and its right child is an edge of T_r.) From the x-coordinate and the y-coordinate assignment, we can see that the edges among the vertices on the right-left path of v other than the right child of v forms a polyline in the drawing which is both x-monotone and y-monotone. Hence each of the vertices on this polyline is visible from v. Similarly, the vertices on the left-right path of v other than the left child of v are visible from v. Thus all such edges can be drawn using straight-line segments without any edge crossings and hence the following lemma holds.

Lemma 9 *Let G be a label-constrained outerplanar graph. Then Algorithm* **Draw-Graph** *finds a straight-line grid drawing of G.*

Figure 1(d) illustrates the straight-line grid drawing of G in Figure 1(a). We are now ready to give a proof of Theorem 1.

Proof of Theorem 1: By Lemma 9, the drawing of G is a straight-line grid drawing. The height of the drawing of G is the label of the root of the dual tree of G. By Lemma 2, the height of the drawing of G is $O(\log n)$. The width of the drawing is $O(n)$. Therefore the area of the drawing is $O(n \log n)$. One can easily see that the drawing of G can be found in linear time. □

4 Recognition Algorithm

In this section we give a linear-time algorithm for recognition of a label-constrained outerplanar graph.

Let G be a maximal outerplanar graph and let T_r be a rooted ordered binary dual tree of G taking a vertex r of degree 1 or 2 as the root. (Note that r corresponds to an inner face f_r of G having an edge on the outer face.) From the definition of the vertex labeling of a binary tree in Section 2, one can easily see that $L_r(T_r)$ can be computed by a bottom-up traversal of T_r in linear time. The verification whether $L_r(T_r)$ is a flat labeling can also be done in linear time. In case $L_r(T_r)$ is not a flat labeling, $L_p(T_p)$ might be a flat labeling where T_p is a rooted ordered binary dual tree of G rooted at a vertex p of degree one or two other than r. Therefore, in order to examine whether G is a label-constrained outerplanar graph or not, we have to compute the vertex labeling $L_p(T_p)$ of the ordered binary dual tree T_p of G rooted at each vertex p with degree one or degree two in the dual tree T of G and verify whether $L_p(T_p)$ is a flat labeling

or not. Hence, the recognition of a label-constrained outerplanar graph takes $O(n^2)$ time by a naive approach. In the rest of this section we show that the recognition can be done in linear time.

Before presenting our recognition algorithm, we present the following observation. Figures 3(b) and 3(c) illustrate the rooted ordered binary trees T_r and T_p of a maximal outerplanar graph G where r and p correspond to the faces f_r and f_p of G, respectively. Note that the vertex i is the left child of p in T_r in Figure 3(b), but it is the right child of p in T_p as illustrated in Figure 3(c). Again c is the parent of p in T_r but c is a child of p in T_p. Thus we can not get the rooted ordered binary dual tree T_p of G immediately from T_r by simply choosing the vertex p of T_r as the new root without taking care about the ordering of the children of each vertex as well as the parent-child relationship between pairs of vertices. However from a close observation, one can see that the ordering of the children is changed only for the vertices on the r-p path of T_r. Furthermore, the parent-child relationship between a pair of vertices is also changed only for the vertices on the r-p path of T_r. For the rest of the vertices of T_r, both the ordering of the children of a vertex and the parent-child relationship between a pair of vertices remain unchanged in T_p. Let x and y be a pair of vertices on the r-p path of T_r such that x is the parent of y in T_r. Then y is the parent of x in T_p. The change in the ordering of the children of a vertex on the r-p path of T_r can be described by the following three lemmas.

Lemma 10 *Let G be a maximal outerplanar graph and let T_r be a rooted ordered binary dual tree of G. Let s be the right (left) child of r in T_r. Let p be a vertex of T other than r such that the degree of p is one or two, and the vertex s is a child of r in T_p. Then s is the left (right) child of r in T_p. (See Figures 5 (a) and (b).)*

Lemma 11 *Let G be a maximal outerplanar graph and let T_r be a rooted ordered binary dual tree of G. Let s be a vertex of T_r other than r such that the degree of s is one or two, and let t be the right (left) child of s in T_r. Then t is the left (right) child of s in T_s. (See Figures 5 (c) and (d).)*

Lemma 12 *Let G be a maximal outerplanar graph and let T_r be a rooted ordered binary dual tree of G. Let x be a degree three vertex such that s is the parent of x, p is the left child of x and q is the right child of x in T_r. Then the following two conditions hold.*

(i) Let y be a descendant of x in the left subtree of x in T_r such that the degree of y is one or two. Then q is the left child of x and s is the right child of x in T_y. (See Figures 5 (e) and (f).)

