M.Sc.Engg. Thesis

# Minimum-Layer Drawings of Trees

Submitted by

Muhammad Jawaherul Alam
Student no.: 040805062P

Department of Computer Science and Engineering
BUET, Dhaka - 1000, Bangladesh

Submitted to

Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
Masters of Science in Computer Science and Engineering

July, 2010

The thesis titled "**Minimum-Layer Drawings of Trees**", submitted by Muhammad Jawaherul Alam, Roll No. 040805062P, Session April 2008, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering and approved as to its style and contents. Examination held on July 24, 2010.

## Board of Examiners

# Candidate's Declaration

This is to certify that the work presented in this thesis entitled "**Minimum-Layer Drawings of Trees**" is the outcome of the investigation carried out by me under the supervision of Professor Dr. Md. Saidur Rahman in the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka. It is also declared that neither this thesis nor any part thereof has been submitted or is being currently submitted anywhere else for the award of any degree or diploma.

Muhammad Jawaherul Alam
Candidate

# Contents

# List of Figures

# Acknowledgments

I would like to thank, first of all, my supervisor Professor Dr. Md. Saidur Rahman for not a few reasons. He not only introduced me to this wonderful and fascinating world of *graph theory* and *graph drawing*, but also provided me with numerous resources and helpful directions. Without his constant supervision and words of encouragement, this thesis would not have been possible at all.

I also thank the members of my thesis committee: Prof. Dr. Md. Monirul Islam, Prof. Dr. M. Kaykobad, Prof. Dr. Md. Abul Kashem Mia and Prof. Dr. Md. Rafiqul Islam for their invaluable suggestions. This thesis is done 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.

I would like to express my deep gratitude to all the members of our research group for their valuable suggestions and continual encouragements. My special thanks goes to Mr. Debajyoti Mondal and Ms. Rahnuma Islam, my colleagues in the research group. My parents also supported me and encouraged me to the best of their ability. My heart-felt gratitude goes to them.

# Abstract

A layered drawing of a planar graph $G$ is a planar straight-line drawing of $G$, where the vertices are placed on a set of horizontal lines, called layers. A minimum-layer drawing of $G$ is a layered drawing of $G$ with the minimum number of layers among all the layered drawings of $G$. To the best of our knowledge, no polynomial-time algorithm is known till now to construct a minimum-layer drawing of a general planar graph. Even for a restricted class of layered drawings, where the edges are between vertices on adjacent layers, the problem of recognizing graphs to admit such a drawing is in NP-complete. Therefore, layered drawings of graphs are often studied for special classes of planar graphs.

In this thesis we address the problem of minimum-layer drawing of trees. For a tree $T$ with pathwidth $h$, a linear-time algorithm is already known and it produces a drawing of $T$ on $\lceil 3h/2 \rceil$ layers. However, the drawing obtained by this algorithm is not necessarily a minimum-layer drawing of $T$. A necessary condition for a tree to admit a $k$-layer drawing is also known and this condition is also sufficient for $k \leq 2$. However for $k > 2$, there is no such necessary and sufficient characterization. The problem of finding a minimum-layer drawing of a tree is known to be in NP-hard with some constraints for the placement of vertices. There are also some polynomial-time algorithms for obtaining a minimum-layer drawing of $T$ with some other constraints on the placement of vertices. However for the general version of the problem, there is neither any polynomial-time algorithm or any hardness result known so far. In this paper we give a linear-time algorithm for obtaining a minimum-layer drawing of a tree. We first give a linear-time algorithm to obtain a minimum-layer drawing of a rooted tree with the additional constraint that the root of the tree is visible from outside the bounding box of the drawing. Using this algorithm, we then give an algorithm to find a minimum-layer drawing of a tree in linear time.

# Chapter 1

# Introduction

A *graph* is a mathematical tool that consists of a set of *vertices* and a set of *edges*, where each edge joins a pair of vertices. In computer science a graph is often used to model a relational structure; a configuration that consists of a collection of entities and the pairwise relations between these entities. We model each entity of the problem at hand with a vertex and the relationship among a pair of entities with an edge between them. Thus a graph can represent any information that can be modeled as objects and relationships between pairs of them. Such a relational structure is omnipresent not only in the field of computer science but also in various problems arising from many diverse application areas. Thus graphs have become powerful mathematical tools in a wide variety of areas in computer science, electrical, architectural and other field of engineering, genetics, bioinformatics, molecular biology, chemistry, VLSI technology, even in geology and social sciences. For almost each of these applications areas, it is often required that the information represented by the graph model is conveniently visualized. This yields to the advent of the field of "Graph Drawing".

*Graph Drawing* is a relatively new area in Computer Science. The *drawing of a graph* constructs a geometric representation of an embedding of the graph in the plane so that the graph and its contents (i.e., the entities and their relationships) are easily traceable. Thus the field of graph drawing is motivated by its abundant applications for information visualization and also for VLSI circuit design, social network analysis, cartography, and bioinformatics, many of which make use of information visualization. Apart from these, graph drawing, particularly automated generation of the drawings of graphs, nowadays finds inducive applications in software engineering (data-flow diagrams, subroutine-call graphs, object-oriented class hierarchies etc.), databases (ER-diagrams), information systems (organizational charts), real-time systems (Petri-nets, state-transition diagrams), Decision support systems (PERT networks, activity trees), electrical and VLSI circuit design (layout design and circuit schematics), artificial intelligence (knowledge-representation diagrams), logic programming (SLD-trees), biology and phylogenetics (evolutionary trees), medical sciences (concept lattices), chemistry (molecular drawings), civil engineering (architectural floorplan) and cartography (map schematics) [3, 21]. For example, [21] shows how a graph drawing technology can be used to generate a VLSI layout satisfying different optimization requirements.

In most of the cases, the geometric representations of graphs generated by graph drawing algorithms are constrained by some predefined geometric or aesthetic properties. A "good" diagram helps in convenient understanding of the underlying system, but a "poor" diagram can be ambiguous, confusing and misleading. The "goodness" of a diagram depends on the application at hand. Different applications requires different desirable geometric or aesthetic properties and hence they impose different constrains on the drawing of a graph. These results in a number of different drawing styles and conventions for drawings of graphs.

A *layered drawing* [27, 29] of a graph $G$ is a drawing of $G$ where the vertices of $G$ are placed on a set of horizontal lines called *layers*, and the edges of $G$ are drawn as straight-line segments. Layered Drawings of graphs finds its application in several areas like VLSI layouts [17], DNA-mapping [30] and information visualization [3, 15]. In some of the application areas, it is often desired to obtain a layered drawing of a graph that occupies the minimum number of layers. In this thesis, we address this problem for a class of graphs called trees. We give a linear-time algorithm to obtain a layered drawing of a given tree on the minimum number of layers.

We first address the problem of minimizing the number of layers in a "root-visible drawing" of a rooted tree. Let $T$ be a rooted tree with the root vertex $r$. A *root-visible drawing* of $T$ is a layered drawing $\Gamma$ of $T$ where there is a point $p$ below all the layers in $\Gamma$ such that a straight-line from $p$ to $r$ does not create any edge crossing with $\Gamma$. We give an algorithm to obtain a root-visible drawing of a rooted tree on the minimum number of layers. We use this algorithm to obtain a minimum-layer drawing of a tree as follows. Given a tree $T$, we find a suitable root $r$ of $T$ such that a minimum-layer root-visible drawing of $T$ with the root $r$ is also a minimum-layer drawing of $T$. We will detail the concepts of the above mentioned algorithms in the later chapters of this thesis. In this introductory chapter, we only focus on some preliminary concepts of layered drawings of graphs and also the application of layered graph drawings in various fields. The rest of this chapter is organized as follows. In Section 1.1, we define layered drawings of graph, which is the central topic of the whole thesis. Section 1.2 depicts some interesting applications of layered graph drawings. Section 1.3 presents the scope of this thesis with a brief overview of the previous results related to the scope and the new results described in this thesis.

## 1.1 Layered Drawings of Planar Graphs

In this section, we give the definition of "layered drawings" of a planar graph. We also define some restricted classes of layered drawings of a planar graph.

A *layered drawing* of a planar graph $G$ is a drawing of $G$ where the vertices of $G$ are placed on a set horizontal lines called *layers*, and each edge of $G$ is drawn as a straight-line segments between its end-vertices without creating any crossing with the other edges. Let $\Gamma$ be a layered drawing of a planar graph $G$. $\Gamma$ is called a *short drawing* of $G$ if the end-vertices of each edge of $G$ is placed on the same or adjacent layers in $\Gamma$. $\Gamma$ is called a *proper drawing* of $G$ if the end-vertices of each edge of $G$ lie on adjacent layers in $\Gamma$. $\Gamma$ is called an *upright drawing* of $G$ if the end-vertices of each edge of $G$ are placed at different layers in $\Gamma$.

Figure 1.1 illustrates different variants of layered drawings by three drawings of the same graph. The drawing in Figure 1.1(a) is proper while those in Figure 1.1(b) and (c) are not. The drawings in Figure 1.1(a) and (c) are upright but that in Figure 1.1(b) is not. The drawings in Figure 1.1(a) and (c) are two short drawings but that in Figure 1.1 is not a short drawing. Finally, the drawings in Figure 1.1(a) and (c) are 4-layer planar drawings and that in Figure 1.1(b) occupies three layers.

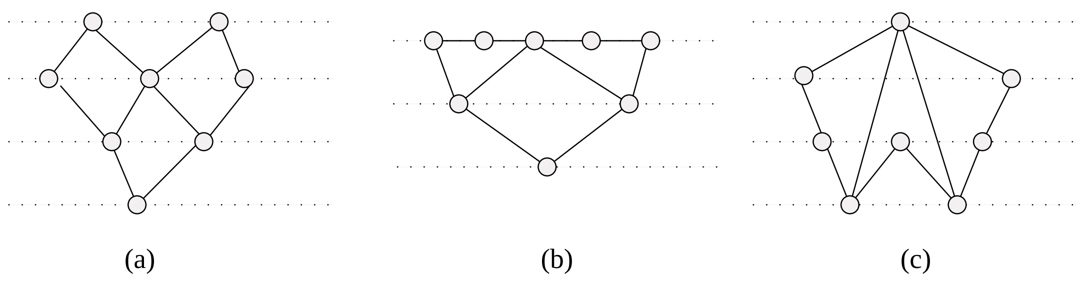


Figure 1.1: Different variants of layered drawings on the same graph.

## 1.2 Applications of Layered Drawings

Layered drawings have found important applications in several areas like VLSI layouts [17], DNA-mapping [30] and information visualization [3, 15]. We present a few applications of layered drawings in the remainder of this section.

### 1.2.1 VLSI Layout

In the *standard cell* technology employed during the VLSI layout design process, the VLSI modules are placed on some constant number of previously fixed rows so that they can be lined up in rows on the integrated circuit. A layered drawing of a graph can be used to obtain a layout of an interconnection network on a standard cell. For example, Figure 1.2(a) represents an interconnection network $C$. We first obtain a graph $G$ from $C$ in Figure 1.2(b), where each vertex of $G$ represents a module in $C$ and each edge of $G$ represents an interconnection between a pair of modules in $C$. From a layered drawing $\Gamma$ of $G$ as illustrated in Figure 1.2(c), we can easily obtain a layout of $C$ in a standard cell as illustrated in Figure 1.2(d), where the number of rows in the layout is equal to the number of layers in $\Gamma$.

### 1.2.2 Flowchart Diagrams

A *flowchart* is a diagram, that represents an algorithm or process, showing the steps as boxes of various kinds, and their order by connecting these with arrows. This diagrammatic representation can give a step-by-step solution to a given problem. Data is represented in these boxes, and the arrows connecting them represent the flow or the direction of the flow of data. Flowcharts are used in analyzing, designing, documenting or managing a process or program

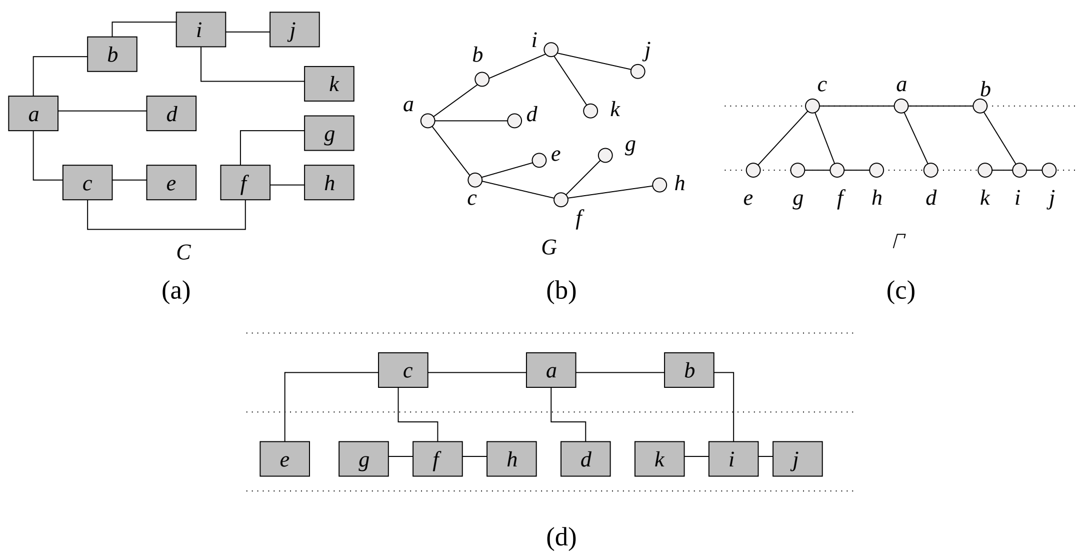


Figure 1.2: (a) An interconnection network $C$, (b) a graph $G$ obtained from $C$, (c) a layered drawing $\Gamma$ of $G$, and (d) a layout of $C$ in a standard cell.

in various fields. Figure 1.3 represents a flowchart of a simple algorithms to find the factorial of a number. Knuth first presented a graph drawing algorithm for automated generation of flowcharts [16]. Today there are many automated algorithms for flowchart design that make use of graph drawing techniques [4, 32].

### 1.2.3 Working Principles of Protocols

Layered Graph drawing also aids in understanding the working principles of different protocols and security models in computer systems and networks. For example, the Bell-La Padula multilevel security model imposes that a process running at security level $k$ can read from objects at its own level or lower and can write to objects at its own level or higher [13]. Viewed as a layered graph drawing problem, we simply need that no edge is drawn vertically downward. The model can be understood from its drawing in Figure 1.4. Note that, this model uses layered drawings of directed graphs.

There are also numerous applications of layered drawings in various fields other than the above mentioned ones. Layered graph drawing algorithms have important applications to key computer technologies such as computer networks (depicting the structure of the physical network), software engineering

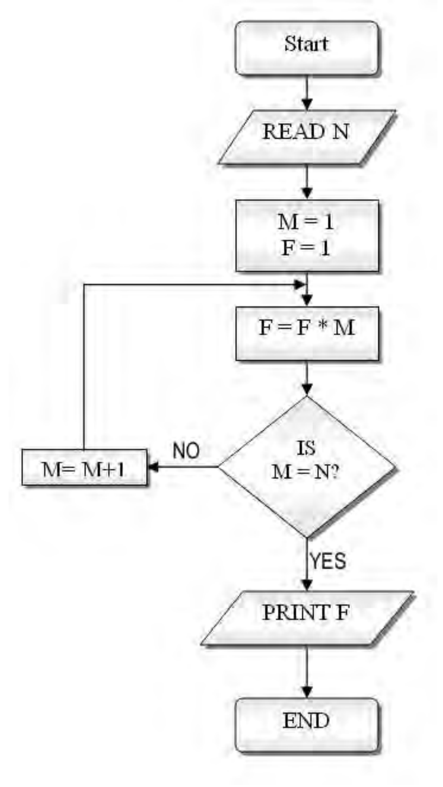


Figure 1.3: A flowchart for computing the factorial of a number.

