M.Sc. Engineering Thesis

Bi-Chromatic Point Set Embeddings of Trees with Fewer Bends

by

Khaled Mahmud Shahriar

Submitted To

Department of Computer Science and Engineering
in partial fulfilment of the requirements for the degree of
Master of Science in Computer Science and Engineering

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka 1000

March 29, 2008

The thesis titled "**Bi-Chromatic Point Set Embeddings of Trees with Fewer Bends**", submitted by Khaled Mahmud Shahriar, Roll No. 040405014P, Session April 2004, 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 March 29, 2008.

Board of Examiners

1. ______________________________

Dr. Md. Saidur Rahman — Chairman
Professor & Head — (Supervisor & Ex-officio)
Department of CSE
BUET, Dhaka 1000

2. ______________________________

Dr. M. Kaykobad — Member
Professor
Department of CSE
BUET, Dhaka 1000

3. ______________________________

Dr. Mahmuda Naznin — Member
Assistant Professor
Department of CSE
BUET, Dhaka 1000

4. ______________________________

Dr. Masud Hasan — Member
Assistant Professor
Department of CSE
BUET, Dhaka 1000

5. ______________________________

Dr. Md. Elias — Member
Professor — (External)
Department of Mathematics
BUET, Dhaka 1000

Candidate's Declaration

This is to certify that the work entitled "Bi-Chromatic Point Set Embeddings of Trees with Fewer Bends" is the outcome of the investigation carried out by me under the supervision of Dr. Md. Saidur Rahman in the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka-1000. It is also declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

Khaled Mahmud Shahriar
Candidate

Contents

List of Figures

Acknowledgments

All praise be to ALLAH. I thank ALLAH (SWT) for providing me the knowledge, capability and perseverance to persist and successfully complete this thesis. Among His means, I express my utmost gratefulness and gratitude to my supervisor Dr. Md. Saidur Rahman, Professor, Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology. It has been a great opportunity to work under a world renowned researcher like him. His vast experience and in-depth knowledge in graph drawing and graph theory, scholarly guidance with precious advice and witty encouragement have been a great impetus to the accomplishment of my thesis work.

I also want to thank the members of my thesis committee: Professor Dr. M. Kaykobad, Professor Dr. Md. Elias, Dr. Mahmuda Naznin and Dr. Masud Hasan for their valuable suggestions.

I express my gratitude towards Tanveer Awal for the support he has extended while writing this thesis. My special thanks also goes to Md. Mostofa Ali Patwary who generously spent his time and effort in reviewing my initial writings.

It would be inappropriate not to mention the contribution of the members of our research group. They listened to all of my presentations and gave me valuable advices. I want to thank them also.

Finally, I would like to express my ever gratefulness to my parents and all others who have contributed to this thesis work. May ALLAH reward them all in here and hereafter.

Abstract

Let G be a planar graph such that each vertex in G is colored by either red or blue color. Assume that there are n_r red vertices and n_b blue vertices in G. Let S be a set of fixed points in the plane such that $|S| = n_r + n_b$ where n_r points in S are colored by red color and n_b points in S are colored by blue color. A bichromatic point-set embedding of G on S is a crossing free drawing of G such that each red vertex in G is mapped to a red point in S and each blue vertex in G is mapped to a blue point in S and each edge is drawn as a polygonal curve. In this thesis, we study the problem of computing bichromatic point-set embeddings of trees with fewer bends per edge on some special configurations of point-sets. Let S be such that no two points in S have same x-coordinates. Assume an ordering l of the points in S by increasing x-values. S is called a consecutive point-set when all the points of same color appear consecutively in l. S is called an alternating point-set when red and blue points alternate in l. In this thesis, we show that any tree G has a bichromatic point-set embedding on a point set S with at most one bend per edge if S is either a consecutive point-set or an alternating point-set and such an embedding can be found in linear time.

Chapter 1

Introduction

A graph consists of a set of vertices and a set of edges, each joining two vertices. Graphs may be used to represent any information that can be modeled as objects and relationships between the objects. A drawing of a graph can be thought of as a diagram consisting of some points on the plane corresponding to the vertices of the graph together with some line segments corresponding to the edges connecting the points. A graph when drawn gives a sort of visualization of the information represented by the graph. One of the objectives of graph drawing is to obtain such representation of a graph that makes the underlying structure of the graph easily understandable, and moreover the drawing should satisfy some criteria that arise from the application point of view. In this thesis, we deal with a significant graph drawing problem known as the bichromatic point-set embedding of planar graphs with fewer bends per edge. In this chapter, we discuss the applications of point-set embeddings of planar graphs. We also review the previous results regarding point-set embedding with minimum number of bends per edge and present the objectives of the thesis. We start with Section 1.1 by giving a precise definition of bichromatic point-set embedding problem. Section 1.2 describes some practical applications of point-set embedding problem. Section 1.3 reviews the previous works in this field. Section 1.4 addresses the scope of this thesis. In Section 1.5, we present the summary of the thesis.

1.1 Bichromatic Point-Set Embedding

Let $G = (V, E)$ be a planar graph where V and E are the set of vertices and edges, respectively. Let $V = V_r \bigcup V_b$, where it is assumed that the vertices in V_r are colored red and the vertices in V_b are colored blue. Let S be a set of points in the plane such that $|S| = |V_r| + |V_b|$ and S contains $|V_r|$ red points and $|V_b|$ blue points. A *bichromatic point-set embedding* of G on S is a crossing free drawing of G such that each red vertex $v_r \in V_r$ is mapped to a red point $p_r \in S$ and each blue vertex $v_b \in V_b$ is mapped to a blue point $p_b \in S$ and each edge is drawn as a polygonal curve. For example, Figure 1.1(a) shows a planar graph G with four red vertices and three blue vertices. Throughout the thesis we draw a red vertex by a white circle and a blue vertex by a black circle. For ease of illustration vertices are numbered and the number is shown inside each vertex. Figure 1.1(b) shows a point-set S with four red points and three blue points. Figure 1.1(c) illustrates a bichromatic point-set embedding of G on S where the number inside each point represents the vertex mapped to that point. Figure 1.1(c) shows that all the edges are not drawn as straight lines in such an embedding. Some edges are drawn as polylines with bends to maintain the planarity of the drawing. A bend is a point where a line changes its direction.

Bichromatic point-set embedding of a plananr graph is not unique i.e. for a given graph G and a point-set S, there may be more than one bichromatic point-set embedding of G on S distinguished by different mappings of vertices of G on points in S. For example, Figure 1.2 represents two bichromatic point-set embeddings of the graph G in Figure 1.1(a) on the point-set of Figure 1.1(b). Now consider the drawing in Figure 1.2(a). There is at least one edge (the edge connecting the vertices a and b) that contains two bends. We call this drawing a bichromatic point-set embedding with at most two bends per edge. On the other hand, we call the drawing in Figure 1.2(b) a bichromatic point-set embedding with at most one bend per edge since number of bends on any edge is at most one. It

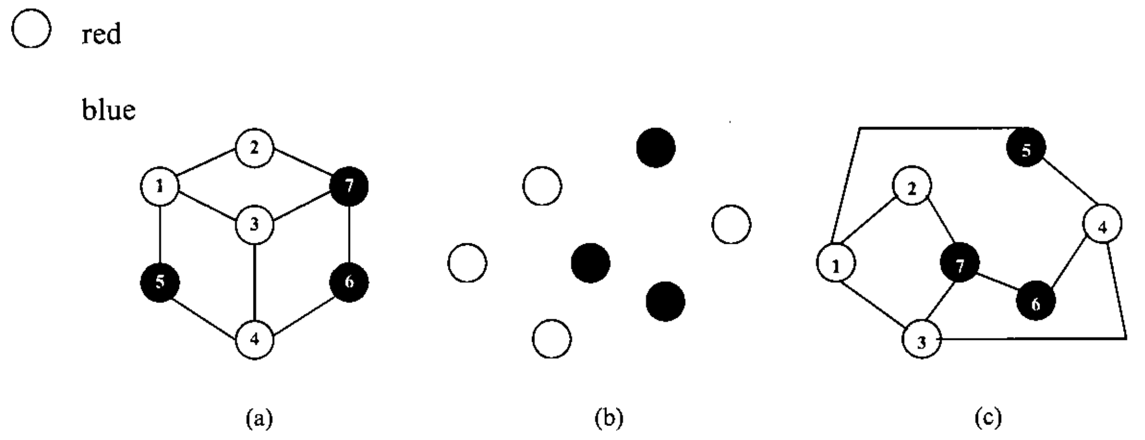


Figure 1.1 (a) A planar graph G, (b) a point-set S, and (c) a bichromatic point-set embedding of G on S.

is desirable both from practical and theoretical point of view to compute a bichromatic point-set embedding that contains fewer bends per edge. Therefore, we say the drawing in Figure 1.2(b) is better and preferable than that in Figure 1.2(a).

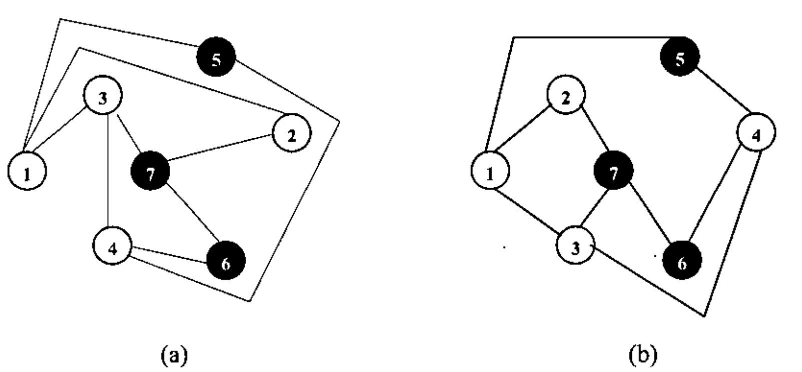


Figure 1.2 Two bichromatic point-set embeddings of a graph. (a) With at most two bends per edge, and (b) with at most one bend per edge.

The general version of the problem of computing bichromatic point-set embedding is known as the *k-chromatic point-set embedding* problem. In the k-chromatic point-set embedding problem, the input is a planar graph $G = (V, E)$ and a point-set S such that vertices in G and points in S are colored using k different colors; k may be any value in the range $1 \leq k \leq |V|$. For any color c, number of vertices in G of color c equals to the number of points of color c in S. The desired output is a crossing free drawing of G such that each vertex in G is mapped to a point of the same color in S and each edge is drawn as a polygonal curve. In the next section, we discuss some of the practical applications of k-chromatic point-set embedding problem.

1.2 Applications

The problem of finding k-chromatic point-set embeddings of planar graphs has practical applications in drawing graphs with semantic constraints [S02] and also in VLSI design [SY02]. We mention here an application in VLSI circuit implementations. Consider the circuit C in Figure 1.3(a) comprising of three different types of basic gates. Figure 1.3(b) shows a circuit board B where the three different types of gates have been prefabricated. The problem is to implement the given circuit C on the circuit board B. The implementation involves mapping each gate from C to a gate of same type in B and then making connections among the mapped gates in B according to the circuit C. Obviously no two connecting wires should overlap while specifying the connections. Figure 1.3(c) represents one such implementation. We can model this problem by considering the circuit C as a graph G where each vertex represents a gate and each edge represents a connection between two gates. Gates of any particular type are represented by vertices having same color. The circuit board B can be modeled as a set of points S on the plane where each point represents a prefabricated gate on the board. A particular type of gate is represented by points with the same color. Subsequently the problem of implementing C on B reduces to the problem of computing k-chromatic point-set embedding of G on S. The value of k in Figure 1.3 is three since there are three different types of gates in the given circuit.

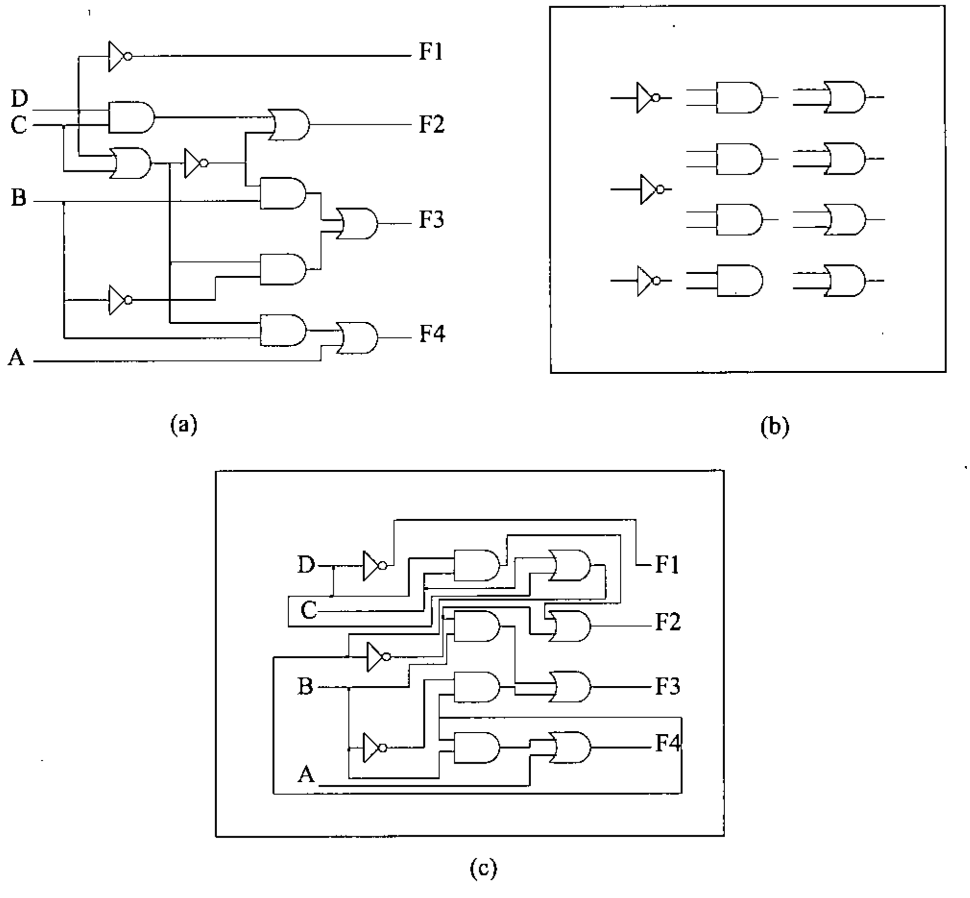


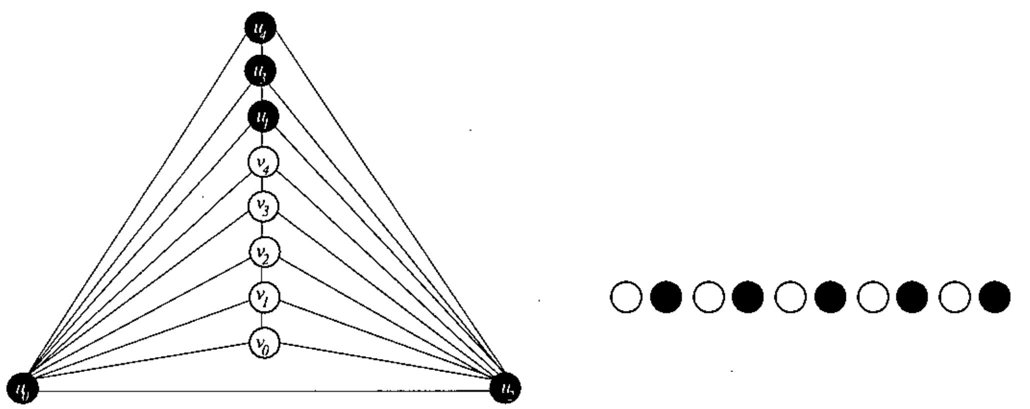
Figure 1.3 (a) A digital circuit C, (b) a circuit board B, and (c) implementation of C on B.

1.3 Previous Results

In this section, we review the previous works regarding k-chromatic point-set embedding and bichromatic point-set embedding of planar graphs.

In the k-chromatic point-set embedding problem, the parameter k denotes the number of different colors that have been used to color the vertices of the given graph G. The minimum value for k is 1 when all the vertices in G are of same color. The maximum possible value for k is $|V|$ when no two vertices in G have same color. For $k = 1$, it has been shown in [C03] that the problem of determining whether a planar graph G has an

1-chromatic point-set embedding (i.e. k=1) without bends is NP-complete. Hence researchers have focussed on finding k-chromatic point-set embedding by introducing bends on the edges and tried to determine the number of bends per edge that will be sufficient to compute such drawing. For $k = n$ where n denotes the number of vertices in the given graph, it has been shown in [PW01] that $O(n)$ bends per edge are required for n-chromatic point-set embedding of a planar graph; hence the number of bends per edge increases linearly with the number of vertices in G for the maximum value of k. On the other hand, for $k = 1$, [KW02] has shown that any planar graph has *1-chromatic point-set embedding* with at most two bends per edge. Surprisingly for the next immediate value of k i.e. when $k = 2$, [GLT06] have shown that there exits instances of planar graphs that require linear number of bends per edge for bichromatic point-set embedding. Figure 1.4(a) shows such a planar graph. [GLT06] have proved that any bichromatic point-set embedding of the graph in Figure 1.4(a) on the point-set in Figure 1.4(b) must contain atleast one edge that has linear number of bends. However there are smaller classes



(a) (b)

Figure 1.4 Bichromatic point-set embedding of the planar graph G in (a) on the point-set S in (b) requires linear number of bends per edge.

of planar graphs that admit bichromatic point-set embedding with constant number of bends per edge. [GLT06] have presented algorithms to compute bichromatic point-set embeddings of paths and cycles with at most one bend per edge and that of caterpillars with at most two bends per edge and [GDL06] have proved that every outerplanar graph has bichromatic point-set embedding with at most 5 bends per edge. Interestingly, it is also possible to find bichromatic point-set embeddings with constant number of bends per edge by working on restricted configurations of point-sets as [GDL07] have shown that every planar graph has k-chromatic point-set embeddings with at most $3k + 7$ bends per edge on consecutive point-sets. Therefore, there is a motivation from theoretical interest to find out what may be the other classes of planar graphs and special point-set configurations that admit k-chromatic point-set embeddings with fewer number of bends per edge.

1.4 Scope of the Thesis

The class of planar graphs we have considered in this thesis is "tree". We have tried to find out how many bends per edge are sufficient for bichromatic point-set embeddings of trees on some special configurations of point-sets, namely consecutive point-sets and alternating point-sets. Let a given point-set S be such that no two points in S have same x-coordinate. Assume an ordering l of the points in S by increasing x-values. S is called a *consecutive point-set* when all the points of same color appear consecutively in l. S is called an *alternating point-set* when points of two different colors alternate in l. In this thesis, we study the problem of computing bichromatic point-set embeddings of trees on consecutive point-sets and alternating point-sets with at most one bend per edge and also developing linear-time algorithms to compute such drawings.

1.5 Summary

The main results of this thesis are as follows.

1. We give a linear-time algorithm for computing bichromatic point-set embeddings of trees on consecutive point-sets with at most one bend per edge.

2. We present a linear-time algorithm for computing bichromatic point-set embeddings of trees on alternating point-sets with at most one bend per edge.

The thesis is organized as follows. Chapter 2 defines basic terminologies relevant to the graphs, graph algorithms and point-set embedding problems to understand our research work. Chapter 3 describes the algorithm that computes bichromatic point-set embedding of trees on a consecutive point-set in linear time. Chapter 4 shows a linear-time algorithm for finding bichromatic point-set embedding of trees on an alternating point-set in linear time. Finally, Chapter 5 gives the conclusion.

Chapter 2

Preliminaries

In this chapter, we give definitions of some basic terms along with some discussion on complexity theory. Definitions that are not given here are discussed as they are needed. In Section 2.1, we start by giving the definitions of some basic terms that are related to and used throughout this thesis. Section 2.2 describes the terms related to bichromatic point-set embeddings of planar graphs. Section 2.3 defines the complexity of an algorithm. Finally, Section 2.4 summarizes this chapter.

2.1 Basic Terminology

In this section, we provide definitions of some graph-theoretical terms used throughout the remainder of this thesis. For readers interested in graph theory, we refer to [NR04, BETT99] and [We01].

2.1.1 Graphs

Let $G = (V, E)$ be a simple graph with vertex set V and edge set E. We denote the set of vertices of G by $V(G)$ and the set of edges of G by $E(G)$. We denote an edge joining vertices v_i, v_j of G by (v_i, v_j). If $(v_i, v_j) \in E$, then two vertices v_i, v_j are said to be *adjacent* in

G; edge (v_i, v_j) is then said to be *incident* to vertices v_i and v_j; v_i is a *neighbor* of v_j. The *degree* of a vertex v in G is the number of edges incident to v in G. Figure 2.1 depicts a simple graph G, where each vertex in $V(G) = \{v_1, v_2, v_3, v_4, v_5\}$ is drawn by small black circle and each edge in $E(G) = \{(v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_1), (v_1, v_5), (v_2, v_5), (v_3, v_5), (v_4, v_5)\}$ is drawn by a line segment.

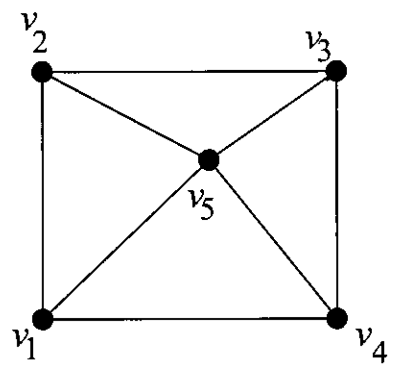


Figure 2.1 A simple graph with five vertices and eight edges.

A *path* in G is an ordered list of distinct vertices $(v_1, v_2, \ldots, v_{q-1}, v_q) \in V$ such that $(v_{i-1}, v_i) \in E$ for all $2 \le i \le q$ [We01]. The length of a path is one less than the number of vertices on the path. A path or walk is *open* if $v_0 \neq v_q$. A path is *closed* if $v_1 = v_q$. A *closed path* containing at least one edge is called a *cycle*. For a path P, $V_{in}(P)$ denotes the internal vertices of P, i.e., all the vertices except the endpoints of P. In Figure 2.1, (v_1, v_2, v_3) is a path and (v_1, v_2, v_5, v_1) is a cycle.