(ii) Let z be a descendant of x in the right subtree of x in T_r such that the degree of z is one or two. Then s is the left child of x and p is the right child of x in T_z. (See Figures 5 (g) and (h).)

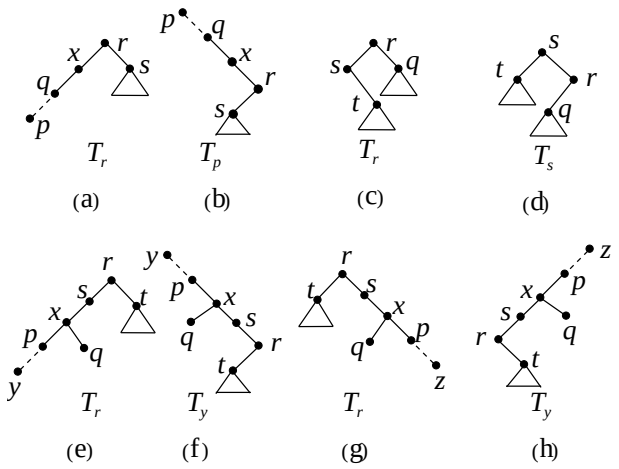

Figure 5: Illustration for Lemmas 10, 11 and 12.

The proofs of Lemmas 10, 11 and 12 are immediate from the definition of the ordering of the children of a vertex in the rooted ordered binary dual tree.

Let x be a vertex of T_r. Let p be the parent of x, u be the left child of x and v be the right child of x. Since the degree of x is at most three, there are at most three connected components in $T - x$. We call the connected component containing u the *left subtree* of x. Similarly the connected component that contains v is called the *right subtree* of x and the connected component that contains p is called the *ancestor subtree* of x. We call any of the left subtree, the right subtree and the ancestor subtree of x a *subtree* of x. The *rooted left subtree* of x is the rooted tree obtained by taking u as the root of the left subtree of x. Similarly we define the *rooted right subtree* and the *rooted ancestor subtree* of x to be the rooted trees obtained by taking v as the root of the right subtree of x and by taking p as the root of the ancestor subtree of x, respectively. We now have the following lemma whose proof is immediate from the fact that labeling is done in a bottom-up approach.

Lemma 13 *Let T_r be a rooted binary tree and let x be a vertex of T_r. Then for any two vertices y and z with degree one or two in the same subtree of x, $L_y(x) = L_z(x)$.*

Thus although there are $O(n)$ possible rooted trees obtained from a binary tree, each vertex of a tree can have at most three different labels. Let x be a vertex in a rooted tree T_r. Then for any vertex y in the left subtree of x, the value of $L_y(x)$ is called the *left-label* of x in T_r. Similarly we define the *right-label* of x in T_r and *ancestor-label* of x in T_r. In the recognition algorithm, we first compute these three labels for each vertex of T_r by two linear-time traversals of T_r. We then verify whether any of these three labels induces any cross path at each vertex of T_r. Finally, we verify for each vertex x whether $L_x(T_x)$ is a

flat-labeling or not by two more linear-time traversals of T_r. The detail of the recognition algorithm is given below.

Let T_r be a rooted tree and let x be a vertex of T_r. Let p be a vertex with degree one or two in T_r. Then by Lemma 13, the value of $L_p(x)$ is equal to the left-label of x if p is in the left subtree of x; $L_P(x)$ is equal to the right-label of x if p is in the right subtree of x; otherwise $L_p(x)$ is equal to the ancestor-label of x. From this observation, we can compute all the three labels of each vertex of T_r by two linear-time traversals of T_r as described in the following lemma.

Lemma 14 *Let T_r be a rooted ordered binary tree. Then one can compute the left-label, the right-label and the ancestor-label of each vertex in T_r in linear time.*

Proof: We compute the three labels of each vertex in T_r by two linear-time traversals of T_r. We first compute the ancestor-label of each vertex in T_r by a bottom-up traversal of T_r. We next compute the left-label and the right-label of each vertex by a top-down traversal of T_r as follows.