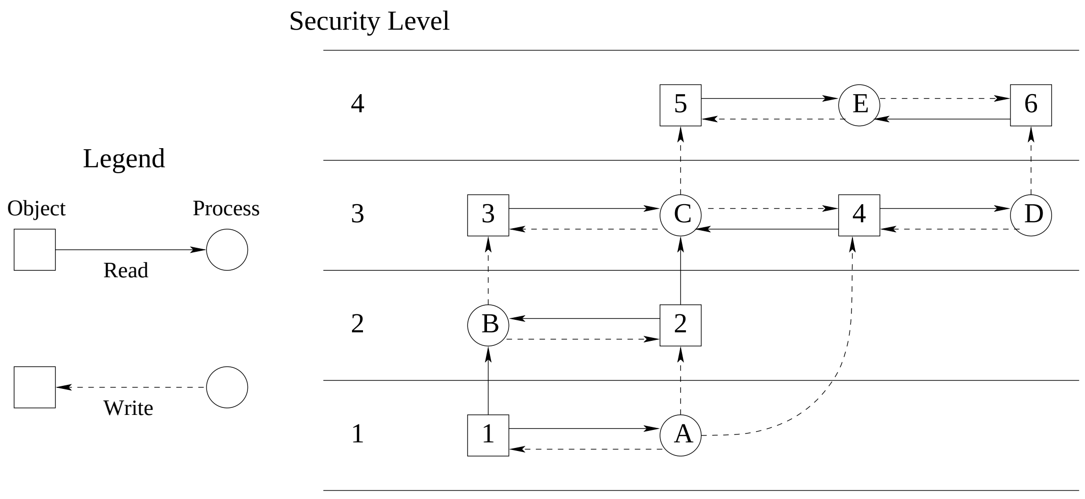


Figure 1.4: The Bell-La Padula multilevel security model, [13].

(data flow diagrams, subroutine-call graphs, program nesting trees), databases (entity-relationship diagrams), information systems (organization charts), real-time systems (state transition diagrams), artificial intelligence (knowledge representation diagrams) etc. Further applications can be found in other science and engineering disciplines, such as medical science (concept lattices), chemistry (molecular drawings), civil engineering (floorplan maps) and cartography (map schematics) [23].

## 1.3 Scope of This Thesis

In this section, we describe the scope of this thesis. We first define "minimum-layer drawings of trees", which is the central topic of this thesis. We also mention the previous results on this topic and the results obtained in this thesis.

A *minimum-layer drawing* of a tree $T$ is a layered drawing $\Gamma$ of $T$, where the number of layers in $\Gamma$ is the minimum among all the possible layered drawings of $T$. Figure 1.5(a) illustrates a tree $T$, Figure 1.5(b) and (c) depict two different layered drawings of $T$ occupying four layers and two layers, respectively. One can observe that at least two layers are required for any layered drawing of $T$, and hence the drawing in Figure 1.5(c) is a minimum-layer drawing of $T$. The problem of obtaining a minimum-layer drawing of a tree finds its motivation in many application areas of layered drawings [3, 15, 17, 30]. For example a minimum-layer drawing of a tree can be employed to minimize the number of rows in the design of a standard cell layout for a circuit.

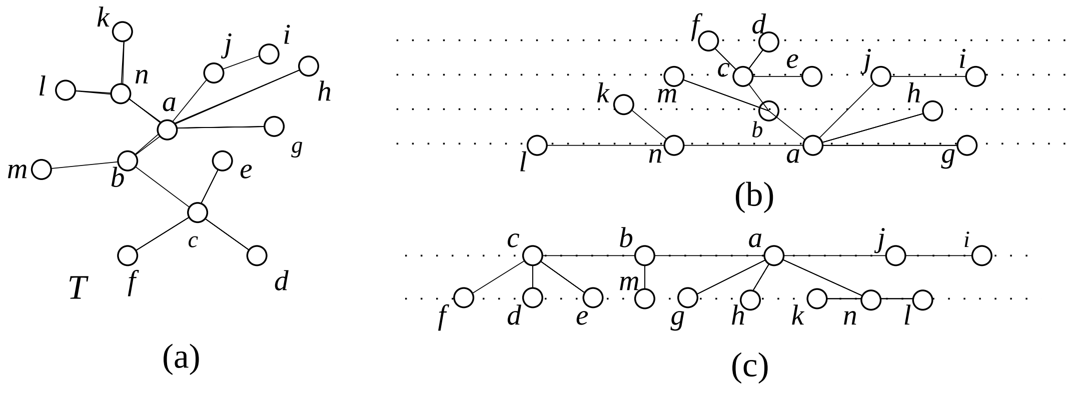


Figure 1.5: (a) A tree $T$, (b) a layered drawing of $T$, and (c) a minimum-layer drawing of $T$.

### 1.3.1 Previous Results

In 1988 de Fraysseix proposed the $k$-lines drawability problem which asks whether all planar graphs can be drawn on $k$ parallel lines lying on the surface of a cylinder [10]. Felsner *et al.* [10] gave a necessary condition for a tree to admit a layered drawing on $k$ layers for any fixed $k$. Linear-time algorithms are known [6, 10] for recognizing trees that admit layered drawings on at most three layers. Suderman gave an algorithm for drawing a tree $T$ on $\lceil 3h/2 \rceil$ layers, where $h$ is the pathwidth of $T$ [26].

There had also been some results on minimizing layers for special variants of layered drawings of trees. Dujmovic *et al.* [8] presented an algorithm to test whether a tree admits a proper drawing for a given value of $k$. By a slight modification of the algorithm, one can also test whether a tree admits an upright drawings on $k$ layers on $k$ layers or not. Using this algorithm, one can obtain polynomial-time algorithms for minimizing the number of layers in a proper or an upright drawing of a tree. Figure 1.6(b) illustrates a proper drawing of the tree of Figure 1.6(a) on the minimum number of layers. Figure 1.6(b) illustrates an upright drawing of the tree in Figure 1.6(a) on the minimum number of layers. There had also been some other works on proper, upright and short drawings of trees [1, 11, 26, 27].

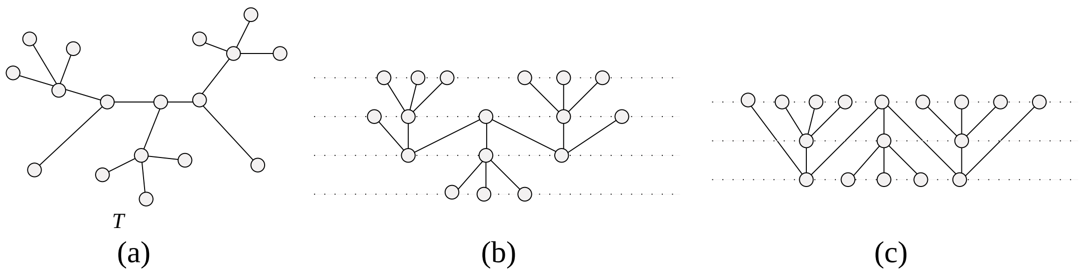


Figure 1.6: (a) A tree $T$, (b) a proper drawing of $T$ on the minimum number of layers, and (c) an upright drawing of $T$ on the minimum number of layers.

Recently, a linear-time algorithm has been given by Alam *et al.* [2] to obtain an "upward drawing" of a rooted tree $T$ on the minimum number of layers. An *upward drawing* of a rooted tree $T$ is a layered drawing of $T$ where no vertex is placed on a layer above its parent. Figure 1.7(b) illustrates an upward drawing of the rooted tree in Figure 1.7(a) on the minimum number of layers. For an unrooted tree $T$, the algorithm in [2] can also choose a vertex $w$ efficiently as illustrated in Figure 1.7(c) such that an upward drawing of $T$ rooted at $w$ occupies the minimum number of layers among all the upward drawings of $T$ rooted at any vertex of $T$. Figure 1.7(d) illustrates an upward drawing of $T$ rooted at $w$ that occupies the minimum number of layers among all the upward drawings of $T$ rooted at any vertex of $T$.

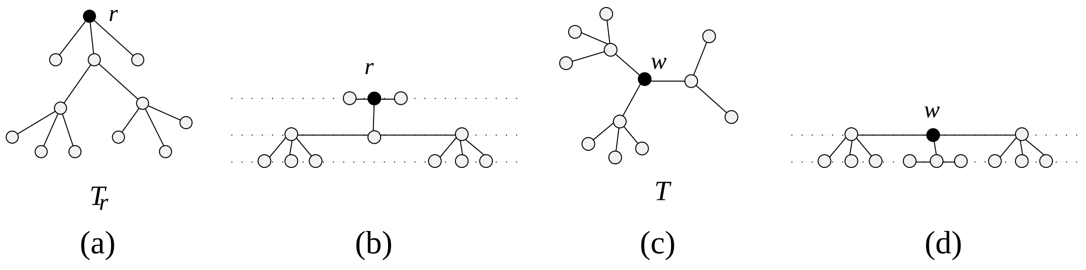


Figure 1.7: (a) A rooted tree $T_r$, (b) an upward drawing of $T_r$ on the minimum number of layers, (c) an unrooted tree $T$, and (d) an upward drawing of $T$ that occupies the minimum number of layers among all the upward drawings of $T$.

On the other hand, some other variants of finding a minimum-layer drawing of a tree are shown to be NP-hard [14, 18] where the drawing needs to satisfy some constraints on the placement of the vertices. However, for the general version of the problem, there is neither any polynomial-time algorithm nor any hardness result known so far.

### 1.3.2 Results in this Thesis

In this thesis, we first give a linear-time algorithm to obtain a “root-visible drawing” of a rooted tree on the minimum number of layers. Using this algorithm we then give a linear-time algorithm to obtain a minimum-layer drawing of a tree. For a brief summary, we now list the results presented in this thesis as follows.

- We give a linear-time algorithm for minimizing the number of layers in a “root-visible drawing” of a rooted tree. Let $T$ be a rooted tree with the root vertex $r$. A *root-visible drawing* of $T$ is a layered drawing $\Gamma$ of $T$ where there is a point $p$ below all the layers in $\Gamma$ such that a straight-line from $p$ to $r$ does not create any edge crossing with $\Gamma$.

- We give an algorithm that finds a suitable root vertex $r$ of a tree $T$ in linear time such that a "root-visible drawing" of $T$ with the root $r$ is also a minimum-layer drawing of $T$. We then use the algorithm for root-visible drawing to obtain a minimum-layer drawing of $T$.

## 1.4 Thesis Organization

The rest of this thesis is organized as follows. In Chapter 2, we give some basic terminology of graph theory and algorithmic theory. Chapter 3 presents a linear-time algorithm to obtain a minimum-layer root-visible drawing of a rooted tree. In Chapter 4, we prove that the root-visible drawing obtained by the algorithm in Chapter 3 is a minimum-layer root-visible drawing of the given rooted tree. We also give a linear-time algorithm for minimum-layer drawing of a tree in this chapter. Finally, Chapter 5 concludes the thesis with a summary of the results and the mention of some future works.

# Chapter 2

# Preliminaries

In this chapter we define some basic terminology of graph theory, graph drawing and algorithm theory, that we will use throughout the rest of this thesis. Definitions which are not included in this chapter will be introduced as they are needed. We cover, in Section 2.1, some definitions of standard graph-theoretical terms. We devote Section 2.2 to define terms related to planar graphs. Section 2.3 defines some drawing convention that are used in the algorithms discussed in this thesis. Finally, we introduce the notion of time complexity of algorithms in Section 2.4.

## 2.1 Basic Terminology

In this section we give some definitions of standard graph-theoretical terms used throughout this thesis. For readers interested in more details of graph theory we refer to [20, 21, 31].

### 2.1.1 Graphs

A *graph* $G$ is a tuple $(V, E)$ which consists of a finite set $V$ of *vertices* and a finite set $E$ of *edges*; each edge being an unordered pair of vertices. Figure 2.1 depicts a graph $G = (V, E)$ where each vertex in $V = \{v_1, v_2, \ldots, v_6\}$ is drawn as a small circle and each edge in $E = \{e_1, e_2, \ldots, e_8\}$ is drawn by a line segment.

We denote an edge joining two vertices $u$ and $v$ of the graph $G = (V, E)$ by $(u, v)$ or simply by $uv$. If $uv \in E$ then the two vertices $u$ and $v$ of the graph $G$ are said to be *adjacent*; the edge $uv$ is then said to be *incident* to the vertices

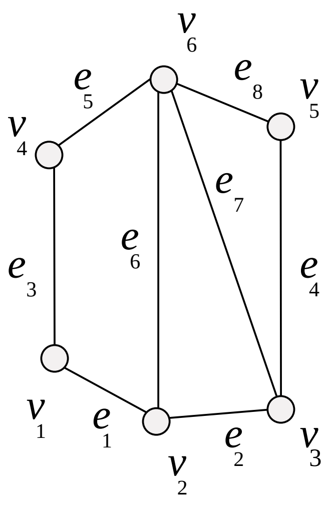


Figure 2.1: A graph with six vertices and eight edges.

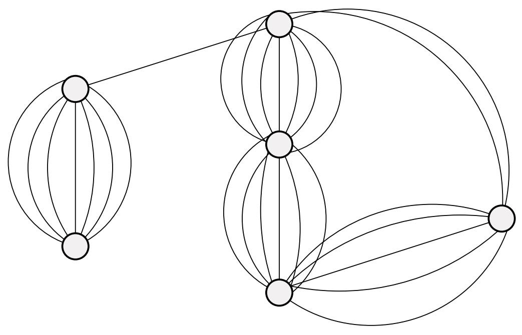
$u$ and $v$; also the vertex $u$ is said to be a neighbor of the vertex $v$ (and vice versa). The *degree* of a vertex $v$ in $G$, denoted by $d(v)$ or $deg(v)$, is the number of edges incident to $v$ in $G$. In the graph shown in Figure 2.1 vertices $v_1$ and $v_2$ are adjacent, and $d(v_6) = 4$, since four of the edges, namely $e_5, e_6, e_7$ and $e_8$ are incident to $v_6$.

### 2.1.2 Simple Graphs and Multigraphs

If a graph $G$ has no "multiple edges" or "loops", then $G$ is said to be a *simple graph*. *Multiple edges* join the same pair of vertices, while a *loop* joins a vertex with itself. The graph in Figure 2.1 is a simple graph.

A graph in which loops and multiple edges are allowed is called a *multigraph*. Multigraphs can arise from various applications. One example is the "call graph" that represents the telephone call history of a network. The graph in Figure 2.2(a) is a call graph that represents the call history among six subscribers. Note that there is no loop in this graph. Figure 2.2(b) illustrates another multigraph with multiple edges and loops.

Often it is clear from the context that the graph is simple. In such cases, a simple graph is called a *graph*. In the remainder of thesis we will only be concerned about simple graphs.



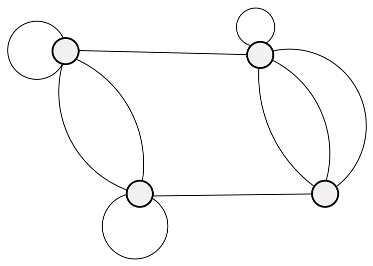
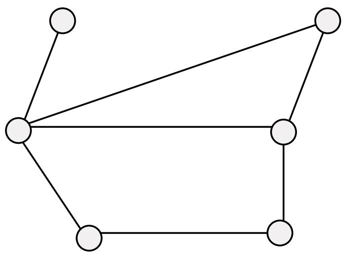


Figure 2.2: Multigraphs.

## 2.1.3 Directed and Undirected Graphs

In a *directed graph*, the edges do have a direction but in an *undirected graph*, the edges are undirected. Strictly speaking, each edge in a directed graph should be represented by a 2-tuple while for an undirected graph it should be represented by a 2-member subset of the vertex set. In Figure 2.3(a) and (b), we show an undirected and a directed graphs respectively. In this thesis, we will mean an undirected graph when we say "a graph" unless mentioned otherwise.



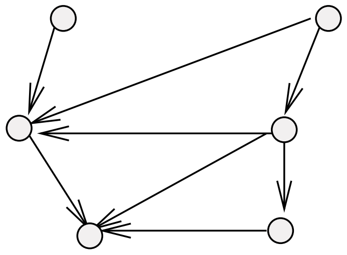


Figure 2.3: Undirected and directed graphs.

## 2.1.4 Subgraphs

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 two vertices in $V'$, then

$G'$ is said to be the *subgraph induced by* $V'$. Figure 2.4 depicts a subgraph of $G$ in Figure 2.1.

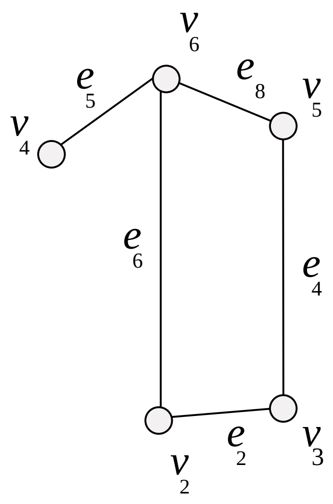


Figure 2.4: A subgraph of the graph in Figure 2.1.

We often construct new graphs from old ones by deleting some vertices or edges. If $v$ is a vertex of a given graph $G = (V, E)$, then $G - v$ is the subgraph of $G$ obtained by deleting the vertex $v$ and all the edges incident to $v$. More generally, if $V'$ is a subset of $V$, then $G - V'$ is the subgraph of $G$ obtained by deleting the vertices in $V'$ and all the edges incident to them. Then $G - V'$ is a subgraph of $G$ induced by $V - V'$. Similarly, if $e$ is an edge of a $G$, then $G - e$ is the subgraph of $G$ obtained by deleting the edge $e$. More generally, if $E' \subseteq E$, then $G - E'$ is the subgraph of $G$ obtained by deleting the edges in $E'$.