2.1.2 Connectivity

A graph G is *connected* if for any two distinct vertices v_i, v_j of G there is a path between v_i and v_j in G. A graph which is not connected is called a *disconnected graph*. A *component* of a graph is a maximal connected subgraph. The graph in Figure 2.2(a) is connected since there is a path between any two distinct vertices of the graph. On the other hand, the graph in Figure 2.2(b) is disconnected since there is no path between v_1 and v_2. The graph in Figure 2.2(b) has two components G_1 and G_2 indicated by dotted lines.

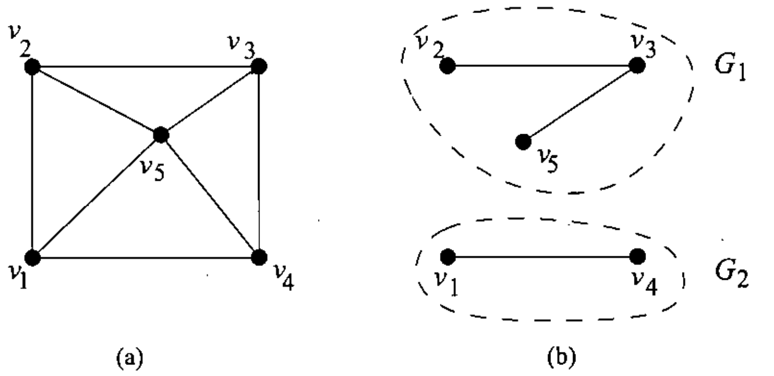


Figure 2.2 (a) A connected graph, and (b) a disconnected graph with two connected components.

2.1.3 Trees

A *tree* is a connected graph without any cycle. Figure 2.3 is an example of a tree. The vertices in a tree are usually called *nodes*. A *rooted tree* is a tree in which one of the nodes is distinguished from others. The distinguished node is called the *root* of tree. The root of a tree is usually drawn at the top. In Figure 2.3, the root is v_0. In a rooted tree G with root v_0, if the last edge on the path from v_0 to a vertex u is (u, v), then v is called the *parent* of u and u is a *child* of v. For example, in Figure 2.3, vertex v_1 is the parent of v_4 and v_4 is a child of v_1.

2.1.4 Planar Graphs and Plane Graphs

A graph is *planar* if it can be embedded in the planes so that no two edges intersect geometrically except at a vertex to which they are both incident. Note that a planar graph may have an exponential number of embeddings. Figure 2.4 shows two planar

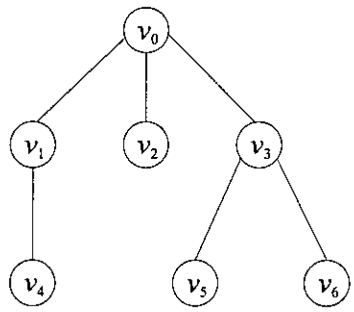


Figure 2.3 A tree.

embeddings of the same planar graph.

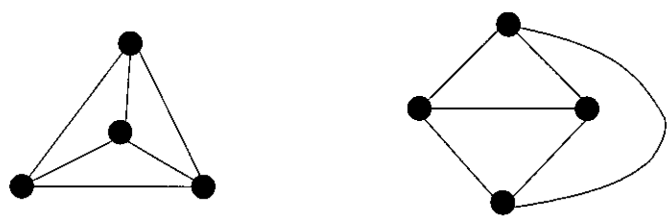


Figure 2.4 Two planar embeddings of the same graph.

2.2 Bichromatic Point-Set Embedding

2.2.1 2-colored Planar Graphs

A *2-coloring* of a planar graph $G = (V, E)$ is a partition of $V(G)$ into two nonempty disjoint sets V_r and V_b where the vertices in V_r and V_b are *red vertices* and *blue vertices* respectively. Given a vertex $v \in V(G)$, we denote the color of v by $c(v)$. We say a graph G is *2-colored* if G has a 2-coloring. Figure 1.1(a) shows a 2-*colored* graph G.

2.2.2 Point-sets

By the term *point-set*, we refer to a set of fixed points in the Euclidian plane. In the rest of the paper, we assume that for any given point-set S, no two points of S have same x-coordinates (if this is not the case for some point-set S, we can rotate the plane to achieve distinct x-coordinates for all the points in S). We use $x(p)$ to denote the x-coordinate of point $p \in S$. Let $|S| = n$ and $p_0, p_1, \ldots, p_{n-1}$ be the points of S ordered according to their x-coordinates i.e. $x(p_0) < x(p_1) < \ldots < x(p_{n-1})$. For any two points p, q in S, we say p is to the *left* of q (or q is to the *right* of p) if $x(p) < x(q)$. For any point $p \in S$, we use $next(p)$ to denote the point $q \in S$ where q is immediately to the right of p in the ordering of points in S by increasing x values.

2.2.3 2-colored Point-set

A 2-*coloring* of a point-set S is a partition of S into two nonempty disjoint sets S_r and S_b where the points in S_r and S_b are colored *red* and *blue* respectively. Given a point $p \in S$, we denote the color of p by $c(p)$. We say a point-set S is 2-*colored* if S has a 2-coloring. Figure 1.1(b) shows a 2-*colored* point-set S.

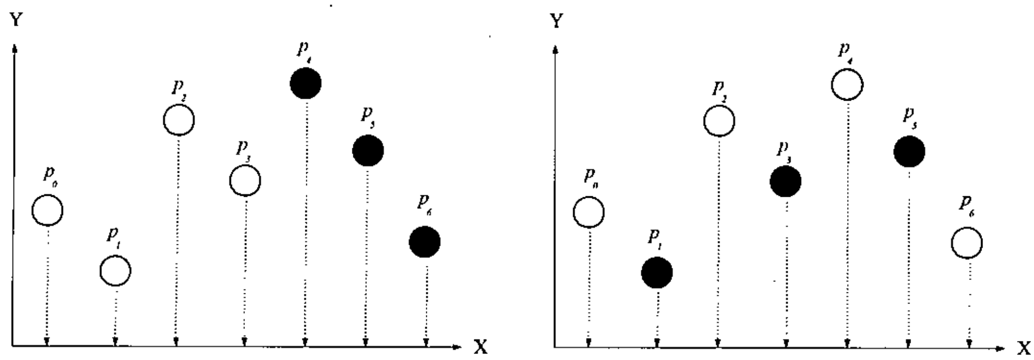
2.2.4 Bichromatic Point-set Embedding

Let $G = (V, E)$ be a 2-colored planar graph where $V = V_r \bigcup V_b$ and $S = S_r \bigcup S_b$ be a 2-colored point-set in the plane such that: (i) vertices in V_r and points in S_r are colored red, (ii) vertices in V_b and points in S_b are colored blue and (iii)$|V_b| = |S_b|$ and $|V_r| = |S_r|$. We say that S is *compatible* with G. A *bichromatic point-set embedding* of G on S is a crossing free drawing of G such that: (i) each vertex $v \in V(G)$ is mapped to a point $p \in S$ where $c(p) = c(v)$, and (ii) each edge $e \in E(G)$ is drawn as a polygonal chain λ. A point shared by any two consecutive segments of a polygonal chain λ is called a *bend* of e. Figure 1.1(c) shows a bichromatic point-set embedding of the 2-colored planar graph

G in Figure 1.1(a) on the 2-colored point-set S in Figure 1.1(b) where S is compatible with G.

2.2.5 Consecutive and Alternating Point-Set

We say a 2-colored point-set S a *consecutive point-set* if for every pair of points p, q in S where $c(p) = c(q)$, there is no point r in S such that $x(p) < x(r) < x(q)$ and $c(r) \neq c(p) = c(q)$. In other words, in a 2-colored consecutive point-set, points of each color are consecutive according to the x-coordinate ordering. We say a 2-colored point-set S is an *alternating point-set*, if for any point p in S, $c(p) \neq c(next(p))$. In other words, in a 2-colored alternating point-set, colors of points alternate in the x-coordinate ordering. It is obvious that for any alternating point-set σ with n_r red vertices and n_b blue vertices, either $n_r = n_b$ (i.e. when the color of the leftmost point of σ is different from the color of its rightmost point) or $n_r = n_b \pm 1$(i.e. when both the endpoints of σ are of same color). We define a single point (either red or blue) to be an alternating point-set of size 1. Figure 2.5(a) shows a 2-colored consecutive point-set and Figure 2.5(b) shows a 2-colored alternating point-set.



(a) (b)

Figure 2.5 (a) A 2-colored consecutive point-set, and (b) a 2-colored alternating point-set.

2.2.6 RB-Sequence

We call a 2-colored point-set, S an *RB-sequence* denoted by σ when all the points of S are collinear. Let l be the line that passes through the points in σ. We call l a *spine* of σ. A spine defines 2 half planes (called *pages*) sharing line l; the top half plane is called the *top page* and the bottom half plane is called the *bottom page*. Figure 2.6 shows an RB-sequence of size 8.

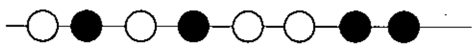


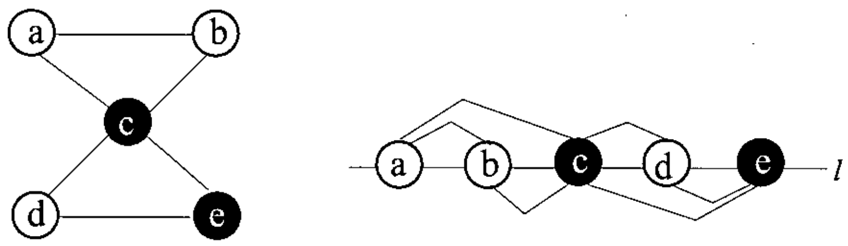
Figure 2.6 An RB-sequence.

2.2.7 Accessibility of Points

Let G be a 2-colored planar graph and σ be a 2-colored RB-sequence compatible with G. Let Γ be a bichromatic point-set embedding of G on σ. Let p be a point on the spine l of σ. We say p is *accessible from top (bottom) page* in Γ if there is no such edge e in Γ that e is drawn through the top (bottom) page and p lies between the endpoints of e on l. For example, consider the 2-colored graph in Figure 2.7(a). Figure 2.7(b) represents a bichromatic point-set embedding of G on some RB-sequence σ. In the drawing of Figure 2.7(b), the point representing the vertex b is not accessible from the top page since b lies between the endpoints of the edge connecting the vertices a and c. Similarly the point representing the vertex d is not accessible from the bottom page. The point that corresponds to the vertex c is accessible from both the pages. It is obvious that the leftmost and rightmost point of σ is always accessible from both the pages irrespective of the drawing Γ.

We have the following observation from [GLT06].

Observation 2.2.1 *Let Γ be a drawing that represents a bichromatic point-set embedding of a 2-colored graph G on an RB-sequence σ. Let p and q be two points on the spine of*



(a) (b)

Figure 2.7 Illustration of accessibility of points. (a) A planar Graph G, and (b) a drawing Γ of G.

σ that are accessible from the same page π. Then it is possible to connect p and q with a polygonal chain λ with at most one bend such that λ is entirely contained in π and λ does not cross any other edge of Γ.

Proof. Since p and q are both accessible from π, there is no edge $e = (u, v)$ in γ such that the closed region bounded by $\overline{uv} \bigcup e$ has p inside and q outside. Therefore, p and q can be connected by a polygonal chain with one bend entirely contained in π and that does not cross any edge of σ.

2.2.8 Chromatic Equivalence

Let P and Q be two 2-colored point-sets. We say P is *chromatic equivalent* to Q if the following two conditions hold: (i) $|P| = |Q| (= n)$, and (ii) $c(p_i) = c(q_i)$ for $0 \leq i \leq n-1$ where $p_0, p_1, \ldots, p_{n-1}$ and $q_0, q_1, \ldots, q_{n-1}$ denote the points in P and Q respectively in the order of increasing x-coordinate value. We make the following observation regarding chromatic equivalence of two alternating point-sets.

Observation 2.2.2 *Let P and Q be two alternating point-sets such that $|P| = |Q|$. If the leftmost points of both P and Q have same color, then P is chromatic equivalent to Q.*

We have the following lemma that relates bichromatic point-set embeddings on two chromatic equivalent point-sets.

Lemma 2.2.3 *Let $G = (V, E)$ be a 2-colored planar graph and S be a 2-colored point-set compatible with G. Let σ be an RB-sequence chromatic equivalent to S. If G has a bichromatic point-set embedding on σ with at most one bend per edge then G admits a bichromatic point-set embedding on S with at most one bend per edge.*

Proof. The proof is constructive. Let Γ be the drawing that represents a bichromatic point-set embedding of G on σ. Let l be the spine of σ and $q_0, q_1, \ldots, q_{n-1}$ be the points of σ ordered on l from left to right where $|V| = n$. The drawing of each edge in Γ is contained either within the top plane or the bottom plane as defined by l since number of bends on any edge is at most one. Let v_i be the vertex of G that is mapped on q_i in σ $(0 \leq i < n)$. Let $p_0, p_1, \ldots, p_{n-1}$ be the points in S. Since σ is chromatic equivalent to S, it follows that $c(q_i) = c(p_i) = c(v_i)$ for $0 \leq i < n$. We map vertex v_i of G on $p_i \in S$ and draw the edges (v_i, v_{i+1}) if exist, as straight line segments. The remaining edges are the edges that are drawn using one bend in Γ. Now using the techniques in [KW02], we draw those edges connecting points of S with at most one bend and without any edge crossings. Figure 2.8 illustrates the techniques described. Thus we have a bichromatic point-set embedding of G on S. Since the technique described in [KW02] draws each edge in constant amount of time, it follows that construction of bichromatic point-set embedding of G on S from the point-set embedding of G on σ requires linear time. □

2.3 Algorithms and Complexity

In this section, we briefly introduce some terminologies related to complexity of algorithms. For interested readers, we refer the book of Cormen *et. al.* [CLRS04].

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

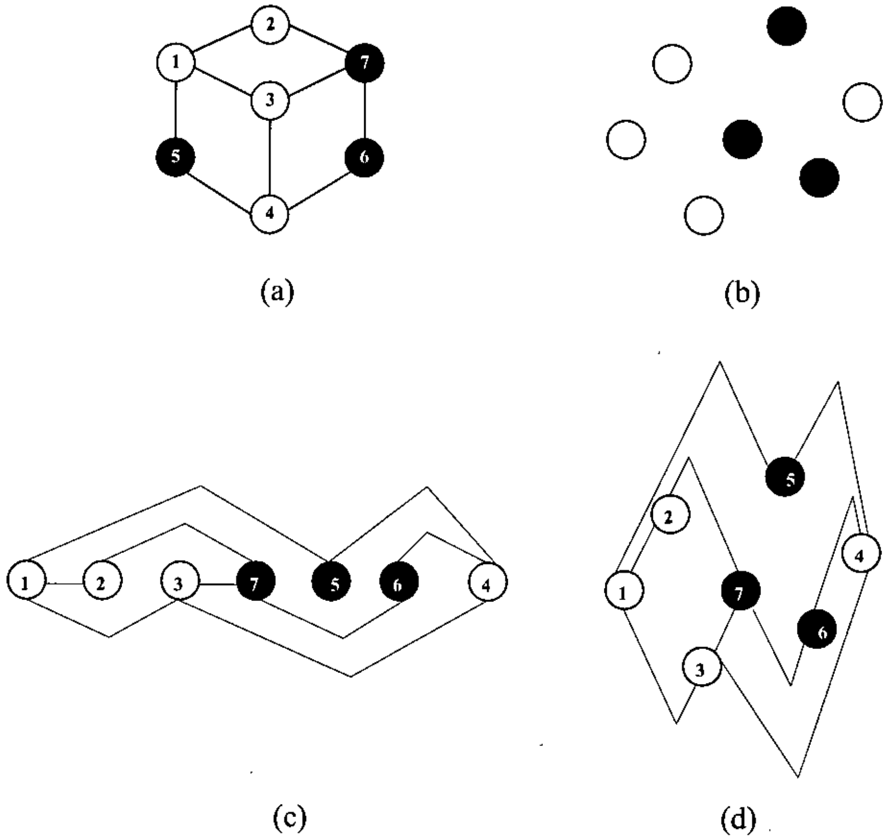


Figure 2.8 An illustration for the proof of Lemma 2.2.3. (a) A 2-colored graph G, (b) a point-set S, (c) a bichromatic point-set embedding of G on a consecutive RB-sequence σ, and (d) a bichromatic point-set embedding of G on S.

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.

2.3.1 The Notation $O(n)$

In analyzing the complexity of the 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 $g(n)$ be 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 n_0 such that $0 \leq f(n) \leq c_1 g(n)$ for all $n \geq n_0$. Thus the running time of an algorithm may be bounded from above by phrasing like "takes time $O(n^2)$" meaning that the upper bound of the algorithm is $O(n^2)$.

2.3.2 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 $O(n)$, $O(n \log n)$, $O(n^{100})$ etc. The remaining algorithms are usually referred to as *exponential* or *nonpolynomial*. 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. For planar graphs, the number of edges $m = O(n)$ and the number of faces $f = O(n)$.

2.4 Summary

In this chapter, we have defined some basic graph-theoretical terms as well as some special terms related to bichromatic point-set embedding. Besides we have also discussed different terms related to complexity of algorithms that will be used in the later sections.

Chapter 3

Embedding on Consecutive Point-Set

In this chapter, we prove that trees admit bichromatic point-set embeddings on consecutive point-sets with at most one bend per edge and such an embedding can be found in linear time. We first present in Section 3.1 a linear-time algorithm that computes a bichromatic point-set of any given tree G on an arbitrary consecutive RB-sequence σ with at most one bend per edge. Then in Section 3.2 we obtain a bichromatic point-set embedding of G on any consecutive point-set S with at most one bend per edge from the bichromatic point-set embedding of G on σ. Finally, Section 3.3 summarizes this chapter.

3.1 Bichromatic Point-Set Embedding on Consecutive RB-sequence

In this section, we describe a linear-time algorithm that computes a bichromatic point-set embedding of a 2-colored tree G with at most one bend per edge on a consecutive RB-sequence σ compatible with G. We call this algorithm **Consecutive-Embedding**. Without loss of generality, we assume that the leftmost point in σ is red; this implies that the blue points are to the right of the red points in σ. Let there be n_r red vertices and n_b blue vertices in G and $|V| = n_r + n_b = n$. We assume any red vertex of G as its root and denote it by v_0.

The rest of the section is organized as follows. In Section 3.1.1, we illustrate the Algorithm **Consecutive-Embedding**. In Section 3.1.2, we verify correctness and time complexity of the Algorithm.

3.1.1 Algorithm Consecutive-Embedding

Given a 2-colored tree G and a consecutive RB-sequence σ, Algorithm **Consecutive-Embedding** finds a bichromatic point-set embedding of G on S by mapping vertices of G on points of S in an incremental way. At each step an unmapped vertex of G is mapped on some point in S in such a way that allows mapping of future vertices without any edge crossing and with atmost one bend per edge. We use the following notations to illustrate the drawing algorithm. Let γ_k denotes the drawing after some step k, $k \geq 0$. For example, Figure 3.1(b) shows the drawing γ_4 after step 4 for the input graph in Figure 3.1(a).

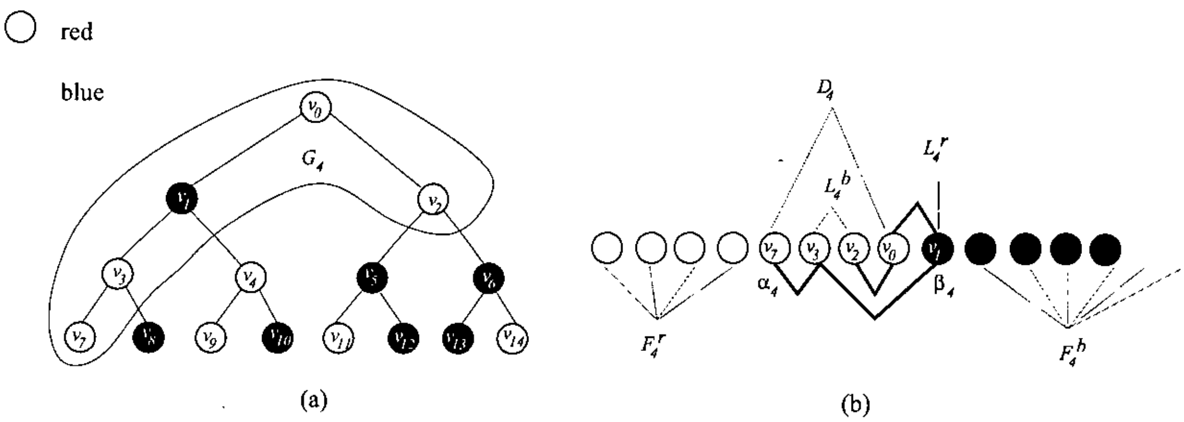


Figure 3.1 (a) A 2-colored tree G, and (b) the drawing γ_4 and the sets F_4^r, F_4^b, L_4^r, L_4^b and D_4.

We denote by G_k the subgraph of G that has been drawn in γ_k; We call any vertex v in $V(G)\setminus V(G_k)$ an *unmapped* vertex and vertices in G_k *mapped* vertices. A mapped vertex v of G_k is a *live* vertex if it has at least one unmapped neighbor; otherwise v is called a *dead* vertex. We use $U_k(v)$ to denote the set of unmapped neighbors of any live vertex v; v is an *R-live* vertex if v has at least one neighbor u such that $u \in U_k(v)$ and $c(u)$ is red; v is called a *B-live* vertex if v has at least one neighbor u such that $u \in U_k(v)$ and $c(u)$ is blue.

