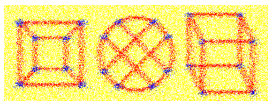

Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 15, no. 2, pp. 177–204 (2011)

Minimum-Area Drawings of Plane 3-Trees

Debajyoti Mondal Rahnuma Islam Nishat Md. Saidur Rahman Muhammad Jawaherul Alam

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

Abstract

A straight-line grid drawing of a plane graph G is a planar drawing of G, where each vertex is drawn at a grid point of an integer grid and each edge is drawn as a straight-line segment. The height, width and area of such a drawing are respectively the height, width and area of the smallest axis-aligned rectangle on the grid which encloses the drawing. A minimum-area drawing of a plane graph G is a straight-line grid drawing of G where the area is the minimum. It is NP-complete to determine whether a plane graph G has a straight-line grid drawing with a given area or not. In this paper we give a polynomial-time algorithm for finding a minimum-area drawing of a plane 3-tree. Furthermore, we show a $\lfloor \frac{2n}{3} - 1 \rfloor \times 2\lceil \frac{n}{3} \rceil$ lower bound for the area of a straight-line grid drawing of a plane 3-tree with $n \geq 6$ vertices, which improves the previously known lower bound $\lfloor \frac{2(n-1)}{3} \rfloor \times \lfloor \frac{2(n-1)}{3} \rfloor$ for plane graphs. We also explore several interesting properties of plane 3-trees.

Keywords. Graph drawing, Minimum area, Minimum layer, Plane 3-tree, Lower bound.

Submitted:	Reviewed:	Revised:	Accepted:
July 2010	October 2010	November 2010	November 2010
	Final: December 2010	Published: February 2011	
Article type: Regular		Communicated by: G. Liotta	

E-mail addresses: debajyoti_mondal_cse@yahoo.com (Debajyoti Mondal) nishat.buet@gmail.com (Rahnuma Islam Nishat) saidurrahman@cse.buet.ac.bd (Md. Saidur Rahman) jawaherul@gmail.com (Muhammad Jawaherul Alam)

1 Introduction

A *plane graph* is a planar graph with fixed planar embedding. In a *straight-line grid drawing* Γ of a plane graph G, each vertex of G is drawn at a grid point of an integer grid and each edge of G is drawn as a straight-line segment. The *area* of Γ is measured by the size of the smallest rectangle with sides parallel to the axes which encloses Γ. The *width* W of Γ is the width of such a rectangle and the *height* H of Γ is the height of such a rectangle. The area is usually described as $W \times H$. A *minimum-area drawing* of a plane graph G is a straight-line grid drawing of G where the area is the minimum. Figure 1(a) depicts a plane graph G and Figure 1(c) depicts a minimum-area drawing of G.

Wagner [28], Fary [18] and Stein [26] independently proved that every planar graph G has a straight-line drawing. A natural question arises: what is the minimum size of a grid required for a straight-line grid drawing? For a given plane graph G with $n \geq 3$ vertices, de Fraysseix *et al.* [12] and Schnyder [25] independently showed that G has a straight-line grid drawing on area $(2n-4) \times (n-2)$ and $(n-2) \times (n-2)$, respectively. Recently, Brandenburg [7] has improved the upper bound of straight-line grid drawing to $\frac{4}{3}n \times \frac{2}{3}n$ area. The problem of finding minimum-area drawings for plane graphs has been shown to be NP-hard by Krug and Wagner [22]. Furthermore, they presented an iterative approach to compactify planar straight-line grid drawings. Frati and Patrignani [20] proved that $2n^2/9 + O(n)$ area is sufficient and $n^2/9 + \Omega(n)$ area is necessary for planar straight-line grid drawings of "nested triangles graphs".

Researchers have also concentrated their attention on minimizing one dimension of the drawing where the other dimension of the drawing is unbounded [1, 10, 15, 19, 27]. Such drawings are known as "layered drawings". A *layered drawing* of a plane graph G is a planar drawing of G, where the vertices are drawn on a set of horizontal lines called *layers* and the edges are drawn as straight line segments. A *minimum-layer drawing* of a plane graph G is a layered drawing of G where the number of layers is the minimum. Figure 1(a) depicts a plane graph G and Figure 1(b) depicts a minimum-layer drawing of G. Chrobak and Nakano [8] gave a linear-time algorithm to obtain a straight-line grid drawing of a plane graph G with n vertices where one dimension of the drawing is bounded by $\lfloor \frac{2n-1}{3} \rfloor$. So, it is obvious that any plane graph G admits a layered drawing on $\lfloor \frac{2n-1}{3} \rfloor$ layers.

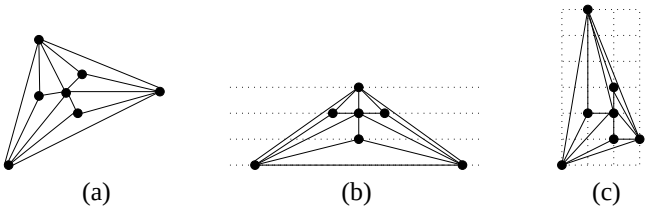

Figure 1: (a) A plane graph G, (b) a minimum-layer drawing of G and (c) a minimum-area drawing of G.

In this paper, we consider the problem of finding minimum-area drawings of a subclass of planar graphs called "plane 3-trees". A *plane 3-tree* G_n with $n \geq 3$ vertices is a plane graph for which the following (a) and (b) hold: (a) G_n is a triangulated plane graph; (b) if $n > 3$, then G_n has a vertex whose deletion gives a plane 3-tree G_{n-1}. Many researchers have shown their interest on plane 3-trees for a long time [2, 4, 14, 16]. In this paper, we explore some interesting properties of plane 3-trees which leads to a polynomial-time algorithm to obtain their minimum-area drawings. We also show that, there exists a plane 3-tree with $n \geq 6$ vertices for which $\lfloor \frac{2n}{3} - 1 \rfloor \times 2\lceil \frac{n}{3} \rceil$ area is necessary for any planar straight-line grid drawing. As a side result, we give an $O(nh_m^4)$ time algorithm to compute a minimum-layer drawing of a plane 3-tree G, where h_m is the minimum number of layers required for any layered drawing of G. Note that, Dujmović *et al.* gave a $f(h) \times n$ time algorithm that can decide whether a given graph G with n vertices admits a planar drawing in h layers or not [15]. The running time of their algorithm is dominated by the cost of finding a "path decomposition" of G. To the best of our knowledge, the algorithm currently known to obtain a "path decomposition" of a graph with "treewidth" $\leq l$, takes at least $\Omega(n^{4l+3})$ time [5]. Clearly, one can obtain minimum-layer drawings for plane 3-trees using the technique presented in [15] but it takes at least $\Omega(n^{15})$ time, since the "treewidth" of plane 3-trees is three.

An outline of our algorithm to compute a minimum-layer drawing of a plane 3-tree is presented here. Let G_n be a plane 3-tree with n vertices and h be a positive integer. Since any plane graph admits a layered drawing on $\lfloor \frac{2n-1}{3} \rfloor$ layers [8], we test whether G_n can be drawn on h layers or not, by iterating h from 1 to $\lfloor \frac{2n-1}{3} \rfloor$. For each h from 1 to $\lfloor \frac{2n-1}{3} \rfloor$, we use dynamic programming to test whether G_n has a drawing on h layers. We show that any plane 3-tree G_n with $n > 3$ vertices has an inner vertex p which is the common neighbor of all the three outer vertices of G_n. The vertex p, along with the three outer vertices of G_n, divides the interior region of G_n into three new regions. We prove that the subgraphs enclosed by those three regions are also plane 3-trees. For each feasible y-coordinate assignment of the outer vertices of G_n, these subgraphs are the three subproblems of our testing problem. We define the result of the testing problem in terms of the test results of the subproblems. Figure 2(a) depicts a plane 3-tree G where p is the common neighbor of the three outer vertices a, b, c of G. Figures 2(b) and (c) show the subproblems of the input graph G for two different placements of p. We divide and test the subproblems recursively and store the test results of the subproblems in a table to compute the minimum number of layers h_m among all the possible layered drawings of G. Figure 2(d) illustrates that G does not admit a layered drawing for the layer assignment of the vertex p as in Figure 2(b). Figure 2(e) is the drawing of G corresponding to the drawings of the subproblems illustrated in Figure 2(c). We can obtain a minimum-area drawing of G in a similar method.

The rest of the paper is organized as follows. Section 2 describes some definitions and presents preliminary results. Section 3 introduces some interesting properties of plane 3-trees. Section 4 presents an $O(nh_m^4)$ time algorithm to

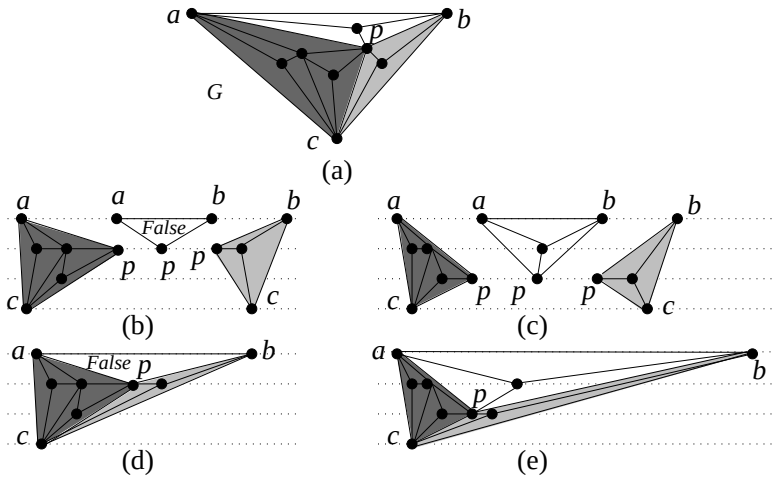

Figure 2: Illustration of the algorithm for minimum-layer drawings.

compute a minimum-layer drawing of a plane 3-tree G_n with n vertices where h_m is the minimum number of layers required for any layered drawing of G. Section 5 illustrates an $O(n^9 \log n)$ time algorithm to obtain a minimum-area drawing of G_n. Section 6 gives a lower bound on the area requirements for straight-line drawings of plane 3-trees. Finally, Section 7 concludes with discussions suggesting future works. An early version of this paper has been presented at [23].