Assume that we are traversing a vertex x during the top-down traversal of T_r. Let p be the parent of x in T_r, u be the left child of x in T_r and v be the right child of x in T_r. Let y be any vertex in the left subtree of x. Then by Lemmas 10, 11 and 12, v becomes the left child and p becomes the right child of x in T_y. Then the left-label of x is computed from the ancestor-label of v and the left-label of p in T_r if x is the left child of p in T_r. On the other hand the left-label of x is computed from the ancestor-label of v and the right-label of p if x is the right child of p. Again if z is any vertex in the right subtree of x, then by Lemmas 10, 11 and 12, p is the left child of x and u is the right child of x in T_z. Then we can compute the right-label of x in the similar way as the computation of its left-label. Clearly each of the bottom-up and the top-down traversals of T_r takes linear time. □

Let x be a vertex in a rooted tree T_r. Lemma 13 implies that for any two vertices y and z in the same subtree of x, $L_y(x) = L_z(x)$. We now have the following lemma that gives a similar result on cross paths induced at x. The proof of this lemma is immediate from the fact that labeling is done in a bottom-up approach.

Lemma 15 *Let T_r be a rooted ordered binary dual tree and let x be a vertex of T_r. Then for any two vertices y and z with degree one or two in the same subtree of x, $L_y(x)$ induces a cross path at x if and only if $L_z(x)$ induces a cross path at x.*

Let x be a vertex in T_r. We say that the *left-label of x induces a cross path* at x if $L_y(x)$ induces a cross path at x where y is a vertex in the left subtree of x with degree one or two. Similarly the *right-label of x induces a cross path* at x if $L_y(x)$ induces a cross path at x where y is a vertex in the right subtree of x with degree one or two. Again the *ancestor-label of x induces a cross path* at x if $L_y(x)$ induces a cross path at x where y is a vertex in the ancestor subtree of x with degree one or two. We now have the following lemma.

Lemma 16 *Let T_r be a rooted ordered binary dual tree. Assume that the left-label, the right-label and the ancestor-label of each vertex of T_r has been computed. Then for any vertex x of T_r, one can verify in constant time whether each of the left-label, the right-label and the ancestor-label of x induces any cross path at x.*

Proof: Let p be the parent of x, u be the left child of x and v be the right child of x in T_r. Assume that u_l, u_r are the left and the right child of u, respectively; and v_l, v_r are the left and the right child of v, respectively. Also assume that q is the parent of p and s is the other child of p. Then the ancestor-label of x induces a left-right path at x if the ancestor-label of x is equal to the ancestor-label of u and the ancestor-label of u is equal to the ancestor-label of u_r. Similarly the ancestor-label of x induces a right-left path at x if the ancestor-label of x is equal to the ancestor-label of v and the ancestor-label of v is equal to the ancestor-label of v_l. The ancestor-label of x induces a cross path at x if it induces either a left-right path or a right-left path at x.

We now show that we can also verify whether the left-label and the right-label of x induce any cross path at x in constant time. We have the following four cases to consider.

Case 1: p is the left child of q and x is the left child of p.

Let y be any vertex of degree one or two in the left subtree of x in T_r. Then by Lemmas 10, 11 and 12, v is the left child and p is the right child of x in T_y as illustrated in Figure 6(a)–(b). Furthermore, v_r is the right child of v and s is the left child of p in T_y. Thus the left-label of x induces a left-right path if the left-label of x is equal to the ancestor-label of v and the ancestor-label of v is equal to the ancestor-label of v_r. Similarly the left-label of x induces a right-left path if the left-label of x is equal to the left-label of p and the left-label of p is equal to the ancestor-label of s. Thus, the left-label of x induces a cross path if it induces either a left-right path or a right-left path at x.

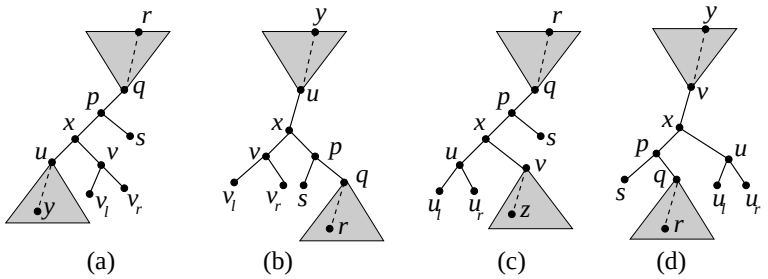

Figure 6: Illustration for the proof of Lemma 16.