Let $\sigma_k \subseteq \sigma$ denotes the set of points representing the vertices of G_k in γ_k. For a vertex v of G, we use $p(v)$ to denote the point of σ that represents v. The leftmost and rightmost points of σ_k will be denoted by α_k and β_k respectively. We say any point p of $\sigma \backslash \sigma_k$ is a *free* point; A *free* point p is a *free red* point if $c(p)$ is red; otherwise p is a *free blue* point if $c(p)$ is blue. The set of free red points and free blue points of $\sigma \backslash \sigma_k$ will be denoted by F_k^r and F_k^b respectively. Any point p of σ_k will be called an *R-live* point , *B-live* point or *dead* point if it represents an R-live vertex, a B-live vertex or a dead vertex, respectively in γ_k. The set of R-live, B-live and dead points of σ_k will be denoted by L_k^r, L_k^b and D_k respectively. Figure 3.1 illustrates the given definitions. Figure 3.1(b) shows the drawing γ_4 for the input graph in Figure 3.1(a). In Figure 3.1(b), the point $p(v_1)$ is an R-live point since vertex v_1 has an unmapped neighbor v_4 where $c(v_4)$ is red; the point $p(v_2)$ is a B-live point since vertex v_2 has neighbors v_5 and v_6 that are unmapped vertices and are colored blue; points $p(v_0)$ and $p(v_7)$ are dead points since both vertices v_0 and v_7 have no unmapped neighbors.

At the end of each step, the resulting drawing satisfies the following step invariant properties.

Property 1: All points in F_k^r are to the left of α_k and all points in F_k^b are to the right of β_k.

Property 2: All points in L_k^r are accessible from the bottom page and all points in L_k^b are accessible from the top page.

Property 3: G_k is a connected subgraph of G that contains the vertex v_0 and the drawing γ_k represents a bichromatic point-set embedding of G_k on σ_k with at most one bend per edge.

We now describe the operations at different steps.

At step $k = 0$, the root vertex v_0 where $c(v_0)$ is red, is mapped to the right most free red point f_r of σ. Figure 3.2 illustrates the operations in step 0.

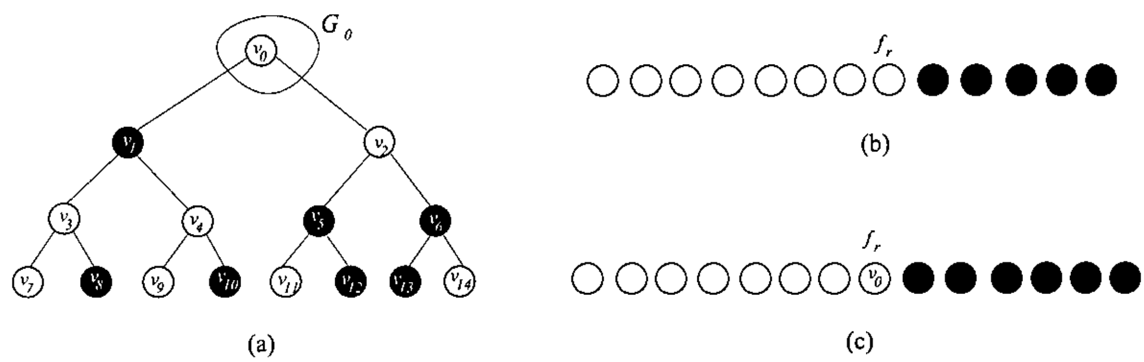


Figure 3.2 An illustration for step 0 of Algorithm **Consecutive-Embedding**. (a) A tree G, (b) a consecutive RB-sequence σ, and (c)the drawing γ_0.

We now prove that the drawing γ_0 satisfies the step invariant properties.

The drawing γ_0 satisfies Property 1: According to our assumption, all the blue points are to the right of the red points in the consecutive RB-sequence σ. We map the root of G to the rightmost free red point f_r of σ. By definition, $\alpha_0 = \beta_0 = f_r$. Hence all the remaining free red points are to the left of α_0 and free blue points are to the right of β_0.

The drawing γ_0 satisfies Property 2: Since no edge is added, it follows that all the points are accessible from both top and bottom pages. Therefore, Property 2 holds trivially.

The drawing γ_0 satisfies Property 3: Since G_0 consists of the single vertex v_0 and no edges, hence it is trivial to show that Property 3 is satisfied.

We now specify the operations to perform at each step k, $k > 0$. We use induction to prove that the resulting drawing γ_k maintains the invariant properties. We consider γ_0 as the base case and as induction hypothesis we assume that the output of the previous step γ_{k-1} satisfies the specified invariant properties. At any step k, we identify the following two cases.

Case 1: There is at least one R-live point in σ_{k-1}.

In this case, $L^r_{k-1} \neq \phi$. Let l_r be the leftmost point in L^r_{k-1}. Let l_r represents the vertex v of G; hence v is an R-live vertex. Consider any vertex u such that $u \in U_{k-1}(v)$ and $c(u)$ is red. We map u to the rightmost point f_r of F^r_{k-1} and add the edge (u, v) connecting points l_r and f_r through the bottom page. Figure 3.3 illustrates case 1.

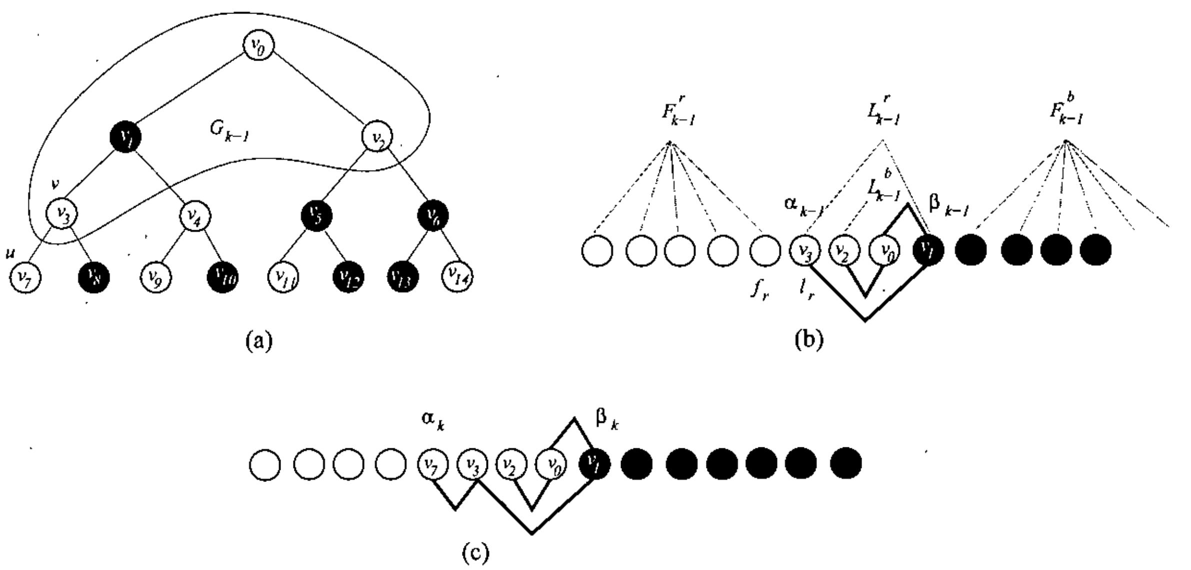


Figure 3.3 An illustration for case 1 of Algorithm **Consecutive-Embedding**. (a) A 2-colored tree G, (b) the drawing γ_{k-1}, and (c) the drawing, γ_k.

We now show that the drawing γ_k satisfies the desired invariant properties.

The drawing γ_k satisfies Property 1: By induction hypothesis, points in F^r_{k-1} are to the left of α_{k-1}. Since f_r is the rightmost point in F^r_{k-1}, it follows that $f_r = \alpha_k$ and $F^r_k = F^r_{k-1} \backslash \{f_r\}$; therefore, points in F^r_k are to the left of α_k. On the other hand, $\beta_{k-1} = \beta_k$ and $F^b_{k-1} = F^b_k$. It follows that points in F^b_k are to the right of β_k.

The drawing γ_k satisfies Property 2: We first show that points in L^r_k are accessible from the bottom page. If there are no R-live points in σ_k, i.e. $L^r_k = \phi$ then this property holds trivially. Otherwise consider a point $p \in L_k^r$ such that p is not accessible from the bottom page in γ_k. Let v_p be the vertex of G represented by p. Hence v_p is an R-live vertex in G_k. Since f_r is the leftmost point in σ_k, f_r is accessible from both the pages in γ_k. It follows that $p \neq f_r$. Since u is mapped on f_r, $v_p \neq u$. Then v_p must be an R-live vertex of G_{k-1}. It follows that p is an R-live point of σ_{k-1}. Therefore, by Property 2 of induction hypothesis, p is accessible from bottom page in γ_{k-1}. Consequently it must be the addition of the edge (u, v) that makes p inaccessible from bottom page in γ_k. The endpoints of the edge (u, v) are the points f_r and l_r. Since f_r is the leftmost point of σ_k, f_r is to the left of p. Also l_r is to the left of p since both l_r and p are in L_{k-1}^r and l_r is the leftmost point of L_{k-1}^r. Since both the endpoints of edge (u, v) lie to the left of p in γ_k, it follows that the edge (u, v) does not modify the accessibility of p from bottom page in γ_k. Therefore, p is accessible from bottom page in γ_k which is a contradiction. Hence all points in L_k^r are accessible from the bottom page.

Next we show that points in L_k^b are accessible from the top page. If σ_k contains no B-live points, i.e. $L_k^b = \phi$ then this property holds trivially. Otherwise consider a point $p \in L_k^b$ such that p is not accessible from top page in γ_k. Let v_p be the vertex of G represented by p. Hence v_p is a B-live vertex in G_k. Since f_r is the leftmost point in σ_k, f_r is accessible from both the pages in γ_k. It follows that $p \neq f_r$. Since u is mapped on f_r, $v_p \neq u$. Then v_p must be a B-live vertex of G_{k-1}. It follows that p is a B-live point of σ_{k-1}. Therefore, by property 2 of induction hypothesis, p is accessible from top page in γ_{k-1}. Consequently it must be the addition of the edge (u, v) that makes p inaccessible from top page in γ_k. But since we draw the edge (u, v) through the bottom page, it cannot modify the accessibility of p from top page. It follows that p is accessible from top page in γ_k which is a contradiction. Therefore, all points in L_k^b are accessible from the top page.

The drawing γ_k satisfies Property 3: According to the operation specified, $V(G_k) = V(G_{k-1}) \bigcup \{u\}$. Since G_{k-1} is a connected graph that contains the vertex v_0(by induction hypothesis) and u is a neighbor of some vertex $v \in V(G_{k-1})$, if follows that G_k is also connected and v_0 is in G_k. Therefore, it remains to show that the edge (u, v) does not create any edge crossings and contains at most one bend. Since $l_r \in L^r_{k-1}$, by Property 2 it is accessible from bottom page in γ_{k-1}. The point $f_r \in F^r_{k-1}$ is to the left of the leftmost point of σ_{k-1} (by Property 1); hence f_r is accessible from both the pages in γ_{k-1}. Therefore, from Observation 2.2.1, f_r and l_r can be connected with a polygonal chain through the bottom page that contains at most one bend and does not cross any other edge in γ_{k-1}. Hence γ_k represents a bichromatic point-set embedding of G_k on σ_k.

Case 2: There is no R-live point but at least one B-live point in σ_{k-1}.

In this case, $L^r_{k-1} = \phi$ and $L^b_{k-1} \neq \phi$. Consider the rightmost point l_b of L^b_{k-1}. Let l_b represents the vertex v of G; hence v is a B-live vertex. Consider any vertex u such that $u \in U_{k-1}(v)$ and $c(u)$ is blue. We map u to the leftmost free blue point f_b of F^b_{k-1} and add the edge (u, v) connecting points l_b and f_b through the top page. Figure 3.4(b) illustrates case 2.

We now show that the required invariants are maintained by the drawing γ_k.

The drawing γ_k satisfies Property 1: By induction hypothesis, points in F^b_{k-1} are to the right of β_{k-1} and f_b is the leftmost point in F^b_{k-1}. Hence $f_b = \beta_k$ and $F^b_k = F^b_{k-1} \backslash \{f_b\}$; therefore, points in F^b_k are to the right of β_k. On the other hand, $\alpha_{k-1} = \alpha_k$ and $F^r_{k-1} = F^r_k$. It follows that points in F^r_k are to the left of α_k.

The drawing γ_k satisfies Property 2: We first show that points in L^r_k are accessible from bottom page. If there are no R-live points in σ_k, i.e. $L^r_k = \phi$ then Property 2 holds trivially. Otherwise it must be the case that $L^r_k = \{f_b\}$ since $L^r_{k-1} = \phi$. Since f_b is the rightmost point in σ_k, it is accessible from both the pages in γ_k. Thus points in L^r_k are accessible from bottom page in γ_k.

We now show that points in L^b_k are accessible from the top page. If there are no B-live points in σ_k, i.e. $L^b_k = \phi$ then this property holds trivially. Otherwise consider a point $p \in L^b_k$ such that p is not accessible from top page in γ_k. Let v_p be the vertex of G

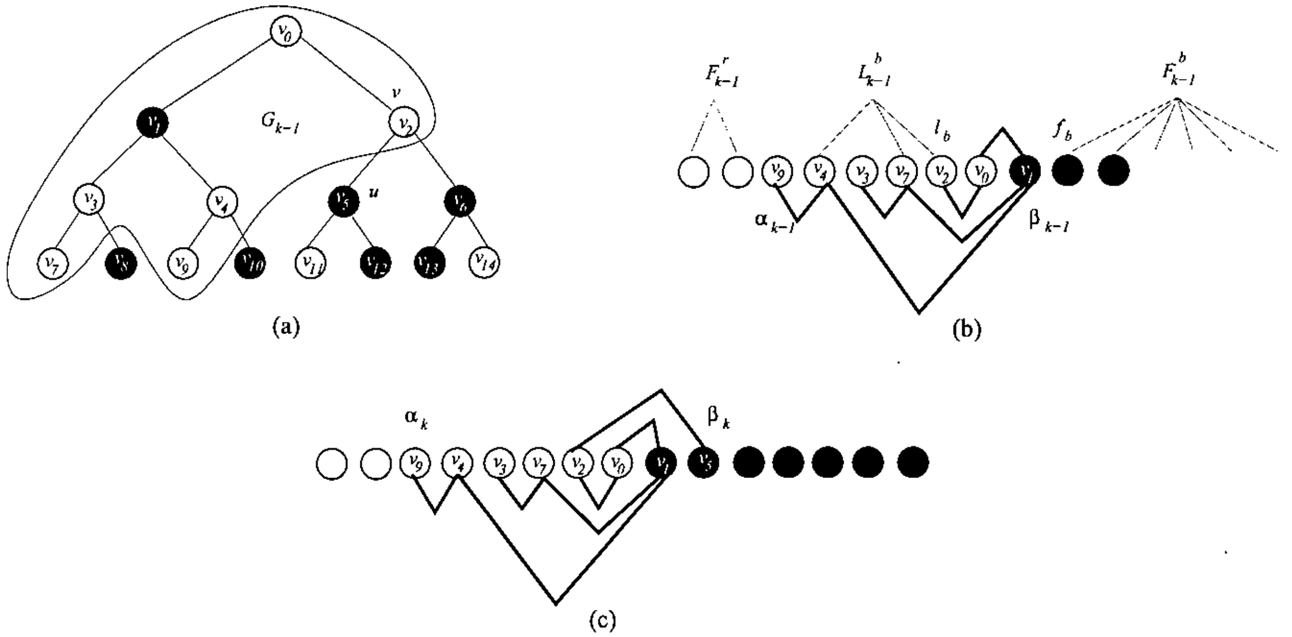


Figure 3.4 An illustration for case 2 of Algorithm **Consecutive-Embedding**. (a) A 2-colored tree G, (b) the drawing γ_{k-1}, and (c) the drawing γ_k.

represented by p. Hence v_p is a B-live vertex in G_k. Since f_b is the rightmost point in σ_k, f_r is accessible from both the pages in γ_k. It follows that $p \neq f_b$. Since u is mapped on f_b, $v_p \neq u$. Then v_p must be a B-live vertex of G_{k-1}. It follows that p is a B-live point of σ_{k-1}. Therefore, by Property 2 of induction hypothesis, p is accessible from top page in γ_{k-1}. Consequently it must be the addition of the edge (u, v) that makes p inaccessible from top page in γ_k. The endpoints of the edge (u, v) are the points f_b and l_b. Since f_b is the rightmost point of σ_k, f_b is to the right of p. Also l_b is to the right of p since both l_b and p are in L^b_{k-1} and l_b is the rightmost point of L^b_{k-1}. Since both the endpoints of edge (u, v) lie to the right of p in γ_k, it follows that the edge (u, v) does not modify the accessibility of p from top page in γ_k. Therefore, p is accessible from top page in γ_k which is a contradiction. Hence all points in L^b_k are accessible from the top page.

The drawing γ_k satisfies Property 3: According to the operation specified, $V(G_k) = V(G_{k-1}) \bigcup \{u\}$. Since G_{k-1} is a connected graph that contains the vertex v_0(by induction hypothesis) and u is a neighbor of some vertex $v \in V(G_{k-1})$, if follows that the graph G_k is also connected and v_0 is in G_k. Therefore, it remains to show that the edge (u, v) does not create any edge crossings and contains at most one bend. Since $f_b \in L^b_{k-1}$, by Property 2 it is accessible from the top page in γ_{k-1}. The point $f_b \in F^b_{k-1}$ is to the right of the rightmost point of σ_{k-1} (by Property 1); hence f_b is accessible from both the pages in γ_{k-1}. Therefore, from Observation 2.2.1, l_b and f_b can be connected with a polygonal chain through the top page that contains at most one bend and does not cross any other edge in γ_{k-1}. Hence γ_k represents a bichromatic point-set embedding of G_k on σ_k.

This concludes the illustration of Algorithm **Consecutive-Embedding**. We now give a formal presentation of Algorithm **Consecutive-Embedding**. Before that we need to describe the data structures that we use in the formal description of Algorithm **Consecutive-Embedding**. We represent a 2-colored tree G using an array of $2|V|$ lists; for each vertex $v \in V$, there are two seperate lists to store the set of red children and the set of blue children of v. We use A_G to denote this representation of G. For example, Figure 3.5(b) shows the representation for the 2-colored tree in Figure 3.5(a). The set of R-live points at any step is stored in a linked list. We denote this list as R_σ. Each element of R_σ holds a pointer to an R-live vertex. The first and last elements of R_σ correspond to the leftmost and rightmost R-live points respectively. R_σ can be accessed from both front and end. We store the set of B-live points in a similiar linked list that we denote as B_σ. Initially the lists R_σ and B_σ are empty. Mapping of vertices to points is stored in an array of size $|\sigma|$. We denote this array as M_σ. The i^{th} element of M holds the vertex mapped to the i_{th} point of σ. Figure 3.6 illustrates the data structures. Figure 3.6(b) shows the drawing γ_k computed after some step $k (k > 0)$ for the input graph G in Figure 3.6(a). Figure 3.6(c) shows A_G after step k. Note that for each vertex $v \in V$, A_G holds the lists of unmapped red and blue children of v since whenever we map a vertex, we remove that node from the list of its parent. Figure 3.6(d) illustrates lists R_σ, B_σ and the array M_σ. We are now ready to present a formal description of the Algorithm

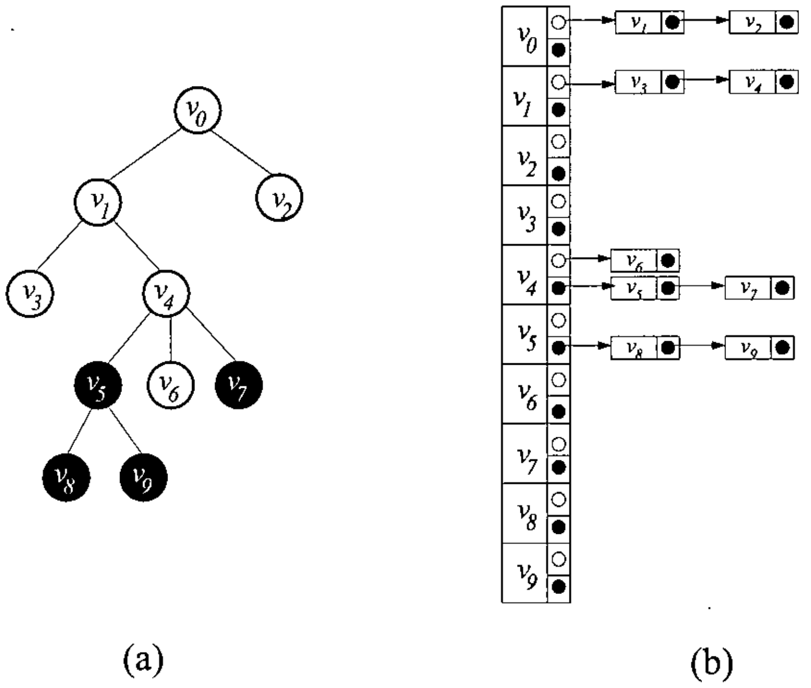


Figure 3.5 (a) A 2-colored tree G, and (b) A_G.

Consecutive-Embedding.