2 Preliminaries

In this section we give some relevant definitions that will be used throughout the paper and present some preliminary results.

Let $G = (V, E)$ be a connected simple graph with vertex set V and edge set E. The *degree* of a vertex v is the number of neighbors of v in G. We denote by *degree(v)* the degree of the vertex v. A *subgraph* of a graph $G = (V, E)$ is a graph $G' = (V', E')$ such that $V' \subseteq V$ and $E' \subseteq E$. If G' contains all the edges of G that join vertices in V', then G' is called the *subgraph induced by* V'. A graph G is *connected* if for any two distinct vertices u and v there is a path between u and v in G. A graph which is not connected is called a *disconnected* graph. The *connectivity* $\kappa(G)$ of a graph G is the minimum number of vertices whose removal results in a disconnected graph or a single-vertex graph. We say that G is *k-connected* if $\kappa(G) \geq k$. We call a set of vertices in a connected graph G a *separator* or a *vertex-cut* if the removal of the vertices in the set results in a disconnected or single-vertex graph.

A *tree* is a connected graph without any cycle. A *rooted tree* T is a tree in which one of the vertices is distinguished from the others. The distinguished vertex is called the *root* of the tree T and every edge of T is directed away from

the root. If v is a vertex in T other than the root, the *parent* of v is the vertex u such that there is a directed edge from u to v. When u is the parent of v, v is called a *child* of u. A vertex in T, which has no children, is called a $leaf$. Any vertex which is not a leaf in T is an *internal* vertex. A *descendant* of u is a vertex v other than u such that there is a directed path from u to v. Let i be any vertex of T. Then we define a *subtree* $T(i)$ *rooted at* i as a subgraph of T induced by vertex i and all the descendants of i. An *ordered rooted tree* is a rooted tree where the children of any vertex are ordered counter-clockwise.

A graph is *planar* if it can be embedded in the plane without edge crossing 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 the *faces*. The unbounded region is called the *outer face* and all the other faces are called the *inner faces*. The vertices on the outer face are called the *outer vertices* and all the other vertices are called the *inner vertices*. If all the faces of a plane graph G are triangles, then G is called a *triangulated plane graph*. For a cycle C in a plane graph G, we denote by $G(C)$ the plane subgraph of G inside C (including C). A plane graph G with $n \geq 3$ vertices is called a *plane* 3*-tree* if the following (a) and (b) hold:

(a) G is a triangulated plane graph;

(b) if $n > 3$, then G has a vertex x whose deletion gives a plane 3-tree G' of $n-1$ vertices.

Note that, vertex x may be an inner vertex or an outer vertex of G. We denote a plane 3-tree of n vertices by G_n. Examples of plane 3-trees are shown in Figure 3; G_6 is obtained from G_7 by removing the inner vertex c of degree three. Then G_5 is obtained from G_6 by deleting the inner vertex b of degree three. G_4 is obtained from G_5 by deleting the outer vertex g of degree three and G_3 is obtained in a similar way.

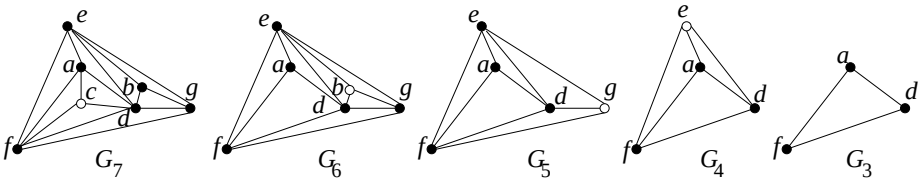

Figure 3: Examples of plane 3-trees.

3 Properties of Plane 3-Trees

In this section we introduce some properties of plane 3-trees. The following results are known on plane 3-trees.

Lemma 1 [4] *Let G_n be a plane 3-tree with n vertices where $n > 3$. Then the following (a) and (b) hold. (a) G_n has an inner vertex x of degree three such that the removal of x gives the plane 3-tree G_{n-1}. (b) G_n has exactly one inner vertex y such that y is the neighbor of all the three outer vertices of G_n.*

By Lemma 1(b) for any plane 3-tree G_n, $n > 3$, there is exactly one inner vertex y which is the common neighbor of all the outer vertices of G_n. We call vertex y the *representative vertex* of G_n.

A *separating triangle* of a triangulated plane graph G is a triangle in G whose interior and exterior contain at least one vertex each. Let G_n be a plane 3-tree and C be a triangle in G_n, then we prove that $G_n(C)$ is also a plane 3-tree as in the following lemma.

Lemma 2 *Let G_n be a plane 3-tree with $n > 3$ vertices and C be any triangle of G_n. Then the subgraph $G_n(C)$ is a plane 3-tree.*

We use the following two facts to prove Lemma 2.

Fact 3 [17] *Any triangulated graph with more than three vertices is a triconnected graph.*

Fact 4 *Let G_n be a triangulated plane graph and C be a separating triangle of G_n where $n > 3$. Then each of the three vertices on C must have degree at least four in G_n.*

Proof. Since G_n and $G_n(C)$ are triangulated and $n > 3$, they are triconnected by Fact 3. Therefore each of the three vertices on C has degree at least three in $G_n(C)$. Suppose for a contradiction that at least one of the vertices w on C has degree exactly three in G_n as illustrated in Figure 4. Since $G_n(C)$ is triconnected with more than three vertices, two of the neighbors of w are on C and the other neighbor is inside C. Since w has no neighbor outside C, we can disconnect the exterior vertices of C from the interior vertices of C by deleting the other two vertices on C except w. This implies that G_n has a vertex-cut of two vertices, and hence G_n would not be triconnected, a contradiction. □

We are now ready to give a proof of Lemma 2.

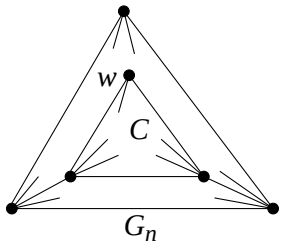

Figure 4: Illustration for the proof of Fact 4.

Proof of Lemma 2. The proof is trivial for the case when the triangle C is

not a separating triangle. If C is the outer face of G, then $G_n(C)$ is itself a plane 3-tree; otherwise C is a triangle whose interior contains no vertex and $G_n(C)$ is a plane 3-tree by definition. We now consider the case when C is a separating triangle in G. Since G_n is triangulated, $G_n(C)$ is triangulated. Then, it is sufficient to prove that we can delete inner vertices of degree three recursively from $G_n(C)$ to obtain the cycle C. By Lemma 1, G_n has an inner vertex of degree three whose deletion gives a plane 3-tree G_{n-1}. We delete such inner vertices of G_n recursively which are outside of $G_n(C)$. Assume that after deleting k vertices we have no inner vertex of degree three outside $G_n(C)$ and let the resulting plane 3-tree be G_{n-k}. As we never deleted the outer vertices of G_n and the inner vertices of $G_n(C)$, C is also a separating triangle of G_{n-k}. There must be an inner vertex of degree three in G_{n-k} by Lemma 1. That vertex must be an inner vertex of $G_{n-k}(C)$ since each of the three outer vertices of $G_{n-k}(C)$ has degree at least four in G_{n-k} by Fact 4. We now delete all the inner vertices of degree three of $G_{n-k}(C)$ recursively in such a way that at each deletion the resulting graph remains a plane 3-tree. By definition after deleting m such inner vertices of $G_{n-k}(C)$ recursively we get a plane 3-tree G_{n-k-m}. Suppose for a contradiction that there is no inner vertex of degree three in $G_{n-k-m}(C)$. We first consider the case when C has no interior vertex which implies that we have recursively deleted the inner vertices of degree three of $G_n(C)$ to get the triangle C and $G_n(C)$ is certainly a plane 3-tree. We next consider the case where C still contains at least one interior vertex. Then $G_{n-k-m}(C)$ has more than three vertices and there is no inner vertex of degree three in G_{n-k-m}. Hence G_{n-k-m} would not be a plane 3-tree by Lemma 1(a), a contradiction. □

Let p be the representative vertex and a, b, c be the outer vertices of G_n. The vertex p, along with the three outer vertices a, b and c, form three triangles $\{a, b, p\}$, $\{b, c, p\}$ and $\{c, a, p\}$ as illustrated in Figure 5. We call those three triangles the *nested triangles around* p.

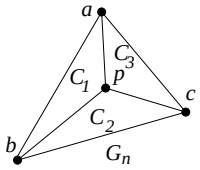

Figure 5: Nested triangles around p.

We now define the *representative tree* of G_n as an ordered rooted tree T_{n-3} satisfying the following two conditions (a) and (b).

(a) if $n = 3$, T_{n-3} consists of a single vertex.

(b) if $n > 3$, then the root p of T_{n-3} is the representative vertex of G_n and the subtrees rooted at the three counter-clockwise ordered children q_1, q_2 and q_3 of p in T_{n-3} are the representative trees of $G_n(C_1)$, $G_n(C_2)$ and $G_n(C_3)$, respectively, where C_1, C_2 and C_3 are the three nested triangles around p in counter-clockwise order.

Figure 6 illustrates the representative tree T_{n-3} of the plane 3-tree G_n. Note that the "4-block trees" [21] and the tree of the "tree decomposition" [5] are quite similar to the representative trees for the plane 3-trees.

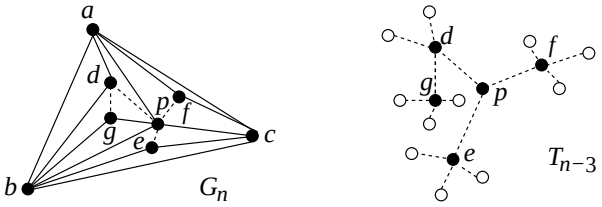

Figure 6: Representative tree T_{n-3} of G_n.

We now prove that T_{n-3} is unique for G_n in the following lemma.