### 2.1.5 Paths and Cycles

A *walk*, $w = v_0, e_1, v_1, \ldots, v_{l-1}, e_l, v_l$, in a graph $G$ is an alternating sequence of vertices and edges of $G$, beginning and ending with a vertex, in which each edge is incident to the two vertices immediately preceding and following it. The vertices $v_0$ and $v_l$ are said to be the end-vertices of the walk $w$.

If the vertices $v_0, v_1, \ldots, v_l$ are distinct (except possibly $v_0$ and $v_l$), then the walk is called a *path* and usually denoted either by the sequence of vertices $v_0, v_1, \ldots, v_l$ or by the sequence of edges $e_1, e_2, \ldots, e_l$. The length of the path is $l$, one less than the number of vertices on the path. For any two vertices $u$ and $v$ of $G$, a $u, v$-path in $G$ is a path whose end-vertices are $u$ and $v$.

A walk or path $w$ is *closed* if the end-vertices of $w$ are the same. A closed path containing at least one edge is called a *cycle*.

## 2.1.6 Connectivity

A graph $G$ is *connected* if for any two distinct vertices $u$ and $v$ of $G$, there is a path between $u$ and $v$. A graph which is not connected is called a *disconnected graph*. A *(connected) component* of a graph is a maximal connected subgraph. The graph in Figure 2.5(a) is a connected graph since there is a path between every pair of distinct vertices of the graph. On the other hand, the graph in Figure 2.5(b) is a disconnected graph since there is no path between, say, $v_1$ and $v_5$. The graph in Figure 2.5(b) has two connected components as indicated by the dotted lines. Note that every connected graph has only one component; the graph itself.

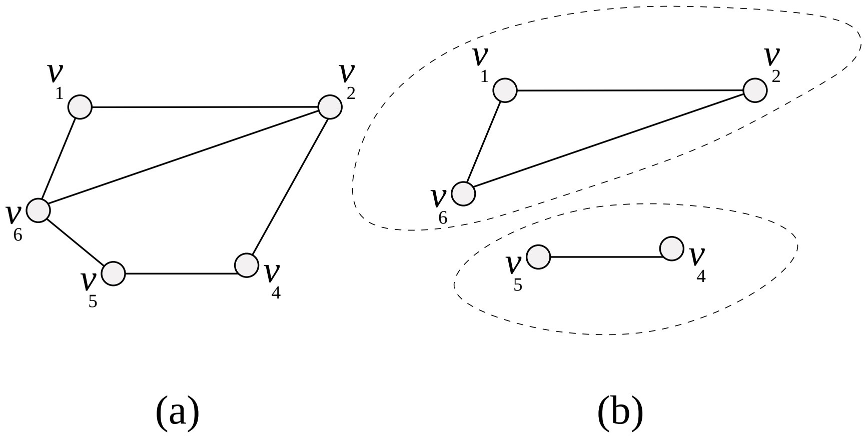


Figure 2.5: (a) A connected graph (b) a disconnected graph with two connected components.

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 $K_1$. We say that $G$ is *$k$-connected* if $\kappa(G) \geq k$. 2-connected and 3- connected graphs are also called biconnected and triconnected graphs, respectively. A *block* is a maximal biconnected subgraph of $G$. 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. If a vertex-cut contains exactly one vertex then we call the vertex a *cut vertex*.

A *tree* $T$ is a connected graph which contains no cycle. A vertex $u$ of $T$ having degree one in $T$ is called a *leaf* of $T$. A vertex $u$ of $T$ having degree greater than one in $T$ is called an *internal vertex* of $T$. A tree $T$ is called a *rooted tree* if one of the vertices $r$ of $T$ is considered as the *root* of $T$. In the following, we use the notation $T_r$ to denote a rooted tree with the root vertex $r$. The *parent* of a vertex $v$ in $T_r$ is the vertex that precedes $v$ in the $r, v$-path in $T_r$. All the neighbors of $v$ in $T_r$ other than its parent are called the *children* of $v$ in $T_r$. An *ancestor* of a vertex $v$ in $T_r$ is such a vertex $u$ of $T_r$ that $u$ is on the $r, v$-path in $T_r$. A *descendant* of $v$ in $T_r$ is such a vertex $u$ of $T_r$ that the $r, u$-path in $T_r$ contains $v$. A *subtree* of $T_r$ rooted at a vertex $v$ of $T_r$ is the subgraph of $T_r$ induced by the descendants of $v$ in $T_r$.

## 2.2 Planar Graphs

In this section we give some definitions related to planar graphs used in the remainder of the thesis. For readers interested in planar graphs we refer to [20].

### 2.2.1 Planar Graphs and Plane Graphs

A *planar drawing* of a graph $G$ is a two-dimensional drawing of $G$ in which no pair of edges intersect with each other except at their common end-vertex. A planar graph is a graph that has at least one planar drawing. A *planar embedding* of a graph $G$ is a data structure that defines a clockwise (or counter clockwise) ordering of the neighbors of each vertex of $G$ that corresponds to a planar drawing of the graph. Note that a planar graph may have an exponential number of embedding. Figure 2.6 shows two planar embeddings of the same planar graph.

A *plane graph* is a planar graph with a fixed planar embedding. A plane graph divides the plane into connected regions called *faces*. A finite plane graph $G$ has one unbounded face and it is called the *outer face* of $G$.

### 2.2.2 Dual Graphs

For a plane graph $G$, we often construct another graph $G^*$ called the (*geometric*) *dual* of $G$ as follows. A vertex $v_i^*$ is placed in each face $F_i$ of $G$; these are the

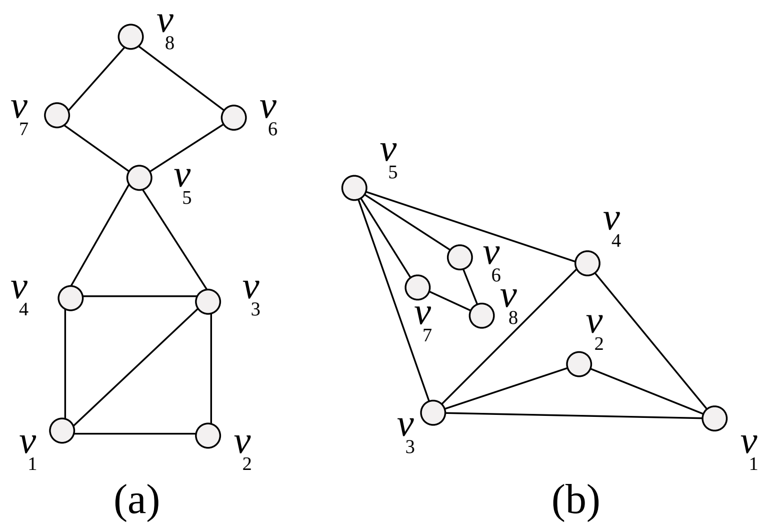


Figure 2.6: Two planar embeddings of the same planar graph.

vertices of $G^*$. Corresponding to each edge $e$ of $G$ we draw an edge $e^*$ which crosses $e$ (but no other edge of $G$) and joins the vertices $v_i^*$ which lie in the faces $F_i$ adjoining $e$; these are the edges of $G^*$. The construction is illustrated in Figure 2.7; the vertices $v_i^*$ are represented by small white circles, and the

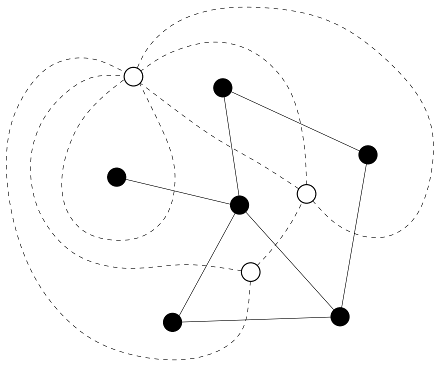


Figure 2.7: A plane graph $G$ and its dual graph $G^*$.

edges $e^*$ of $G^*$ by dotted lines. $G^*$ is not necessarily a simple graph even if $G$ is simple. Clearly the dual $G^*$ of a plane graph $G$ is also plane. One can easily observe the following lemma.

**Lemma 2.2.1** *Let $G$ be a connected plane graph with $n$ vertices, $m$ edges and $f$ faces, and let the dual $G^*$ have $n^*$ vertices, $m^*$ edges and $f^*$ faces; then $n^* = f$, $m^* = m$, and $f^* = n$.*

Clearly the dual of the dual of the plane graph $G$ is the original graph $G$. However a planar graph may give rise to two or more geometric duals since the plane embedding is not necessarily unique.

## 2.3 Drawing Conventions

In this section we introduce some conventional drawing styles, which are found suitable in different application domain. The different drawing styles vary owing to different representations of vertices and edges. Depending on the purpose and objective, the vertices are typically represented with points or boxes and edges are represented with simple Jordan curves [21]. A few of the most important drawing styles are introduced below.

### 2.3.1 Planar Drawings

A drawing $\Gamma$ of a graph $G$ is planar if no two edges intersect with each other except at their common end-vertices. In Figure 2.8(a) and (b), we show a planar and a non-planar drawing of the same graph.

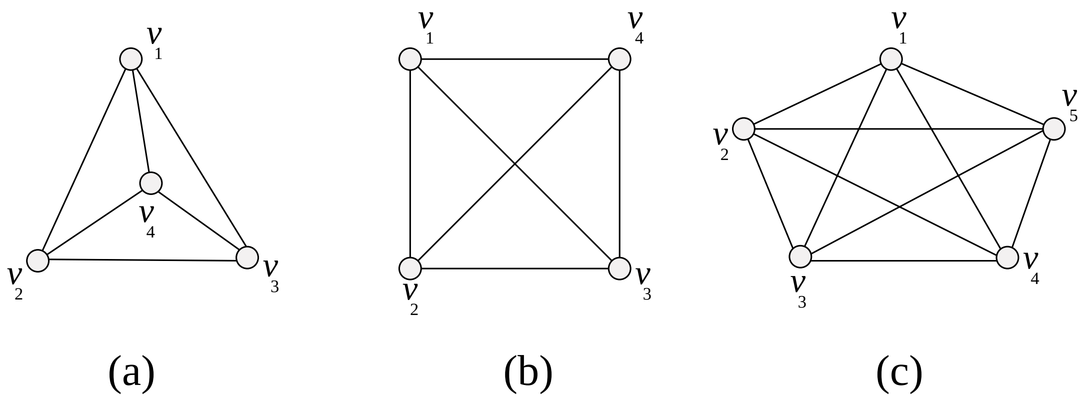


Figure 2.8: (a) A planar drawing, (b) a non-planar drawing of the graph drawn in (a), and (c) a graph which does not have a planar drawing.

Planar drawings of graphs are more convenient than non-planar drawings because, as shown empirically in [22], the presence of edge-crossings in a drawing of a graph make it more difficult for a person to understand the information being modeled. Unfortunately, not all graphs have a planar drawing. Figure 2.8(c) is an example of one such graph.

### 2.3.2 Straight-line Drawings

A *straight-line drawing* of a graph $G$ is a drawing of $G$ in which each edge is drawn as a straight line segment, as illustrated in Figure 2.9. Wagner [28], Fary

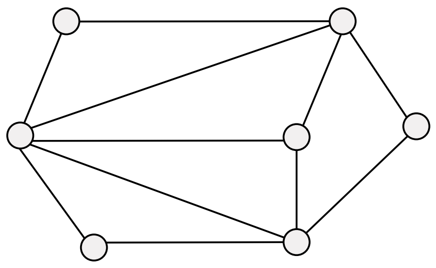


Figure 2.9: A straight line drawing.

[9] and Stein [25] independently proved that every planar graph has a straight line drawing.

### 2.3.3 Layered Drawings

A *layered drawing* of a planar graph $G$ is a drawing of $G$ where the vertices of $G$ are placed on a set horizontal lines called *layers*, and each edge of $G$ is drawn as a straight-line segments between its end-vertices without creating any crossing with the other edges. Let $\Gamma$ be a layered drawing of a planar graph $G$. $\Gamma$ is called a *short drawing* of $G$ if the end-vertices of each edge of $G$ is placed on the same or adjacent layers in $\Gamma$. $\Gamma$ is called a *proper drawing* of $G$ if the end-vertices of each edge of $G$ lie on adjacent layers in $\Gamma$. $\Gamma$ is called an *upright drawing* of $G$ if the end-vertices of each edge of $G$ are placed at different layers in $\Gamma$.

### 2.3.4 Grid Drawings

A drawing of a graph is called a *grid drawing* if the vertices are all located at grid points of an integer grid as illustrated in Figure 2.10. Note that this a special class of layered drawings where not only the vertical but also the vertical position of the vertices are at integer distance from each other. This drawing approach also overcomes the following problems in graph drawing with real number arithmetic [21].

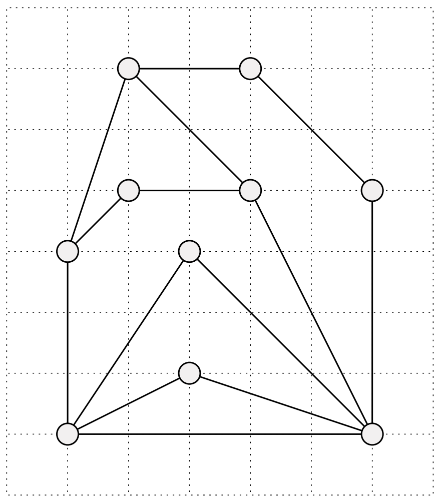


Figure 2.10: A grid drawing.

(i) When the embedding has to be drawn on a raster device, real vertex coordinates have to be mapped to integer grid points, and there is no guarantee that a correct embedding will be obtained after rounding.

(ii) Many vertices may be concentrated in a small region of the drawing. Thus the embedding may be messy, and line intersections may not be detected.

(iii) One cannot compare area requirement for two or more different drawings using real number arithmetic, since any drawing can be fitted in any small area using magnification.

The *size* of an integer grid required for a grid drawing is measured by the size of the smallest rectangle on the grid which encloses the drawing. The *width* $W$ of the grid is the width of the rectangle and the *height* $H$ of the grid is the height of the rectangle. The grid size is usually described as $W \times H$.

It is a very challenging problem to draw a plane graph on a grid of the minimum size. In recent years, several works are devoted to this field [5, 7, 24]; for example, every plane graph of $n$ vertices has a straight line grid drawing on a grid size $W \times H \leq (n-1) \times (n-1)$.

## 2.4 Complexity of Algorithms

In this section we briefly introduce some terminologies related to *complexity* of algorithms. For interested readers, we refer the book of Garey and Johnson

[12].

The most widely accepted complexity measure for an algorithm is the *running time*, which is expressed by the number of operations it performs before producing the final answer. The number of operations required by an algorithm is not the same for all problem instances. Thus, we consider all inputs of a given size together, and we define the *complexity of the algorithm for that input size* to be the worst case behavior of the algorithm on any of these inputs. Then the running time is a function of size $n$ of the input.

### 2.4.1 The Notation $O(n)$

In analyzing the complexity of an algorithm, we are often interested only in the "asymptotic behavior", that is, the behavior of the algorithm when applied to very large inputs. To deal with such a property of functions we shall use the following notations for asymptotic running time. Let $f(n)$ and $g(n)$ are the functions from the positive integers to the positive reals, then we write $f(n) = O(g(n))$ if there exists positive constants $c_1$ and $c_2$ such that $f(n) \leq c_1 g(n) + c_2$ for all $n$. Thus the running time of an algorithm may be bounded from above by phrasing like "takes time $O(n^2)$".

### 2.4.2 Polynomial Algorithms

An algorithm is said to be *polynomially bounded* (or simply *polynomial*) if its complexity is bounded by a polynomial of the size of a problem instance. Examples of such complexities are $O(n)$, $O(nlogn)$, $O(n^{100})$, etc. The remaining algorithms are usually referred as *exponential* or *non-polynomial.* Examples of such complexity are $O(2^n)$, $O(n!)$, etc. When the running time of an algorithm is bounded by $O(n)$, we call it a *linear-time* algorithm or simply a *linear* algorithm.

### 2.4.3 NP-complete Problems

There are a number of interesting computational problems for which it has not been proved whether there is a polynomial time algorithm or not. Most of them are "NP-complete", which we will briefly explain in this section.