Algorithm Consecutive-Embedding(A_G, M_σ)

{A_G represents a 2-colored rooted tree G and M_σ represents a 2-colored consecutive RB-sequence.}

begin

Let there be n_r red vertices and n_b blue vertices in G;

$f_r := n_r - 1$; {f_r always holds the index of the rightmost free red point in σ. We assume the leftmost point of σ is indexed 0.}

$f_b := n_r$; {F_b always holds the index of the leftmost free blue point in σ.}

Set R_σ and B_σ to **NIL**; {Initially both the lists are empty.}

{Let vertex v_0 be the root of G. At first we embed v_0.}

$M_\sigma[f_r] := v_0$;

$f_r := f_r - 1$;

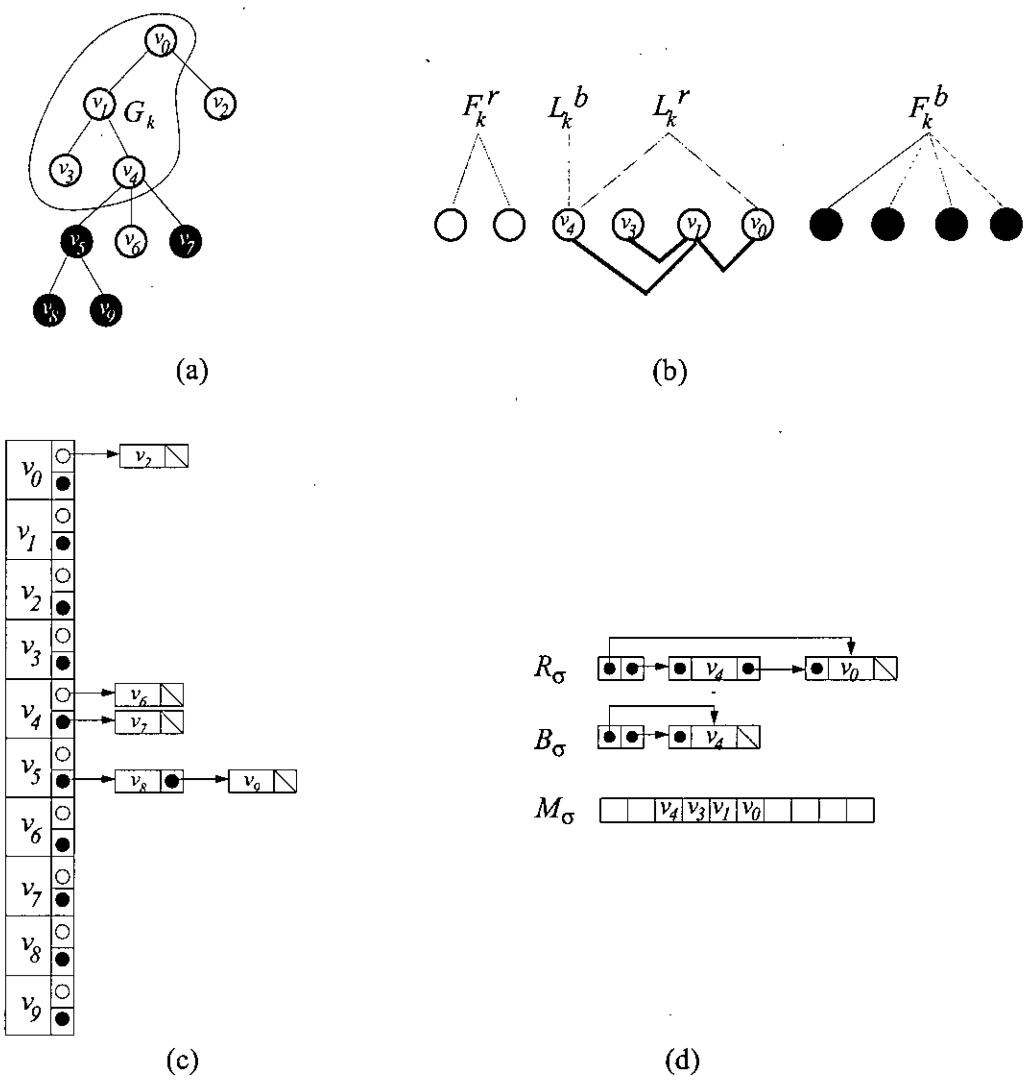


Figure 3.6 (a) A 2-colored tree G, (b) the drawing γ_k, (c) A_G after step k, and (d) states of the lists R_σ, B_σ and the array M_σ after step k.

if v_0 has at least on red child in A_G $\{v_0$ is an R-live vertex$\}$

then Add v_0 to the front of R_σ;

if v_0 has at least on blue child in A_G $\{v_0$ is a B-live vertex$\}$

then Add v_0 to the front of B_σ;

for k= 1 to $n-1$ do

  if R_σ is not empty **then**

{There is at least one R-live point in σ. This is case 1.1.}

begin

Let v be the vertex stored in the first element of R_σ;

{Hence the leftmost R-live point represents v.}

Let u be the first red child of v in A_G.

$M_\sigma[f_r] := u$;

$f_r := f_r - 1$;

Remove u from the list of red children of v in A_G.

if v had no red child left in A_G **then** remove v from R_σ.

if u has at least one red child in A_G **then** store u at the front of R_σ.

if u has at least one blue child in A_G **then** store u at the front of B_σ.

end

else if R_σ is empty but B_σ is not empty **then**

{There is no R-live point but at least one B-live point in σ. This is case 1.2.}

begin

Let v be the vertex stored in the last element of B_σ;

{Hence the rightmost B-live point represents v.}

Let u be the first blue child of v in A_G;

$M_\sigma[f_b] := u$;

$f_b := f_b + 1$;

Remove u from the list of blue children of v in A_G;

if v had no blue child left in A_G **then** remove v from B_σ;

if u has at least one red child in A_G **then** store u at the end of R_σ;

if u has at least one blue child in A_G **then** store u at the end of B_σ;

end

end.

3.1.2 Correctness and Time Complexity

In this section, we verify the correctness and time complexity of Algorithm **Consecutive-Embedding**. We first prove the following lemma on the correctness of the Algorithm **Consecutive-Embedding**.

Lemma 3.1.1 *Algorithm* **Consecutive-Embedding** *computes a bichromatic point-set embedding of a 2-colored tree G on a consecutive RB-sequence σ with at most one bend per edge.*

Proof. First of all we need to show that the Algorithm **Consecutive-Embedding** terminates. At step 0 we map the root v_0 of G on some free point p of σ where $c(p) = c(v_0)$. In subsequent steps, we map vertices of G that are not already mapped. Since there are finite number of vertices in G, the Algorithm must terminate after some finite steps. Now it remains to show that when the Algorithm **Consecutive-Embedding** terminates, the resultant drawing represents a bichromatic point-set embedding of G with at most one bend per edge. Let us assume the Algorithm **Consecutive-Embedding** terminates after some step k, $k \geq 0$. It follows that there are no live points in σ_k otherwise the Algorithm would have continued according to case 1 or case 2. Let γ_k be resulting drawing. Since γ_k satisfies the step invariant properties, it follows that γ_k represents a bichromatic point-set embedding of some graph G_k with at most one bend per edge where G_k is a connected subgraph of G. Therefore, we need to show that G_k is the graph G i.e. the set $V(G)\backslash V(G_k) = \phi$. In other words, we need to ensure that no vertex in G is left unmapped in γ_k. Now for contradiction assume $V(G)\backslash V(G_k) \neq \phi$. Let v be a vertex of $V(G)\backslash V(G_k)$. Since σ_k has no live points, there is no vertex $u \in V(G_k)$ such that v is a neighbor of u in G; otherwise u would have been a live vertex and the point representing u would be a live point. It follows that G has more than one component which is a contradiction since G is connected. Therefore, vertices such as v cannot exist and $V(G)\backslash V(G_k) = \phi$. Thus the Algorithm **Consecutive-Embedding** finds a bichromatic point-set embedding of G

on σ with at most one bend per edge. □

We now have the following lemma on the time complexity of Algorithm **Consecutive-Embedding**

Lemma 3.1.2 *Algorithm* **Consecutive-Embedding** *runs in linear time.*

Proof. In each step of the Algorithm **Consecutive-Embedding**, we embed a vertex of G on a point of σ. Hence **Consecutive-Embedding** requires $O(|V(G)|)$ steps to compute a bichromatic point-set embedding of G. From the formal description of Algorithm **Consecutive-Embedding** in Section 3.1.1, one can readily observe that operations in each of the steps take constant time. Thus Algorithm **Consecutive-Embedding** runs in linear time. □

3.2 Bichromatic Point-Set Embedding on Consecutive Point-Set

In this section, we prove the existence of bichromatic point-set embedding of trees on consecutive point-sets with at most one bend per edge. We in fact prove the following theorem.

Theorem 3.2.1 *Let $G = (V, E)$ be a 2-colored tree. Let S be a 2-colored consecutive point-set compatible with G. G has a bichromatic point-set embedding on S with at most one bend per edge. Moreover such a drawing can be computed in linear time.*

Proof. The proof is constructive. Let σ be any RB-sequence chromatic equivalent to S. It follows that σ is also consecutive and compatible with G. Using the Algorithm **Consecutive-Embedding** we construct a bichromatic point-set embedding of G on σ with at most one bend per edge; by Lemma 3.1.2, this takes linear time. Then using the technique used in the proof of Lemma 2.2.3, we compute a bichromatic point-set embedding of G on S with at most one bend per edge from bichromatic point-set embedding of G on σ and this also takes linear amount of time. Thus it requires linear time to construct a bichromatic point-set embedding of G on S. □

3.3 Summary

In this chapter, we have proved the existence of bichromatic point-set embeddings of trees on consecutive point-sets with at most one bend per edge. We have described a linear-time algorithm that finds a bichromatic point-set embedding of a 2-colored tree on a consecutive RB-sequence with at most one bend per edge. Then using such drawing we have shown how to construct a bichromatic point-set embedding of the given tree on any consecutive point-set with at most one bend per edge in linear time.

Chapter 4

Embedding on Alternating Point-Set

In this chapter, we prove that trees admit bichromatic point-set embeddings on alternating point-sets with at most one bend per edge and such an embedding can be found in linear time. We first present a linear-time algorithm that computes a bichromatic point-set of any given tree G on an arbitrary alternating RB-sequence σ with at most one bend per edge in Section 4.1. Then in Section 4.2 we obtain bichromatic point-set embedding of G on any alternating point-set S with at most one bend per edge from the bichromatic point-set embedding of G on σ. Finally, Section 4.3 summarizes this chapter.

4.1 Bichromatic Point-Set embedding on Alternating RB-sequence

In this section, we describe a linear-time algorithm that computes a bichromatic point-set embedding of a 2-colored tree G on an alternating RB-sequence with at most one bend per edge. We call this algorithm **Alternating-Embedding**. Algorithm **Alternating-Embedding** uses a procedure that we call **Tree-Embed**. Therefore, we first illustrate Procedure **Tree-Embed** in Section 4.1.1. Then we present Algorithm **Alternating-Embedding** in Section 4.1.2. Finally, in Section 4.1.3 we verify correctness and time complexity of the Algorithm.

4.1.1 Procedure Tree-Embed

Procedure **Tree-Embed** has two inputs: (i) a 2-colored tree G with a designated root v_0, and (ii) an integer value identified as *level*. The output of **Tree-Embed** is a drawing γ that satisfies the following conditions: (i) γ represents a bichromatic point-set embedding of either G or some connected subgraph G_s of G on an arbitrary alternating point-set σ, and (ii) v_0 is mapped to the leftmost point of σ. Whether the output is a drawing of G or G_s is determined by the configuration of the input graph G as well as the value of *level*.

This procedure works in a step by step fashion. At the end of each step a connected subgraph of G is embedded on an arbitrary RB-sequence such that the resulting RB-sequence is alternating. We use G_k to denote the subgraph of G that is embedded after step k and σ_k to denote the RB-sequence representing vertices in G_k, for $k \geq 0$. Let γ_k denotes the drawing after step k. Figure 4.1(b) shows drawing γ_5 for the input graph in Figure 4.1(a).

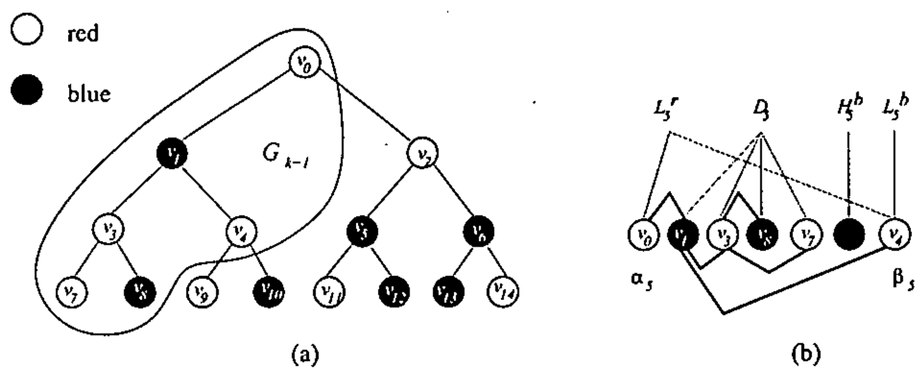


Figure 4.1 (a) A 2-colored tree G, and (b) the drawing γ_5 after step 5 and the sets L_5^r, L_5^b, H_5^b and D_5.

We use the following notations to illustrate the operations inside the procedure. We say a vertex v in $V(G)\backslash V(G_k)$ is an *unmapped* vertex and vertices in G_k are *mapped* vertices.

A mapped vertex v of G_k is a *live* vertex if it has at least one unmapped neighbor; else v is called a *dead* vertex. We use $U_k(v)$ to denote the set of unmapped neighbors of any live vertex v; v is an *R-live* vertex if there is at least one vertex u such that $u \in U_k(v)$ and $c(u)$ is red; v is called a *B-live* vertex if there is at least one such vertex u that $u \in U_k(v)$ and $c(u)$ is blue.

For a vertex v of G_k, we use $p(v)$ to denote the point of σ_k that represents v. The leftmost and rightmost points of σ_k will be denoted by α_k and β_k respectively. A point of σ_k will be called a *R-live* point, *B-live* point or *dead* point if it represents a R-live vertex, a B-live vertex or a dead vertex, respectively in γ_k. The set of R-live, B-live and dead points in σ_k will be denoted by L_k^r, L_k^b and D_k respectively. Any point of σ_k that does not represent a vertex is called a *hole*; a hole can be either red or blue. The set of blue holes and red holes in σ_k will be denoted by H_k^b and H_k^r respectively. For example, in Figure 4.1(b), point $p(v_0)$ is a R-live point since vertex v_0 has an unmapped neighbor v_2 where $c(v_2)$ is red; point $p(v_4)$ is both an R-live point and a B-live point as one of its unmapped neighbors v_9 is red and another unmapped neighbor v_{10} is blue; points $p(v_1)$, $p(v_3)$, $p(v_7)$ and $p(v_8)$ are dead points.

We define the following seven types for σ_k.

Type I: We say σ_k is of Type I when the following conditions are hold: (i) the rightmost point of σ_k is red, i.e. $c(\beta_k)$ is red; (ii) σ_k has at least one B-live point, i.e. $L_k^b \neq \phi$; (iii) σ_k has no red hole or blue hole, i.e. $H_k^r = \phi$ and $H_k^b = \phi$.

Type II: We say σ_k is of Type II when the following conditions are hold: (i) the rightmost point of σ_k is red, i.e. $c(\beta_k)$ is red; (ii) σ_k has at least one B-live point, i.e. $L_k^b \neq \phi$; (iii) σ_k has no red hole, i.e. $H_k^r = \phi$; (iv) σ_k has at least one blue hole, i.e. $H_k^b \neq \phi$.

Type III: We say σ_k is of Type III when the following conditions are hold: (i) the rightmost point of σ_k is red, i.e. $c(\beta_k)$ is red; (ii) σ_k has no B-live point, i.e. $L_k^b = \phi$; (iii) σ_k has at least one R-live point, i.e. $L_k^r \neq \phi$; (iv) σ_k has no red hole, i.e. $H_k^r = \phi$.

Type IV: We say σ_k is of Type IV when the following conditions are hold: (i) the rightmost point of σ_k is red, i.e. $c(\beta_k)$ is red; (ii) σ_k has no B-live point, i.e. $L_k^b = \phi$; (iii) σ_k has no R-live point, i.e. $L_k^r = \phi$; (iv) σ_k has no red hole, i.e. $H_k^r = \phi$.

Type V: We say σ_k is of Type V when the following conditions are hold: (i) the rightmost point of σ_k is blue, i.e. $c(\beta_k)$ is blue; (ii) σ_k has at least one R-live point, i.e. $L_k^r \neq \phi$; (iii) σ_k has no red hole or blue hole, i.e. $H_k^r = \phi$ and $H_k^b = \phi$.

Type VI: We say σ_k is of Type VI when the following conditions are hold: (i) the rightmost point of σ_k is blue, i.e. $c(\beta_k)$ is blue; (ii) σ_k has no R-live point, i.e. $L_k^r = \phi$; (iii) σ_k has at least one B-live point, i.e. $L_k^b \neq \phi$; (iv) σ_k has no red hole or blue hole, i.e. $H_k^r = \phi$ and $H_k^b = \phi$.

Type VII: We say σ_k is of Type VII when the following conditions are hold: (i) the rightmost point of σ_k is blue, i.e. $c(\beta_k)$ is blue; (ii) σ_k has no B-live point, i.e. $L_k^b = \phi$; (iii) σ_k has no R-live point, i.e. $L_k^r = \phi$; (iv) σ_k has no red hole or blue hole, i.e. $H_k^r = \phi$ and $H_k^b = \phi$.

We define *horizontal flip* of γ_k as rotating γ_k by an angle 180 degree with respect to any line perpendicular to the spine of σ_k. Figure 4.2(c) illustrates horizontal flip of the drawing γ_k in Figure 4.2(b). We have the following observation regarding horizontal flip operation.

Observation 4.1.1 *Let γ_k be a drawing that represents a bichromatic point-set embedding of some graph G_k on an alternating point-set σ_k. Let γ_k^h be the drawing obtained by horizontal flip of γ_k. Then γ_k^h also represents a bichromatic point-set embedding of G_k on σ_k. Furthermore for any vertex v of G_k, if $p(v)$ is accessible from some page π in γ_k, $p(v)$ is also accessible from π in γ_k^h.*

Proof. Let v be any vertex in $V(G_k)$ and mapped to some point $p \in \sigma_k$ where $c(v) = c(p)$. Since horizontal flip operation does not change the mapping of any vertex, it follows that v is mapped to p also in γ_k^h. Also horizontal flip operation does not add or delete any edge of γ_k. Thus γ_k^h represents a bichromatic point-set embedding of G_k.

Moreover, if an edge e of G_k is contained in top (bottom) page in γ_k, e also remains in the top (bottom) page in γ_k^h. Thus horizontal flip operation does not change accessibility of any point. □

Next we define *vertical flip* of γ_k as rotating γ_k by an angle 180 degree with respect to the spine of σ_k. Figure 4.2(d) illustrates vertical flip of the drawing γ_k in Figure 4.2(b). We have the following observations regarding vertical flip operation.

Observation 4.1.2 *Let γ_k be a drawing that represents a bichromatic point-set embedding of some graph G_k on an alternating point-set σ_k. Let γ_k^v be the drawing obtained by vertical flip of γ_k. Then γ_k^v also represents a bichromatic point-set embedding of G_k on σ_k. Furthermore for any vertex v of G_k, if $p(v)$ is not accessible from top page in γ_k, $p(v)$ is not accessible from bottom page in γ_k^v and vice versa.*

Proof. Let v be a vertex in $V(G_k)$ and mapped to some point $p \in \sigma_k$ where $c(v) = c(p)$. Since vertical flip operation does not change the mapping of any vertex, it follows that v is mapped to p also in γ_k^v. Thus γ_k^v represents a bichromatic point-set embedding of G_k. Also vertical flip operation does not add or delete any edge of γ_k. Now for each edge e of G_k, if e is drawn in top (bottom) page in γ_k, then e must be in the bottom (top) page in γ_k^v. Thus any point p of σ_k that is not accessible from top (bottom) page in γ_k becomes inaccessible from bottom (top) page in γ_k^h. □

We define *inversion* of any 2-colored graph G as changing the color of each of the vertex in G such that each blue vertex of G becomes a red vertex and each red vertex becomes a blue vertex. Similarly inversion of any 2-colored point-set σ is defined as changing color of each blue point to red and each red point to blue. One can observe that the point-set obtained after inverting an alternating RB-sequence is also an alternating RB-sequence. Figure 4.3 illustrates the inversion operation. We have the following observation regarding inversion operation.

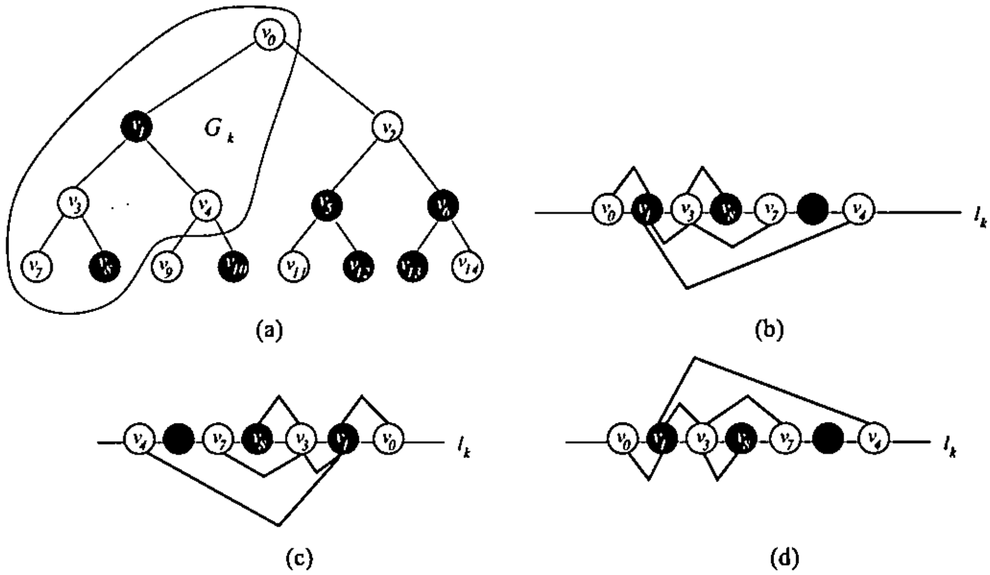


Figure 4.2 (a) A 2-colored tree G, (b) the drawing γ_k (after some step k, $k \geq 0$), (c) γ_k after horizontal flip, and (d) γ_k after vertical flip.

Observation 4.1.3 *Let γ_k be a drawing that represents a bichromatic point-set embedding of some graph G_k on an alternating point-set σ_k. Let G_k^t be the graph obtained after inversion of G_k and σ_k^t be the point-set obtained after inversion of σ_k. Then γ_k represents a bichromatic point-set embedding of G_k^t on alternating RB-sequence σ_k^t.*

Proof. Let v be a vertex in $V(G_k)$ and mapped to some point $p \in \sigma_k$. It follows that $c(v) = c(p)$. Let $c(v)$ is red in G_k. Hence $c(v)$ is blue in G_k^t. Similarly $c(p)$ is blue in $p \in \sigma_k^t$. Also inversion operation does not add or delete any edge of G_k. Therefore, γ_k represents a bichromatic point-set embedding of G_k^t. □

We are now ready to define the invariants that are maintained at the end of each step of the drawing algorithm. The invariants are as follows.