Lemma 5 *Let G_n be any plane 3-tree with $n \geq 3$ vertices. Then G_n has a unique representative tree T_{n-3} with exactly $n-3$ internal vertices and $2n-5$ leaves.*

Proof. The case $n = 3$ is trivial since the representative tree of G_3 is a single vertex. We may thus assume that G has four or more vertices. By Lemma 1(b) G_n has exactly one representative vertex. Let p be that representative vertex of G_n and C_1, C_2, C_3 be the three nested triangles around p. By Lemma 2, $G_n(C_1)$, $G_n(C_2)$ and $G_n(C_3)$ are plane 3-trees. Let n_1, n_2 and n_3 be the number of vertices in $G_n(C_1)$, $G_n(C_2)$ and $G_n(C_3)$, respectively. Then by the induction hypothesis, T_{n_1-3}, T_{n_2-3} and T_{n_3-3} are the unique representative trees of $G_n(C_1)$, $G_n(C_2)$ and $G_n(C_3)$, respectively. We now assign p as the parent of q_1, q_2 and q_3, where q_1, q_2 and q_3 are the roots of T_{n_1-3}, T_{n_2-3} and T_{n_3-3}, respectively. Since p is the unique representative vertex of G_n, the choice for the root of T_{n-3} is unique. Since G_n has n vertices and any inner vertex of G_n except p belongs to exactly one of $G_n(C_1)$, $G_n(C_2)$ and $G_n(C_3)$, the total number of vertices in T_{n_1-3}, T_{n_2-3} and T_{n_3-3} is $n_1-3+n_2-3+n_3-3 = n-4$. Thus the new tree T_{n-3} with root p has $n-4+1 = n-3$ internal vertices. Since T_{n_1-3}, T_{n_2-3} and T_{n_3-3} are ordered trees and q_1, q_2 and q_3 are ordered counter-clockwise around p, T_{n-3} is also an ordered tree. Furthermore one can easily observe that, the leaves represent only the internal faces of G_n. Since the number of internal faces of G_n is $2n-5$ by Euler's Theorem, T_{n-3} has $2n-5$ leaves. □

Now we have the following lemma whose proof is immediate from the definition of the representative tree and from Lemma 5.

Lemma 6 *Let T_{n-3} be the representative tree of a plane 3-tree G_n with $n \geq 3$ vertices and let $T(i)$ be a subtree rooted at a vertex i of T_{n-3}. Then there exists a unique triangle C in G_n such that $T(i)$ is the representative tree of $G_n(C)$.*

By Lemma 6, for any vertex p of T_{n-3}, there is a unique triangle in G_n which we denote as C_p for the rest of this article. Furthermore, if p is the root of T_{n-3},

then C_p is the outer face of G_n; if p is a leaf of T_{n-3}, then C_p is an inner face of G_n and if p is an internal vertex in T_{n-3}, then C_p is a separating triangle in G_n. Let L be the set of leaves in T_{n-3} and let a, b and c be the outer vertices of G_n. Then $T_{n-3} - L$ is a spanning tree of $G_n - \{a, b, c\}$ where each vertex p of $T_{n-3} - L$ is mapped to the representative vertex of $G_n(C_p)$, as illustrated in Figure 6. Thus for the rest of this article, we shall often use an internal vertex p of T_{n-3} and the representative vertex of $G_n(C_p)$ interchangeably. We shall also denote by $T(p)$ the representative tree of $G_n(C_p)$. Figures 7(a) and (b) illustrate $G_n(C_p)$ and its representative tree $T(p)$, respectively.

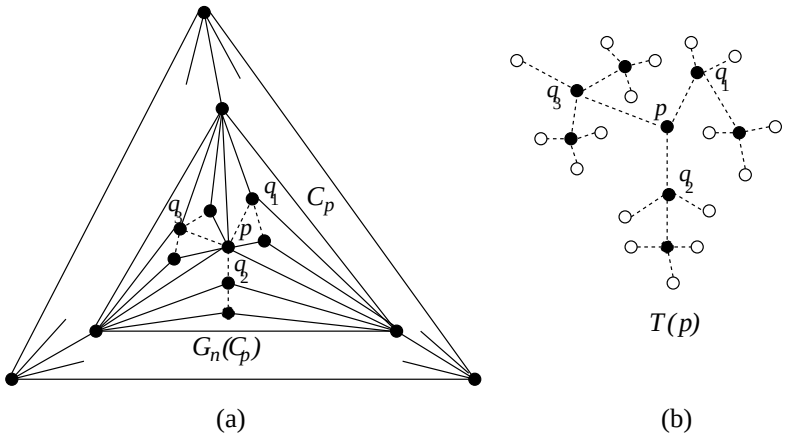

Figure 7: (a) Illustration of $G_n(C_p)$ and (b) the representative tree $T(p)$.

We now have the following lemma.

Lemma 7 *For any plane 3-tree G_n with $n \geq 3$ vertices, the representative tree T_{n-3} of G_n can be found in time $O(n)$.*

Proof. To construct T_{n-3} we first find the representative vertex p of G_n. We keep a list for each inner vertex u of G_n. For each outer vertex v_i of G_n, $i \in \{1, 2, 3\}$, we add v_i in the list of u if u is adjacent to v. One can easily observe that, only the list of the representative vertex p will contain the three outer vertices of G_n. Thus we can find p in time $O(\sum_{i=1}^{3} degree(v_i))$. Let C_{q_1}, C_{q_2}, C_{q_3} be the nested triangles around p. We can find the three children q_1, q_2 and q_3 of p by updating the lists as follows. Since the lists are already updated for all the outer vertices of $G_n(C_{q_1})$, $G_n(C_{q_2})$ and $G_n(C_{q_3})$ except p, we only need to update the lists by adding p to the list of u if u is adjacent to p. Thus the three children of p can be found in time $O(degree(p))$. We then continue updating the lists recursively to find the other vertices of T_{n-3}. Once the lists are updated by a vertex, we do not consider that vertex later to update the lists. The process of updating the lists for each vertex v takes $O(degree(v))$ time and hence the total time of constructing the representative tree is $O(\sum_{v \in V} degree(v)) = O(n)$ since G_n is planar. □

The proof of Lemma 7 leads to a linear-time algorithm to construct the representative tree of a plane 3-tree.

4 Minimum-Layer Drawings

In this section we consider the problem of finding minimum-layer drawings of plane 3-trees.

In a layered drawing of a plane graph G, the vertices are drawn on a set of horizontal lines called layers and the edges are drawn as straight line segments. We assume that the layers are aligned parallel to the x-axes with different y-coordinates and the y-coordinates of the layers are defined as follows. We denote by $y(l)$ the y-coordinate of a layer l. Let $\{l_1, l_2, ..., l_n\}$ be a set of n layers where $y(l_1) < y(l_2) < ... < y(l_n)$, then $y(l_i) = i$, $1 \leq i \leq n$. Thus for the rest of this article, we denote a layer assignment of a vertex v by a y-coordinate assignment of v.

Chrobak *et al.* [8] showed that the upper bound for one dimension of a straight-line grid drawing of any plane graph G with n vertices is $\lfloor \frac{2n-1}{3} \rfloor$. So, it is obvious that any plane 3-tree G admits a layered drawing on $\lfloor \frac{2n-1}{3} \rfloor$ layers. Therefore we assume that, G admits a layered drawing on h layers and iterate h from 1 to $\lfloor \frac{2n-1}{3} \rfloor$. For each iteration, we check whether G is drawable on h layers or not. The first h within which G is drawable is the minimum number of layers h_m required to draw G.

A brute force approach to solve this problem is to assign all possible combinations of y-coordinates to the vertices of G and check whether there is any edge crossing. However, if the total number of vertices is n and the number of layers is h, there are n^h different assignments possible. This exponential time makes the approach impractical for large n and h. We now present a dynamic programming approach to solve the problem. We first give an algorithm **Minimum-Layer** to generate all the feasible y-coordinate assignments of the vertices of G iterating h from 1 to $\lfloor \frac{2n-1}{3} \rfloor$. Then we give an algorithm **Feasibility-Checking** to check whether G admits a layered drawing on h layers for a particular y-coordinate assignment of its outer vertices. For convenience, we describe Algorithm **Feasibility-Checking** before Algorithm **Minimum-Layer**. At the end of this section we give pseudocodes for both of the algorithms. We now formally define the input and the output of the decision problem *Feasibility Checking*.

Input: A plane 3-tree G and y-coordinate assignments of the three outer vertices a, b and c of G.

Output: If G admits a layered drawing with the given y-coordinates of a, b and c, the output is $True$, and $False$ otherwise.

Let T be the representative tree of a plane 3-tree G and v_y be the y-coordinate of any vertex v. For any vertex p of T, we denote by Γ_p a layered drawing of $G(C_p)$ and by $F_p(a_y, b_y, c_y)$ the Feasibility Checking problem of p where a_y, b_y, c_y are the y-coordinates of the three outer vertices a, b, c of $G(C_p)$, respectively. We solve this Feasibility Checking problem using dynamic programming by characterizing the "optimal substructure" and "overlapping

subproblems" properties of the problem which are the two key ingredients for the dynamic programming to be applicable [9]. Characterizing *optimal substructure* means showing that the optimal solution of the problem consists of the optimal solutions of the subproblems. To show the optimal substructure property of the Feasibility Checking problem, we need the following two lemmas.