Similarly if z is any vertex of degree one or two in the right subtree of x in T_r. Then by Lemmas 10, 11 and 12, p is the left child and u is the right child of x in T_z as illustrated in Figure 6(c)–(d). Furthermore, q is the right child of p and u_l is the left child of u in T_z. Thus the right-label of x induces a left-right path if the right-label of x is equal to the left-label of p and the left-label of p is

equal to the left-label of q. Similarly the right-label of x induces a right-left path if the right-label of x is equal to the ancestor-label of u and the ancestor-label of u is equal to the ancestor-label of u_l. Thus, the right-label of x induces a cross path if it induces either a left-right path or a right-left path at x.

Case 2: p is the left child of q and x is the right child of p.

Let y be any vertex in the left subtree of x and let z be any vertex in the right subtree of x in T_r. Then one can find the ordering of the children of x and the ordering of the children of the children of x in T_y and T_z by Lemmas 10, 11 and 12 in a similar way as in Case 1.

The left-label of x induces a left-right path if the left-label of x is equal to the ancestor-label of v and the ancestor-label of v is equal to the ancestor-label of v_r. Similarly the left-label of x induces a right-left path if the left-label of x is equal to the right-label of p and the right-label of p is equal to the left-label of q. The left-label of x induces a cross path if it induces either a left-right path or a right-left path.

The right-label of x induces a left-right path if the right-label of x is equal to the right-label of p and the right-label of p is equal to the ancestor-label of s. Similarly the right-label of x is equal to the ancestor-label of u and the ancestor-label of u is equal to the ancestor-label of u_l. The right-label of x induces a cross path if it induces either a left-right path or a right-left path.

Case 3: p is the right child of q and x is the left child of p.

In this case, we can verify whether the left-label and the right-label of x induce any cross path at x in a similar way as in Case 1. The only difference is that we need to use the right-label of q instead of the left-label of q (since p is now the right child of q, rather than the left child.)

Case 4: p is the right child of q and x is the right child of p.

In this case, we can verify whether the left-label and the right-label of x induce any cross path at x in a similar way as in Case 2. The only difference is that we need to use the right-label of q instead of the left-label of q (since p is now the right child of q, rather than the left child.) □

Lemma 16 implies that once we compute the left-label, the right-label and the ancestor-label of each vertex of a rooted tree T_r, we can verify in linear time whether the left-label, the right-label and the ancestor-label of x induce any cross path at x by a traversal of T_r. Let x be a vertex of T_r. Then we say that the *ancestor-label of x is flat* if the rooted ancestor subtree induces a flat label. Similarly, the *left-label of x is flat* if the rooted left subtree induces a flat label. Again, the *right-label of x is flat* if the rooted right subtree induces a flat label. We now have the following three lemmas, whose proofs are trivial.

Lemma 17 *Let T_r be a rooted ordered binary dual tree and let x be a vertex in T_r. Let u be the left child of x and let v be the right child of x in T_r. Then the ancestor-label of x is flat if the ancestor-label of u is flat, the ancestor-label of v is flat and the ancestor-label of x does not induce any cross path at x.*

Lemma 18 *Let T_r be a rooted ordered binary dual tree and let x be a vertex in T_r. Let p be the parent of x and let v be the right child of x in T_r. Then the left-label of x is flat if one of the following conditions (i)–(ii) hold.*

(i) x is the left child of p, the ancestor-label of v is flat, the left-label of p is flat and the left-label of x does not induce any cross path at x.

(ii) x is the right child of p, the ancestor-label of v is flat, the right-label of p is flat and the left-label of x does not induce any cross path at x.

Lemma 19 *Let T_r be a rooted ordered binary dual tree and let x be a vertex in T_r. Let p be the parent of x and let u be the left child of x in T_r. Then the right-label of x is flat if one of the following conditions (i)–(ii) hold.*

(i) x is the left child of p, the left-label of p is flat, the ancestor-label of u is flat and the right-label of x does not induce any cross path at x.

(ii) x is the right child of p, the right-label of p is flat, the ancestor-label of u is flat and the right-label of x does not induce any cross path at x.

We are now ready to present our linear-algorithm to recognize whether a given outerplanar graph is a label-constrained outerplanar graph or not.