Property 1: If $L_k^r \neq \phi$, all points in L_k^r are accessible from the bottom page.

Property 2: If $L_k^b \neq \phi$, all points in L_k^b are accessible from the top page.

Property 3: If $H_k^b \neq \phi$, there is no B-live point to the left of any blue hole and all

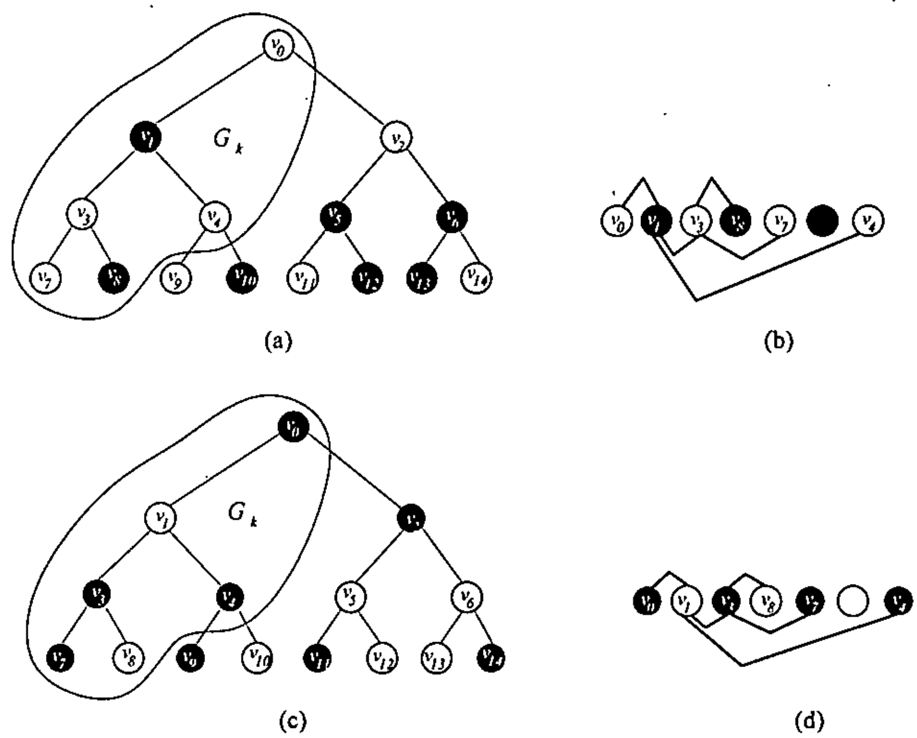


Figure 4.3 (a) A 2-colored tree G, (b) the drawing γ_k, after some step $k > 0$, (c) G after inversion, and (d) γ_k after inversion.

points in H_k^b are accessible from the top page.

Property 4: σ_k is an alternating RB-sequence and type of σ_k is either I, II, III, IV, V, VI or VII.

Property 5: G_k is a connected subgraph of G such that G_k contains the root v_0 of G and γ_k is a bichromatic point-set embedding of G_k on σ_k with at most one bend per edge. Moreover, v_0 is represented by the leftmost point of σ_k.

We now specify the operations as performed inside Procedure **Tree-Embed**.

At step $k = 0$, we embed the root vertex v_0 of G. We take any point p_0 on the plane such that $c(p_0) = c(v_0)$. We assume that the point p_0 is on x-axis. We map v_0 on p_0. Figure 4.4 illustrates the operation in step 0. We now show that the invariants are maintained after this step. Since no edge is added, properties 1, 2 and 3 are hold trivially.

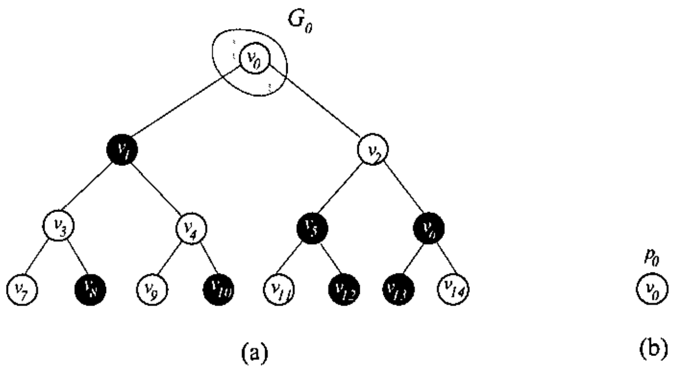


Figure 4.4 An illustration for step 0 of Procedure **Tree-Embed**. (a) A 2-colored tree G, and (b) the drawing γ_0.

σ_0 contains the single point p_0 and hence is an alternating RB-sequence of unit length; also $\alpha_0 = \beta_0 = p_0$. We now determine type of σ_0. From the operation specified, $H_0^r = \phi$ and $H_0^b = \phi$. There can be the following cases.

(i) $c(p_0)$ is red and p_0 is a B-live point: In this case, σ_0 is of Type I since $c(\beta_0)$ is red, $L_0^b = \{p_0\} \neq \phi$, $H_0^r = \phi$, and $H_0^b = \phi$.

(ii) $c(p_0)$ is red, p_0 is not a B-live point and p_0 is an R-live point: In this case, σ_0 is of Type III since $c(\beta_0)$ is red, $L_0^r = \{p_0\} \neq \phi$, $L_0^b = \phi$, and $H_0^r = \phi$.

(iii) $c(p_0)$ is red, p_0 is a dead point: In this case, σ_0 is of Type IV since $c(\beta_0)$ is red, $L_0^r = \phi$, $L_0^b = \phi$, and $H_0^r = \phi$.

(iv) $c(p_0)$ is blue and p_0 is an R-live point: In this case, σ_0 is of Type V since $c(\beta_0)$ is blue, $L_0^r = \{p_0\} \neq \phi$, $L_0^b = \phi$, $H_0^r = \phi$, and $H_0^b = \phi$.

(v) $c(p_0)$ is blue, p_0 is not an R-live point and p_0 is a B-live point: In this case, σ_0 is of Type VI since $c(\beta_0)$ is blue, $L_0^r = \phi$, $L_0^b = \{p_0\} \neq \phi$, $H_0^r = \phi$, and $H_0^b = \phi$.

(vi) $c(p_0)$ is blue, p_0 is a dead point: In this case, σ_0 is of Type VII since $c(\beta_0)$ is red, $L_0^r = \phi$, $L_0^b = \phi$, $H_0^r = \phi$, and $H_0^b = \phi$.

For example, in Figure 4.4, $c(p_0)$ is red and p_0 is a B-live vertex since v_0 has an unmapped neighbor v_1 where v_1 is blue; therefore, σ_0 is of Type I. Thus for all possible input combinations, σ_0 is within the defined types of I-VII. Therefore, Property 4 holds for γ_0.

Since G_0 contains only the vertex v_0 and p_0 is the only point of σ_0, if follows that γ_0 satisfies Property 5.

We now specify the operations at a subsequent step k, $k > 0$. We use induction to prove that the resulting drawing γ_k maintains the invariant properties. We consider γ_0 as the base case and as induction hypothesis we assume that the output of the previous step γ_{k-1} satisfies the specified invariant properties. At any step k ($k > 0$), We identify the following seven cases.

Case 1: σ_{k-1} is of Type I, i.e. $c(\beta_{k-1})$ is red, $L^b_{k-1} \neq \phi$, $H^r_{k-1} = \phi$ and $H^b_{k-1} = \phi$.

We add a point p_b on the spine of σ_{k-1} such that p_b is to the right of β_{k-1} and $c(p_b)$ is blue. Let l_b be the rightmost point in L^b_{k-1} and v be the vertex of G mapped on l_b. Hence v is a B-live vertex and there is a vertex $u \in U_{k-1}(v)$ such that $c(u)$ is blue. We map u on p_b and draw the edge (v, u) connecting points l_b and p_b through the top page. Figure 4.5 illustrates this case. We now prove that γ_k satisfies the invariants.

The drawing γ_k satisfies Property 1: If σ_k contains no R-live points, i.e. $L^r_k = \phi$ then Property 1 holds trivially. Otherwise consider a point $p_l \in L^r_k$ such that p_l is not accessible from bottom page in γ_k. Let v_l be the vertex of G represented by p_l. Hence v_l is an R-live vertex in G_k. Since p_b is the rightmost point in σ_k, p_b is accessible from both the pages in γ_k. It follows that $p_l \neq p_b$. Since u is mapped on p_b, $v_l \neq u$. Then v_l must be an R-live vertex of G_{k-1}. It follows that p_l is an R-live point of σ_{k-1}. Therefore, by Property 1 of induction hypothesis, p_l is accessible from bottom page in γ_{k-1}. Consequently it must be the addition of the edge (u, v) that makes p_l inaccessible from bottom page in γ_k. But since we draw the edge (u, v) through the top page, it cannot make p_l inaccessible from bottom page. It follows that p_l is accessible from bottom page in γ_k which is a contradiction. Therefore, all points in L^r_k are accessible from the bottom page.

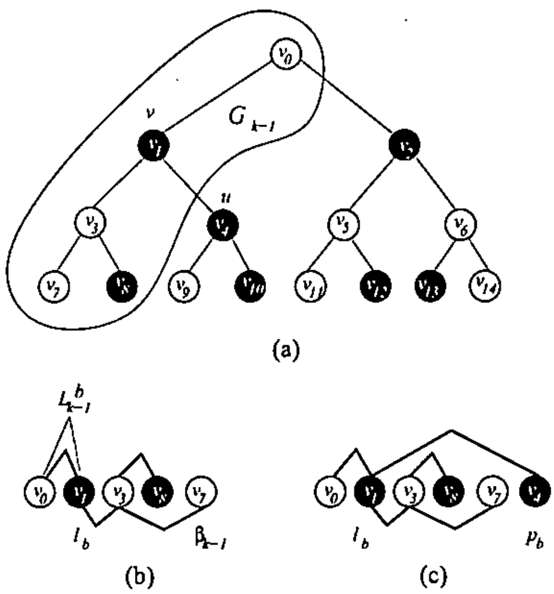


Figure 4.5 An illustration for case 1 of Procedure **Tree-Embed**. (a) A 2-colored tree G, (b) the drawing γ_{k-1}, and (c) the drawing γ_k.

The drawing γ_k *satisfies Property* 2: If there are no B-live points in σ_k, i.e. $L_k^b = \phi$ then Property 2 holds trivially. Otherwise consider a point $p_l \in L_k^b$ such that p_l is not accessible from the top page in γ_k. Let v_l be the vertex of G represented by p_l. Hence v_l is a B-live vertex in G_k. Since p_b is the rightmost point in σ_k, p_b is accessible from both the pages in γ_k. It follows that $p_l \neq p_b$. Since u is mapped on p_b, $v_l \neq u$. Then v_l must be a B-live vertex of G_{k-1}. It follows that p_l is a B-live point of σ_{k-1}. Therefore, by Property 2 of induction hypothesis, p_l is accessible from top page in γ_{k-1}. Consequently it must be the addition of the edge (u, v) that makes p_l inaccessible from top page in γ_k. The endpoints of the edge (u, v) are p_b and l_b. Since p_b is the rightmost point of σ_k, p_b is to the right of p_l. Also l_b is to the right of p_l since both l_b and p_l are in L_{k-1}^b and l_b is the rightmost point of L_{k-1}^b. Since both the endpoints of edge (u, v) lie to the right of p_l in γ_k, it follows that the edge (u, v) cannot make p_l inaccessible from top page. Therefore, p_l is accessible from top page in γ_k which is a contradiction. Hence all points in L_k^b are accessible from the top page.

The drawing γ_k satisfies Property 3: Since σ_{k-1} contains no blue holes and no new blue hole is created by the operation defined for this step, it follows that $H_k^b = \phi$. Therefore, Property 3 is maintained trivially.

The drawing γ_k satisfies Property 4: By induction hypothesis σ_{k-1} is an alternating RB-sequence where $c(\beta_{k-1})$ is red. Since the point p_b is to the right of β_{k-1} and $c(p_b)$ is blue, it follows that $\sigma_k = \sigma_{k-1} \bigcup \{p_b\}$ is also an alternating RB-sequence where $\beta_k = p_b$. We now determine the type of σ_k. From the operations specified, $H_k^r = H_{k-1}^r = \phi$ and $H_k^b = H_{k-1}^b = \phi$. Now there may be the following cases.

(i) $L_k^r \neq \phi$: In this case, σ_k is of Type V since $c(\beta_k)$ is blue, $L_k^r \neq \phi$, $H_k^r = \phi$ and $H_k^b = \phi$.

(ii) $L_k^r = \phi$ and $L_k^b \neq \phi$: In this case, σ_k is of Type VI since $c(\beta_k)$ is blue, $L_k^r = \phi$, $L_k^b \neq \phi$, $H_k^r = \phi$ and $H_k^b = \phi$.

(iii) $L_k^r = \phi$ and $L_k^b = \phi$: In this case, σ_k is of Type VII since $c(\beta_k)$ is blue, $L_k^r = \phi$, $L_k^b = \phi$, $H_k^r = \phi$ and $H_k^b = \phi$.

Therefore, for all possible input combinations, Type of σ_k is either of V, VI and VII. Hence Property 4 holds for γ_k.

The drawing γ_k satisfies Property 5: According to the operation specified, $V(G_k) = V(G_{k-1}) \bigcup \{u\}$. Since G_{k-1} is a connected graph that contains the vertex v_0(by induction hypothesis) and u is a neighbor of some vertex $v \in V(G_{k-1})$, if follows that the graph G_k is also connected and v_0 is in G_k. Therefore, it remains to show that the edge (u, v) does not create any edge crossing and contains at most one bend. Since $l_b \in L_{k-1}^b$, by Property 2 it is accessible from the top page in γ_{k-1}. The point p_b is taken such that p_b is to the right of the rightmost point of σ_{k-1}; hence p_b is accessible from both (top and bottom) pages. Therefore, according to Observation 2.2.1, l_b and p_b can be connected with such a polygonal chain through the top page that contains at most one bend and does not cross any other edge in γ_{k-1}. Hence γ_k represents a bichromatic point-set embedding of G_k on σ_k.

Case 2: σ_{k-1} is of Type II, i.e. $c(\beta_{k-1})$ is red, $L^b_{k-1} \neq \phi$, $H^b_{k-1} \neq \phi$ and $H^r_{k-1} = \phi$.

Let l_b be the leftmost point in L^b_{k-1} and v be the vertex represented by l_b. Consider any vertex u such that $u \in U_{k-1}(v)$ and $c(u)$ is blue. Let G_v and G_u be the components of G obtained by deleting the edge (u, v) from G where G_u contains v and G_u contains u. One can observe that $V(G_{k-1}) \subseteq V(G_v)$ and all the vertices in G_u are unmapped vertices. Moreover, G_u is also a 2-colored tree. We designate u as the root of G_u and invoke Procedure **Tree-Embed**$(G_u, u, 1)$ (note that this is a recursive call) with graph G_u and level value of 1 as input. Let γ_u be the returned drawing. We use σ_u to denote the alternating RB-sequence associated with γ_u. Let L^r_u, L^b_u, H^r_u and H^b_u denotes the set of R-live points, B-live points, red holes and blue holes respectively in σ_u. Now we identify three sub cases.

Case 2.1: The rightmost point in σ_u is red and there are no live points in σ_u.

Note that this case arises when **Tree-Embed**$(G_u,\ u,\ 1)$ terminates in the way as specified in case 4.2. It follows that γ_u satisfies the invariant properties 1-5 and thus represents a bichromatic point-set embedding of G_u on σ_u (by Property 5). Moreover, σ_u is of Type IV. Since u is designated as the root of G_u, it follows that u is mapped to the leftmost point of σ_u (by Property 5). Let h_b be the rightmost point in H^b_{k-1}. We now perform the following operations. We first flip γ_u horizontally. As a result the rightmost point in σ_u now represents the vertex u; let β_u denotes this point. We insert the drawing γ_u between the points h_b and $next(h_b)$. Finally we add the edge (u, v) connecting the points l_b and β_u through the top page. Figure 4.6 illustrates this case. From the operations specified, if follows that $L^r_k = L^r_{k-1} \bigcup L^r_u$, $L^b_k = L^b_{k-1} \bigcup L^b_u$, $H^b_k = H^b_{k-1} \bigcup H^b_u$ and $\sigma_k = \sigma_{k-1} \bigcup \sigma_u$. We now show that the drawing after this step satisfies the invariants.

The drawing γ_k satisfies Property 1: Since σ_u contains no R-live points, it follows that $L^r_k = L^r_{k-1}$. If there are no R-live points in σ_k, i.e. $L^r_k = \phi$ then Property 1 holds trivially. Otherwise consider a point $p_l \in L^r_k$ such that p_l is not accessible from the bottom page in γ_k. Since p_l is also in L^r_{k-1}, by induction hypothesis, p_l is accessible from bottom

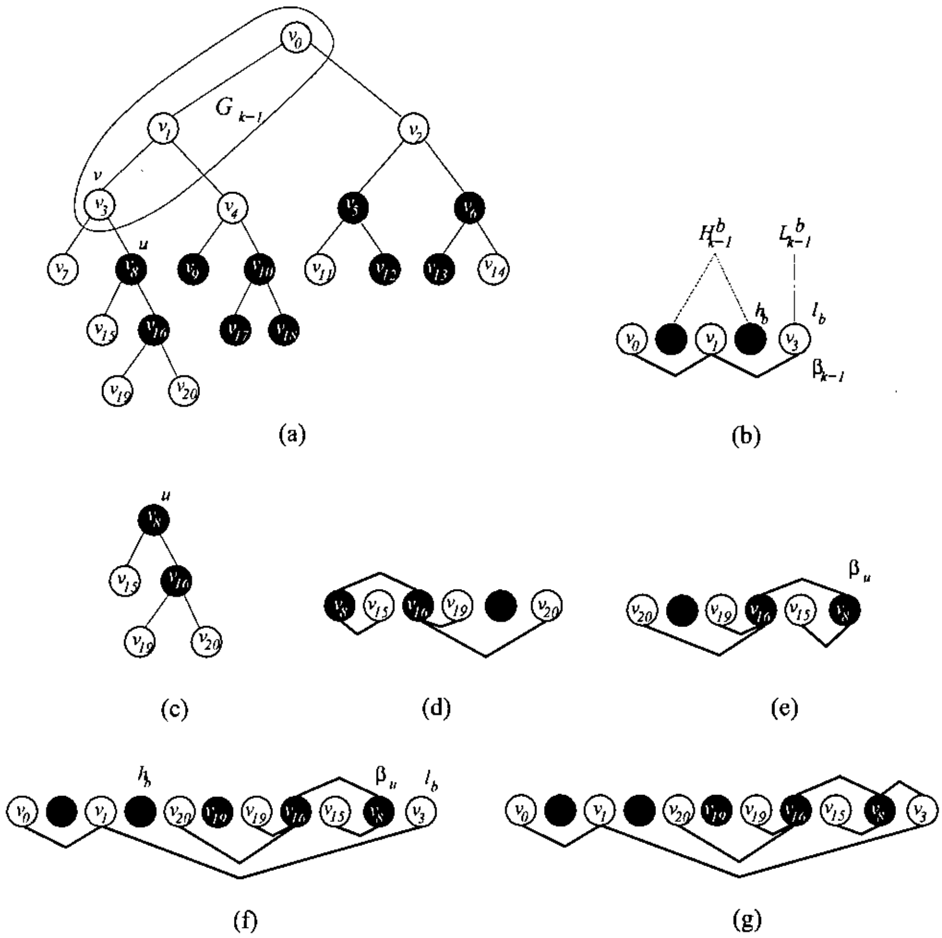


Figure 4.6 An illustration for case 2.1 of Procedure **Tree-Embed**. (a) A 2-colored tree G,(b) the drawing γ_{k-1}, (c) subgraph G_u of G, (d) the drawing γ_u, (e) the drawing after horizontal flip of γ_u, (f) insertion of γ_u, and (g) the drawing γ_k.

page in γ_{k-1} (Property 1). Therefore, it must be some edge e added to γ_{k-1} that makes p_l inaccessible from bottom page where p_l lies between the endpoints of e. From the drawing operation specified, e is either an edge of γ_u or the edge (u, v). Since p_l in not in σ_u, p_l cannot lie between the endpoints of any edge in γ_u. Hence e is not an edge of γ_u. Moreover, e cannot be the edge (u, v) that connects the points l_b and β_u since the edge (u, v) is drawn through the top page and therefore cannot make p_l inaccessible from bottom page in γ_k. Hence no edge such as e exists and p_l is accessible from bottom page in γ_k. Therefore, γ_k satisfies Property 1.

The drawing γ_k satisfies Property 2: Since σ_u contains no B-live points, it follows that $L_b^k = L_{k-1}^b$. If there are no B-live points in σ_k, i.e. $L_k^b = \phi$ then Property 2 holds trivially. Otherwise consider a point $p_l \in L_k^b$ such that p_l is not accessible from the top page in γ_k. Since p_l is also in L_{k-1}^b, by induction hypothesis, p_l is accessible from top page in γ_{k-1}. Therefore, it must be some edge e added to γ_{k-1} that makes p_l inaccessible from top page where p_l lies between the endpoints of e. From the drawing operation specified, e is either an edge of γ_u or the edge (u, v). Since p_l in not in σ_u, p_l cannot lie between the endpoints of any edge in γ_u. Hence e is not an edge of γ_u. Then e must be the edge (u, v) that connects the points l_b and β_u. Since both l_b and p_l are in L_{k-1}^b and l_b is the leftmost of point of L_{k-1}^b, it follows that l_b is to the left of p_l. Since we insert γ_u between the points h_b and $next(h_b)$ in σ_{k-1} and β_u is a point of σ_u, hence any point to the right of h_b in σ_{k-1} is also to the right of β_u in σ_k. It follows that p_l is to the right of β_u in σ_k since p_l is right to h_b in σ_{k-1} (by Property 3). Since both l_b and β_u are to the left of p_l, the edge (u, v) cannot cause p_l to be inaccessible from top page. Hence e cannot be the edge (u, v). Consequently no edge such as e exists and p_l is also accessible from top page in γ_k. It follows that all points in L_k^b are accessible from top page.