The state of algorithms consists of the current values of all the variables and the location of the current instruction to be executed. A *deterministic algorithm* is one for which each state, upon execution of the instruction, uniquely determines at most one of the following state (next state). All computers, which exist now, run deterministically. A problem $Q$ is in the *class P* if there exists a deterministic polynomial-time algorithm which solves $Q$. In contrast, a *non-deterministic algorithm* is one for which a state may determine many next states simultaneously. We may regard a non-deterministic algorithm as having the capability of branching off into many copies of itself, one for the each next state. Thus, while a deterministic algorithm must explore a set of alternatives one at a time, a non-deterministic algorithm examines all alternatives at the same time. A problem $Q$ is in the *class NP* if there exists a non-deterministic polynomial-time algorithm which solves $Q$. Clearly, $P \subseteq NP$.

Among the problems in *NP* are those that are hardest in the sense that if one can be solved in polynomial-time then so can every problem in *NP*. These are called *NP-complete* problems. The class of *NP*-complete problems has the following interesting properties.

(a) No *NP*-complete problem can be solved by any known polynomial algorithm.

(b) If there is a polynomial algorithm for any *NP*-complete problem, then there are polynomial algorithms for all *NP*-complete problems.

Sometimes we may be able to show that, if problem $Q$ is solvable in polynomial time, all problems in *NP* are so, but we are unable to argue that $Q \in NP$. So $Q$ does not qualify to be called *NP*-complete. Yet, undoubtedly $Q$ is as hard as any problem in *NP*. Such a problem $Q$ is called *NP-hard*.

# Chapter 3

# Root-Visible Drawings

## 3.1 Introduction

A *root-visible drawing* of a rooted tree $T$ is a layered drawing $\Gamma$ of $T$ where there is a point $p$ below all the layers in $\Gamma$ such that a straight-line from $p$ to the root of $T$ does not create any edge crossing with $\Gamma$. A root-visible drawing of $T$ that occupies the minimum number of layers among all the root-visible drawings of $T$ is called a *minimum-layer root-visible* drawing of $T$. Figure 3.1(a) illustrates a rooted tree $T$ where the vertex 1 is the root of $T$. Fig. 3.1(b) and (c) depict two different root-visible drawings of $T$ occupying three layers and two layers, respectively. One can observe that at least two layers are required for any root-visible drawing of $T$, and hence the drawing in Fig. 3.1(c) is a minimum-layer root-visible drawing of $T$. In this chapter we give a linear-time algorithm to obtain a root-visible drawing of a rooted tree $T$. In the next chapter, we will prove that the drawing obtained by this algorithm is a minimum-layer root-visible drawing of $T$.

Our algorithm is roughly outlined as follows. Let $T$ be a rooted tree with the root vertex $r$. To obtain a root-visible drawing of $T$ we first partition $T$ into several smaller rooted trees through a bottom-up computation. We then compute a minimum-layer root-visible drawing $\Gamma$ of $T$ such that the drawing of the partitions of $T$ in $\Gamma$ are minimum-layer root-visible drawings of the corresponding partitions. If $T$ consists of only one vertex then the vertex itself is the only partition of $T$. Otherwise, we compute the partitions of $T$ from the partitions of the subtrees of $T$. Similarly, we obtain the drawing of $T$ using the

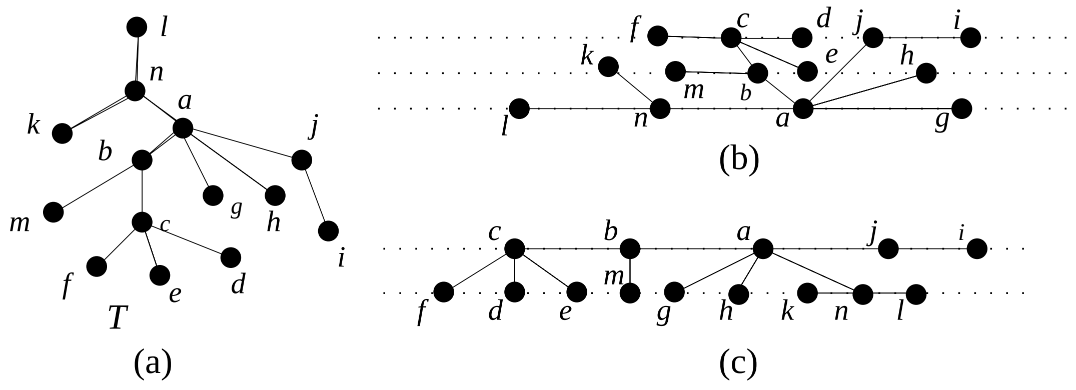


Figure 3.1: (a) A rooted tree $T$ with the root $l$, (b) a root-visible drawing of $T$, and (c) a minimum-layer root-visible drawing of $T$.

drawings of the subtrees of $T$. We thus construct a minimum-layer root-visible drawing of $T$. Figure 3.2(a) depicts a rooted tree $T$ and Figure 3.2(b) illustrates the partitions of $T$ and a root-visible drawing of $T$.

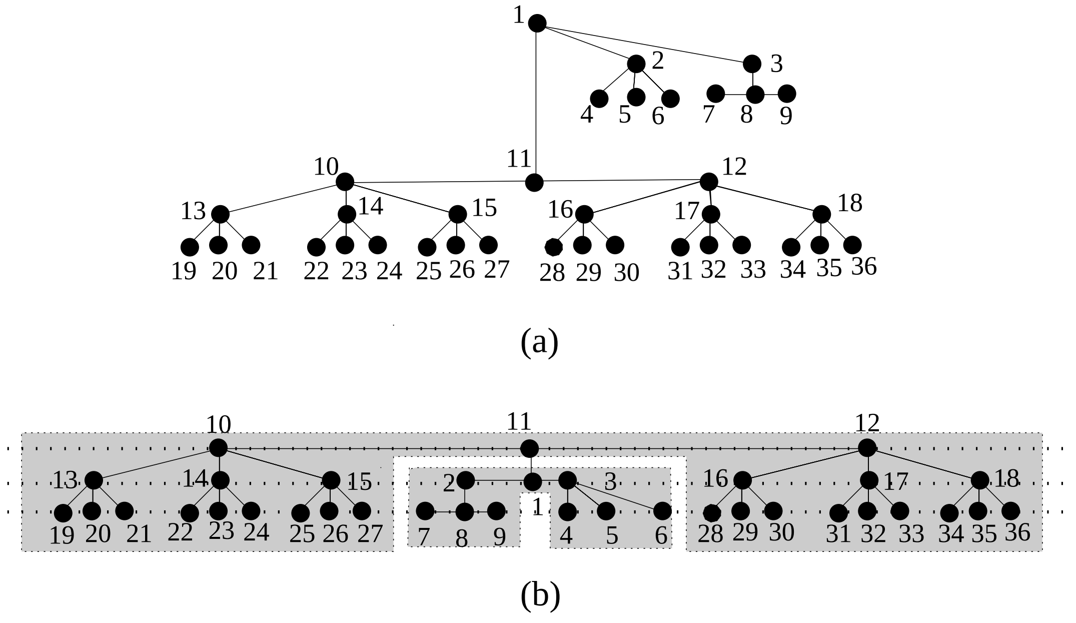


Figure 3.2: (a) A rooted tree $T$, and (b) a root-visible drawing of $T$ and the partitions of $T$.

The rest of this chapter is organized as follows. Section 2 describes some definitions and presents preliminary results. Section 3 gives an algorithm for obtaining a root-visible drawing of a rooted tree $T$. Section 4 proves the time complexity of the drawing algorithm described in Section 3. Finally, Section 6 concludes the chapter.

## 3.2 Preliminaries

In this section, we define some basic terminologies that will be used throughout the chapter. For the graph theoretic terminologies not illustrated here, see [21].

Let $G = (V, E)$ be a simple graph with the vertex set $V$ and the edge set $E$. A vertex $u \in V$ is adjacent to a vertex $v \in V$ if there is an edge $(u, v) \in E$. The *degree* of a vertex $v$ is the number of vertices adjacent to $v$ in $G$. We denote by $deg(v)$ the degree of $v$ in $G$. For $V' \subseteq V$, $G - V'$ denotes the graph obtained from $G$ by deleting all vertices in $V'$ together with all edges incident to them. For a subgraph $G'$ of $G$, we denote by $G - G'$ the graph obtained from $G$ by deleting all vertices in $G'$ together with all edges incident to them. A graph $G$ is *connected* if for any two distinct vertices $u$ and $v$ there is a path between $u$ and $v$ in $G$. The maximal connected subgraphs of a graph $G$ is called components of $G$.

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 the *child* of $u$. A vertex $v$ of $T$ is called a *leaf* if it has no children; otherwise $v$ 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 $i$.

Let $\Gamma$ be a layered drawing of $T$ and $k$ be the number of layers in $\Gamma$. We denote by $l_i$, $1 \leq i \leq k$, the $i$-th layer of $\Gamma$ starting from the topmost layer. For a subgraph $T'$ of $T$, $\Gamma(T')$ denotes the drawing of $T'$ contained in $\Gamma$. The *bounding box* of $\Gamma$ is the smallest rectangle enclosing $\Gamma$ with one side parallel to the layers of $\Gamma$. Let $v$ be a vertex in $T$. Then $v$ has *left-visibility* in $\Gamma$ if there is a point $p$ to the left of $v$ on the same layer as $v$ outside the bounding box of $\Gamma$ such that the straight-line between $v$ and $p$ does not create any edge crossing with $\Gamma$. Similarly, we define the *right-visibility* of $v$ in $\Gamma$. The vertex $v$

has *side-visibility* in $\Gamma$ if $v$ has either left-visibility or right-visibility. Again, the vertex $v$ has *top-visibility* in $\Gamma$ if there is a point $p$ above the bounding box of $\Gamma$ such that the straight-line between $v$ and $p$ does not create any edge crossing with $\Gamma$. Similarly, we define the *bottom-visibility* of $v$ in $\Gamma$. For example, in the drawing of Fig. 3.1(b), the vertices $k$, $h$, $i$ and $a$ have left-visibility, right-visibility, top-visibility and bottom-visibility, respectively. The vertices $k$, $h$ and $i$ have side-visibility, but the vertex $a$ does not have side-visibility. We now have the following fact.

**Fact 3.2.1** *Let $\Gamma$ be a layered drawing of a tree $T$ on $k$ layers such that a vertex $x$ of $T$ has top-visibility in $\Gamma$. Then $T$ admits a layered drawing $\Gamma'$ on $k$ layers where $x$ has bottom-visibility in $\Gamma'$. Moreover, if a vertex $y$ of $T$ has side-visibility in $\Gamma$, then $y$ has top-visibility and bottom-visibility in $\Gamma$.*

We also have the following lemma that gives a visibility property of a layered drawing of a tree.

**Lemma 3.2.2** *Let $x$ and $y$ be any two vertices of a tree $T$ and let $\Gamma$ be a layered drawing of $T$ on $k$ layers such that for each component $C$ of $T-P$, the vertex of $C$ adjacent to a vertex of $P$ has top-visibility in $\Gamma(C)$. Then there is a layered drawing $\Gamma'$ of $T$ on at most $k+1$ layers where $x$ has top-visibility and $y$ has side-visibility in $\Gamma'$.*

**Proof.** Let $P = \langle v_0(=x), e_1, v_1, \ldots, e_l, v_l(=y)\rangle$ be the unique path from $x$ to $y$ in $T$. Then each component of $T-P$ has a vertex adjacent to a vertex $v_i$, $0 \leq i \leq l$, of $P$. We now construct a layered drawing $\Gamma'$ of $T$ on $k+1$ layers where $x$ has top-visibility and $y$ has side-visibility as follows. We denote by $l_i$, $1 < i \leq k+1$, the $i$-th layer of $\Gamma'$ starting from the topmost layer. We first put the vertices $v_i$, $0 \leq i \leq l$, on layer $l_1$ such that the vertex $v_i$ is placed to the left of $v_{i+1}$, $1 \leq i < l$. We next place the drawings $\Gamma(C)$ of each component $C$ of $T-P$ on $l_2$ to $l_{k+1}$ layers and draw the edge that connects $C$ with the corresponding vertex on $P$. It is easy to observe that the order of placement of the vertices of $P$ gives an order of the placement of the components of $T-P$ such that the drawing can be completed without any edge crossing. $\mathcal{Q.E.D.}$

Figure 3.3(a) illustrates a layered drawing $\Gamma$ of a tree $T$ on 7 layers where the vertex $x$ has top-visibility and the vertex $y$ does not have side-visibility.

Figure 3.3(b) illustrates the layered drawing $\Gamma'$ of $T$ on 8 layers where $x$ still has top-visibility and $y$ has side-visibility.

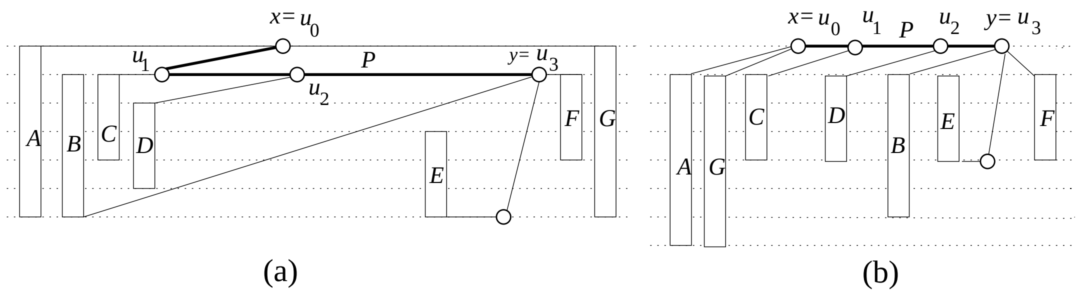


Figure 3.3: Illustration for the proof of Lemma 3.2.2.

Let $T$ be a rooted tree with the root vertex $r$. A *root-visible drawing* of $T$ is a layered drawing $\Gamma$ of $T$ where $r$ has bottom-visibility in $\Gamma$. A root-visible drawing of $T$ is called a *minimum-layer root-visible drawing* of $T$ if the number of layers used in the drawing is the minimum among all the possible root-visible drawings of $T$.

Let $L = \langle x_1, x_2, \ldots, x_k \rangle$ be a list of $k$ integers. Then $k$ is the *cardinality* of $L$ which is denoted by $|L|$. $L$ is called a *decreasing list of integers* if $x_1 > x_2 > \ldots > x_k$. Let $L$ be a decreasing list of integers. Then $x_1$ is the *largest integer in* $L$ and is denoted by $L_1$. Similarly $x_k$ is the *smallest integer in* $L$ and is denoted by $L_f$. We now define the notion of "inserting an integer" $n$ into $L$. Let $l$ be the smallest integer greater than or equal to zero such that $n+l \notin L$. Then *inserting* $n$ *into* $L$ yields a decreasing list $L'$ of integers where $L' = \langle x_1, \ldots, x_{i-1}(> n+l), n+l \rangle$. The *index of insertion* is the cardinality of $L'$. For example, inserting 4 into the decreasing list of integers $L = \langle 9, 8, 5, 4, 3, 1 \rangle$ would yield the new decreasing list of integers $L' = \langle 9, 8, 6 \rangle$. Here the index of insertion is 3.

Let $T$ be a rooted tree with the root vertex $r$. We denote by $V(T)$ the vertex set of $T$. Let $P(T) = \langle\ T_1,\ T_2,\ \ldots,\ T_k\ \rangle$ be a list of rooted trees where $V(T_i)$ induces a connected subgraph of $T$ for $1 \le i \le k$ and none of the trees is empty except for $T_k$. We call $P(T)$ a *linear partition* of $T$ if the following conditions hold.

(i) abc $\bigcup_{i=1}^{k} V(T_i) = V(T)$ and $V(T_i)$ is disjoint with $V(T_j)$ for $1 \le i \ne j \le k$.

(ii) $r$ is the root of $T_k$ if $T_k$ is not empty; otherwise $r$ is the root of $T_{k-1}$.

(iii) The parent of any vertex $v$ in $T_i$, $1 \le i \le k$, is also the parent of $v$ in $T$.

(iv) For $2 \leq i \leq k-1$, there is a vertex $v_i$ of $T_i$, which is the parent of the root $u_{i-1}$ of $T_{i-1}$ in $T$. We call $v_i$ the *leg* of $T_i$. If $T_k$ is not empty, then the parent $v_k$ of the root $u_{k-1}$ in $T$ is also in $T_k$ and is called the *leg* of $T_k$. Note that $T_1$ does not have a leg.

Let $P(T) = \langle\ T_1,\ T_2,\ \ldots,\ T_k\ \rangle$ be a linear partition of a rooted tree $T$. We denote by $u_i$, $1 \leq i \leq k$, the root of $T_i$ and by $v_i$, $2 \leq i \leq k$, the leg of $T_i$.