Let G be an outerplanar graph and let T be the dual tree of G. We first take an arbitrary vertex r as the root of T to obtain a rooted tree T_r. We then compute the ancestor-label, the left-label and the right-label of each vertex of T_r by two linear-time traversals of T_r as described in the proof of Lemma 14. We next verify whether the ancestor-label, the left-label and the right-label of each vertex x of T_r induce any cross path at x by another linear-time traversal of T_r as described in Lemma 16. We now verify whether the ancestor-label, the left-label and the right-label of each vertex is flat or not by two more linear-time traversal of T_r as follows. The first of this traversals is a bottom-up traversal of T_r, where we verify for each vertex x of T_r whether the ancestor-label of x is flat or not by Lemma 17. Then in a top-down traversal of T_r, we verify whether the left-label and the right-label of each vertex x of T_r is flat or not by Lemmas 18 and 19. Finally, we recognize whether G is a label-constrained outerplanar graph or not by verifying whether $L_P(T_p)$ is a flat labeling or not for vertex p of degree one or two in T as follows. Since the degree of p is one or two; at least one of the left child, the right child and the parent of p is empty in T_r. If the parent of p is empty then the ancestor-label of p represents the label of p with respect to p. Similarly, the label of p with respect to p is represented by the left-label of p if the left child of p is empty and by the right-label of p if the right child of p is empty. Thus $L_p(T_p)$ is a flat labeling for a vertex p of degree one or two if at least one of the following conditions (i)–(iii) hold.

(i) The parent of p is empty and the ancestor-label of p is flat.

(ii) The left child of p is empty and the left-label of p is flat.

(iii) The right child of p is empty and the right-label of p is flat.

Clearly this takes constant time for each vertex p with degree one or two. Thus the overall complexity of the algorithm is linear. We thus have the following theorem.

Theorem 2 *Let G be an outerplanar graph. Then one can verify in linear time whether G is a label-constrained outerplanar graph or not.*

5 Conclusion

In this paper we introduced a subclass of outerplanar graphs, which we call label-constrained outerplanar graphs. A graph in this class has a straight-line grid drawing on a grid of $O(n \log n)$ area, and the drawing can be found in linear time. We gave an algorithm to recognize a label-constrained outerplanar graph in linear time. Our drawing algorithm is based on a very simple and natural labeling of a tree. The labeling might also be adopted for solving some other tree-related problems.

Recently Frati [6] showed that the area bound for a straight-line grid drawing of an outerplanar graph G is $O(dn \log n)$, where d is the maximum degree of G. This immediately gives an $O(n \log n)$ area bound for straight-line grid drawing of G if the maximum degree of G is bounded by a constant. But the maximum degree of an outerplanar graph is not always bounded by a constant. A trivial outerplanar graph may have the maximum degree $n-1$ as illustrated in Figure 7(a) (although it requires $O(n)$ area for a straight-line grid drawing as illustrated in Fig. 7(b)). We have introduced a non-trivial subclass of outerplanar graphs, which we call (2^p-1)-graphs for $p \geq 2$, has the maximum degree $d = O(n^{0.5})$. A (2^p-1)-graph G consists of several blocks, called (2^p-1)-blocks.

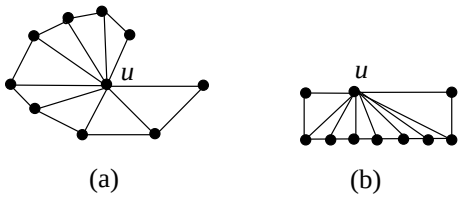

Figure 7: (a) A trivial outerplanar graph G with maximum degree $n-1$, and (b) a straight-line grid drawing of G with $O(n)$ area.

Each such block consists of a path P, and two distinct vertices u and v such that

(i) the path P is v_{-f}, v_{-f+1}, …, v_{-1}, v_0, v_1, …, v_{f-1}, v_f with 2^p-1 vertices, where $f = 2^{p-1}-1$, and

(ii) vertices u and v are adjacent, and u is adjacent to all the vertices v_{-f}, v_{-f+1}, …, v_{-1}, v_0 and v is adjacent to all the vertices v_0, v_1, …, v_{f-1}, v_f.