The drawing γ_k satisfies Property 3: Let H_u^b denotes the set of blue holes, if exists in σ_u. It follows that $H_k^b = H_{k-1}^b \bigcup H_u^b$. If σ_k contains no blue holes, i.e. $H_k^b = \phi$ then Property 3 holds trivially. Otherwise, we first show that there is no B-live point to the left of any blue hole in σ_k. Consider any blue hole $p_h \in H_k^b$. p_h is either in H_{k-1}^b or in H_u^b. We first examine the case when $p_h \in H_{k-1}^b$. By Property 3 of induction hypothesis, no point of L_{k-1}^b is to left of p_h in σ_{k-1}. Since $L_k^b = L_{k-1}^b$, it follows that there is no B-live point to the left of p_h in σ_k. Next we examine the case when $p_h \in H_u^b$. Assume there is a point $p_l \in L_k^b$ to left of p_h in σ_k. Since γ_u contains no B-live points, it follows that $p_l \in L_{k-1}^b$. Since we insert γ_u between points h_b and $next(h_b)$ in σ_{k-1} and p_h is in σ_u, it follows that p_l must also be to left of h_b in σ_{k-1}. It contradicts the induction hypothesis that there is no B-live point to left of $h_b \in H^b_{k-1}$ in σ_{k-1}. Therefore, no point such as p_l exists and hence all B-live points are to the right of any blue hole of σ_k.

We next prove that any blue hole of σ_k is accessible from top page in γ_k. Consider a point $p_h \in H^b_k$ such that p_h is not accessible from the top page in γ_k. p_h is either in H^b_{k-1} or in H^b_u. Consider the case when $p_h \in H^b_{k-1}$. By induction hypothesis, p_h is accessible from top page in γ_{k-1}. Therefore, it must be some edge e added to γ_{k-1} that makes p_h inaccessible from top page where p_h lies between the endpoints of e. From the drawing operation specified, e is either an edge of γ_u or the edge (u, v). Since p_h in not in σ_u, p_h cannot lie between the endpoints of any edge in γ_u. Hence e is not an edge of γ_u. Then e must be the edge (u, v) that connects the points l_b and β_u. Since l_b is in L^b_{k-1}, by Property 3 of induction hypothesis l_b is to the right of p_h in σ_{k-1}. The point h_b is the rightmost blue hole in σ_{k-1}; therefore, h_b is to right of p_h. Since we insert γ_u between the points h_b and $next(h_b)$ in σ_{k-1} and β_u is a point of σ_u, hence any point to the left of h_b in σ_{k-1} is also to the left of β_u in σ_k. It follows that p_l is to the left of β_u in σ_k. Now since both l_b and β_u are to the right of p_h, the edge (u, v) cannot cause p_h to be inaccessible from top page in γ_k. Hence e cannot be the edge (u, v). Consequently no edge such as e exists and p_h is also accessible from top page in γ_k.

Now consider the case when $p_h \in H^b_u$. Since drawing γ_u satisfies the invariants, it follows that p_h is accessible from top page in γ_u. Moreover, the horizontal flip operation does not change the top page accessibility of p_h in γ_u (according to Observation 4.1.1). Since we insert γ_u between the points h_b and $next(h_b)$ in σ_{k-1}, therefore, to make p_h inaccessible from the top page in γ_k, there must be such an edge e in γ_{k-1} that one endpoint of e is to the left of h_b and another is to the right of h_b. But such an edge makes the point $h_b \in H^b_{k-1}$ inaccessible from top page in γ_{k-1}. This contradicts the induction hypothesis that h_b is accessible from top page in γ_{k-1}(Property 3). Therefore, no edge such as e exists and p_h is accessible from top page in γ_k. Therefore, all the blue holes in γ_k are accessible from the top page.

The drawing γ_k satisfies Property 4: From the operation specified, it follows that σ_u is an alternating RB-sequence where the leftmost point is blue and the rightmost point is red. Hence after the horizontal flip operation of γ_u, the leftmost point of σ_u is red and the rightmost point of σ_u is blue. Since we insert the drawing γ_u between the points h_b and $next(h_b)$ in σ_{k-1} where $c(h_b)$ is blue and $c(next(h_b))$ is red, it follows that σ_k is also an alternating RB-sequence and $\beta_k = \beta_{k-1}$. We now identify type of σ_k. Since σ_u is of Type IV, it follows that $L_u^r = \phi$, $L_u^b = \phi$, and $H_u^r = \phi$. From the operations specified, $H_k^r = H_{k-1}^r \bigcup H_u^r = \phi$. There may be the following cases.

(i) $L_k^b \neq \phi$ and $H_k^b = \phi$: In this case, σ_k is of Type I since $c(\beta_0)$ is red, $L_k^b \neq \phi$, $H_k^r = \phi$ and $H_k^b = \phi$.

(ii) $L_k^b \neq \phi$ and $H_k^b \neq \phi$: In this case, σ_k is of Type II since $c(\beta_k)$ is red, $L_k^b \neq \phi$, $H_k^r = \phi$ and $H_k^b \neq \phi$.

(iii) $L_k^b = \phi$ and $L_k^r \neq \phi$: In this case, σ_k is of Type III since $c(\beta_k)$ is red, $L_k^r \neq \phi$, $L_k^b = \phi$ and $H_k^r = \phi$.

(iv) $L_k^b = \phi$ and $L_k^r = \phi$: In this case, σ_k is of Type IV since $c(\beta_k)$ is red, $L_k^r = \phi$, $L_k^b = \phi$, and $H_k^r = \phi$.

Therefore, Property 4 holds for γ_k.

The drawing γ_k satisfies Property 5: From the operations specified it follows that $V(G_k) = V(G_{k-1}) \bigcup V(G_u)$. By induction hypothesis, G_{k-1} is connected. Moreover, G_u is also a connected subgraph of G. Since $v \in V(G_{k-1})$, $u \in V(G_u)$ and the edge (u, v) is in γ_k, it follows that G_k is also connected. Now it remains to show that the edge (u, v) does not create any edge crossings and contains at most one bend. Since $l_b \in L_{k-1}^b$, by induction hypothesis, l_b is accessible from the top page in γ_{k-1}. β_u is the rightmost point of σ_u and thus accessible from both the pages in γ_u. Therefore, to make β_u inaccessible from top page after insertion of γ_u between h_b and $next(h_b)$, there must be an edge e in γ_{k-1} such that one endpoint of e is to the left of h_b and the other is to the right of h_b. But such an edge also makes h_b inaccessible from top page in γ_{k-1} and thus contradicts the induction hypothesis that γ_{k-1} maintains Property 3. Hence both l_b and β_u are accessible from the top page and therefore can be connected with such a polygonal chain through the top page that does not cross any other edge and may contain at most one bend (according to Observation 2.2.1). Therefore, Property 5 holds for γ_k.

Case 2.2: The rightmost point in σ_u is blue and there are no live points and holes in σ_u.

Note that this case arises when **Tree-Embed**(G_u, u, 1) terminates in the way as specified in case 7.2. It follows that γ_u satisfies the invariant properties 1-5 and thus represents a bichromatic point-set embedding of G_u on σ_u (by Property 5). Moreover, σ_u is of Type VII. Since u is designated as the root of G_u, it follows that u is mapped to the leftmost point of σ_u (by Property 5). Let h_b be the rightmost point in H^b_{k-1}. We now perform the following operations. We first flip γ_u horizontally. As a result the rightmost point in σ_u now represents the vertex u; let β_u denotes this point. Next we insert the drawing γ_u between the points h_b and $next(h_b)$ in γ_{k-1} and then remove the point h_b. Finally we add the edge (u, v) connecting the points l_b and β_u through the top page. Figure 4.7 illustrates this case.

From the operations specified, if follows that $L^r_k = L^r_{k-1} \bigcup L^r_u$, $L^b_k = L^b_{k-1} \bigcup L^b_u$, $H^b_k = H^b_{k-1} \bigcup H^b_u \backslash \{h_b\}$ and $\sigma_k = \sigma_{k-1} \bigcup \sigma_u$. We now show that the drawing after this step satisfies the invariants.

The drawing γ_k satisfies Property 1: Since σ_u contains no R-live points, it follows that $L^r_k = L^r_{k-1}$. If there are no R-live points in σ_k, i.e. $L^r_k = \phi$ then Property 1 holds trivially. Otherwise consider a point $p_l \in L^r_k$ such that p_l is not accessible from the bottom page in γ_k. Since p_l is also in L^r_{k-1}, by Property 1 of induction hypothesis, p_l is accessible from bottom page in γ_{k-1}. Therefore, it must be some edge e added to γ_{k-1} that makes p_l inaccessible from bottom page where p_l lies between the endpoints of e. From the drawing operation specified, e is either an edge of γ_u or the edge (u, v). Since p_l in not in σ_u, p_l cannot lie between the endpoints of any edge in γ_u. Hence e is not an edge of

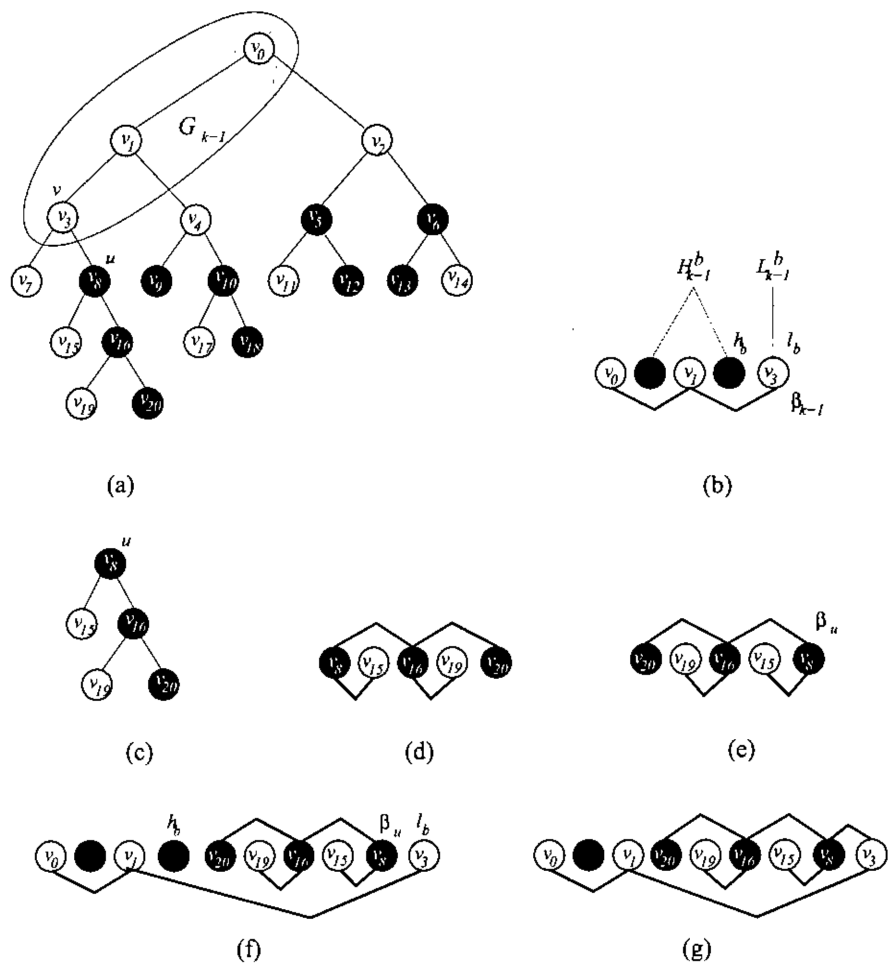


Figure 4.7 An illustration for case 2.2 of Procedure **Tree-Embed**. (a) A 2-colored tree G, (b) the drawing γ_{k-1}, (c) subgraph G_u of G, (d) the drawing γ_u, (e) the drawing after horizontal flip of γ_u, (f) insertion of γ_u between h_b and $next(h_b)$, and (g) the drawing γ_k.

γ_u. Moreover, e cannot be the edge (u, v) that connects the points l_b and β_u since the edge (u, v) is drawn through the top page and therefore cannot make p_l inaccessible from bottom page in γ_k. Hence no edge such as e exists and p_l is accessible from bottom page in γ_k. It follows that γ_k satisfies Property 1.

The drawing γ_k satisfies Property 2: Since σ_u contains no B-live points, it follows that $L_b^k = L_{k-1}^b$. If there are no B-live points in σ_k, i.e. $L_k^b = \phi$ then Property 2 holds trivially. Otherwise consider a point $p_l \in L_k^b$ such that p_l is not accessible from top page in γ_k.

Since p_l is also in L^b_{k-1}, by Property 2 of induction hypothesis, p_l is accessible from top page in γ_{k-1}. Therefore, it must be some edge e added to γ_{k-1} that makes p_l inaccessible from top page where p_l lies between the endpoints of e. From the drawing operation specified, e is either an edge of γ_u or the edge (u, v). Since p_l in not in σ_u, p_l cannot lie between the endpoints of any edge in γ_u. Hence e is not an edge of γ_u. Then e must be the edge (u, v) that connects the point l_b and β_u. Since both l_b and p_l are in L^b_{k-1} and l_b is the leftmost of point of L^b_{k-1}, it follows that l_b is to the left of p_l. Since the point h_b is in H^b_{k-1}, by Property 3 of induction hypothesis, h_b is to the left of p_l. Since we insert γ_u between the points h_b and $next(h_b)$ in σ_{k-1} and β_u is a point of σ_u, hence any point to the right of h_b in σ_{k-1} is also to the right of β_u in σ_k. It follows that p_l is to the right of β_u in σ_k since p_l is right to h_b in σ_{k-1} (by Property 3). Since both l_b and β_u are to the left of p_l, the edge (u, v) cannot cause p_l to be inaccessible from top page. Hence e cannot be the edge (u, v). Consequently no edge such as e exists and p_l is also accessible from top page in γ_k. It follows that all points in L^b_k are accessible from top page.

The drawing γ_k satisfies Property 3: There are no blue holes in σ_u. Moreover, since we remove the point $h_b \in H^b_{k-1}$, it follows that $H^b_k = H^b_{k-1} \setminus \{h_b\}$. If σ_k contains no blue holes, i.e. $H^b_k = \phi$ then Property 3 holds trivially. Otherwise we first show that there is no B-live point to the left of any blue hole in σ_k. Consider any blue hole $p_h \in H^b_k$. It follows that $p_h \in H^b_{k-1}$. By Property 3 of induction hypothesis, no point of L^b_{k-1} is to left of p_h in σ_{k-1}. Since $L^b_k = L^b_{k-1}$, it follows that there is no B-live point to the left of p_h in σ_k.

We next prove that any blue hole of σ_k is accessible from top page in γ_k. Consider a point $p_h \in H^b_k$ such that p_h is not accessible from the top page in γ_k. Since p_h is also in H^b_{k-1}, by Property 3 of induction hypothesis, p_h is accessible from top page in γ_{k-1}. Therefore, it must be some edge e added to γ_{k-1} that makes p_h inaccessible from top page where p_h lies between the endpoints of e. From the drawing operation specified, e is either an edge of γ_u or the edge (u, v). Since p_h in not in σ_u, p_h cannot lie between the endpoints of any edge in γ_u. Hence e is not an edge of γ_u. Then e must be the edge (u, v) that connects the point l_b and β_u. Since l_b is in L^b_{k-1}, by Property 3 of induction hypothesis, l_b is to the right of p_h in σ_{k-1}. The point h_b is the rightmost blue hole in σ_{k-1}; therefore, h_b is to right of p_h. Since we insert the drawing γ_u between the points h_b and $next(h_b)$ and β_u is a point of σ_u, hence β_u is to the right of h_b. It follows that β_u is to right of p_h. Since both l_b and β_u are to the right of p_h, the edge (u, v) cannot cause p to be inaccessible from top page. Hence e cannot be the edge (u, v). Consequently no edge such as e exists and $p_b h$ is also accessible from top page in γ_k.

The drawing γ_k satisfies Property 4: From the operation specified, it follows that σ_u is an alternating RB-sequence where both the leftmost and rightmost points of σ_u are blue. As a result, after horizontal flip of γ_u, the leftmost and the rightmost points of σ_u are still blue. Since σ_{k-1} is an alternating RB-sequence and $c(h_b)$ is blue, it follows that both the points $prev(h_b)$ and $next(h_b)$ are red. Thus after insertion of γ_u between the points h_b and $next(h_b)$ in γ_{k-1} and then removal of point h_b, the resultant point-set σ_k is also an alternating RB-sequence where $\beta_k = \beta_{k-1}$. We now identify type of σ_k. Since σ_u is of Type VII, it follows that $L^r_u = \phi$, $L^b_u = \phi$, $H^r_u = \phi$ and $H^b_u = \phi$. Moreover, $H^r_k = H^r_{k-1} \bigcup H^r_u = \phi$. Now there may be the following cases.

(i) $L^b_k \neq \phi$ and $H^b_k = \phi$: In this case, σ_k is of Type I since $c(\beta_k)$ is red, $L^b_k \neq \phi$, $H^r_k = \phi$ and $H^b_k = \phi$.

(ii) $L^b_k \neq \phi$ and $H^b_k \neq \phi$: In this case, σ_k is of Type II since $c(\beta_k)$ is red, $L^b_k \neq \phi$, $H^r_k = \phi$ and $H^b_k \neq \phi$.

(iii) $L^b_k = \phi$ and $L^r_k \neq \phi$: In this case, σ_k is of Type III since $c(\beta_k)$ is red, $L^r_k \neq \phi$, $L^b_k = \phi$ and $H^r_k = \phi$.

(iv) $L^b_k = \phi$ and $L^r_k = \phi$: In this case, σ_k is of Type IV since $c(\beta_k)$ is red, $L^r_k = \phi$, $L^b_k = \phi$ and $H^r_k = \phi$.

Therefore, Property 4 holds for γ_k.

The drawing γ_k satisfies Property 5: From the operations specified, it follows that

$V(G_k) = V(G_{k-1}) \bigcup V(G_u)$. By induction hypothesis, G_{k-1} is connected. Moreover, G_u is a connected subgraph of G. Since $v \in V(G_{k-1})$, $u \in V(G_u)$ and we add the edge (u, v), it follows that G_k is also connected. Now it remains to show that the edge (u, v) does not create any edge crossings and contains at most one bend. Since $l_b \in L^b_{k-1}$, by induction hypothesis, l_b is accessible from the top page in γ_{k-1}. β_u is the rightmost point of σ_u and thus accessible from both the pages in γ_u. Therefore, to make β_u inaccessible from top page after insertion of γ_u between h_b and $next(h_b)$, there must be an edge e in γ_{k-1} such that one endpoint of e is to the left of h_b and the other is to the right of h_b. But such an edge also makes h_b inaccessible from top page in γ_{k-1} and thus contradicts the induction hypothesis that γ_{k-1} maintains Property 3. Hence both l_b and β_u are accessible from the top page and therefore can be connected with such a polygonal chain through the top page that does not cross any other edge and may contain at most one bend (according to Observation 2.2.1). Therefore, Property 5 holds for γ_k.

Case 2.3: The rightmost point in σ_u is blue and σ_u has no R-live points or holes but contains at least one B-live point.

Note that this case arises when **Tree-Embed**(G_u, u, 1) terminates in the way as specified in case 6.2. It follows that γ_u satisfies the invariant properties 1-5 and thus represents a bichromatic point-set embedding of a graph G_s on σ_u (by Property 5) where G_s is a connected subgraph of G_u and G_s contains the vertex u. Moreover, σ_u is of Type VI. Let h_b be the rightmost point in H^b_{k-1}. Now we perform the following operations. We first flip γ_u horizontally. As a result the rightmost point in σ_u now represents the vertex u; let β_u denotes this point. Next we insert the drawing γ_u between the points h_b and $next(h_b)$ in σ_{k-1} and then remove the point h_b. Finally we add the edge (u, v) connecting the points l_b and β_u through the top page. Figure 4.8 illustrates this case.

From the operations specified, if follows that $L^r_k = L^r_{k-1} \bigcup L^r_u$, $L^b_k = L^b_{k-1} \bigcup L^b_u$, $H^b_k = H^b_{k-1} \bigcup H^b_u \setminus \{h_b\}$ and $\sigma_k = \sigma_{k-1} \bigcup \sigma_u$. We now show that the drawing after this step satisfies the invariants.

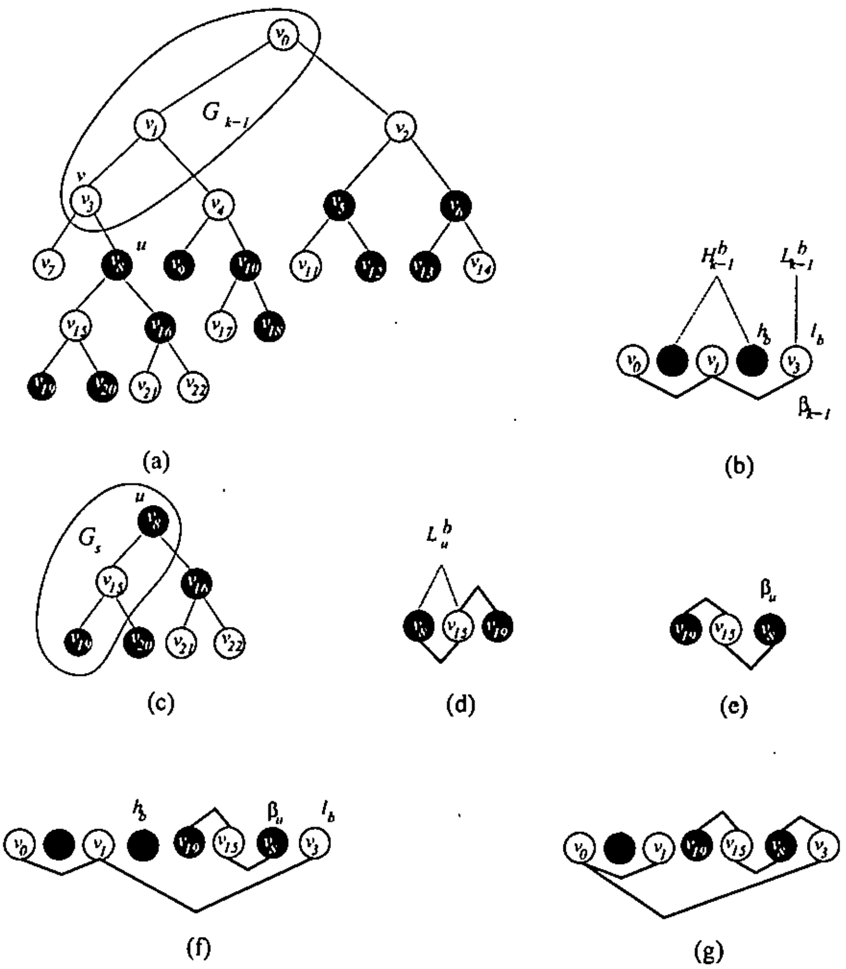


Figure 4.8 An illustration for case 2.3 of Procedure **Tree-Embed**. (a) A 2-colored tree G, (b) the drawing γ_{k-1}, (c) graph G_u; shaded area denotes the graph G_s mapped on γ_u, (d) the drawing γ_u, (e) the drawing after horizontal flip of γ_u, (f) insertion of γ_u between h_b and $next(h_b)$, and (g) the drawing γ_k.