## 3.3 Root-Visible Drawings of Trees

In this section we give an algorithm for obtaining a root-visible drawing of a rooted tree $T$.

We define a "label" of $T$ to be a decreasing list of integers and denote it by $L(T)$. Let us assume that $L(T) = \langle x_1, x_2, \ldots, x_k \rangle$. In Theorem 3.3.1, we find a linear partition $P(T) = \langle\ T_1,\ T_2,\ \ldots,\ T_k\ \rangle$ of $T$ such that $k = |L(T)|$. We also construct a root-visible drawing $\Gamma$ of $T$ where for each partition $T_i$, $1 \leq i \leq k$, $\Gamma(T_i)$ is a root-visible drawing of $T_i$ on $x_i$ layers.

**Theorem 3.3.1** *Let $T$ be a rooted tree with the root $r$. Then one can define a label $L(T) = \langle x_1, x_2, \ldots,\ x_k \rangle$ and a linear partition $P(T) = \langle\ T_1,\ T_2,\ \ldots,\ T_k \rangle$ of $T$ such that $T$ admits a layered drawing $\Gamma$ on $x_1$ layers satisfying conditions (a)–(d).*

*(a) For $1 \leq i \leq k$, $\Gamma(T_i)$ occupies $x_i$ layers.*

*(b) For $2 \leq i \leq k$, $v_i$ is on the $l_1$ layer in $\Gamma(T_i)$.*

*(c) For $1 \leq i \leq k$, $u_i$ has bottom-visibility and is either on $l_1$ or $l_{x_i}$ layer in $\Gamma(T_i)$.*

*(d) If $T_k$ is not empty, then $r$ has side-visibility in $\Gamma(T_k)$; otherwise there is a point on the $l_2$ layer in $\Gamma(T_{k-1})$ that has both bottom-visibility and visibility from $r$.*

**Proof.** Let $n$ be the number of vertices in $T$. If $n = 1$, $r$ is the only vertex in $T$ and we define $L(T) = \langle 1 \rangle$. It is easy to see that the linear partition $P(T) = \langle T \rangle$ satisfies the claim. We thus assume that $n > 1$ and for any rooted tree $T'$ with less than $n$ vertices, one can find a linear partition satisfying (a)–(d).

Let $c_1$, $c_2$, …, $c_p$ be the children of $r$ in $T$ and let $L(T_{c_1})$, $L(T_{c_2})$, …, $L(T_{c_p})$ be the labels of the subtrees $T_{c_1}$, $T_{c_2}$, …, $T_{c_p}$ of $T$ rooted at the vertices $c_1$, $c_2$, …, $c_p$, respectively. We assume that $L(T_{c_1}) = \langle x_1, \ldots, x_k \rangle$. Thus by induction hypothesis, there is a layered drawing $\Gamma_1$ of $T_{c_1}$ on $x_1$ layers and a linear partition $P(T_{c_1}) = \langle T_1, \ldots, T_k \rangle$ of $T_{c_1}$ satisfying the conditions (a)–(d). For notational convenience we denote by $z_i$ the value of $L(T_{c_i})_1$, $1 \leq i \leq p$. Without loss of generality, we also assume that $z_1 \geq z_2 \geq \ldots \geq z_p$ and if $z_i = z_{i+1}$ for some $1 \leq i \leq p-1$, then $|L(T_i)| \geq |L(T_{i+1})|$. We now have the following cases to consider.

*Case 1.* $z_2 < x_k$. Since $x_k > z_1 \geq z_2 \geq \ldots \geq z_p$, by induction hypothesis and by Fact 3.2.1 each of the trees $T_{c_i}$, $2 \leq i \leq p$, admits a layered drawing $\Gamma_i$ on less than $x_k$ layers with the top-visibility of the root. We now define a linear partition $P(T)$ of $T$ as $P(T) = \langle T_1, \ldots, T_{k-1}, T'_k = T_k \cup (T - T_{c_1}) \cup \{(r, u_k)\} \rangle$. Note that the root and leg of all the partitions in $P(T)$ are the same as the corresponding partitions in $P(T_{c_1})$ other than the roots of $T_k$ and $T'_k$ where the root of $T'_k$ is $r$. We now obtain a layered drawing $\Gamma$ of $T$ from $\Gamma_1$ satisfying the conditions (a)–(d) as follows. Without loss of generality, we assume that $u_k$ is placed on the $l_1$ layer and has right-visibility in $\Gamma_1(T_k)$. We place $r$ on the $l_1$ layer of $\Gamma_1(T_k)$ to the right of $u_k$ and add the edge $(r, u_k)$ using a straight-line segment without edge crossings. Finally we place the drawings $\Gamma_i$ for $2 \leq i \leq p$ on $l_2$ to $lx_k$ layers in $\Gamma_1(T_k)$ and add the edges $(r, c_i)$ for $2 \leq i \leq p$ to complete the drawing. It is easy to see that $P(T)$ and $\Gamma$ satisfy the conditions (a)–(d). An example for this case is illustrated in Fig. 3.4.

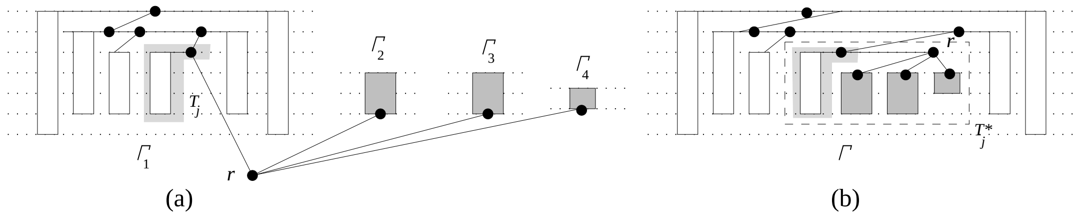


Figure 3.4: Illustration for Case 1.

*Case 2.* $z_2 \in L(T_{c_1})$. We first assume that $z_2 = x_k$. If $z_2 > z_3$ and $|L(T_{c_2})| = 1$, then we define $L(T) = \langle x_1, \ldots, x_k, 0 \rangle$. We now define a linear partition $P(T)$ of $T$ as follows. $P(T) = \langle T_1, \ldots, T_{k-1}, T'_k = T_k \cup (T - T_{c_1}) \cup \{(r, u_k)\}, T^0 \rangle$. Here $T^0$ represents an empty tree. We now obtain a layered drawing $\Gamma$ of $T$ from

$\Gamma_1$ satisfying the conditions (a)–(d) as follows. Without loss of generality, we assume that $u_k$ is placed on the $l_1$ layer and has right-visibility in $\Gamma_1(T_k)$. We place $r$ on the $l_1$ layer of $\Gamma_1(T_k)$ to the right of $u_k$ and add the edge $(r, u_k)$ using a straight-line segment without edge crossings. Since $L(T_{c_2}) = \langle x_k \rangle$, $T_{c_2}$ admits a layered drawing $\Gamma_2$ on $x_k$ layers keeping the side-visibility of $c_2$. We thus place $\Gamma_2$ on the $x_k$ layers of $\Gamma_1(T_k)$ to the right of $r$, possibly after mirroring with respect to $y$-axis and add the edge $(r, c_2)$ using a straight-line segment without edge crossings. Again since $x_k = z_2 > z_3 \geq \ldots \geq z_p$, by induction hypothesis and by Fact 3.2.1, each of the trees $T_{c_i}$, $3 \leq i \leq p$, admits a layered drawing $\Gamma_i$ on less than $x_k$ layers with the top-visibility of the root. We thus finally place the drawings $\Gamma_i$ for $3 \leq i \leq p$ in $\Gamma_1(T_k)$ on the $l_2$ to $l_{x_k}$ layers and add the edges $(r, c_i)$ for $3 \leq i \leq p$ to complete the drawing. It is easy to see that $P(T)$ and $\Gamma$ satisfies the conditions (a)–(d). An example for this case is illustrated in Fig. 3.5.

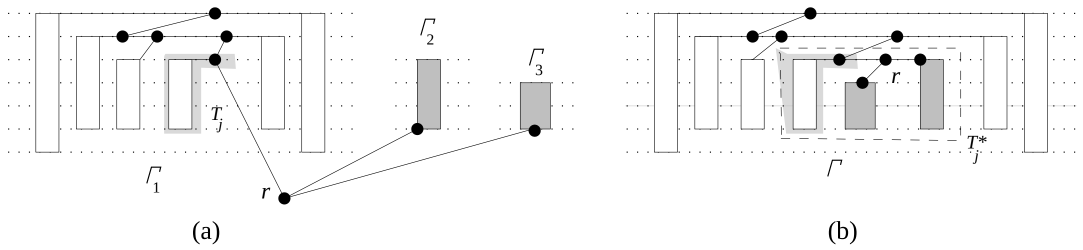


Figure 3.5: Illustration for Case 2 when $z_2 = x_k$, $z_2 > z_3$ and $|L(T_{c_2})| = 1$.

On the other hand, if $z_2 > x_k$ or if $z_2 = z_3$ or $|L(T_{c_2})| > 1$, then $L(T)$ is obtained by inserting $z_2 + 1$ into $L(T_{c_1})$. Let $j$ be the index of insertion. We now define a linear partition $P(T)$ of $T$ as follows. $P(T_r) = \langle T_1, \ldots, T_{j-1}, T'_j = (T - \bigcup_{i=1}^{j-1} T_i) \cup \{(u_{j-1}, v_j)\} \rangle$. We now obtain a layered drawing $\Gamma$ of $T$ from $\Gamma_1$ satisfying the conditions (a)–(d) as follows. We first define a label $L' = \langle x_1, \ldots, x_{j-1}, x_j + 1 \rangle$ and a linear partition $P' = \langle T_1, \ldots, T_{j-1}, T^*_j = T_{c_1} - \bigcup_{i=1}^{j-1} T_i) \rangle$ of $T_{c_1}$. By Lemma 3.2.2, $T^*_j$ admits a layered drawing on $x_j + 1$ layers where the leg vertex of $T_j$ (if any) has top-visibility and $c_1$ has side-visibility. Thus $T_{c_1}$ admits a layered drawing that satisfies conditions (a)–(d) for $L'$ and $P'$. We then obtain a desired layered drawing of $T$ in a similar way as in Case 1. An example for this case is illustrated in Fig. 3.6.

*Case 3.* $z_2 \notin L(T_{c_1})$ *and* $L(T_{c_2})_1 > x_k$. We have the following subcases.

*Subcase 3A.* $|L(T_{c_2})| > 1$. In this case $L(T)$ is obtained by inserting $z_2 + 1$ into

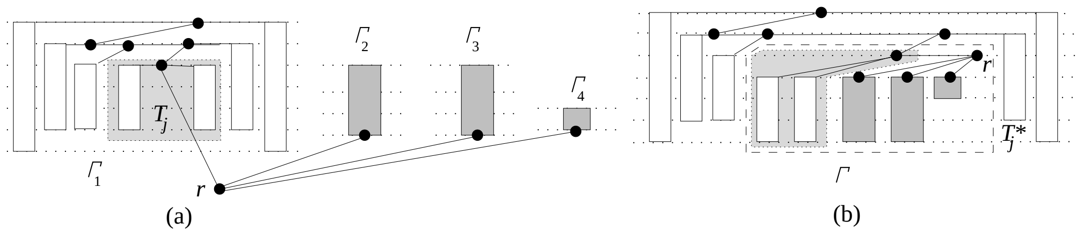


Figure 3.6: Illustration for Case 2 when $z_2 = z_3$ or $|L(T_{c_2})| > 1$.

$L(T_{c_1})$. Let $j$ be the index of insertion. We now define a linear partition $P(T)$ of $T$ as follows. $P(T_r) = \langle T_1, \ldots, T_{j-1}, T'_j = (T - \bigcup_{i=1}^{j-1} T_i) \cup \{(u_{j-1}, v_j)\}\rangle$. We now obtain a layered drawing $\Gamma$ of $T$ from $\Gamma_1$ satisfying the conditions (a)–(d) as follows. We first define a label $L' = \langle x_1, \ldots, x_{j-1}, z_2\rangle$ and a linear partition $P' = \langle T_1, \ldots, T_{j-1}, T^*_j = T_{c_1} - \bigcup_{i=1}^{j-1} T_i)\rangle$ of $T_{c_1}$. By Lemma 3.2.2, $T^*_j$ admits a layered drawing on at most $z_2$ layers where the leg vertex of $T_j$ (if any) has top-visibility and $c_1$ has side-visibility. Thus $T_{c_1}$ admits a layered drawing $\Gamma'_1$ that satisfies conditions (a)–(d) for $L'$ and $P'$. We next place $r$ on the $l_1$ layer of $\Gamma'_1(T^*_j)$. By induction hypothesis and by Fact 3.2.1, each of the trees $T_{c_i}$, $2 \le i \le p$, admits a layered drawing $\Gamma_i$ on at most $z_2$ layers with the top-visibility of the root. We thus finally place the drawings $\Gamma_i$ for $2 \le i \le p$ on the $l_2$ to $l_{z_2+1}$ layers of $\Gamma'_1(T^*_j)$ and add the edges $(r, c_i)$ for $2 \le i \le p$ to complete the drawing. It is easy to see that $P(T)$ and $\Gamma$ satisfies the conditions (a)–(d). An example for this subcase is illustrated in Fig. 3.7.

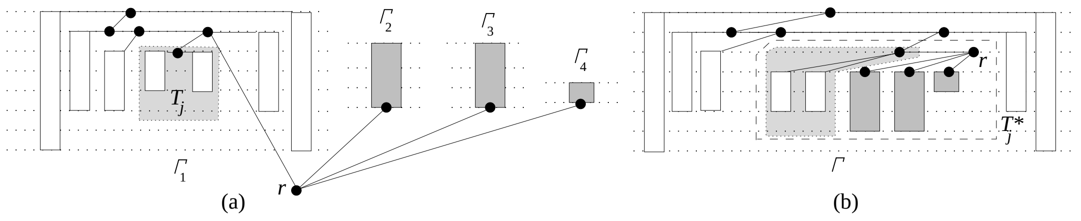


Figure 3.7: Illustration for Subcase 3A.

*Subcase 3B. $|L(T_{c_2})| = 1$ and $z_2 = z_3 = z_4$.* In this case $L(T)$ is obtained by inserting $z_2+1$ into $L(T_{c_1})$. We can obtain $P(T)$ and $\Gamma$ satisfying the conditions (a)–(d) in a similar way as described in Case 3A. We then place these drawings to the right of the drawing of $(T_{c_2})$, between the two drawings $\Gamma_1(T_{j-1})$ and $\Gamma_1(T_j)$ on $l_2$ to $l_{z_2}$ layers of $\Gamma_1(T_{j-1})$. We finally add the edges $(r,\, c_i)$, $3 \le i \le p$

with straight-line segments without edge crossings to complete the drawing. An example for this subcase is illustrated in Fig. 3.8.

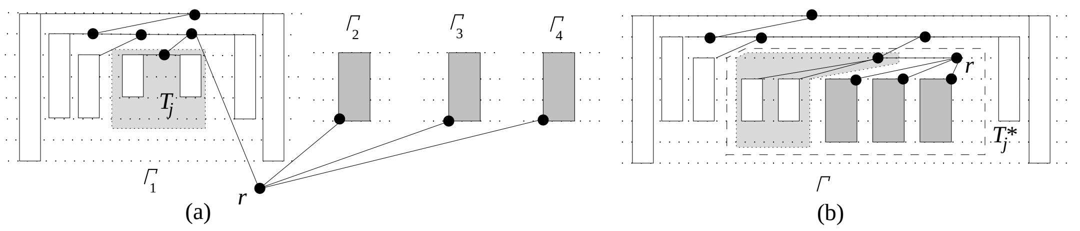


Figure 3.8: Illustration for Subcase 3B.