Lemma 8 *Let G be a plane 3-tree with representative vertex p. Let Γ_p be a layered drawing of G and let $\Gamma(C_p)$ be the layered drawing of C_p in Γ_p. Let $\Gamma'(C_p)$ be another layered drawing of C_p where a, b and c have the same y-coordinates as in $\Gamma(C_p)$. Then G has a layered drawing Γ'_p having $\Gamma'(C_p)$ as the drawing of C_p.*

Proof. The case for $n = 3$ is trivial since for this case Γ'_p coincides with $\Gamma'(C_p)$. We may thus assume that n is greater than three and the claim holds for any plane 3-tree of less than n vertices. Let l be the layer that contains vertex p and let p_y be the y-coordinate of p in Γ_p. The layer l intersects $\Gamma'(C_p)$ at two points (x_1, p_y) and (x_2, p_y), $x_1 \neq x_2$. We place p on l in between x_1 and x_2 to obtain $\Gamma'(C_{q_1})$, $\Gamma'(C_{q_2})$ and $\Gamma'(C_{q_3})$ where C_{q_1}, C_{q_2} and C_{q_3} are the nested triangles around p. By induction hypothesis $G(C_{q_1})$, $G(C_{q_2})$ and $G(C_{q_3})$ admit layered drawings Γ'_{q_1}, Γ'_{q_2} and Γ'_{q_3} which contain the drawings $\Gamma'(C_{q_1})$, $\Gamma'(C_{q_2})$ and $\Gamma'(C_{q_3})$, respectively. Clearly, one can obtain Γ'_p by patching Γ'_{q_1}, Γ'_{q_2} and Γ'_{q_3} inside $\Gamma'(C_{q_1})$, $\Gamma'(C_{q_2})$ and $\Gamma'(C_{q_3})$, respectively, as illustrated in Figure 8. □

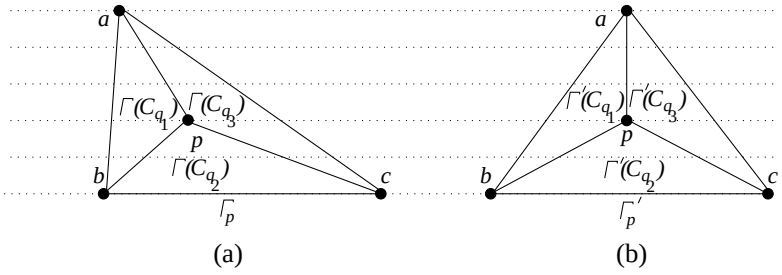

Figure 8: Illustration for the proof of Lemma 8. (a) Layered drawing Γ_p of G and (b) layered drawing Γ'_p of G.

Now we have the following lemma.

Lemma 9 *Let G be a plane 3-tree with the representative tree T. Let p be any internal vertex of T with the three children q_1, q_2, q_3 in T and let a, b, c be the three outer vertices of $G(C_p)$. Then $G(C_p)$ admits a layered drawing Γ_p for the assignment (a_y, b_y, c_y) if and only if Γ_{q_1}, Γ_{q_2} and Γ_{q_3} admit layered drawings for the assignments (a_y, b_y, p_y), (b_y, c_y, p_y) and (c_y, a_y, p_y), respectively, where $min(a_y, b_y, c_y) < p_y < max(a_y, b_y, c_y)$.*

Proof. The necessity is trivial, and proof of the sufficiency can be obtained in a similar technique as described in the proof of Lemma 8. □

We can readily find the "overlapping subproblems" property of the *Feasibility Checking* problem. *Overlapping subproblem* occurs when a recursive algorithm visits the same problem more than once. Figure 9 illustrates this property for the *Feasibility Checking* problem where the overlapping subproblems are shown by dotted rectangles and bold rectangles. We now have the following theorem

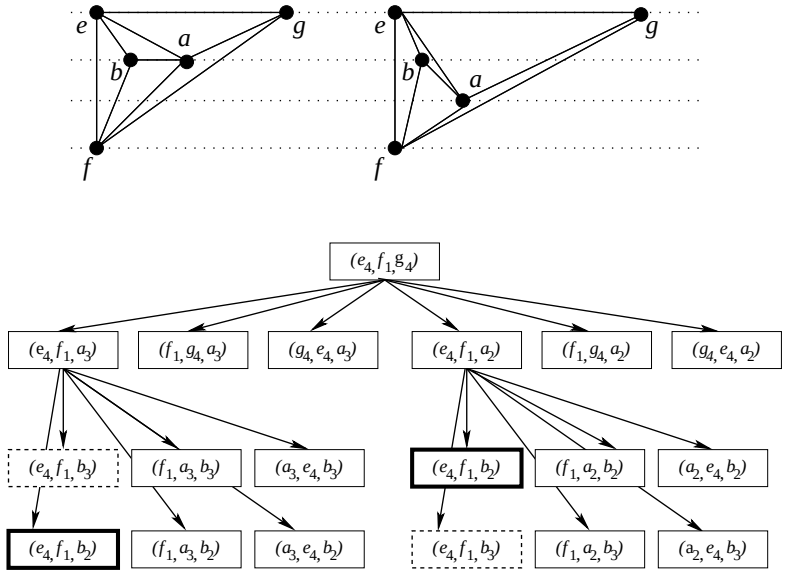

Figure 9: Overlapping Subproblems.

which leads to a recursive solution of the Feasibility Checking problem.

Theorem 4.1 *Let G be a plane 3-tree and let p be any vertex of the representative tree T of G. Let a, b, c be the three outer vertices of $G(C_p)$ and q_1, q_2, q_3 be the three children of p if p is an internal vertex of T. Let $F_p(a_y, b_y, c_y)$ denote the Feasibility Checking problem of p where a_y, b_y, c_y are the y-coordinates of a, b, c. Then $F_p(a_y, b_y, c_y)$ has the following recursive formula.*

$$F_p(a_y, b_y, c_y) = \begin{cases} False & \text{if } \{max\{a_y, b_y, c_y\} - min\{a_y, b_y, c_y\} = 0\}; \\ True & \text{if } \{max\{a_y, b_y, c_y\} - min\{a_y, b_y, c_y\} \geq 1\} \\ & \text{where } p \text{ is a leaf;} \\ False & \text{if } \{max\{a_y, b_y, c_y\} - min\{a_y, b_y, c_y\} \leq 1\} \\ & \text{where } p \text{ is an internal vertex;} \\ \bigvee_{p_y}\{F_{q_1}(a_y, b_y, p_y) \wedge F_{q_2}(b_y, c_y, p_y) \wedge F_{q_3}(c_y, a_y, p_y)\} & \\ \quad \text{where } \{min\{a_y, b_y, c_y\} < p_y < max\{a_y, b_y, c_y\}\}, & \\ \quad \text{otherwise.} & \end{cases}$$

Proof. Consider the case when $max\{a_y, b_y, c_y\} - min\{a_y, b_y, c_y\} = 0$. Then we assign $F_p(a_y, b_y, c_y) = False$ since a triangle cannot be drawn on a single layer. The next case is $max\{a_y, b_y, c_y\} - min\{a_y, b_y, c_y\} \geq 1$ when p is a leaf. Then

we assign $F_p(a_y, b_y, c_y) = True$ since two layers are sufficient to draw a triangle. The next case is $max\{a_y, b_y, c_y\} - min\{a_y, b_y, c_y\} \leq 1$ when p is an internal vertex. Then we assign $F_p(a_y, b_y, c_y) = False$ for this case since the outer face needs two layers to be drawn and the inner vertex p cannot be placed on any of them. The remaining case is $max\{a_y, b_y, c_y\} - min\{a_y, b_y, c_y\} > 1$ when p is an internal vertex. Then we define $F_p(a_y, b_y, c_y)$ recursively by Lemma 9. □

We associate a table $FC_i[1{:}\lfloor\frac{2n+2}{3}\rfloor, 1{:}\lfloor\frac{2n+2}{3}\rfloor, 1{:}\lfloor\frac{2n+2}{3}\rfloor]$ for each vertex i of the representative tree T of G, where the solution of $F_i(a_y, b_y, c_y)$ is stored in $FC_i[a_y, b_y, c_y]$. To store the computed y-coordinates of the vertices of G, we maintain another table $\mathrm{Y}_i[1{:}\lfloor\frac{2n+2}{3}\rfloor, 1{:}\lfloor\frac{2n+2}{3}\rfloor, 1{:}\lfloor\frac{2n+2}{3}\rfloor]$ for each vertex i of T. Each entry $\mathrm{Y}_i[a_y, b_y, c_y]$ is computed as follows.

$$Y_i[a_y, b_y, c_y] = \begin{cases} False & \text{if } FC_i[a_y, b_y, c_y] = \text{False;} \\ True & \text{if } i \text{ is a leaf and } FC_i[a_y, b_y, c_y] = \text{True;} \\ i_y & \text{if } i \text{ is an internal vertex and } FC_i[a_y, b_y, c_y] = \text{True.} \end{cases}$$

Let G be a plane 3-tree with the outer vertices a, b, c and p be the representative vertex of G. If $\mathrm{Y}_p[a_y, b_y, c_y]$ is $False$, then G has no layered drawing for the given y-coordinate assignment a_y, b_y, c_y. If the entry is $True$, then G has no inner vertex and G has a layered drawing for the given y-coordinate assignment a_y, b_y, c_y. Otherwise, G has a layered drawing for the given y-coordinate assignment a_y, b_y, c_y and the entry $\mathrm{Y}_p[a_y, b_y, c_y]$ contains the y-coordinate of the representative vertex p.

To obtain the y-coordinate assignment of each internal vertex of G, we check the entry $\mathrm{Y}_p[a_y, b_y, c_y]$. If the entry contains a y-coordinate of the representative vertex p, we check the entries $\mathrm{Y}_{q_1}[a_y, b_y, p_y]$, $\mathrm{Y}_{q_2}[b_y, c_y, p_y]$ and $\mathrm{Y}_{q_3}[c_y, a_y, p_y]$ to get the y-coordinates of the three children of p. We push $\mathrm{Y}_{q_1}[a_y, b_y, p_y]$, $\mathrm{Y}_{q_2}[b_y, c_y, p_y]$ and $\mathrm{Y}_{q_3}[c_y, a_y, p_y]$ on a stack and pop one entry for further exploration recursively. This is similar to the traversal of the representative tree T of G in preorder, that is, first traversing the root of T, then traversing the left, middle and right subtrees one after another. When the stack is empty, y-coordinates for all the vertices of G are obtained. Since T has $n-3$ internal vertices by Lemma 5, this process takes $O(n)$ time.