The drawing γ_k *satisfies Property* 1: Since σ_u contains no R-live points, it follows that $L_k^r = L_{k-1}^r$. If there are no R-live points in σ_k, i.e. $L_k^r = \phi$ then Property 1 holds trivially. Otherwise consider a point $p_l \in L_k^r$ such that p_l is not accessible from bottom page in γ_k. Since p_l is also in L_{k-1}^r, by Property 1 of induction hypothesis, p_l is accessible from bottom page in γ_{k-1}. Therefore, it must be some edge e added to γ_{k-1} that makes p_l inaccessible from bottom page where p_l lies between the endpoints of e. From the drawing operation specified, e is either an edge of γ_u or the edge (u, v). Since p_l in not in σ_u, p_l cannot lie between the endpoints of any edge in γ_u. Hence e is not an edge of γ_u. Moreover, e cannot be the edge (u, v) that connects the points l_b and β_u since the edge (u, v) is drawn through the top page and therefore cannot make p_l inaccessible from bottom page in γ_k. Hence no edge such as e exists and p_l is accessible from bottom page in γ_k. Thus γ_k satisfies Property 1.

The drawing γ_k satisfies Property 2: Let L_u^b denotes the set of B-live points in σ_u. It follows that $L_k^b = L_{k-1}^b \bigcup L_u^b$. Consider any point $p_l \in L_k^b$ such that p_l is not accessible from the top page in γ_k. p_l is either in L_{k-1}^b or in L_u^b. Consider the case when $p_l \in L_{k-1}^b$. By Property 2 of induction hypothesis, p_l is accessible from top page in γ_{k-1}. Therefore, it must be some edge e added to γ_{k-1} that makes p_l inaccessible from top page where p_l lies between the endpoints of e. From the drawing operation specified, e is either an edge of γ_u or the edge (u, v). Since p_l in not in σ_u, p_l cannot lie between the endpoints of any edge in γ_u. Hence e is not an edge of γ_u. Then e must be the edge (u, v) that connects the points l_b and β_u. Since l_b is the rightmost point of L_{k-1}^b, l_b is to the left of p_l in σ_{k-1}. The point h_b is a blue hole in σ_{k-1}; therefore, by Property 3, h_b is to left of p_l. Since we insert the drawing γ_u between the points h_b and $next(h_b)$ and β_u is a point of σ_u, hence any point to the right of h_b in σ_{k-1} is also to the right of β_u in σ_k. It follows that p_l is to the right of β_u in σ_k. Now since both l_b and β_u are to the left of p_h, the edge (u, v) cannot cause p_l to be inaccessible from top page in γ_k. Hence e cannot be the edge (u, v). Consequently no edge such as e exists and p_l is also accessible from top page in γ_k.

Now consider the case when $p_l \in H_u^b$. Since drawing γ_u satisfies the invariants, it follows that p_l is accessible from top page in γ_u. Moreover, the horizontal flip operation does not change the top page accessibility of p_l in γ_u (according to Observation 4.1.1). Since we insert γ_u between the points h_b and $next(h_b)$ in σ_{k-1}, therefore, to make p_l inaccessible from the top page in γ_k, there must be such an edge e in γ_{k-1} that one endpoint of e is to the left of h_b and another is to the right of h_b. But such an edge makes the point $h_b \in H^b_{k-1}$ inaccessible from top page in γ_{k-1}. This contradicts the induction hypothesis that h_b is accessible from top page in γ_{k-1}. Therefore, no edge such as e exists and p_l is accessible from top page in γ_k. Therefore, each B-live point in γ_k is accessible from the top page.

The drawing γ_k satisfies Property 3: There are no blue holes in σ_u. Moreover, since we remove the point $h_b \in H^b_{k-1}$, it follows that $H^b_k = H^b_{k-1} \backslash \{h_b\}$. If σ_k contains no blue holes, i.e. $H^b_k = \phi$ then Property 3 holds trivially. Otherwise we first show that there is no B-live points to the left of any blue hole in σ_k. Consider any blue hole $p_h \in H^b_k$ and any B-live point $p_l \in L^b_{k-1}$. Now p_bl is either in L^b_{k-1} or in L^b_u. We first examine the case when $p_l \in L^b_{k-1}$. Since $p_h \in H^b_{k-1}$, by Property 3 of induction hypothesis, p_h is to the left of p_l. . Next consider the case when $p_l \in H^b_u$. Since h_b is the rightmost point in H^b_{k-1}, it follows that p_h is to the left of h_b. Since we insert the drawing γ_u between the points h_b and $next(h_b)$, therefore, points in σ_u are to the right of h_b. It follows that p_bl is to the right of p_h. Hence points in L^b_k are to the right of points in H^b_k.

We next prove that any blue hole of σ_k is accessible from top page in γ_k. Consider a point $p_h \in H^b_k$ such that p_h is not accessible from the top page in γ_k. Since p_h is also in H^b_{k-1}, p_h is accessible from top page in γ_{k-1} (by Property 3). Therefore, it must be some edge e added to γ_{k-1} that makes p_h inaccessible from top page where p_h lies between the endpoints of e. From the drawing operation specified, e is either an edge of γ_u or the edge (u, v). Since p_h in not in σ_u, p_h cannot lie between the endpoints of any edge in γ_u. Hence e is not an edge of γ_u. Then e must be the edge (u, v) that connects the point l_b and β_u. Since l_b is in L^b_{k-1}, by Property 3 of induction hypothesis, l_b is to the right of p_h in σ_{k-1}. The point h_b is the rightmost blue hole in σ_{k-1}; therefore, h_b is to right of p_h. Since we insert the drawing γ_u between the points h_b and $next(h_b)$ and β_u is a point of σ_u, hence β_u is to the right of h_b. It follows that β_u is to right of p_h. Since both l_b and β_u are to the right of p_h, the edge (u, v) cannot cause p to be inaccessible from top page. Hence e cannot be the edge (u, v). Consequently no edge such as e exists and p_bh is also accessible from top page in γ_k.

The drawing γ_k *satisfies Property* 4: From the operations specified, it follows that σ_u is an alternating RB-sequence where both the leftmost and rightmost points of σ_u are blue. As a result, after horizontal flip of γ_u, the leftmost and the rightmost point of σ_u are still blue. Since σ_{k-1} is an alternating RB-sequence and $c(h_b)$ is blue, it follows that both the points $prev(h_b)$ and $next(h_b)$ are red. Thus after insertion of γ_u between the points h_b and $next(h_b)$ in γ_{k-1} and then removal of point h_b, the resultant point-set σ_k is also an alternating RB-sequence where $\beta_k = \beta_{k-1}$. We now identify type of σ_k. Since σ_u is of Type VI, it follows that $L_u^r = \phi$, $L_u^b \neq \phi$, $H_u^r = \phi$ and $H_u^b = \phi$. From the operations specified, $H_k^r = H_{k-1}^r \bigcup H_u^r = \phi$. There may be the following cases.

(i) $L_k^b \neq \phi$ *and* $H_k^b = \phi$: In this case, σ_k is of Type I since $c(\beta_k)$ is red, $L_k^b \neq \phi$, $H_k^r = \phi$ and $H_k^b = \phi$.

(ii) $L_k^b \neq \phi$ *and* $H_k^b \neq \phi$: in this case σ_k is of Type II since $c(\beta_k)$ is red, $L_k^b \neq \phi$, $H_k^r = \phi$ and $H_k^b \neq \phi$.

(iii) $L_k^b = \phi$ *and* $L_k^r \neq \phi$: in this case σ_k is of Type III since $c(\beta_k)$ is red, $L_k^r \neq \phi$, $L_k^b = \phi$ and $H_k^r = \phi$.

(iv) $L_k^b = \phi$ *and* $L_k^r = \phi$: in this case σ_k is of Type IV since $c(\beta_k)$ is red, $L_k^r = \phi$, $L_k^b = \phi$ and $H_k^r = \phi$.

Therefore, Property 4 holds for γ_k.

The drawing γ_k *satisfies Property* 5: From the operations specified, it follows that $V(G_k) = V(G_{k-1}) \bigcup V(G_s)$. By induction hypothesis, G_{k-1} is connected. Moreover, G_s is also a connected subgraph of G. Since $v \in V(G_{k-1})$ and $u \in V(G_s)$ and the edge (u, v) is in γ_k, it follows that G_k is also connected. Now it remains to show that the edge (u, v) does not create any edge crossing and contains at most one bend. Since $l_b \in L_{k-1}^b$, by Property 2 of induction hypothesis, l_b is accessible from the top page in γ_{k-1}. After horizontal flip of γ_u, the vertex u is represented by the rightmost point β_u of σ_c and thus accessible from both the pages in γ_u. Therefore, to make β_u inaccessible from top page after insertion of γ_u between h_b and $next(h_b)$, there must be an edge e in γ_{k-1} such that one endpoint of e is to the left of h_b and the other is to the right of h_b. But such an edge also makes h_b inaccessible from top page in γ_{k-1}, therefore, contradicts the induction hypothesis that γ_{k-1} maintains Property 3. Hence both l_b and β_u are accessible from the top page and thus can be connected with a polygonal chain through the top plain and the edge may contain at most one bend (by Observation 2.2.1).

Case 3: σ_{k-1} is of Type III, i.e. $c(\beta_{k-1})$ is red, $L^b_{k-1} = \phi$, $L^r_{k-1} \neq \phi$ and $H^r_{k-1} = \phi$.

We add two points p_b and p_r on the spine of σ_{k-1} such that $x(\beta_{k-1}) < x(p_b) < x(p_r)$, $c(p_b)$ is blue and $c(p_r)$ is red. Let l_r be the rightmost point in L^r_{k-1} and v be the vertex represented by l_r. Hence v is an R-live vertex and there is a vertex $u \in U_{k-1}(v)$ such that $c(u)$ is red. We map u on p_r and draw the edge (u, v) connecting points l_r and p_r through the bottom page. Figure 4.9 illustrates this case.

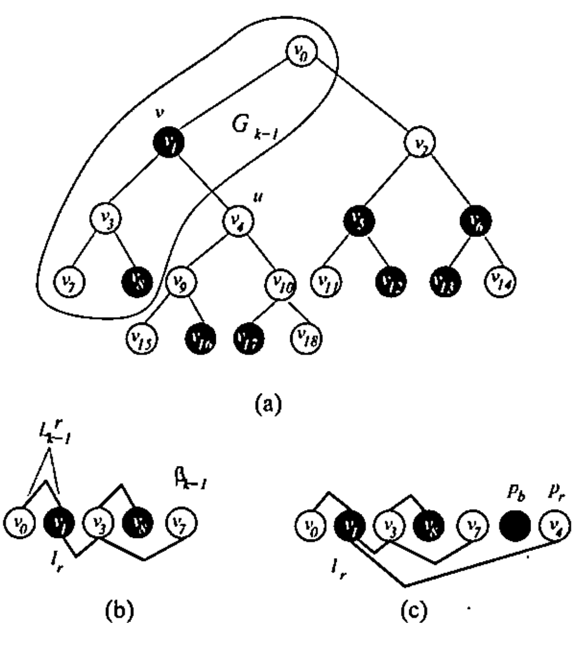


Figure 4.9 An illustration for case 3 of Procedure **Tree-Embed**. (a) A 2-colored tree G, (b) the drawing γ_{k-1}, and (c) the drawing γ_k.

We now show that the drawing after this step satisfies the invariants.

The drawing γ_k satisfies Property 1: If σ_k contains no R-live points, i.e. $L_k^r = \phi$ then Property 1 holds trivially. Otherwise consider a point $p_l \in L_k^r$ such that p_l is not accessible from bottom page in γ_k. Let v_l be the vertex of G represented by p_l. Hence v_l is an R-live vertex in G_k. Since p_r is the rightmost point in σ_k, p_r is accessible from both the pages in γ_k. It follows that $p_l \neq p_r$. Since u is mapped on p_r, $v_l \neq u$. Then v_l must be an R-live vertex of G_{k-1}. It follows that p_l is an R-live point of σ_{k-1}. Therefore, by Property 1 of induction hypothesis, p_l is accessible from bottom page in γ_{k-1}. Consequently it must be the edge (u, v) that makes p_l inaccessible from bottom page i.e. p_l lies between the points l_r and p_r. Now since both p_l and l_r are in L_{k-1}^r and l_r is the rightmost point of L_{k-1}^r, it follows that l_r is to the right of p_l. Again p_r is the rightmost point of σ_k and hence is to the right of p_l. Since both l_r and p_r are to the right of p_l, the edge (u, v) cannot make p_l inaccessible from any page. Therefore, no point such as p_l exists. Hence all points in L_k^r are accessible from the bottom page in γ_k.

The drawing γ_k satisfies Property 2: If there are no B-live points in σ_k, i.e. $L_k^b = \phi$ then Property 2 holds trivially. Otherwise $L_k^b = \{p_r\}$ since $L_{k-1}^b = \phi$. Since p_r is the rightmost point of σ_k, it is accessible from both the pages in γ_k. Therefore, Property 2 holds for γ_k.

The drawing γ_k satisfies Property 3: According to the operation specified, the point p_b is a blue hole in σ_k. Hence $H_k^b = H_{k-1}^b \bigcup \{p_b\}$. Since $L_{k-1}^b = \phi$, it follows that either $L_k^b = \phi$ or $L_k^b = \{p_r\}$. Since p_r is the rightmost point in σ_k, therefore, irrespective of whether p_r is in L_k^b or not, there is no B-live point to the left of any blue hole in σ_k.

We next prove that any blue hole of σ_k is accessible from top page in γ_k. Consider a point $p_h \in H_k^b$ such that p_h is not accessible from the top page in γ_k. p_h is either in H_{k-1}^b or $p_h = p_b$. Consider the case when $p_h \in H_{k-1}^b$. By Property 3 of induction hypothesis, p_h is accessible from top page in γ_{k-1}. Therefore, it must be the edge (u, v) that makes p_h inaccessible from top page in γ_k. But it is not possible since the edge (u, v) is drawn through the bottom page. Hence p_h must be accessible from top page in γ_k. We then examine the case when $p_h = p_b$. Let e be the edge that makes p_h inaccessible from top page in γ_k i.e. p_b lies between the endpoints of e. e is either an edge of γ_{k-1} or the edge (u, v). Since p_b is to the right of β_{k-1}, both the endpoints of any edge in γ_{k-1} are to the left of p_b. Therefore, e cannot be an edge of γ_{k-1}. Also e cannot be the edge (u, v) since the edge is drawn through the bottom page. Hence no edge such as e exists and p_h is accessible from top page in γ_k. Thus Property 3 holds for σ_k.

The drawing γ_k satisfies Property 4: By induction hypothesis σ_{k-1} is an alternating RB-sequence where $c(\beta_{k-1})$ is red. From the way the points p_b and p_r are taken, one can observe that $\sigma_k = \sigma_{k-1} \bigcup \{p_b, p_r\}$ is also an alternating RB-sequence where $\beta_k = p_r$. We now determine the type of σ_k. From the operations specified, $H_k^r = H_{k-1}^r = \phi$ and $H_k^b = H_{k-1}^b \bigcup \{p_b\}$. There may be the following cases.

(i) $L_k^b \neq \phi$: In this case, σ_k is of Type II since $c(\beta_k)$ is red, $L_k^b \neq \phi$, $H_k^r = \phi$ and $H_k^b \neq \phi$.

(ii) $L_k^b = \phi$ *and* $L_k^r \neq \phi$: In this case, σ_k is of Type III since $c(\beta_k)$ is red, $L_k^r \neq \phi$, $L_k^b = \phi$ and $H_k^r = \phi$.

(iii) $L_k^r = \phi$ *and* $L_k^b = \phi$: In this case, σ_k is of Type IV since $c(\beta_k)$ is red, $L_k^r = \phi$, $L_k^b = \phi$ and $H_k^r = \phi$.

Therefore, γ_k satisfies Property 4.

The drawing γ_k satisfies Property 5: According to the operation specified, $V(G_k) = V(G_{k-1}) \bigcup \{u\}$. Since G_{k-1} is a connected graph that contains the vertex v_0(by induction hypothesis) and u is a neighbor of some vertex $v \in V(G_{k-1})$, if follows that the graph G_k is also connected and v_0 is in G_k. Therefore, it remains to show that the edge (u, v) does not create any edge crossing and contains at most one bend. Since $l_r \in L_{k-1}^r$, by Property 2 it is accessible from the bottom page in γ_{k-1}. The point p_r is to the right of the rightmost point of σ_{k-1}; hence p_r is accessible from both the pages (top and bottom). Therefore, l_r and p_r can be connected with a polygonal chain through the bottom page that contains at most one bend and does not cross any other edge in γ_{k-1} (by Observation

2.2.1. Hence γ_k represents a bichromatic point-set embedding of G_k on σ_k.

Case 4: σ_{k-1} is of Type IV, i.e. $c(\beta_{k-1})$ is red, $L^b_{k-1} = \phi$, $L^r_{k-1} = \phi$ and $H^r_{k-1} = \phi$.

In this case, there are no live points in σ_{k-1} which implies that G has no unmapped vertices. At this point, the procedure terminates. Let γ_G and σ_G represents the output drawing and output point-set respectively where $\gamma_G = \gamma_{k-1}$ and $\sigma_G = \sigma_{k-1}$. We distinguish the following two sub cases determined by the value of *level*(the other input of the Procedure **Tree-Embed**).

Case 4.1: *level*= 0. In this case, we check the number of times G has been inverted inside this instance of the procedure. It should be noted that G is inverted in each intermediate step i whenever σ_{i-1} is of Type VI and the value of input *level*= 0(refer to case 6.2). If G has been inverted odd number of times, we invert G and σ_G once more. Then the procedure terminates and returns the drawing γ_G.

Case 4.2: *level*= 1. In this case, the procedure simply terminates and returns the drawing γ_G.

Case 5: σ_{k-1} is of Type V, i.e. $c(\beta_{k-1})$ is blue, $L^r_{k-1} \neq \phi$, $H^r_{k-1} = \phi$ and $H^b_{k-1} = \phi$.

We add a point p_r on the spine of σ_{k-1} such that p_r is to the right of β_{k-1} and $c(p_r)$ is red. Let l_r be the rightmost point in L^r_{k-1} and v be the vertex of G mapped on l_r. Hence v is an R-live vertex and there is a vertex $u \in U_{k-1}(v)$ such that $c(u)$ is red. We map u on p_r and draw the edge (u, v) connecting points l_r and p_r through the bottom page. Figure 4.10 illustrates this case.

We now prove that γ_k satisfies the invariants.

The drawing γ_k satisfies Property 1: If there are no R-live points in σ_k, i.e. $L^r_k = \phi$ then Property 1 holds trivially. Otherwise consider a point $p_l \in L^r_k$ such that p_l is not accessible from the bottom page in γ_k. Let v_l be the vertex of G represented by p_l. Hence v_l is an R-live vertex in G_k. Since p_r is the rightmost point in σ_k, p_r is accessible from both the pages in γ_k. It follows that $p_l \neq p_r$. Since u is mapped on p_r, $v_l \neq u$. Then v_l must be an R-live vertex of G_{k-1}. It follows that p_l is an R-live point of σ_{k-1}.

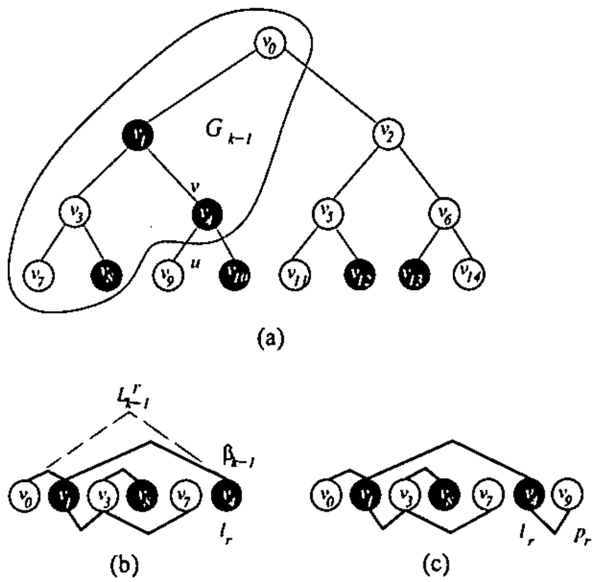


Figure 4.10 An illustration for case 5 of Procedure **Tree-Embed**. (a) A 2-colored tree G, (b) the drawing γ_{k-1}, and (c) the drawing γ_k.