*Subcase 3C.* $|L(T_{c_2})| = 1$ *and* $z_2 \neq z_3$. In this case $L(T)$ is obtained by inserting $z_2$ into $L(T_{c_1})$. Let $j$ be the index of insertion. We now define a linear partition $P(T)$ of $T$ as $P(T_r) = \langle T_1, \ldots, T_{j-1}, T'_j = (T - \bigcup_{i=1}^{j-1} T_i) \cup \{(u_{j-1}, v_j)\}\rangle$. We now obtain a layered drawing $\Gamma$ of $T$ from $\Gamma_1$ satisfying the conditions (a)–(d) as follows. We first place $r$ on the $l_{z_2+1}$ layer of $\Gamma_1(T_{j-1})$ and add the edge $(r,\, u_k)$ with a straight-line segment without edge crossings. By induction hypothesis, $T_{c_2}$ admits a layered drawing on $z_2$ layers with the side-visibility of $c_2$. We next place this drawing (possibly after mirroring with respect to $x$ and $y$ axis) between the two drawings $\Gamma_1(T_{j-1})$ and $\Gamma_1(T_j)$ so that $c_2$ has right-visibility. We then add the edge $(r,\, c_2)$ with a straight-line segment without edge crossings. Again since $z_2 > z_3 \geq \ldots \geq z_p$, by induction hypothesis, $T_{c_i}$, $3 \leq i \leq p$ admits a layered drawing on at most $z_2 - 1$ layers with the bottom-visibility of $c_i$. We then place these drawings to the right of the drawing of $(T_{c_2})$, between the two drawings $\Gamma_1(T_{j-1})$ and $\Gamma_1(T_j)$ on $l_2$ to $l_{z_2}$ layers of $\Gamma_1(T_{j-1})$. We finally add the edges $(r,\, c_i)$, $3 \leq i \leq p$ with straight-line segments without edge crossings to complete the drawing. It is easy to see that $P(T)$ and $\Gamma$ satisfies the conditions (a)–(d). An example for this subcase is illustrated in Fig. 3.9.

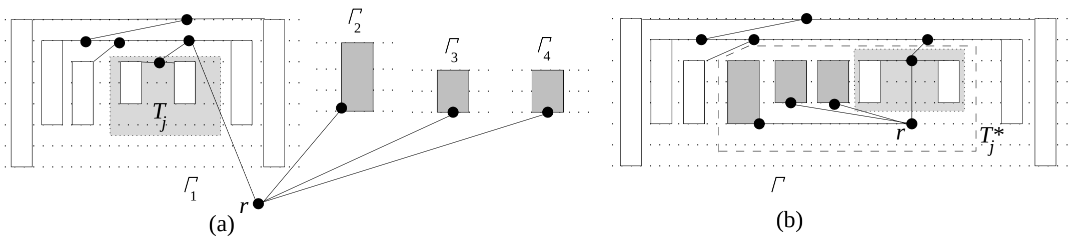


Figure 3.9: Illustration for Subcase 3C.

*Subcase 3D.* $|L(T_{c_2})| = 1$ *and* $z_2 = z_3 \neq z_4$ *and* $z_2 - 1 \in L(T_{c_1})$. In this case

$L(T)$ is obtained by first inserting $z_2$ and then inserting $z_2 - 1$ into $L(T_{c_1})$. Let $j$ be the index of insertion for the first insertion. We now define a linear partition $P(T)$ of $T$ as $P(T_r) = \langle T_1, \ldots, T_{j-1}, T'_j = (T - \bigcup_{i=1}^{j-1} T_i) \cup \{(u_{j-1}, v_j)\}, T^0\rangle$, where $T^0$ represents an empty tree. We obtain a layered drawing of $T$ from $\Gamma_1$ satisfying (a)–(d) as follows. We first obtain a drawing $\Gamma'$ of $T - T_{c_3}$ in the same way as in Subcase 3C. By induction hypothesis, $T_{c_3}$ admits a layered drawing on $z_2 = z_3$ layers with the side-visibility of $c_3$. We place this drawing (possibly after mirroring with respect to $y$-axis) on $l_2$ to $l_{z_2+1}$ layers of $\Gamma'(T_{j-1})$ so that the edge $(r, c_3)$ can be drawn with a straight-line segment without edge-crossings. An example for this subcase is illustrated in Fig. 3.10.

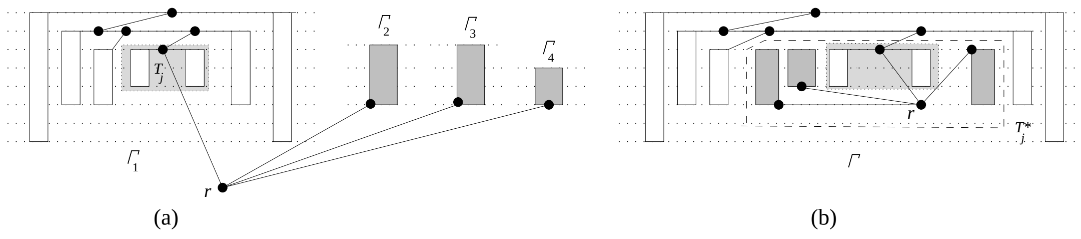


Figure 3.10: Illustration for Subcase 3D.

*Subcase 3E. $|L(T_{c_2})| = 1$ and $z_2 = z_3 \neq z_4$ and $z_2 - 1 \notin L(T_{c_1})$.* In this case $L(T)$ is obtained by first inserting $z_2$ and then inserting $z_2 - 1$ into $L(T_{c_1})$. Let $j$ be the index of insertion for the first insertion. We now define a linear partition $P(T)$ of $T$ as $P(T_r) = \langle T_1, \ldots, T_{j-1}, T'_j = (T - \bigcup_{i=1}^{j-1} T_i) \cup \{(u_{j-1}, v_j)\}, T^0\rangle$, where $T^0$ represents an empty tree. We now obtain a layered drawing $\Gamma$ of $T$ from $\Gamma_1$ satisfying the conditions (a)–(d) as follows. We first define a label $L' = \langle x_1, \ldots, x_{j-1}, z_2 - 1\rangle$ and a linear partition $P' = \langle T_1, \ldots, T_{j-1}, T^*_j = T_{c_1} - \bigcup_{i=1}^{j-1} T_i)\rangle$ of $T_{c_1}$. By Lemma 3.2.2, $T^*_j$ admits a layered drawing on at most $z_2 - 1$ layers where the leg vertex of $T_j$ (if any) has top-visibility and $c_1$ has side-visibility. Thus $T_{c_1}$ admits a layered drawing $\Gamma'_1$ that satisfies conditions (a)–(d) for $L'$ and $P'$. We now place $r$ on the $l_1$ layer of $\Gamma'_1(T^*_j)$ and add the edge $(r, u_k)$ with a straight-line segment without edge crossings. By induction hypothesis, $T_{c_2}$ and $T_{c_3}$ admit a layered drawings on $z_2$ layers with the side-visibility of $c_2$ and $c_3$, respectively. We next place the drawing of $T_{c_2}$ (possibly after mirroring with respect to $x$ and $y$ axis) between the two drawings $\Gamma'_1(T_{j-1})$ and $\Gamma'_1(T^*_j)$ so that $c_2$ has right-visibility and is placed on the $l_{z_2+1}$ layer of $\Gamma'_1(T_{j-1})$. We then place the drawing of $T_{c_3}$ (possibly after mirroring with respect to $y$-axis) on the $l_2$ to $l_{z_2+1}$ layers of $\Gamma_1'(T_{j-1})$ with the left-visibility of $c_3$. We next add the edges $(r,\, c_2)$ and $(r,\, c_2)$ with straight-line segments without edge crossings. Again since $z_2 = z_3 > z_4 \geq \ldots \geq z_p$, by induction hypothesis and by Fact 3.2.1, $T_{c_i}$, $4 \leq i \leq p$ admits a layered drawing on at most $z_2 - 1$ layers with the top-visibility of $c_i$. We place these drawings between the drawing of $T_{c_3}$ and the drawing of the edge $(r,\, c_2)$, on $l_3$ to $l_{z_2+1}$ layers of $\Gamma_1'(T_{j-1})$. Finally we add the edges $(r,\, c_i)$, $4 \leq i \leq p$ with straight-line segments without edge crossings. An example for this subcase is illustrated in Fig. 3.11. $\mathcal{Q.E.D.}$

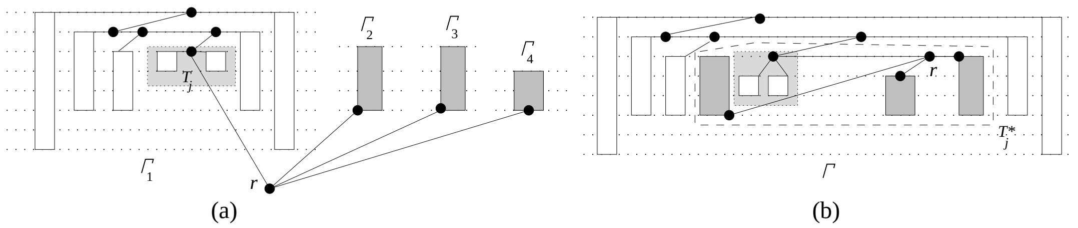


Figure 3.11: Illustration for Subcase 3E.

The proof of Theorem 3.3.1 leads to an algorithm to obtain a layered drawing of a rooted tree $T$ on $L(T)_1$ layers keeping the top-visibility of $r$. We call this Algorithm **Draw_Visible**. The algorithm also gives a linear partition $P(T)$ of $T$ that satisfies the conditions (a)–(d) of Theorem 3.3.1. We call this layered drawing a *valid drawing* and this linear partition a *valid partition* of $T$. Let $\Gamma$ be the valid drawing and $P(T) = \langle\, T_1,\, T_2,\, \ldots,\, T_k \,\rangle$ be the valid partition of $T$. A partition $T_i$ is a *bad partition* if $u_i$ is on the bottommost layer in $\Gamma(T_i)$ for $1 \leq i \leq k-1$. $T_i$ is a *weak partition* if $T_{i-1}$ is a bad partition for $2 \leq i \leq k$. We now have the following corollary whose proof follows from the construction of the valid partition of a rooted tree as described in the proof of Theorem 3.3.1.

**Corollary 3.3.2** *Let $T$ be a rooted tree with the root $r$ and let $L(T) = \langle x_1, x_2, \ldots, x_k\rangle$ and $P(T) = \langle\, T_1,\, T_2,\, \ldots,\, T_k \,\rangle$ be the label and the valid partition of $T$, respectively. Then there exist two subtrees $t_1$ and $t_2$ of $T_i$, $1 \leq i \leq k-1$, rooted at children of $u_i$ such that the labels of $t_1$ and $t_2$ contain $x_i$ unless $T_i$ is a weak partition.*

Summarizing the definition of the label of a rooted tree for various cases in the proof of Theorem 3.3.1, we now give the precise definition of the *label* of a rooted tree $T$ with the root vertex $r$ as follows.

(a) If $r$ is the only vertex in $T$, then $L(T) = \langle 1 \rangle$.

(b) If $c_1, c_2, \ldots, c_p$ are the children of $r$ in $T$ and $T_{c_1}, T_{c_2}, \ldots, T_{c_p}$ are the subtrees of $T$ rooted at the vertices $c_1, c_2, \ldots, c_p$, respectively, then $L(T)$ is defined as follows. (Without loss of generality, we assume that $L(T_{c_1})_1 \geq L(T_{c_2})_1 \geq \ldots \geq L(T_{c_p})_1$ and if $L(T_{c_i})_1 = L(T_{c_{i+1}})_1$ for some $1 \leq i \leq p-1$, then $|L(T_{c_i})| \geq |L(T_{c_{i+1}})|$).

  (i) If $L(T_{c_2})_1 < L(T_{c_1})_f$, then $L(T) = L(T_{c_1})$.

  (ii) If $L(T_{c_2})_1 = L(T_{c_1})_f$, then $L(T)$ is defined as follows.

    A. If $|L(T_{c_2})| = 1$ and $L(T_{c_2})_1 > L(T_{c_3})_1$, then $L(T)$ is obtained by inserting 0 into $L(T_{c_1})$.

    B. Otherwise $L(T)$ is obtained by inserting $L(T_{c_2})_1 + 1$ into $L(T_{c_1})$.

  (iii) If $L(T_{c_2})_1 > L(T_{c_1})_f$ and $L(T_{c_2})_1 \in L(T_{c_1})$, then $L(T)$ is obtained by inserting $L(T_{c_2})_1 + 1$ into $L(T_{c_1})$.

  (iv) If $L(T_{c_2})_1 > L(T_{c_1})_f$ and $L(T_{c_2})_1 \notin L(T_{c_1})$, then $L(T)$ is defined as follows.

    A. If $|L(T_{c_2})| > 1$, then insert $L(T_{c_2})_1 + 1$ into $L(T_{c_1})$ to obtain $L(T)$.

    B. If $|L(T_{c_2})| = 1$ and $L(T_{c_2})_1 = L(T_{c_3})_1 = L(T_{c_4})_1$, then insert $L(T_{c_2})_1 + 1$ into $L(T_{c_1})$ to obtain $L(T)$.

    C. If $|L(T_{c_2})| = 1$ and $L(T_{c_2})_1 \neq L(T_{c_3})_1$, then insert $L(T_{c_2})_1$ into $L(T_{c_1})$ to obtain $L(T)$.

    D. If none of the above condition holds, then insert $L(T_{c_2})_1$ into $L(T_{c_1})$ to obtain a new decreasing list $L'$ of integers. Finally, $L(T)$ is obtained by inserting 0 into $L'$ if $L(T_{c_2})_1 - 1 \notin L(T_{c_1})$; otherwise $L(T)$ is obtained by inserting $L(T_{c_2})_1 - 1$ into $L'$.

## 3.4 Time-complexity of Algorithm Draw_Visible

In this section, we prove that Algorithm **Draw_Visible** runs in linear time. We first have the following lemma.

**Lemma 3.4.1** *The label of a rooted tree $T$ can be computed in linear time. Furthermore a valid partitioning of $T$ can also be computed in linear time.*

**Proof.** We compute the label of $T$ by a bottom-up traversal of $T$ according to the definition. For each vertex $v$ in $T$, while we traverse $v$ we compute the label of $T_v^r$ as follows. We first find at most four children $w_1$, $w_2$, $w_3$ and $w_4$ of $v$ in $T$ such that the following conditions hold.

- For $1 \leq i \leq 3$, either $L(T_{w_i}^r)_1 > L(T_{w_{i+1}}^r)_1$ or $L(T_{w_i}^r)_1 = L(T_{w_{i+1}}^r)_1$ and $|L(T_{w_i}^r)| \geq |L(T_{w_{i+1}}^r)|$.
- For any child $w$ of $v$ in $T$ other than $w_1$, $w_2$, $w_3$ and $w_4$, either $L(T_{w_4}^r)_1 > L(T_w^r)_1$ or $L(T_{w_4}^r)_1 = L(T_w^r)_1$ and $|L(T_{w_4}^r)| \geq |L(T_w^r)|$.

Clearly this can be done in $O(deg(v))$ time. We then compute the label of $T_w^r$ by inserting zero, one or two integers into $L(T_{w_1}^r)$ according to the definition. To do this insertion in constant time we store $L(T_{w_1}^r)$ as an array of size $L(T)_1$ The array contains only zeros and ones where a 1 (0) in the $i$-th position of the array represents the presence (absence) of $i$ in $L(T_{w_1}^r)$ for $1 \leq i \leq L(T_{w_1}^r)_1$. For each position $i$ of the array, we also keep a pointer to the $j$-the position of the array, $1 \leq i \leq j \leq L(T_{w_1}^r)_1$, where $j$ is the minimum integer greater than or equal to $i$ such that the $j$-the position of the array contains a one. Then it is trivial to see that the insertion of an integer into the label of a vertex can be done in constant time.

The overall time-complexity for the computation of the label of $T$ is thus $O(\sum_{v \in V(T)}(deg(v)+1)) = O(n)$.

For computing a valid partitioning of $T$ during this bottom-up traversal, we can store the pointer to the root vertex for each partition. As we compute the label of the subtrees, we can easily update these pointers in constant time. We then compute these partitions by a depth-first traversal of $T$ at the end of the traversal. Thus computing the valid partition also takes linear time. $\mathcal{Q.E.D.}$

We now have the following theorem.

**Theorem 3.4.2** *Algorithm **Draw_Visible** runs in $O(n)$ time on a rooted tree $T$ with $n$ vertices.*

**Proof.** Lemma 3.4.1 implies that the computation of label and valid partition of $T$ takes linear time. Thus it is sufficient to prove that the drawing of $T$ also takes linear time. Let us first ignore the time required for the step of the algorithm when we modify of the drawing of a subtree of $T$ according to Lemma 3.2.2 such that a vertex $a$ has side-visibility in the new drawing while keeping the top-visibility of another vertex $b$. Then from the description of the algorithm, one can observe that the operations performed at each vertex are translation and mirroring of the drawing of the subtrees. To perform these operations, one can associate a translation offset and a scaling factor with each vertex $v$ so that a final top-down traversal of $T$ gives the exact $x$ and $y$ coordinates for each vertex in a similar way as in [9]. The computation of the translation offset and the scaling factor for all the children of each vertex $v$ requires $O(deg(v))$ time in total. On the other hand, the algorithm described in the proof of Lemma 3.2.2 for the above modification of the drawings of the subtrees takes $O(\sum_{v \in P}(deg(v)))$ time where $P$ is the path from $a$ to $b$. It is interesting to note that $a$ and $b$ represents the leg of a partition $T_i$ and the root of another partition $T_j$ of the subtree in this case and all the partitions $T_l$, $i \leq l \leq j$ are merged to a single partition. Thus any vertex $v$ in $P$ is considered for this operation at most once. Hence Algorithm **Draw_Visible** takes $O(\sum_{v \in V(T)}(deg(v))) = O(n)$ time. $\mathcal{Q.E.D.}$

## 3.5 Conclusion

In this chapter, we gave a linear-time algorithm to obtain a root-visible drawing of a rooted tree $T$. In the next chapter we will prove that the drawing obtained by this algorithm is a minimum-layer root-visible drawing of $T$. Using this algorithm, we will also give an algorithm for minimum-layer drawing of a tree in the next chapter.