We now describe Algorithm **Minimum-Layer** which computes the minimum number of layers required to draw G using Algorithm **Feasibility- Checking**. Let T be the representative tree of the plane 3-tree G. We assume that G admits a layered drawing on h layers and iterate h from 1 to $\lfloor\frac{2n-1}{3}\rfloor$. At each iteration we traverse T in preorder and for each vertex i of T, Algorithm **Minimum-Layer** generates all possible y-coordinate assignments for the outer vertices a, b and c of $G(C_i)$ within h layers. For each such assignment a_y, b_y and c_y, Algorithm **Feasibility-Checking** is called to check whether $G(C_i)$ is drawable. The first h within which G is drawable is the minimum number of layers h_m required to draw G. At the end of this section, formal descriptions of Algorithm **Minimum-Layer** and Algorithm **Feasibility-Checking** are given in Algorithm 1 and Algorithm 2, respectively.

Lemma 10 *Let T be the representative tree of a plane 3-tree G and i be any internal vertex of T. Let a, b and c be the outer vertices of $G(C_i)$. Then Algorithm **Minimum-Layer** generates all possible y-coordinate assignments for a, b and c within h layers after the hth iteration.*

Proof. We prove the correctness of the algorithm by induction. For $h = 1$, the assignment is obvious from Line 3. We may thus assume that $h > 1$ and all the y-coordinate assignments within layer 1 to $h - 1$ have been generated and the results have been calculated within $h - 1$ iterations. Now we consider the hth iteration. In Line 8, a_y is assigned layer h and in Line 9 b_y and c_y are assigned all possible y-coordinates within h. Next, b_y is assigned layer h in Line 17 and in Line 18, a_y and c_y are assigned all possible y-coordinates within $h-1$ and h, respectively. Similarly, c_y is assigned layer h in Line 26 and in Line 27, a_y and b_y are assigned all possible y-coordinates within $h - 1$.

Suppose for a contradiction that the y-coordinate assignments a_y, b_y and c_y have not been generated after the hth iteration. Clearly $max\{a_y, b_y, c_y\}$ cannot be less than h, since all the y-coordinate assignments within layer 1 to $h-1$ have been generated by induction. We may thus assume that $max\{a_y, b_y, c_y\} = h$. One can observe that the hth iteration ensures the generation of all possible y-coordinate assignments such that $max\{a_y, b_y, c_y\} = h$, a contradiction. □

We now analyze the complexity of Algorithm **Minimum-Layer**.

Theorem 4.2 *Given a plane 3-tree G with n vertices, Algorithm **Minimum-Layer** computes the minimum number of layers h_m required to draw G on layers in $O(nh_m^4)$ time.*

Proof. To prove the claim we first calculate the number of times Algorithm **Feasibility-Checking** is called. Since we iterate the number of layers h from 1 to $\lfloor \frac{2n-1}{3} \rfloor + 1$ and at each iteration we traverse T in preorder, the number of times all the vertices of T is considered is $h_m \times n$. For each internal vertex p, Algorithm **Feasibility-Checking** is called for $h \times h$ times in Line 11, $h \times (h-1)$ times in Line 20 and $(h-1) \times (h-1)$ times in Line 29. For all the $n - 3$ internal vertices of T, in each iteration the total number of calls to Algorithm **Feasibility-Checking** by Algorithm **Minimum-Layer** is

$$\begin{aligned} & h_m \times n(h^2 + h(h-1) + (h-1)^2) \\ &= h_m n(h^2 + h^2 - h + h^2 - 2h + 1) \\ &= h_m n(3h^2 - 3h + 1) \\ &= O(nh_m^3) \end{aligned}$$

We store the solutions of the subproblems in the FC tables where each entry of the tables initially contains null to denote that the entry is yet to be filled in. When the subproblem is first encountered during the execution of the recursive algorithm **Feasibility-Checking**, its solution is computed and stored in the table. Each subsequent time the subproblem is encountered, the value stored in the table is looked up and returned. The solutions of the subproblems are computed bottom up and each lookup takes $O(1)$ time. Moreover, p_y can take at most h_m values in Line 5 of Algorithm **Feasibility-Checking**. Therefore, each

call to Algorithm **Feasibility-Checking** takes $O(h_m) \times O(1) = O(h_m)$ time. Since the total number of times Algorithm **Feasibility-Checking** is called, including the recursive calls, is $O(nh_m^3)$ the total running time of this algorithm is $O(nh_m^3) \times O(h_m) = O(nh_m^4)$. We now recall that the construction of the representative tree takes $O(n)$ time by Lemma 7. Thus Algorithm **Minimum-Layer** takes $O(n) + O(nh_m^4) = O(nh_m^4)$ time in total. □

Algorithm 1 *Minimum-Layer(G)*

```
1: Construct the representative tree T of G
2: for each vertex i of T do
3:     FC_i[1,1,1] = False
4: end for
5: {The outer vertices of G(C_i) are a, b and c}
6: for each h from 2 to ⌊(2n−1)/3⌋ + 1 do
7:     for each internal vertex i of T in preorder do
8:         a_y = h
9:         for b_y from 1 to h and c_y from 1 to h do
10:            if FC_i[a_y,b_y,c_y] = null then
11:                Feasibility-Checking (a,b,c)
12:            end if
13:            if i = root && FC_i[a_y,b_y,c_y] = true then
14:                return
15:            end if
16:        end for
17:        b_y = h
18:        for a_y from 1 to h − 1 and c_y from 1 to h do
19:            if FC_i[a_y,b_y,c_y] = null then
20:                Feasibility-Checking (a,b,c)
21:            end if
22:            if i = root && FC_i[a_y,b_y,c_y] = true then
23:                return
24:            end if
25:        end for
26:        c_y = h
27:        for a_y from 1 to h − 1 and b_y from 1 to h − 1 do
28:            if FC_i[a_y,b_y,c_y] = null then
29:                Feasibility-Checking (a,b,c)
30:            end if
31:            if i = root && FC_i[a_y,b_y,c_y] = true then
32:                return
33:            end if
34:        end for
35:    end for
36: end for
```

Algorithm 2 *Feasibility-Checking(a,b,c)*

{The outer vertices of G are a, b and c and p is its representative vertex}
if $FC_p[a_y, b_y, c_y] \neq null$ **then**
    return $FC_p[a_y, b_y, c_y]$
else if $(\max\{a_y, b_y, c_y\} - \min\{a_y, b_y, c_y\} > 1)$ & (p is an internal vertex) **then**
    for $\min\{a_y, b_y, c_y\} < p_y < \max\{a_y, b_y, c_y\}$ **do**
        if (Feasibility-Checking(a, b, p) & Feasibility-Checking(b, c, p) & Feasibility-Checking(c, a, p)) **then**
            $FC_p[a_y, b_y, c_y]$ = True , $\mathrm{Y}_p[a_y, b_y, c_y] = p_y$, **break**
        end if
    end for
else
    Compute $FC_p[a_y, b_y, c_y]$ and $\mathrm{Y}_p[a_y, b_y, c_y]$ by Theorem 4.1
end if

5 Minimum-Area Drawings

In this section we extend the concept of the dynamic programming technique of Section 4 to give an algorithm **Minimum-Area** to obtain a minimum-area drawing of a plane 3-tree G.

We now present an outline of the algorithm. Since the upper bound of the area of straight-line grid drawings of planar graphs is kn^2 with $k \leq 1$, it is obvious that the upper bound for the area of a minimum-area drawing of a plane 3-tree G is at most kn^2 with $k \leq 1$. Since the minimum number of layers required for any straight-line grid drawing of G is h_m, the upper bound for width is $\lceil n^2/h_m \rceil$. Therefore, we assume a height h and a width w and iterate from 1 to n and 1 to $min(\lceil \frac{n^2}{h} \rceil, \lceil \frac{n^2}{h_m} \rceil)$, respectively. At each iteration of h and w we check whether G is drawable on a $w \times h$ grid or not. Algorithm **Minimum-Area** generates all the possible (x, y)-coordinate assignments for the outer vertices of G and checks the drawability of G for each such assignment using Algorithm **Area-Checking**.

For convenience, we describe Algorithm **Area Checking** before Algorithm **Minimum-Area**. At the end of this section we give pseudocodes for both of the algorithms. Here we formally define the input and output of the problem *Area Checking*.

Input: A plane 3-tree G and (x, y)-coordinate assignments of the three outer vertices a, b and c of G.

Output: If G admits a drawing with the given (x, y)-coordinates of a, b and c, the output is $True$ and otherwise it is $False$.

Like the Feasibility Checking problem for minimum-layer drawing, we can characterize the optimal substructure for the problem Area Checking. Let G be

a plane 3-tree with the representative tree T. We denote the x-coordinate and y-coordinate of a vertex v by v_x and v_y, respectively. We denote by $A_p(a_y^x, b_y^x, c_y^x)$ the Area Checking problem of any vertex p of T where a_y^x, b_y^x, c_y^x are the (x, y)-coordinates of the three outer vertices a, b and c of $G(C_p)$. We denote by Γ'_p a minimum-area drawing of $G(C_p)$.

We now prove that the Area Checking problem has the following optimal substructure property.

Lemma 11 *Let G be a plane 3-tree with the representative tree T. Let p be any internal vertex of T with the three children q_1, q_2, q_3 in T and a, b, c be the outer vertices of $G(C_p)$. Then the Area Checking problems of q_1, q_2 and q_3 are the three subproblems of the Area Checking problem of p.*

Proof. The vertex p is an inner vertex of G and therefore, p must be placed inside the outer face of G. Since the (x, y)-coordinates of a, b, c are preassigned and p_x, p_y are the same for the drawings Γ'_{q_1}, Γ'_{q_2} and Γ'_{q_3}, those three drawings can be combined to get the drawing Γ'_p of $G(C_p)$ as illustrated in Figure 10. Thus the solution of the Area Checking problem of p consists of the solutions of the Area Checking problems of q_1, q_2 and q_3; and hence the Area Checking problems of q_1, q_2 and q_3 are the three subproblems of the Area Checking problem of p. □

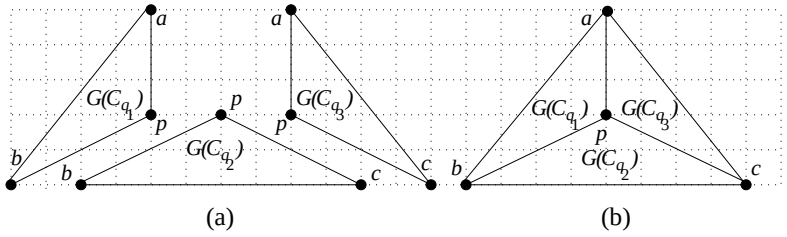

Figure 10: Illustration for the proof of Lemma 11.