We call the vertices u and v the *pole vertices*, and the vertices v_{-1}, v_0 and v_1 as the *left, middle and right base vertices* of the block, respectively. The structure of a $(2^p - 1)$-block is illustrated in Fig. 8(a). Figures 8(b) and 8(c) illustrate a (2^p-1)-block for $p = 2$ and $p = 3$, respectively. A (2^p-1)-graph G is constructed

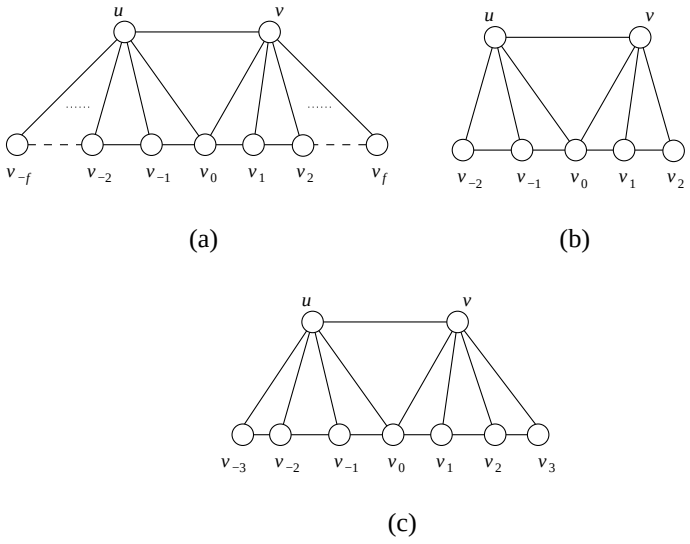

Figure 8: (a) The structure of a $(2^p - 1)$-block, (b)-(c) $(2^p - 1)$-blocks for $p = 2$ and $p = 3$, respectively.

from $2^p - 1$ such blocks as follows. Let us denote these blocks as B_1, B_2, ..., B_{2^p-1}, respectively. For $1 \leq i \leq (2^{p-1} - 1)$, the left and middle base vertices of B_i is identified with the pole vertices of B_{2i}, and the middle and right base vertices of B_i are identified with the pole vertices of B_{2i+1}. Figures 9(a) and 9(b) illustrate the construction of a $(2^p - 1)$-graph for $p = 2$ and $p = 3$, respectively. The number of vertices in G is $n = (2^p - 1)^2 + 2$ and the middle base vertex of each block B_i $(1 \leq i < 2^{p-1})$ has the maximum degree, $d = 2^p + 4$. Therefore, the maximum degree of G is $d = O(n^{0.5})$. A straight-line grid drawing of such a graph by the algorithm of Frati [6] requires $O(n^{1.5} \log n)$ area. However, this subclass of outerplanar graphs is a subclass of label-constrained outerplanar graphs. Hence our algorithm produces an $O(n \log n)$ area drawing for such a graph as illustrated in Fig. 10.

Acknowledgements

We thank the referees for their useful comments which helped us to improve the presentation of the paper.

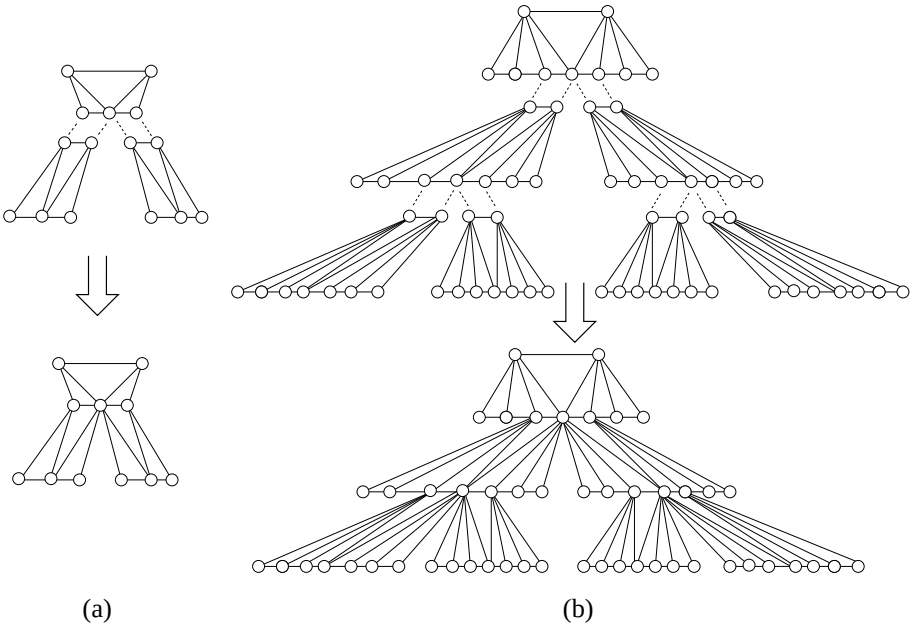

Figure 9: The construction of $(2^p - 1)$-graph for (a) $p = 2$ and (b) $p = 3$.