# Chapter 4

# Minimum-Layer Drawings of Trees

A *layered drawing* of a tree $T$ is a planar straight-line drawing of $T$ where the vertices of $T$ are placed on a set of horizontal lines, called *layers*. We call a layered drawing of $T$ a *minimum-layer drawing* of $T$, when the number of layers used in the drawing is the minimum among all the possible layered drawings of $T$. Figure 4.1(a) illustrates a tree $T$, Fig. 4.1(b) and (c) depict two different layered drawings of $T$ occupying four layers and two layers, respectively. One can observe that at least two layers are required for any layered drawing of $T$, and hence the drawing in Fig. 4.1(c) is a minimum-layer drawing of $T$.

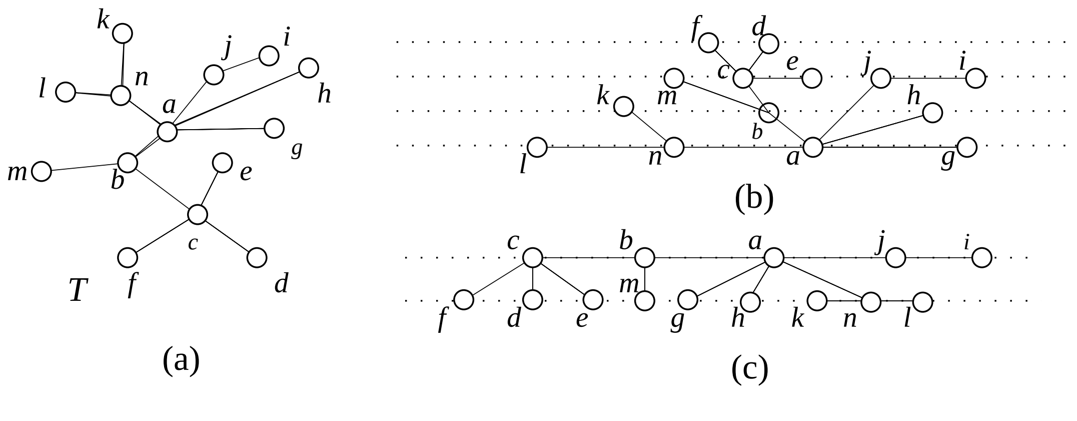


Figure 4.1: (a) A tree $T$, (b) a layered drawing of $T$, and (c) a minimum-layer drawing of $T$.

In 1988 de Fraysseix proposed the $k$-lines drawability problem which asks whether all planar graphs can be drawn on $k$ parallel lines lying on the surface of a cylinder [10]. There are some variants of layered drawings of trees which have been shown to be NP-hard [14, 18]. On the other hand, for some variants of layered drawings of trees, minimum-layer drawings have been achieved [2, 8]. However, for the general version of the problem, there is neither any polynomial-time algorithm nor any hardness result known so far. In this chapter we address the problem of minimum-layer drawing of a tree. We first prove that for a rooted tree $T$, the drawing obtained by Algorithm **Draw_Visible** is a minimum-layer root-visible drawing of $T$. We then use this algorithm to find a minimum-layer drawing of a tree in linear time. Let $T$ be a tree. We first find in linear time, a suitable vertex $r$ of $T$ such that a minimum-layer root-visible drawing of $T$ with the root $r$ is also a minimum-layer drawing of $T$. We then obtain a minimum-layer drawing of $T$ by computing a minimum-layer root-visible drawing of a $T$ rooted at $r$ using Algorithm **Draw_Visible**. Figure 4.2(a) illustrates a tree $T$. Figure 4.2(b) depicts a rooted tree $T'$ obtained from $T$ such that a minimum-layer root-visible drawing of $T'$ is also a minimum layer drawing of $T$. Figure 4.2(c) illustrates a minimum-layer drawing of $T$.

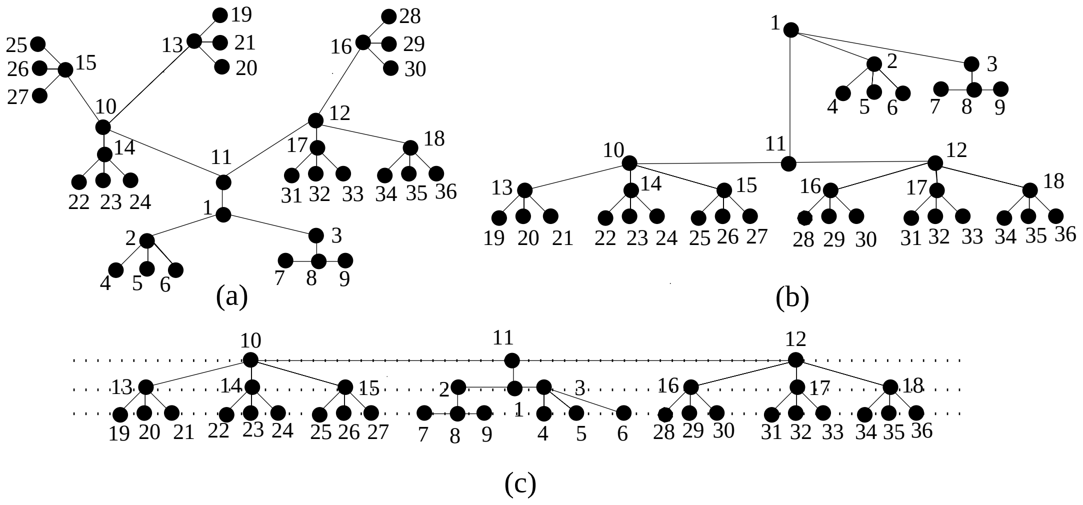


Figure 4.2: (a) A tree $T$, (b) a rooted tree $T'$ formed by taking the vertex 1 of $T$ as the root of $T'$, and (c) a minimum-layer drawing of $T$ and the partitions of $T$.

The rest of this chapter is organized as follows. Section 2 proves that the drawing of a rooted tree $T$ obtained by Algorithm **Draw_Visible** is a minimum-layer root-visible drawing of $T$. Section 3 presents a linear-time algorithm to obtain a minimum-layer drawing of an unrooted tree. Finally, Section 6 concludes the chapter.

## 4.1 Minimum-Layer Root-Visible Drawings

In the previous chapter we gave an algorithm to obtain a root-visible drawing of a rooted tree $T$ on $L(T)_1$ layers. In this section we prove that any root-visible drawing of $T$ requires at least $L(T)_1$ layers. We actually prove a stronger result in this section where we consider a "poly-line drawing" of $T$ and prove that even for a poly-line drawing scenerio, at least $L(T)_1$ layers are required for any root-visible drawing of $T$.

A *poly-line layered drawing* of a tree $T$ is a poly-line drawing of $T$ where the vertices of $T$ as well as the bends on the edges of $T$ are placed on a set of layers. Let $\Gamma$ be a poly-line layered drawing of $T$ and let $v$ be a vertex in $T$. Then $v$ has *poly-line top-visibility* in $\Gamma$ if there is a point $p$ above the bounding box of $\Gamma$ such that a poly-line between $v$ and $p$ does not create any edge crossing with $\Gamma$ with all its bend placed on the layers of $\Gamma$. Similarly, we define the *poly-line bottom-visibility* of $v$ in $\Gamma$. A poly-line layered drawing $\Gamma$ of a rooted tree $T$ is called a *poly-line root-visible drawing* of $T$ if the root vertex of $T$ has poly-line bottom-visibility in $\Gamma$. We now have the following theorem.

**Theorem 4.1.1** *Let $T$ be a rooted tree with the root vertex $r$ and let $L(T) = \langle x_1, x_2, \ldots, x_k \rangle$ and $P(T) = \langle\ T_1,\ T_2,\ \ldots,\ T_k\ \rangle$ be the label of $T$ and the valid partition of $T$, respectively. Denote by $G_i$, $1 \le i \le k$, the rooted tree $T - (\bigcup_{j=1}^{i-1} T_j)$. Then for $1 \le i \le k$, at least $x_i$ layers are required for any poly-line root-visible drawing of $G_i$ unless $T_i$ is a weak partition.*

Before we prove Theorem 4.1.1, we need the following two lemmas.

**Lemma 4.1.2** *[2, 26] Let $T$ be a rooted tree such that at least three subtrees of $T$ require at least $k$ layers in any poly-line layered drawing. Then any poly-line layered drawing of $T$ requires at least $k + 1$ layers.*

**Lemma 4.1.3** *Let $T$ be a tree and let $P = (u_1, u_2, \ldots, u_l)$ be a path in $T$. Let $C_i$, $1 \le i \le l$ be the set of components of $T - P$ such that each component of $C_i$ has a vertex adjacent to $u_i$. Let $u_m$, $u_n$ and $u_p$ be three vertices of $P$ where $m < n < p$. Assume that there exist two components in $C_m$ and two components* *in $C_p$, each requiring at least $k-1$ layers in any poly-line layered drawing with the poly-line bottom-visibility of the vertex adjacent to a vertex of $P$ and there exist two components in $C_n$ each requiring at least $k$ layers in any poly-line layered drawing with the poly-line bottom-visibility of the vertex adjacent to a vertex of $P$. Then $T$ requires at least $k+1$ layers in any poly-line layered drawing with the poly-line top-visibility of $u_p$ and the poly-line bottom-visibility of $u_m$.*

**Proof.** Assume for a contradiction that $T$ admits a layered drawing $\Gamma$ on $k$ layers, with the poly-line top-visibility of $u_p$ and the poly-line bottom-visibility of $u_m$. Since each of the components of $C_n$ requires at least $k$ layers, we need to place the vertex $u_n$ in between the two components of $C_n$ in $\Gamma$. One can observe that to ensure the poly-line top-visibility of $u_p$ and the poly-line bottom-visibility of $u_m$ we need to place the components of $C_p$ on the $k-1$ layers above the incident edges of $u_n$ and the components of $C_m$ on the $k-1$ layers below the incident edges of $u_n$., as illustrated in Figure 4.3. In this situation, either $u_p$ or $u_m$ looses poly-line visibility from $u_n$; and hence the edges $(u_m, u_n)$ and $(u_n, u_p)$ cannot be drawn simultaneously avoiding edge crossing. $\mathcal{Q.E.D.}$

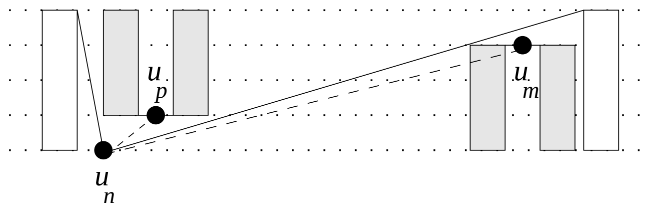


Figure 4.3: Illustration for the Proof of Lemma 4.1.3.

We are now ready to prove Theorem 4.1.1.

**Proof of Theorem 4.1.1.** Let $n$ be the number of vertices in $T$. The case for $n = 1$ is trivial and hence we assume that $n > 1$ and for any rooted tree $T'$ with less than $n$ vertices, the claim is true. Let $c_1$, $c_2$, ..., $c_p$ be the children of $r$ in $T$ and $L(T_{c_1})$, $L(T_{c_2})$, ..., $L(T_{c_p})$ are the labels of the subtrees $T_{c_1}$, $T_{c_2}$, ..., $T_{c_p}$ of $T$ rooted at the vertices $c_1$, $c_2$, ..., $c_p$, respectively. We assume that $L(T_{c_1}) = \langle y_1, \ldots, y_k\rangle$ and $P(T_{c_1}) = \langle t_1, \ldots, t_k\rangle$. We also assume that $H_i = T_{c_1} - (\bigcup_{j=1}^{i-1} t_j)$, $1 \le i \le k$. Thus by induction hypothesis, at least $x_i$ layers are required for any poly-line root-visible drawing of $H_i$, $1 \le i \le k$. For notational convenience we denote by $z_i$ the value of $L(T_{c_i})_1$, $1 \le i \le p$. Without loss of generality, we also assume that $z_1 \geq z_2 \geq \ldots \geq z_p$ and if $z_i = z_{i+1}$ for some $1 \leq i \leq p-1$, then $|L(T_i)| \geq |L(T_{i+1})|$. We now have the following cases to consider.

*Case 1.* $z_2 < x_k$. In this case, $L(T) = \langle x_1 = y_1, \ldots, x_k = y_k$ and $P(T) = \langle T_1 = t_1, \ldots, T_{k-1} = t_{k-1}, T_k = t_k \cup (T - T_{c_1}) \cup \{(r, u_k)\}\rangle$. Since $H_i$ is a subgraph of $G_i$ for $1 \leq i \leq k$, by induction hypothesis any poly-line root-visible drawing of $G_i$ requires at least $x_i$ layers.

*Case 2.* $z_2 \in L(T_{c_1})$. We first assume that $z_2 = y_k > z_3$ and $|L(T_{c_2})| = 1$. Then $L(T) = \langle x_1, \ldots, x_k, 0\rangle$ and $P(T) = \langle T_1 = t_1, \ldots, T_{k-1} = t_{k-1}, T_k = t_k \cup (T - T_{c_1}) \cup \{(r, u_k)\}, T_{k+1} = T^0\rangle$, where $T^0$ represents an empty tree. It is again trivial to see that any poly-line root-visible drawing of $G_i$ requires at least $x_i$ layers for $1 \leq i \leq k$ since $H_i$ is a subgraph of $G_i$.

On the other hand, if $z_2 > y_k$ or if $z_2 = z_3$ or $|L(T_{c_2})| > 1$, then $L(T)$ is obtained by inserting $z_2 + 1$ into $L(T_{c_1})$. Let $j$ be the index of insertion. Then $P(T_r) = \langle T_1 = t_1, \ldots, T_{j-1} = t_{j-1}, T_j = (T - \bigcup_{i=1}^{j-1} t_i) \cup \{(u_{j-1}, v_j)\}\rangle$. For $1 \leq i < j$, $G_i$ contains $H_i$ as a subgraph and thus any poly-line root-visible drawing of $G_i$ requires at least $x_i$ layers. We now show that at least $x_j = y_j + 1$ layers are required for any poly-line root-visible drawing of $G_j$.

For a contradiction, let us assume that $G_j = H_j \cup (T - T_{c_1}) \cup \{(u_k, r)\}$ admits a poly-line root-visible drawing on less than $x_j$ layers. Since $z_2 \in L(T_{c_1})$, there is a partition, say $t_l \in P(T_{c_1})$, $j \leq l \leq k$ such that $y_l = z_2$. Let $j^*$ be the maximum index, $j < j^* \leq l$, such that $t_{j^*}$ is not a weak partition and $H_{j^*} \cup (T - T_{c_1}) \cup \{(u_k, r)\}$ admits a poly-line root-visible drawing on less than $x_{j^*}$ layers. We first assume $z_2 = y_k$, that is $l = k$. Then either $z_2 = z_3 = y_k$ or $|L(T_{c_2})| > 1$. Suppose $z_2 = z_3 = y_k$. Then $H_k \cup (T - T_{c_1}) \cup \{(u_k, r)\}$ has at least three subtrees each of which requires at least $z_2$ layers. Then by Lemma 4.1.2 $H_k \cup (T - T_{c_1}) \cup \{(u_k, r)\}$ requires at least $z_2 + 1$ layers for any poly-line root-visible drawing. Again one can prove in a similar way that if $|L(T_{c_2})| > 1$, then $H_k \cup (T - T_{c_1}) \cup \{(u_k, r)\}$ requires at least $z_2 + 1$ layers for any poly-line root-visible drawing. Thus $j^* > l$. Since $t_{j^*}$ is not a weak partition, by Corollary 3.3.2 and by induction hypothesis, $t_{j^*}$ has at least two subtrees rooted at the children of $u_{j^*}$ that requires at least $y_{j^*}$ layers for any poly-line root-visible drawing. If $t_{j^*}$ is not a bad partition, then $t_{j^*+1}$ is not a weak partition and thus $H_{j^*+1} \cup (T - T_{c_1}) \cup \{(u_k, r)\}$ requires at least $y_{j^*+1} + 1 = y_{j^*}$ layers for any poly-line root-visible drawing. Then by Lemma 4.1.2, $H_{j^*} \cup (T - T_{c_1}) \cup \{(u_k, r)\}$ requires at least $x_{j^*} = y_{j^*} + 1$ layers for any poly-line root-visible drawing, which is a contradiction. On the other hand if $t_{j^*}$ is a bad partition, then from the construction of the valid partition and by induction hypothesis $u_{j^*}$ has two children $a$ and $b$ such that each of the subtrees rooted at $a$ and $b$ requires at least $y_{j^*}$ layers for any poly-line root-visible drawing. Furthermore, a child $u_{j^*-1}$ of $u_{j^*}$ other than $a$ and $b$ has two children $c$ and $d$ such that each of the subtrees rooted at $c$ and $d$ requires at least $y_{j^*} - 1$ layers for any poly-line root-visible drawing. Again since $y_{j^*+1} = y_{j^*} - 1$, $y_{j^*+1} = y_{j^*} - 1$ and $t_{j^*+2}$ is not a weak partition, $u_{j^*+1}$ has a child such that a the subtree rooted at the child requires at least $y_{j^*} - 1$ layers and $H_{j^*+2}$ requires at least $y_{j^*} - 1$ layers for any poly-line root-visible drawings. Then By Lemma 4.1.3, $H_{j^*}$ requires at least $y_{j^*} + 1 = x_{j^*}$ layers for any poly-line layered drawing keeping the bottom-visibility of $r$ and the top-visibility of $u_{j^*+1}$, which is a contradiction. Thus $G_j$ requires at least $x_j$ layers for any poly-line root-visible drawing.