One can easily observe the overlapping subproblem property for the Area Checking problem in a similar way that we used to show the overlapping subproblem property of the Feasibility Checking problem.

By a method similar to the proof of Lemma 10 one can see that Algorithm **Minimum-Area** generates all possible (x, y)-coordinate assignments of the outer vertices of G within $w \times min(\lceil \frac{n^2}{h} \rceil, \lceil \frac{n^2}{h_m} \rceil)$ area. We now prove Theorem 5.1 which states the recursive solution of Area Checking problem.

Theorem 5.1 *Let G be a plane 3-tree with the representative tree T and p be any vertex of T. Let a, b, c be the three outer vertices of $G(C_p)$ and q_1, q_2, q_3 be the three children of p when p is an internal vertex of T. Let $A_p(a_y^x, b_y^x, c_y^x)$ be the Area Checking problem of p where a, b and c have distinct (x, y)-coordinates.*

Then $A_p(a_y^x, b_y^x, c_y^x)$ has the following recursive formula.

$$A_p(a_y^x, b_y^x, c_y^x) = \begin{cases} False & \text{if } \{max\{a_x, b_x, c_x\} - min\{a_x, b_x, c_x\} = 0\} \\ & \vee \{max\{a_y, b_y, c_y\} - min\{a_y, b_y, c_y\} = 0\}; \\ True & \text{if } \{\{max\{a_x, b_x, c_x\} - min\{a_x, b_x, c_x\} \geq 1\} \\ & \wedge \{max\{a_y, b_y, c_y\} - min\{a_y, b_y, c_y\} \geq 1\}\} \\ & \wedge\ p \text{ is a leaf;} \\ False & \text{if } \{\{max\{a_x, b_x, c_x\} - min\{a_x, b_x, c_x\} \leq 1\} \\ & \vee \{max\{a_y, b_y, c_y\} - min\{a_y, b_y, c_y\} \leq 1\}\} \\ & \wedge\ p \text{ is an internal vertex;} \\ \bigvee_{p_x, p_y} \{A_{q_1}(a_y^x, b_y^x, p_y^x) \wedge A_{q_2}(b_y^x, c_y^x, p_y^x) \wedge A_{q_3}(c_y^x, a_y^x, p_y^x)\} & \\ & \text{where } (p_x, p_y) \text{ is inside the triangle with the} \\ & \text{vertices } a, b, c, \text{ otherwise.} \end{cases}$$

Proof. First we consider the case when $max\{a_x, b_x, c_x\} - min\{a_x, b_x, c_x\} = 0 \vee max\{a_y, b_y, c_y\} - min\{a_y, b_y, c_y\} = 0$. Then we assign $A_p(a_y^x, b_y^x, c_y^x) = False$ because a grid of at least area 1×1 is necessary to draw a triangle. The next case is $max\{a_x, b_x, c_x\} - min\{a_x, b_x, c_x\} \geq 1 \wedge max\{a_y, b_y, c_y\} - min\{a_y, b_y, c_y\} \geq 1$ when p is a leaf. Then we assign $A_p(a_y^x, b_y^x, c_y^x) = True$ since area 1×1 is sufficient to draw a triangle. The next case is $max\{a_x, b_x, c_x\} - min\{a_x, b_x, c_x\} \leq 1 \vee max\{a_y, b_y, c_y\} - min\{a_y, b_y, c_y\} \leq 1$ when p is an internal vertex. We assign $A_p(a_y^x, b_y^x, c_y^x) = False$ since the width and height of $G(C_p)$ is at most 1 and p cannot be placed inside C_p. The remaining case is $max\{a_x, b_x, c_x\} - min\{a_x, b_x, c_x\} > 1 \wedge max\{a_y, b_y, c_y\} - min\{a_y, b_y, c_y\} > 1$ when p is an internal vertex. Then we define $A_p(a_y^x, b_y^x, c_y^x)$ recursively according to Lemma 11. □

We associate a table $AC_i[1{:}\lceil \frac{n^2}{h_m} \rceil, 1{:}\lceil \frac{n^2}{h_m} \rceil, 1{:}\lceil \frac{n^2}{h_m} \rceil, 1{:}n, 1{:}n, 1{:}n]$ for each vertex i of the representative tree T of G, where the solution of $A_i(a_y^x, b_y^x, c_y^x)$ is stored in $AC_i[a_y^x, b_y^x, c_y^x]$. To store the computed (x, y)-coordinates of the vertices of G, we maintain two tables $\mathrm{X}_i[1{:}\lceil \frac{n^2}{h_m} \rceil, 1{:}\lceil \frac{n^2}{h_m} \rceil, 1{:}\lceil \frac{n^2}{h_m} \rceil, 1{:}n, 1{:}n, 1{:}n]$ and $\mathrm{Y}_i[1{:}\lceil \frac{n^2}{h_m} \rceil, 1{:}\lceil \frac{n^2}{h_m} \rceil, 1{:}\lceil \frac{n^2}{h_m} \rceil, 1{:}n, 1{:}n, 1{:}n]$ for each vertex i of T. Each entry of the two table X_i and Y_i is computed as follows.

$$X_i[a_x, b_x, c_x, a_y, b_y, c_y] = \begin{cases} False & \text{if } AC_i[a_x, b_x, c_x, a_y, b_y, c_y] = \text{False;} \\ True & \text{if } i \text{ is a leaf and} \\ & AC_i[a_x, b_x, c_x, a_y, b_y, c_y] = \text{True;} \\ i_x & \text{if } i \text{ is an internal vertex and} \\ & AC_i[a_x, b_x, c_x, a_y, b_y, c_y] = \text{True.} \end{cases}$$

$$Y_i[a_x, b_x, c_x, a_y, b_y, c_y] = \begin{cases} False & \text{if } AC_i[a_x, b_x, c_x, a_y, b_y, c_y] = \text{False;} \\ True & \text{if } i \text{ is a leaf and} \\ & AC_i[a_x, b_x, c_x, a_y, b_y, c_y] = \text{True;} \\ i_y & \text{if } i \text{ is an internal vertex and} \\ & AC_i[a_x, b_x, c_x, a_y, b_y, c_y] = \text{True.} \end{cases}$$

Let a, b, c be the outer vertices and p be the representative vertex of G. If $\mathrm{X}_p[a_x, b_x, c_x, a_y, b_y, c_y]$ or $\mathrm{Y}_p[a_x, b_x, c_x, a_y, b_y, c_y]$ is $False$, then G has no

straight-line grid drawing for the given (x, y)-coordinate assignments a_y^x, b_y^x, c_y^x. If the entries are *True*, then G has a straight-line grid drawing with the given (x, y)-coordinate assignments a_y^x, b_y^x, c_y^x. Otherwise, the two entries contain a x-coordinate and a y-coordinate of the representative vertex p, respectively.

We now describe Algorithm **Minimum-Area** which gives a drawing of G with the minimum area, using Algorithm **Area-Checking**. Let T be the representative tree of the plane 3-tree G. We assume a width w and a height h for G. We iterate h from 1 to n and for each h, we iterate w from 1 to $min(\lceil \frac{n^2}{h} \rceil, \lceil \frac{n^2}{h_m} \rceil)$. At each iteration we traverse T in preorder. For each internal vertex i of T, **Minimum-Area** generates all possible (x, y)-coordinate assignments for the outer vertices a, b and c of $G(C_i)$ within area $w \times h$. For each such (x, y)-coordinate assignment of a, b and c, Algorithm **Area-Checking** is called to check whether $G(C_i)$ is drawable. Each time a drawing of G with smaller area is found, the stored area is replaced by the smaller area and at the end of the algorithm, the stored area is the minimum. At the end of this section, formal descriptions of Algorithm **Minimum-Area** and Algorithm **Area-Checking** are given in Algorithm 3 and Algorithm 4, respectively.

We now analyze the complexity of Algorithm **Minimum-Area**.

Theorem 5.2 *Given a plane 3-tree G with $n \geq 3$ vertices, Algorithm **Minimum-Area** gives a minimum-area drawing of G in $O(n^9 \log n)$ time.*

Proof. We iterate height h from 2 to n and for each h, width w is iterated from 2 to $min(\lceil \frac{n^2}{h} \rceil, \lceil \frac{n^2}{h_m} \rceil)$ where h_m is the minimum number of layers required to draw G. So the total number of iterations in Line 7 is

$$\begin{aligned} & h_m \tfrac{n^2}{h_m} + \tfrac{n^2}{h_m+1} + ... + \tfrac{n^2}{n} \\ &= n^2(1 + \tfrac{1}{h_m+1} + ... + \tfrac{1}{n}) \\ &= n^2(1 + \textstyle\sum_{k=h_m+1}^{n} \tfrac{1}{k}) \\ &\leq n^2 + n^2 \times \log \tfrac{n}{h_m} \\ &= O(n^2 \log n) \end{aligned}$$

The first time a feasible drawing is found, we store the area $w \times h$ for that drawing. Each subsequent time a feasible drawing is found, we replace the stored area only if the area $w \times h$ for the current values of w and h is smaller or equal to the stored area. After the algorithm is terminated, the minimum area required to draw G is returned.

Let the representative tree of G be T. For each iteration we traverse T in preorder in Line 8 and for each internal vertex of T, Algorithm **Area-Checking** is called w^2h^2, $w^2h(h-1)$ and $w^2(h-1)^2$ times in Line 12, Line 23 and Line 34, respectively. Since there are $n-3$ internal vertices in T, the total number of calls to Algorithm **Area-Checking** by Algorithm **Minimum-Area** in each iteration is

$$\begin{aligned} & n(w^2h^2 + w^2h(h-1) + w^2(h-1)^2) \\ &= nw^2(3h^2 - 3h + 1) \\ &= O(nw^2h^2) \end{aligned}$$

We store the solutions of the subproblems in the AC tables where each entry of the tables initially contains null to denote that the entry is yet to be filled

in. When the subproblem is first encountered during the execution of the recursive algorithm **Area-Checking**, its solution is computed and stored in the table. Each subsequent time the subproblem is encountered, the value stored in the table is looked up and returned. The solutions of these subproblems are computed bottom up and each lookup takes $O(1)$ time. Moreover, p_y^x can take at most $w \times h$ values in Line 9 of Algorithm **Area-Checking**. Therefore, each call to Algorithm **Area-Checking** takes $O(1) \times O(wh) = O(wh)$ time.