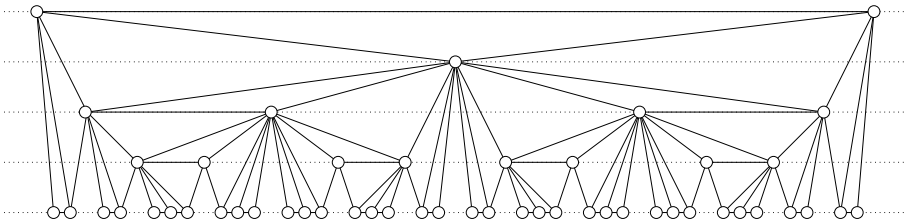

Figure 10: The drawing of a $(2^p - 1)$-graph for $p = 3$ using our algorithm.

References

[1] T. C. Biedl. Drawing outerplanar graphs in $o(n \log n)$ area. In *Proceedings of Graph Drawing 2002*, volume 2528 of *Lect. Notes in Computer Science*, pages 54–65. Springer, 2002.

[2] F. Brandenburg, D. Eppstein, M. Goodrich, S. Kobourov, G. Liotta, and P. Mutzel. Selected open problems in graph drawing. In *Proceedings of Graph Drawing 2003*, volume 2912 of *Lect. Notes in Computer Science*, pages 515–539. Springer, 2004.

[3] H. de Fraysseix, J. Pach, and R. Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10:41–51, 1990.

[4] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice-Hall Inc., Upper Saddle River, New Jersey, 1999.

[5] G. Di Battista and F. Frati. Small area drawings of outerplanar graphs. *Algorithmica*, 54(1):25–53, 2009.

[6] F. Frati. Straight-line drawings of outerplanar graphs in $o(dn \log n)$ area. In *Proceedings of the 19th Canadian Conference on Computational Geometry*, pages 225–228, 2007.

[7] A. Garg and A. Rusu. Area-efficient drawings of outerplanar graphs. In *Proceedings of Graph Drawing 2003*, volume 2912 of *Lect. Notes in Computer Science*, pages 159–165. Springer, 2003.

[8] A. Garg and A. Rusu. A more practical algorithm for drawing binary trees in linear area with arbitrary aspect ratio. In *Proceedings of Graph Drawing 2003*, volume 2912 of *Lect. Notes in Computer Science*, pages 129–134. Springer, 2003.

[9] M. Jünger and P. Mutzel (Eds.). *Graph Drawing Software.* Springer, Berlin, 2004.

[10] M. R. Karim, M. J. Alam, and M. S. Rahman. Straight-line grid drawings of label-constrained outerplanar graphs with $o(n \log n)$ area. In *Proceedings of Workshop on Algorithms and Computation 2009*, volume 5431 of *Lect. Notes in Computer Science*, pages 310–321. Springer, 2009.

[11] M. R. Karim and M. S. Rahman. Straight-line grid drawings of planar graphs with linear area. In *Proceedings of Asia-Pacific Symposium on Visualisation 2007*, pages 109–112. IEEE, 2007.

[12] M. R. Karim and M. S. Rahman. Four-connected spanning subgraphs of doughnut graphs. In *Proceedings of Workshop on Algorithms and Computation 2008*, volume 4921 of *Lect. Notes in Computer Science*, pages 132–143. Springer, 2008.

[13] M. R. Karim and M. S. Rahman. On a class of planar graphs with straight line grid drawings on linear area. *Journal of Graph Algorithms and Applications*, 13(2):153–177, 2009.

[14] M. Kaufmann and D. Wagner (Eds.). *Drawing Graphs: Methods and Models*, volume 2025 of *Lect. Notes in Computer Science*. Springer, Berlin, 2001.

[15] T. Nishizeki and M. S. Rahman. *Planar Graph Drawing*. World Scientific, Singapore, 2004.

[16] W. Schnyder. Embedding planar graphs on the grid. In *Proceedings of First ACM-SIAM Symposium on Discrete Algorithms*, pages 138–148, 1990.