Again if $z_2 > y_k$, we can show that $G_i$ requires at least $x_i$ layers by a similar reasoning as in the previous paragraph.

*Case 3.* $z_2 \notin L(T_{c_1})$ *and* $L(T_{c_2})_1 > x_k$. If $|L(T_{c_2})| > 1$ or $z_2 = z_3 = z_4$, then we can prove the claim by the same reasoning as in the previous paragraph. Again if $|L(T_{c_2})| = 1$ and $z_2 \neq z_3$. We thus assume that $|L(T_{c_2})| = 1$ and $z_2 = z_3 \neq z_4$. Then $L(T)$ is obtained by first inserting $z_2$ and then inserting $z_2 - 1$ into $L(T_{c_1})$ if $z_2 - 1 \in L(T_{c_1})$. On the other hand, $L(T)$ is obtained by first inserting $z_2$ and then inserting 0 into $L(T_{c_1})$ if $z_2 - 1 \notin L(T_{c_1})$. Let $j$ be the index of insertion for the first insertion in both the cases. Then $P(T_r) = \langle T_1, \ldots, T_{j-1}, T'_j = (T - \bigcup_{i=1}^{j-1} T_i) \cup \{(u_{j-1}, v_j)\}, T^0\rangle$, where $T^0$ represents an empty tree. If $z_2 - 1 \notin L(T_{c_1})$, then the claim is trivial to prove since $H_i$ is a subgraph of $G_i$ for $1 \leq i \leq j$ and $x_{j+1} = 0$. On the other hand if $z_2 - 1 \in L(T_{c_1})$, then it is also trivial to prove the claim since $H_i$ is a subgraph of $G_i$ for $1 \leq i \leq j$ and $T_{j+1}$ is a weak partition. $\mathcal{Q.E.D.}$

Theorem 4.1.1 implies that any poly-line root-visible drawing of arooted tree $T$ requires at least $L(T)_1$ layers. This result is a bit surprising. The flexibility of using bends in "polyline drawings" of trees gives an impression that a polyline drawing of a tree $T$ may require less number of layers than that of a straight-line drawing of $T$. However Algorithm **Draw_Visible**, described in the previous section, gives a (straight-line) root-visible drawing of a rooted tree $T$ on $L(T)_1$

layers. Thus $L(T)_1$ represents the minimum number of layers required in a root-visible drawing of $T$ both for the poly-line and for the straight-line scenerio.

## 4.2 Minimum-Layer Drawings of Trees

In this section we address the problem of minimum-layer drawing of a tree. We have the following theorem.

**Theorem 4.2.1** *Let $T$ be a tree. Then one can find a vertex $w$ of $T$ in linear time such that there is a minimum-layer drawing $\Gamma$ of $T$ where $w$ has bottom-visibility.*

**Proof.** We first take an arbitrary vertex $r$ as the root of $T$ to obtain a rooted tree $T$ and compute the label $L(T)$ of $T$. By Theorems 3.3.1 and 4.1.1, $L(T)_1$ represents the minimum number of layers for any layered drawing of $T$ with the bottom-visibility of $r$. Let $c_1$, $c_2$, …, $c_p$ be the children of $r$ in $T$ and $L(T_{c_1})$, $L(T_{c_2})$, …, $L(T_{c_p})$ are the labels of the subtrees $T_{c_1}$, $T_{c_2}$, …, $T_{c_p}$ of $T$ rooted at the vertices $c_1$, $c_2$, …, $c_p$, respectively. For notational convenience we denote by $z_i$ the value of $L(T_{c_i})_1$, $1 \leq i \leq p$. Without loss of generality, we also assume that $z_1 \geq z_2 \geq \ldots \geq z_p$ and if $z_i = z_{i+1}$ for some $1 \leq i \leq p-1$, then $|L(T_i)| > |L(T_{i+1})|$. We now have the following cases to consider.

*Case 1:* $z_1 = z_2 = z_4$. In this case $L(T) = \langle z_1 + 1 \rangle$. By Lemma 4.1.2, $z_1 + 1$ layers are required for any layered drawing of $T$. Furthermore by Theorem 3.3.1, there is a layered drawing of $T$ on $z_1 + 1$ layers with the bottom-visibility of $r$. Therefore we choose $r$ as the desired vertex $w$.

*Case 2:* $z_1 = z_2 > z_3$. If $|L(T_{c_1})| > 1$, then $L(T)_1 = z_1 + 1$. By Lemma 4.1.2, $z_1 + 1$ layers are required for any layered drawing of $T$. Since by Theorem 3.3.1, $T$ admits a layered drawing on $z_1 + 1$ layers with the bottom-visibility of $r$, we can choose $r$ as the desired vertex $w$. If $|L(T_{c_1})| = 1$, then $L(T) = \langle z_1, 0 \rangle$. At least $z_1$ layers are required for any layered drawing of $T$ and by Theorem 3.3.1, there is also a layered drawing of $T$ on $z_1$ layers with the bottom-visibility of $r$. We can thus take $r$ as the desired vertex $w$.

*Case 3:* $z_1 > z_2$. If $|L(T_{c_1})| = 1$, then $L(T) = \langle z_1 \rangle$ and $r$ can be taken as the desired root $w$ since at least $z_1$ layers are required for any layered drawing of $T$ and by Theorem 3.3.1, there is also a layered drawing of $T$ on $z_1$ layers with the bottom-visibility of $r$. We thus assume that $|L(T_{c_1})| > 1$. Let $P(T) = \langle T_1, T_2, \ldots \rangle$ be the valid partition of $T$. Then we choose the root of $T_1$ as the desired vertex $w$. We now show that there is a minimum layer drawing of $T$ with the bottom-visibility of $w$. By Corollary 3.3.2, $T_1$ has at least two subtrees $t_1$ and $t_2$ such that the label of $t_1$ and $t_2$ contains $z_1$. Thus by Theorems 3.3.1 and 4.1.1, minimum-layer root-visible drawings of $t_1$ and $t_2$ occupies $z_1$ layers. Denote $T' = T - T_1$ and take the leg vertex $v_2$ of $T_2$ as the root of $T'$. One can obtain a layered drawing of $T'$ on $z_1$ layers as follows. Let $\Gamma$ be the drawing of $T_{c_1}$ obtained by Algorithm **Draw_Visible**. Then mirroring $\Gamma(T_{c_1} - T_1)$ gives a layered drawing $\Gamma^*$ of $T_{c_1} - T_1$ on at most $z_1 - 1$ layers with the top-visibility of $v_2$. Again since $z_1 > z_2 \geq z_3 \geq \ldots$, by Theorem 3.3.1 and by Fact 3.2.1, each subtree $T_{c_i}$ of $T$ other than $T_{c_1}$ admits a layered drawing $\Gamma_i$ on at most $z_1 - 1$ layers. Thus a layered drawing of $T'$ with the bottom-visibility of $v_2$ can be obtained by placing the drawings $\Gamma_i$ and the drawing $\Gamma^*$ on the bottommost $z_1 - 1$ layers, placing $r$ on the topmost layers and adding the necessary edges. Therefore by Theorem 4.1.1, the label of $T'$ is at most $z_1$. If the label of $T'$ is less than $z_1$, then $L(T_w) = \langle z_1, 0 \rangle$ and by Theorem 3.3.1 there is a layered drawing of $T$ on $z_1$ layers with the bottom-visibility of $w$. If the label of $T'$ is $z_1$, then $L(T_w) = \langle z_1 + 1 \rangle$ and by Theorem 3.3.1 $T$ admits a layered drawing on $z_1 + 1$ layers with the bottom-visibility of $w$. By Lemma 4.1.2, $z_2 + 1$ layers are also necessary for any layered drawing of $T$. $\mathcal{Q.E.D.}$

## 4.3 Conclusion

In this chapter, we first proved that the drawing of a rooted tree $T$ obtained by Algorithm **Draw_Visible** is a minimum-layer root-visible drawing of $T$. Using this algorithm we then gave an algorithm for minimum-layer drawing of a tree.

# Chapter 5

# Conclusion

In this thesis, we addressed the problem of minimum-layer drawing of a tree. We first gave an algorithm to obtain a minimum-layer root-visible drawing of a rooted tree. We then gave another algorithm to find a suitable vertex $r$ of a tree $T$ such that a minimum-layer root visible drawing of $T$ rooted at $r$ gives a minimum-layer drawing of $T$. The drawing obtained by our algorithm also represents a minimum-layer poly-line layered drawing of a tree.

The problem of minimizing the number of layers in a drawing of a tree is motivated by both theoretical interest and practical applications. Although the problem of minimum-layer drawings of trees has attracted much interest of the researchers for the past few years, there had neither been any polynomial-time algorithm nor any hardness result known to date for the general version of the problem. In this respect this thesis addressed a long-standing optimization problem on tree drawings. The practical applications of minimum-layer drawings of trees arise from the requirement in various fields, especially for the information visualization and VLSI circuit design. Visualization of heirarchical structures like organizational charts, software class hierarchies, phylogenetic evolutions etc involves layered drawings of trees. However one expects a visualizing drawing of a hierarchical structure like trees to exhibit the hierarchical property blatantly. Unfortunately, the drawing produced by our algorithm fails to show the inherent hirerchical structure present in the structure of a tree. Again the problem of minimum-layer tree drawing finds its application for the design of a standard cell on the minimum number of rows as illustrated in Section 1.2. However, this algorithm is applicable for this application only for a small class of circuits, particularly it is applicable only when the graph obtained from the circuit to be placed in the standard cell is a tree. Thus the result in this thesis is more interesting and eye-catching for theoretical motivation rather than its practical applications.

The following is a brief list of future works related to our results presented in this thesis.

1. In this thesis we solved the problem of minimizing the number of layers in a layered drawing of a tree. However the drawing produced by our algorithm does not necessarily preserve a given circular ordering of the neighbors of each vertex of the tree. The problem of minimum-layer drawing of an ordered tree is still open and remains as a future work.

2. It is also a future work to obtain minimum-layer drawings of some classes of planar graphs richer than trees either for the unrestricted or for some restricted variation of layered drawing. In this context it may be mentioned that recently Mondal *et al.* have given a polynomial-time algorithm for minimum-layer drawings of a class of planar graphs called "plane 3-trees" [19]. A *plane* 3-*tree* $G_n$ with $n \geq 3$ vertices is a plane graph for which the following conditions (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 $G_{n-1}$. However the problem is still open for other classes of planar graphs.

# References

[1] M. J. Alam, M. M. Rabbi, M. S. Rahman, and M. R. Karim. Upright drawings of graphs on three layers. *Journal of Applied Mathematics and Informatics*, 2010 (to appear).

[2] 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.

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

[4] M. Bohl and M. Rynn. *Tools For Structured and Object-Oriented Design.* Prentice-Hall, Upper Saddle River, NJ, USA, 7th edition, 2007.

[5] M. Chrobak and S. Nakano. Minimum-width grid drawings of plane graphs. *Computational Geometry: Theory and Application*, 11:29–54, 1998.

[6] S. Cornelsen, T. Schank, and D. Wagner. Drawing graphs on two and three lines. *Journal of Graph Algorithms and Applications*, 8(2):161–177, 2004.

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

[8] V. Dujmović, M. R. Fellows, M. T. Hallett, M. Kitching, G. Liotta, C. McCartin, N. Nishimura, P. Ragde, F. A. Rosamond, M. Suderman, S. Whitesides, and D. R. Wood. On the parameterized complexity of layered graph drawing. In *the Proceedings of the 9th Annual European Symposium on Algorithms (ESA)*, volume 2161 of *Lecture Notes in Computer Science*, pages 488–499. Springer-Verlag, 2001.

[9] I. Fary. On straight line representations of planar graphs. *Acta Sci. Math. Szeged.*, 11:229–233, 1948.

[10] S. Felsner, G. Liotta, and S. K. Wismath. Straight-line drawings on restricted integer grids in two and three dimensions. *Journal of Graph Algorithms and Applications*, 7(4):363–398, 2003.

[11] U. Fößmeier and M. Kaufmann. Nice drawings for planar bipartite graphs. In *the Proceedings of the 3rd Italian Conference on Algorithms and Complexity (CIAC 1997)*, pages 122–134. Springer-Verlag, 1997.

[12] M. R. Garey and D. S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-completeness*. W. H. Freeman and Company, New York, USA, 1979.

[13] S. Hansche, J. Berti, and C. Hare. *Official (ISC)2 guide to the CISSP exam*. Auerbach Publications, Boston, MA, USA, 2004.

[14] K. Hayashi and S. Masuda. Complexity results on the minimum width drawing problem for rooted unordered trees. *Electronics and Communications in Japan*, 81(1):32–41, 1998.

[15] M. Kaufmann and D. Wagner. *Drawing Graphs: Methods and Models*, volume 2025 of *Lecture Notes in Computer Science*. Springer-Verlag, London, UK, 2001.

[16] D. E. Knuth. Computer-drawn flowcharts. *Communications of the ACM*, 6(9):555–563, 1963.

[17] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layouts*. Wiley, New York, NY, USA, 1990.

[18] K. Marriott and P. J. Stuckey. NP-completeness of minimal width unordered tree layout. *Journal of Graph Algorithms and Applications*, 8(2):295–312, 2004.

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

[20] T. Nishizeki and N. Chiba. *Planar Graphs: Theory and Algorithms.* North-Holland, Amsterdam, 1988.

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

[22] H. C. Purchase. Which aesthetic has the greatest effect on human understanding? In *the Proceedings of the 5th International Symposium on Graph Drawing (GD 1997)*, volume 1353 of *Lecture Notes in Computer Science*, pages 248–261. Springer-Verlag, 1997.

[23] M. S. Rahman. *Efficient Algorithms for Drawing Planar Graphs.* PhD thesis, Graduate School of Information Sciences, Tohoku University, Sendai, Japan, 1999.

[24] P. Scheffler. A linear algorithm for the pathwidth of trees. In *the Proceedings of the Topics in Combinatorics and Graph Theory*, pages 613–620. Physica-Verlag, 1990.

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

[26] M. Suderman. Pathwidth and layered drawing of trees. *International Journal of Computational Geometry and Applications*, 14(3):203–225, 2004.

[27] M. Suderman. Proper and planar drawings of graphs on three layers. In *the Proceedings of the 15th International Symposium on Graph Drawing (GD 2005)*, volume 3843 of *Lecture Notes in Computer Science*, pages 434–445. Springer-Verlag, 2006.

[28] K. Wagner. Bemerkungen zum vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 46:26–32, 1936.

[29] J. N. Warfield. Crossing theory and hierarchy mapping. *IEEE Transactions on Systems, Man, and Cybernetics*, 7:502–523, 1977.

[30] M. S. Waterman and J. R. Griggs. Interval graphs and maps of DNA. *Bulletin of Mathematical Biology*, 48:189–195, 1986.

[31] D. B. West. *Introduction to Graph Theory*. Prentice-Hall, Upper Saddle River, New Jersey, USA, 2001.

[32] A. P. Wolfman and D. M. Berry. flo - a language for typesetting flowcharts, 1989.

# Index