Hence for each iteration, the number of times Algorithm **Area-Checking** is called including all the recursive calls is $O(nw^2h^2)$. Therefore, the total running time of Algorithm **Area-Checking** is $O(nw^2h^2) \times O(wh) = O(nw^3h^3) = O(n^7)$ since $wh = O(n^2)$. Thus the total time required for all the $O(n^2 \log n)$ iterations is $O(n^2 \log n) \times O(n^7) = O(n^9 \log n)$.

We now recall that the construction of the representative tree takes $O(n)$ time by Lemma 7. Thus Algorithm **Minimum-Layer** takes $O(n) + O(n^9 \log n)$ $= O(n^9 \log n)$ time in total. □

6 Lower Bound

In this section we improve the lower bound on area for straight-line grid drawings of plane graphs. We show that there exist plane 3-trees, for which the improved bound holds.

One of the most famous and long standing conjectures states that any plane graph G with n vertices can be drawn in $\lceil \frac{2n}{3} - 1 \rceil \times \lceil \frac{2n}{3} - 1 \rceil$ area [20]. Frati and Patrignani [20] showed that this bound neglects at least a linear term. They showed that there exists a plane graph with n vertices which requires at least $(\frac{2n}{3} - 1) \times (\frac{2n}{3})$ area where n is a multiple of three. This indicates that the known $(\frac{2n}{3} - 1) \times (\frac{2n}{3} - 1)$ lower bound on area for the straight-line grid drawings of plane graphs can be improved further. The lower bound on area is known to be $\lfloor \frac{2(n-1)}{3} \rfloor \times \lfloor \frac{2(n-1)}{3} \rfloor$ area [8] which we improve to $\lfloor \frac{2n}{3} - 1 \rfloor \times 2\lceil \frac{n}{3} \rceil$ area for $n \geq 6$.

Before showing the graphs for which the improved lower bound on area holds, we describe the "nested triangles graphs". Dolev *et al.* first exhibited the "nested triangles graphs" in 1984, to obtain a lower bound on area $(\frac{2n}{3} - 1) \times (\frac{2n}{3} - 1)$ for straight-line grid drawings of plane graphs where the outer face is fixed [13]. Let t_1, t_2 be two disjoint *3-cycles* in a graph G and Γ be a planar drawing of G. Then t_1 is nested in t_2 in Γ, if t_1 is drawn in the region enclosed by t_2. This relationship is shown by $t_2 > t_1$. We call a planar graph G_t with $n \geq 3$ vertices a *nested triangles graph* if the following (a) and (b) hold:

(a) if $n = 3$, then G_t is a 3-cycle;

(b) if $n > 3$, then G_t is a triangulated plane graph with exactly $n/3$ nested triangles such that $t_{n/3} > ... > t_2 > t_1$.

Algorithm 3 *Minimum-Area(G)*

```
1: Construct the representative tree T of G
2: for each vertex i of T_{n-3} in preorder do
3:     AC_i[1,1,1,1,1,1] = False
4: end for
5: {The outer vertices of G(C_i) are a, b and c; area stores the minimum area}
6: area=n^2
7: for each h from 2 to n and each w from 2 to min(⌈n^2/h⌉, ⌈n^2/h_m⌉) do
8:     for each vertex i of T_{n-3} in preorder do
9:         a_x = w, a_y = h
10:        for 1 ≤ b_x ≤ w , 1 ≤ b_y ≤ h , 1 ≤ c_x ≤ w , 1 ≤ c_y ≤ h do
11:            if AC_i[a_x,b_x,c_x,a_y,b_y,c_y] = null then
12:                Area-Checking (a,b,c)
13:            end if
14:            if i = root && AC_i[a_x,b_x,c_x,a_y,b_y,c_y] = true then
15:                if area ≥ w × h then
16:                    area=wh
17:                end if
18:            end if
19:        end for
20:        b_x = w, b_y = h
21:        for 1 ≤ a_x ≤ w , 1 ≤ a_y ≤ h − 1 , 1 ≤ c_x ≤ w , 1 ≤ c_y ≤ h do
22:            if AC_i[a_x,b_x,c_x,a_y,b_y,c_y] = null then
23:                Area-Checking (a,b,c)
24:            end if
25:            if i = root && AC_i[a_x,b_x,c_x,a_y,b_y,c_y] = true then
26:                if area ≥ w × h then
27:                    area=wh
28:                end if
29:            end if
30:        end for
31:        c_x = w, c_y = h
32:        for 1 ≤ a_x ≤ w , 1 ≤ a_y ≤ h − 1 , 1 ≤ b_x ≤ w , 1 ≤ b_y ≤ h − 1 do
33:            if AC_i[a_x,b_x,c_x,a_y,b_y,c_y] = null then
34:                Area-Checking (a,b,c)
35:            end if
36:            if i = root && AC_i[a_x,b_x,c_x,a_y,b_y,c_y] = true then
37:                if area ≥ w × h then
38:                    area=wh
39:                end if
40:            end if
41:        end for
42:    end for
43: end for
```

Algorithm 4 *Area-Checking(a,b,c)*

```
{a, b, c are outer vertices of G and p is the representative vertex}
if (max{a_x, b_x, c_x} − min{a_x, b_x, c_x} = 0) || (max{a_y, b_y, c_y}−
    min{a_y, b_y, c_y} = 0) then
    AC_p[a_x, b_x, c_x, a_y, b_y, c_y] = False
    X_p[a_x, b_x, c_x, a_y, b_y, c_y] = False
    Y_p[a_x, b_x, c_x, a_y, b_y, c_y] = False
else if (max{a_x, b_x, c_x} − min{a_x, b_x, c_x} > 1) & (max{a_y, b_y, c_y}−
        min{a_y, b_y, c_y} > 1) & (p is an internal node) then
    for all integer points (p_x, p_y) inside the triangle with vertices a,b,c do
        if Area-Checking(a,b,p)& Area-Checking(b,c,p)&
           Area-Checking(c,a,p) then
            AC_p[a_x, b_x, c_x, a_y, b_y, c_y] = True
            X_p[a_x, b_x, c_x, a_y, b_y, c_y] = p_x
            Y_p[a_x, b_x, c_x, a_y, b_y, c_y] = p_y
            break
        end if
    end for
else
    Compute AC_p[a_x, b_x, c_x, a_y, b_y, c_y], X_p[a_x, b_x, c_x, a_y, b_y, c_y]
    and Y_p[a_x, b_x, c_x, a_y, b_y, c_y] by Theorem 5.1
end if
```

Lemma 12 *Let G_t be a nested triangles graph with n vertices and $t = (n/3)$ nested triangles. Then there exists a plane 3-tree G_n^* with n vertices such that G_n^* contains $n/3$ nested triangles.*

Proof. The case $t = 1$ is trivial since G_1 is a triangle which is the plane 3-tree G_3^*. So suppose that $t > 1$ and the lemma holds for all nested triangles graphs having less than t nested triangles. We delete the three outer vertices of G_t to get G_{t-1}. By induction hypothesis, there exists a plane 3-tree G_{n-3}^* with $(n/3) - 1$ nested triangles. Let the outervertices of G_{n-3}^* be d, e and f. We put G_{n-3}^* inside a triangle $\{a, b, c\}$ and add the edges (a, e), (a, d), (a, f), (c, f), (b, f), (b, e) as shown in Figure 11(a). The resulting graph is the required G_n^* if it is a plane 3-tree and contains $n/3$ nested triangles. Since G_{n-3}^* is a plane 3-tree, we can delete its interior vertices recursively in such a way that the resulting graph remains triangulated at each step. We can then delete the vertices d, e and f one after another to obtain the triangle $\{a, b, c\}$. As illustrated in Figures 11(b)–(d), the deletion of d, e, and f one after another keeps the resulting graph triangulated at each step. Thus we can always delete an inner vertex of G_n^* in such a way that at each step the resulting graph remains a plane 3-tree; and hence, G_n^* is a plane 3-tree. Moreover, since the number of nested triangles in G_{n-3}^* is $(n - 3)/3$, the number of nested triangles in G_n^* is $n/3$ in total. □

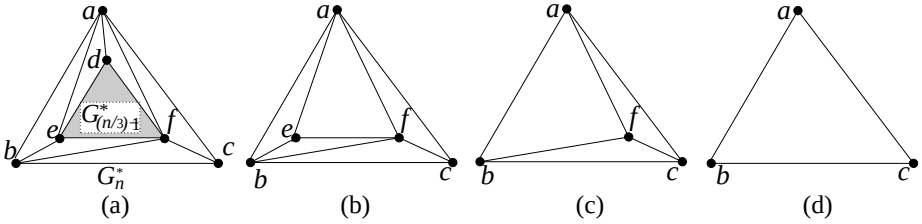

Figure 11: Illustration for the proof of Lemma 12.

Fact 13 [20] *Let Γ be any planar drawing of a graph G, and let t_2 and t_1 be two disjoint 3-cycles of G such that $t_2 > t_1$ in Γ. The height (width) of t_2 in Γ is at least two units bigger than the height (width) of t_1.*

We denote by G'_6, G'_7 and G'_8 the three plane 3-trees depicted by Figures 12(a), (b) and (c), respectively.

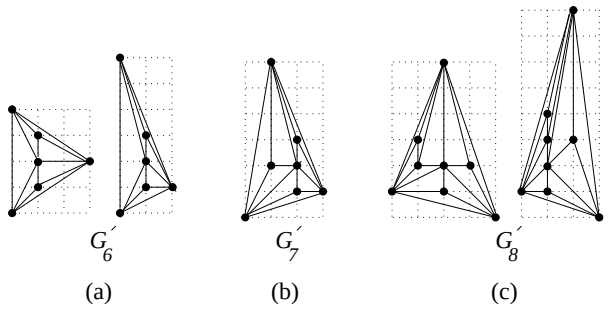

Figure 12: Drawings of G'_6, G'_7 and G'_8.

Fact 14 *The minimum-area straight-line grid drawings for G'_6 requires 2×6 or 3×4 area, G'_7 requires 3×6 area and G'_8 requires 3×8 or 4×6 area.*

Proof. We can prove the fact by case study or by Algorithm **Minimum-Area** presented in Section 5. □

We now have the following theorem for the lower bound on area of plane graphs. The proof of the theorem uses G'_6, G'_7 and G'_8 as the building blocks for the graphs attaining the lower bound with $n \geq 6$ vertices as illustrated in Figure 12. Note that when n is a multiple of three, this bound is the same as the one by Frati and Patrignani [20]. In fact, the graph they used as the building block is G'_6.

Theorem 6.1 *For each $n \geq 6$, there is a n-vertex plane graph G such that the area required to obtain a straight-line grid drawing of G is at least $\lfloor \frac{2n}{3} - 1 \rfloor \times 2 \lceil \frac{n}{3} \rceil$.*

Proof. As an existential proof, we construct plane 3-trees for which the lower bound holds. We form those graphs by enclosing G'_6, G'_7 and G'_8 with area 3×4, 3×6 and 4×6 in the innermost triangle of $G^\star_{n-6}$, $G^\star_{n-7}$ and $G^\star_{n-8}$ where $n = 3m$, $3m+1$ and $3m+2$ for $m \geq 2$, respectively. We enclose the drawings of Figure 12(a) and (c) with area 3×4 and 4×6 since drawings enclosing the alternative drawings of G'_6 and G'_8 will take the same or more area. Therefore the new lower bound for the area $W \times H$ follows from Fact 13–14 and Lemma 12.

$$W \times H = \begin{cases} (\frac{2n-5}{3}) \times (\frac{2n+4}{3}) & \text{if } n = 3m+1 \text{ and } m \geq 2; \\ (\frac{2n-4}{3}) \times (\frac{2n+2}{3}) & \text{if } n = 3m+2 \text{ and } m \geq 2; \\ (\frac{2n-3}{3}) \times (\frac{2n}{3}) & \text{if } n = 3m \text{ and } m \geq 2. \end{cases}$$

It can be easily shown that for all $n \geq 6$, the lower bound for the area of a n-vertex plane graph is $\lfloor \frac{2n}{3} - 1 \rfloor \times 2\lceil \frac{n}{3} \rceil$. □

We conclude this section with the conjecture that for $n > 6$, the $\lfloor \frac{2n}{3} - 1 \rfloor \times 2\lceil \frac{n}{3} \rceil$ lower bound on the area requirement of plane graphs also hold for the class of plane 3-trees shown in Figure 13.

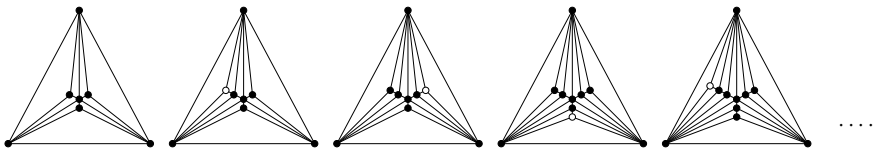

Figure 13: A class of plane 3-trees.

7 Conclusion

We have shown that for a fixed planar embedding of a plane 3-tree G, a minimum-area drawing can be obtained in polynomial time. Since a plane 3-tree G has only linear number of planar embeddings, we can compute the area requirements of all the embeddings of G and determine the planar embedding which gives the best area bound; and thus we can obtain a minimum-area drawing of G in polynomial time when the embedding of G is not fixed.

Since the area minimization problem for plane 3-trees can be solved in polynomial time, it remains open to investigate whether any other computationally hard problem in the area of graph drawing can be solved in polynomial time for plane 3-trees. Many such problems yet to be analyzed can be found in [3, 6, 24]. It is a challenge to find a simpler algorithm for obtaining minimum-area drawings of plane 3-trees and to explore further properties of this subclass of planar graphs. It is also left as a future work to find other classes of planar graphs for which the area minimization problem can be solved in polynomial time.

It is well known that if a decision problem on graphs of small "treewidth" can be defined in "monadic second-order logic", there is a linear-time algorithm for testing the problem [11]. Since the "treewidth" of plane 3-trees is bounded by

three, it would be interesting to study whether the area minimization problem is definable in "monadic second-order logic" or not.

Acknowledgement

This work is done under the project "Minimum-Area Drawings of Plane Graphs", supported by CASR, BUET, in Graph Drawing & Information Visualization Laboratory of the Department of CSE, BUET established under the project "Facility Upgradation for Sustainable Research on Graph Drawing & Information Visualization" supported by the Ministry of Science and Information & Communication Technology, Government of Bangladesh. We thank the anonymous referees for their useful comments and suggestions.

References

[1] M. J. Alam, M. A. H. Samee, M. M. Rabbi, and M. S. Rahman. Minimum-layer upward drawings of trees. *Journal of Graph Algorithms and Applications*, 14(2):245–267, 2010.

[2] S. Arnborg and A. Proskurowski. Canonical representations of partial 2- and 3-trees. *Behaviour & Information Technology*, 32(2):197–214, 1992.

[3] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.

[4] T. Biedl and L. E. R. Velasquez. Drawing planar 3-trees with given face-areas. In *the 17th International Symposium on Graph Drawing (GD 2009)*, volume 5849 of *Lecture Notes in Computer Science*, pages 316–322. Springer, 2010.

[5] H. L. Bodlaender and T. Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *Journal of Algorithms*, 21(2):358–402, 1996.

[6] F. Brandenburg, D. Eppstein, M. T. Goodrich, S. Kobourov, G. Liotta, and P. Mutzel. Selected open problems in graph drawing. In *the 11th International Symposium on Graph Drawing (GD 2003)*, volume 2912 of *Lecture Notes in Computer Science*, pages 515–539. Springer, 2004.

[7] F. J. Brandenburg. Drawing planar graphs on $\frac{8}{9}n^2$ area. In *the International Conference on Topological and Geometric Graph Theory*, volume 31 of *Electronic Notes in Discrete Mathematics*, pages 37–40. Elsevier, 2008.

[8] M. Chrobak and S. Nakano. Minimum width grid drawings of plane graphs. In *the DIMACS International Workshop on Graph Drawing*, volume 894 of *Lecture Notes in Computer Science*, pages 104–110. Springer, 1994.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 1990.

[10] S. Cornelsen, T. Schank, and D. Wagner. Drawing graphs on two and three lines. In *the 10th International Symposium on Graph Drawing(GD 2002)*, volume 2528 of *Lecture Notes In Computer Science*, pages 31–41. Springer, 2002.

[11] B. Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990.

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

[13] D. Dolev, T. Leighton, and H. Trickey. Planar embedding of planar graphs. *Advances in Computing Research*, 2:147–161, 1984.

[14] V. Dujmović, D. Eppstein, M. Suderman, and D. R. Wood. Drawings of planar graphs with few slopes and segments. *Computational Geometry-Theory and Applications*, 38(3):194–212, 2007.

[15] V. Dujmović, M. Fellows, M. Hallett, M. Kitching, G. Liotta, C. McCartin, K. Nishimura, P. Ragde, F. Rosamond, M. Suderman, S. H. Whitesides, and D. Wood. On the parametrized compexity of layered graph drawing. In *the 9th European Symposium on Algorithms (ESA 01)*, volume 2161 of *Lecture Notes in Computer Science*, pages 488–499. Springer, 2001.

[16] V. Dujmović, M. Suderman, and D. R. Wood. Really straight graph drawings. In *the 12th International Symposium on Graph Drawing (GD 2004)*, volume 3383 of *Lecture Notes in Computer Science*, pages 122–132. Springer, 2005.

[17] Colm Ó Dúnlaing. A simple criterion for nodal 3-connectivity in planar graphs. *Electronic Notes in Theoretical Computer Science*, 225:245–253, 2009.

[18] I. Fary. On straight line representation of planar graphs. In *Acta Sci. Math. Szeged*, volume 11, pages 229–233, 1948.

[19] S. Felsner, G. Liotta, and S. K. Wismath. Straight-line drawings on restricted integer grids in two and three dimensions. In *the 9th International Symposium on Graph Drawing (GD 2001)*, volume 2265 of *Lecture Notes In Computer Science*, pages 328–342. Springer, 2002.

[20] F. Frati and M. Patrignani. A note on minimum area straight-line drawings of planar graphs. In *the 15th International Symposium on Graph Drawing (GD 2007)*, volume 4875 of *Lecture Notes in Computer Science*, pages 339–344. Springer, 2008.

[21] G. Kant. A more compact visibility representation. In *the 19th International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 790 of *Lecture Notes In Computer Science*, pages 411 – 424. Springer, 1993.

[22] M. Krug and D. Wagner. Minimizing the area for planar straight-line grid drawings. In *the 15th International Symposium on Graph Drawing (GD 2007)*, volume 4875 of *Lecture Notes In Computer Science*, pages 207–212. Springer, 2008.

[23] D. Mondal, R. I. Nishat, M. S. Rahman, and M. J. Alam. Minimum-area drawings of plane 3-trees. In *the 22nd Canadian Conference on Computational Geometry (CCCG 2010)*, pages 191–194, 2010.

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

[25] W. Schnyder. Embedding planar graphs on the grid. In *the first annual ACM-SIAM Symposium on Discrete Algorithms*, pages 138–148. Society for Industrial and Applied Mathematics, 1990.

[26] K. S. Stein. Convex maps. In *American Mathematical Society*, volume 2, pages 464–466, 1951.

[27] M. Suderman. Proper and planar drawings of graphs on three layers. In *the 13th International Symposium on Graph Drawing (GD 2005)*, volume 3843 of *Lecture Notes In Computer Science*, pages 434–445. Springer, 2005.

[28] K. Wagner. Bemerkungen zum vierfarbenproblem. In *Jahresbericht Deutsche Math*, volume 46, pages 26–32, 1936.