Therefore, by Property 1 of induction hypothesis, p_l is accessible from bottom page in γ_{k-1}. Consequently it must be the addition of the edge (u, v) that makes p_l inaccessible from bottom page in γ_k. The endpoints of the edge (u, v) are p_r and l_r. Since p_r is the rightmost point of σ_k, p_r is to the right of p_l. Also l_r is to the right of p_l since both l_r and p_l are in L^r_{k-1} and l_r is the rightmost point of L^r_{k-1}. Since both the endpoints of edge (u, v) lie to the right of p_l in γ_k, it follows that the edge (u, v) cannot make p_l inaccessible from bottom page. Therefore, p_l is accessible from bottom page in γ_k which is a contradiction. Hence all points in L^r_k are accessible from the bottom page in γ_k. Thus γ_k satisfies Property 1.

The drawing γ_k satisfies Property 2: If σ_k contains no B-live points, i.e. $L^b_k = \phi$ then Property 1 holds trivially. Otherwise consider a point $p_l \in L^b_k$ such that p_l is not accessible from top page in γ_k. Let v_l be the vertex of G represented by p_l. Hence v_l is a B-live vertex in G_k. Since p_r is the rightmost point in σ_k, p_r is accessible from both the pages in γ_k. It follows that $p_l \neq p_r$. Since u is mapped on p_r, $v_l \neq u$. Then v_l must be a B-live vertex of G_{k-1}. It follows that p_l is a B-live point of σ_{k-1}. Therefore, by Property 2 of induction hypothesis, p_l is accessible from top page in γ_{k-1}. Consequently it must be the addition of the edge (u, v) that makes p_l inaccessible from top page in γ_k. But since we draw the edge (u, v) through the bottom page, it cannot make p_l inaccessible from top page. It follows that p_l is accessible from bottom page in γ_k which is a contradiction. Therefore, all points in L_k^b are accessible from the top page. Thus γ_k satisfies Property 2.

The drawing γ_k satisfies Property 3: Since σ_{k-1} contains no blue holes and no new blue hole is added by the operation defined for this step, it follows that $H_k^b = \phi$. Therefore, Property 3 is maintained trivially.

The drawing γ_k satisfies Property 4: By induction hypothesis, σ_{k-1} is an alternating RB-sequence where $c(\beta_{k-1})$ is blue. Since the point p_r is to the right of β_{k-1} and $c(p_r)$ is red, it follows that $\sigma_k = \sigma_{k-1} \bigcup \{p_r\}$ is also an alternating RB-sequence where $\beta_k = p_r$. We now determine the type of σ_k. From the operations specified, $H_k^r = H_{k-1}^r = \phi$ and $H_k^b = H_{k-1}^b = \phi$. There may be the following cases.

(i) $L_k^b \neq \phi$: In this case, σ_k is of Type I since $c(\beta_k)$ is red, $L_k^b \neq \phi$, $H_k^r = \phi$ and $H_k^b = \phi$.

(ii) $L_k^b = \phi$ and $L_k^r \neq \phi$: In this case, σ_k is of Type III since $c(\beta_k)$ is red, $L_k^r \neq \phi$, $L_k^b = \phi$ and $H_k^r = \phi$.

(iii) $L_k^r = \phi$ and $L_k^b = \phi$: In this case, σ_k is of Type IV since $c(\beta_k)$ is red, $L_k^r = \phi$, $L_k^b = \phi$ and $H_k^r = \phi$.

Therefore, for all possible input combinations σ_k is either of types I, III and IV. Thus γ_k satisfies Property 4.

The drawing γ_k satisfies Property 5: According to the operation specified, $V(G_k) = V(G_{k-1}) \bigcup \{u\}$. Since G_{k-1} is a connected graph that contains the vertex v_0 (by induction hypothesis) and u is a neighbor of some vertex $v \in V(G_{k-1})$, if follows that the graph G_k is also connected and v_0 is in G_k. Now it remains to show that the edge (u, v) does not create any edge crossing and contains at most one bend. Since $l_r \in L_{k-1}^r$, by Property 2 it is accessible from the bottom page in γ_{k-1}. The point p_r is taken such that p_r is to the right of the rightmost point of σ_{k-1}; hence p_r is accessible from both the pages. Therefore, l_r and p_r can be connected with a polygonal chain through the bottom page that contains at most one bend and does not cross any other edge in γ_{k-1} (from Observation 2.2.1). Hence γ_k represents a bichromatic point-set embedding of G_k on σ_k.

Case 6: σ_{k-1} is of Type VI, i.e. $c(\beta_{k-1})$ is blue, $L^r_{k-1} = \phi$, $L^b_{k-1} \neq \phi$, $H^r_{k-1} = \phi$ and $H^b_{k-1} = \phi$.

We distinguish two sub case as determined by the value of *level*.

Case 6.1: *level*$= 0$. In this case, we first invert G. It should be noted that later iterations in this instance of the procedure will consider this inverted graph as input. Next we invert σ_{k-1} and then flip γ_{k-1} vertically. The resulting drawing and the point-set are denoted by γ_k and σ_k respectively. Figure 4.11 illustrates the case.

We now prove that γ_k satisfies the invariants.

The drawing γ_k satisfies Property 1: Since points in σ_k are obtained after inverting the points in σ_{k-1}, it follows that $L^r_k = L^b_{k-1}$. By induction hypothesis, points in L^b_{k-1} are accessible from the top page. Now γ_k is obtained by a vertical flip of γ_{k-1}. Therefore, according to Observation 4.1.1, points in L^r_k are accessible from bottom page in γ_k. Hence Property 1 holds for γ_k.

The drawing γ_k satisfies Property 2: Since points in σ_k are obtained after inverting the points in σ_{k-1}, it follows that $L^b_k = L^r_{k-1}$. By induction hypothesis, points in L^r_{k-1} are accessible from the bottom page. Now γ_k is obtained by a vertical flip of γ_{k-1}. Therefore, according to Observation 4.1.2, points in L^b_k are accessible from top page in γ_k. Hence Property 2 holds for γ_k.

The drawing γ_k satisfies Property 3: Since σ_{k-1} does not contain any blue hole and points in σ_k are obtained after inverting the points in σ_{k-1}, it follows that there are no blue holes in σ_k. Now γ_k is obtained by a vertical flip of γ_{k-1}. This operation does not create in any blue hole either. Hence $H^b_k = \phi$ and thus Property 3 trivially holds for γ_k.

The drawing γ_k satisfies Property 4: By Property 4 of induction hypothesis, σ_{k-1}

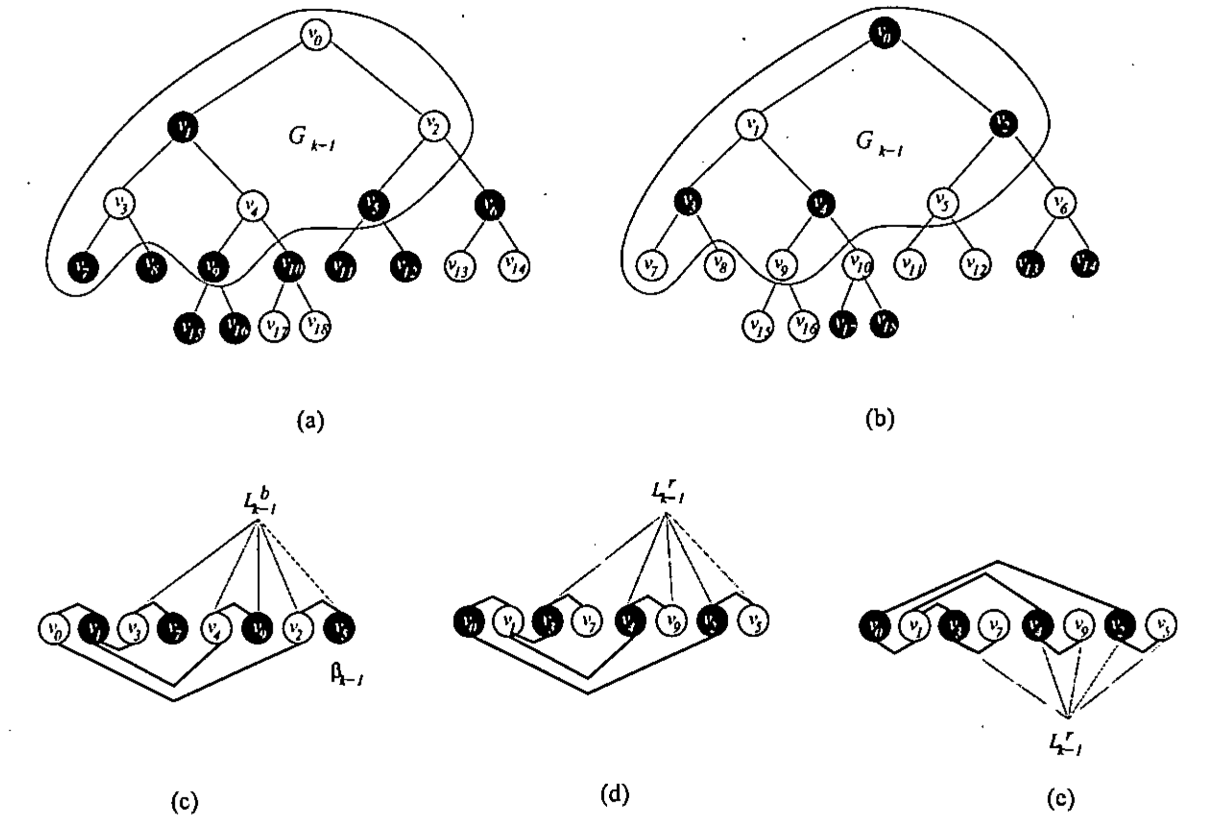


Figure 4.11 An illustration for case 6.1 of Procedure **Tree-Embed**. (a) A 2-colored tree G, (b) the graph G after inversion, (c) the drawing γ_{k-1}, (d) the drawing γ_{k-1} after inversion of σ_{k-1}, and (e) the drawing γ_k obtained by vertical flip of the drawing in (d).

is an alternating RB-sequence. Since σ_k is obtained after inverting the points in σ_{k-1}. Moreover, vertical flip of γ_{k-1} doesn't change the color of any point, it follows that σ_k is also an alternating RB-sequence. We now determine type of σ_k. Since $L_k^r = L_{k-1}^b \neq \phi$, $L_k^b = L_{k-1}^r = \phi$, $H_k^b = H_{k-1}^r = \phi$, $H_k^r = H_{k-1}^b = \phi$ and $c(\beta_k)$ it red, it follows that σ_k is of Type III. Thus γ_k satisfies Property 4.

The drawing γ_k satisfies Property 5: We invert both the input graph G and σ_{k-1} and then flip the drawing γ_{k-1} vertically. Therefore, it follows from Observation 4.1.3 and Observation 4.1.2 that γ_k represents a bichromatic point-set embedding of G_k on σ_k. Moreover, v_0 is still mapped to the leftmost point of σ_k. It follows that γ_k satisfies

Property 5.

Case 6.2: *level*= 1.

In this case, the procedure terminates and returns. Let γ_s and σ_s be the returned drawing and point-set respectively where $\gamma_s = \gamma_{k-1}$ and $\sigma_s = \sigma_{k-1}$. Since σ_s contains B-live points, it follows that there are vertices of G that are left unmapped. Hence γ_s represents a bichromatic point-set embedding of a connected subgraph of G. Let G_s denotes this subgraph. Then $V(G_s) \subset V(G)$. By Property 5, $v_0 \in V(G_s)$ and mapped to the leftmost point of σ_s.

Case 7: σ_{k-1} is of Type VII, i.e. $c(\beta_{k-1})$ is blue, $L^b_{k-1} = \phi$, $L^r_{k-1} = \phi$, $H^r_{k-1} = \phi$ and $H^b_{k-1} = \phi$.

In this case, there are no live points in σ_{k-1} which implies that G has no unmapped vertex. At this point, the procedure terminates. Let γ_G and σ_G represents the output drawing and output point-set respectively where $\gamma_G = \gamma_{k-1}$ and $\sigma_G = \sigma_{k-1}$. We distinguish the following two sub cases determined by the value of *level*(the other input of the Procedure **Tree-Embed**).

Case 7.1: *level*= 0. In this case, we check the number of times G has been inverted inside this instance of the procedure. It should be noted that G is inverted in each intermediate step i whenever σ_{i-1} is of Type VI and the value of input *level*= 0(refer to case 6.2). If G has been inverted odd number of times, we invert G and σ_G once more. Then the procedure terminates and returns γ_G.

Case 7.2: *level*= 1. In this case, the procedure simply terminates and returns the drawing γ_G.

This concludes the description of Procedure **Tree-Embed**. We now give a formal presentation of the Procedure **Tree-Embed**. Before that we need to describe the data structures that we use in the formal description of Procedure **Tree-Embed**. We represent a 2-colored tree G using an array of $2|V|$ lists; for each vertex $v \in V$, there are two seperate lists to store the set of red children and the set of blue children of v. We use A_G to denote this representation of G. For example, Figure 3.5(b) shows the representation for the 2-colored tree in Figure 3.5(a). The set of R-live points at any step is stored in a doubly linked list. We denote this list as R_σ. Each element of R_σ holds a pointer to an R-live vertex. Moreover, elements of R_σ can be accessed from both ends. We use a similiar doubly linked list B_σ to store the set of B-live points. Mapping of vertices to points is also stored in a doubly linked list. We denote this list as M_σ. At the end of some step $k(k > 0)$, each element of M_σ represents a point p of σ_k and holds the vertex mapped to that point. M_σ also allows access from both the ends. We store the set of blue holes in another doubly linked list denoted as H_σ. Each element of H_σ holds a pointer to an element of M_σ that represents a blue hole. Note that in each of these lists the first element corresponds to the leftmost point and the last element corresponds to the rightmost point of the set it represents. Initially the lists R_σ, B_σ, H_σ and M_σ are empty. Figure 4.12 illustrates the data structures. Figure 4.12(b) shows the drawing γ_k computed after some step $k(k > 0)$ inside **Tree-Embed** for the input graph G in Figure 4.12(a). Figure 4.12(c) shows A_G after step k. Note that for each vertex $v \in V$, A_G holds the lists of unmapped red and blue children of v since whenever we map a vertex, we remove that node from the list of its parent. Figure 4.12(d) shows the lists R_σ, B_σ, H_σ and M_σ corresponding to the drawing in Figure 4.12(b). We are now ready to present a formal description of the Procedure **Tree-Embed**.

Procedure Tree-Embed(A_G, v_0, *level*)

{A_G represents a 2-colored rooted tree G, v_0 is the root of G.}

begin

Let T points to the graph currently used by the procedure;

{Initially T points to G. However in subsequent steps T may also point to the graph obtained by inversion of G;

Set T to A_G;

Set R_σ, B_σ, H_σ and M_σ to **NIL**; {Initially all the lists are empty.}

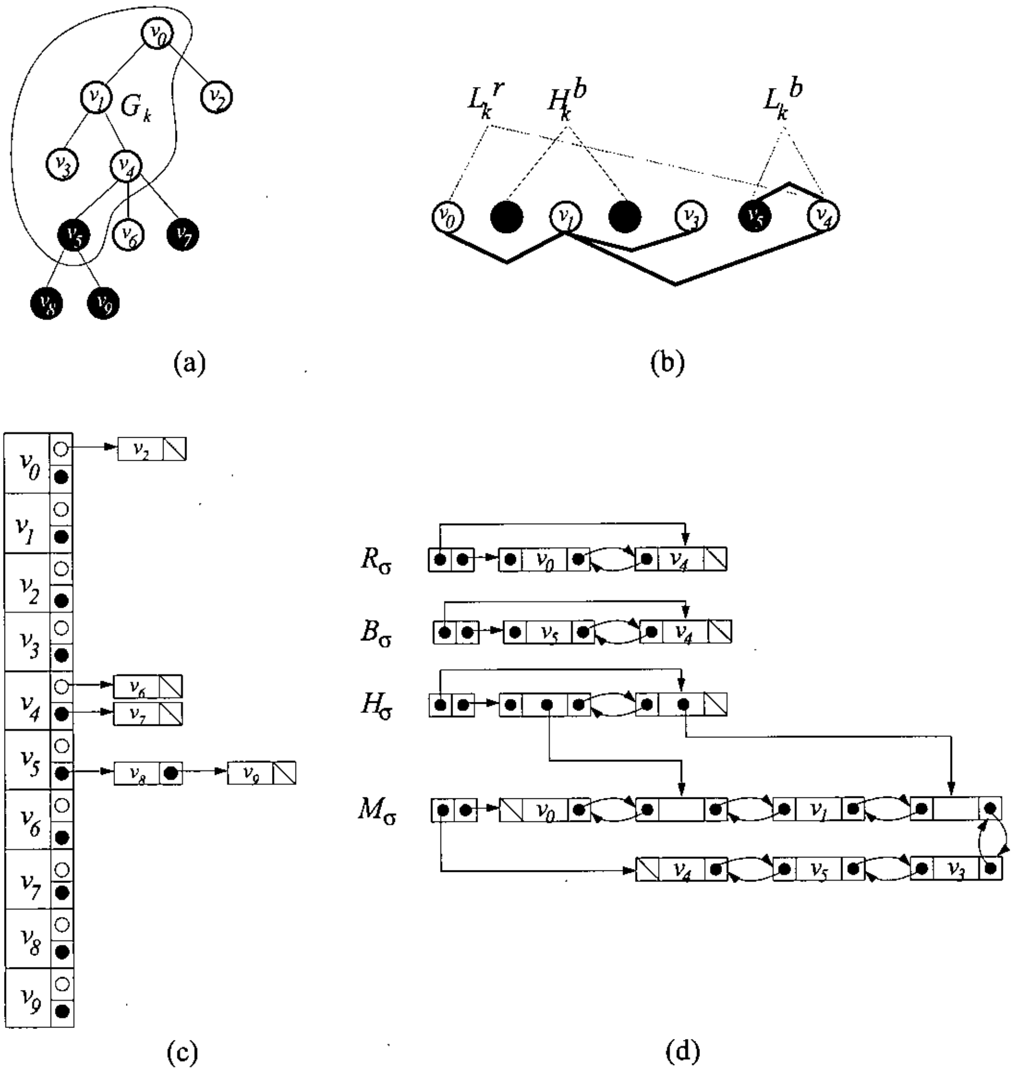


Figure 4.12 (a) A 2-colored tree G, (b) the drawing γ_k, (c) A_G after step k, and (d) states of the lists R_σ, B_σ, H_σ and R_σ after step k.

if *level*$= 0$ **then** Set A_G^T to inversion of A_G;

$\{A_G^T$ represents the graph which is the inversion of $G.\}$

Set *invert*$:= 0$; $\{$*invert* holds number of times G is inverted; c.f. case 6.2.$\}$

Set $k := 0$; $\{k$ holds the current iteration index.$\}$

$\{$We first embed the root v_0 of $G.\}$

Add v_0 to the end of M_σ; $\{$This corresponds to point $p_0.\}$

if v_0 has at least on red child in T {v_0 is an R-live vertex.}

then Add v_0 to the end of R_σ;

if v_0 has at least on blue child in T {v_0 is a B-live vertex.}

then Add v_0 to the end of B_σ;

Set *Type* according to $c(v_0)$ and status of R_σ and B_σ;

{An empty list implies that there is no point of the corresponding type in σ.}

Set $k := k + 1$;

while true do

begin

  if *Type* is I **then** {c.f. case 1}

  begin

    Let v be the vertex stored in the last element of B_σ;

    {Hence the rightmost B-live point represents v.}

    Let u be the first blue child of v in T;

    Add u to the end of M_σ;

    Remove u from the list of blue children of v in T;

    if v has no more blue child in T **then** remove v from B_σ;

    if u has at least one red child in T **then** store u at the end of R_σ;

    if u has at least one blue child in T **then** store u at the end of B_σ;

    Set *Type* according to $c(v_0)$ the status of R_σ and B_σ;

  end

  else if *Type* is II **then** {c.f. case 2.}

  begin

    Let v be the vertex stored in the first element of B_σ.

    {Hence the leftmost B-live point represents v.}

    Let u be the first blue child of v in T;

    Let h be the last element of H_σ.

{Hence h points to the rightmost blue hole of M_σ.}

Tree-Embed(T, u, 0); {Recursive invocation on subtree rooted at u.}

Let R_u, B_u, H_u and M_u be the lists returned from the procedure;

{R_u, B_u, H_u and M_u represents the sets L_u^r, L_u^b, H_u^b and σ_u respectively.}

Let $Type_u$ holds the type of σ_u;

{We assume this is also returned by the procedure.}

{Now consider the three subcases.}

if $Type_u$ is IV **then** {c.f. case 2.1. In this case, the lists R_u and B_u are empty.}

begin

Let *temp=val(h)*; { *val(h)* is the value contained in h.}

Set *next(val(h)* to *tail(M_u)* and *next(head(M_u))* to *next(temp)*;

{This operation is equivalent to perform a horizontal flip of σ_u and then insert it to the next of rightmost blue hole in σ_{k-1}.}

Remove u from the list of blue children of v in T;

if v has no more blue child in T **then** remove v from B_σ;

if H_u is not empty then **Add** H_u to the end of H_σ;

Set *Type* according to the status of R_σ, B_σ and H_σ;

end

else if $Type_u$ is VII **then**

{c.f. case 2.2. In this case, the lists R_u, B_u and H_u are empty.}

begin

Let *temp=val(h)*; { *val(h)* is the value contained in h.}

Set *next(val(h)* to *tail(M_u)* and *next(head(M_u))* to *next(temp)*;

{This operation is equivalent to perform a horizontal flip of σ_u and then insert it to the next of rightmost blue hole in σ_{k-1}.}

Remove h from H_σ;

Remove u from the list of blue children of v in T;

if v has no more blue child in T **then** remove v from B_σ;

Set *Type* according to the status of R_σ, B_σ and H_σ;

end

else if $Type_u$ is VII **then**

{c.f. case 2.3. In this case, the lists R_u and H_u are empty.}

begin

Let *temp=val(h)*; { *val(h)* is the value contained in h}

Set *next(val(h)* to *tail(M_u)* and *next(head(M_u))* to *next(temp)*;

{This operation is equivalent to perform a horizontal flip of σ_u and then insert it to the next of rightmost blue hole in σ_{k-1}.}

Remove h from H_σ;

Remove u from the list of blue children of v in T;

Set *head(B_σ)* to *tail(B_u)* and *next(head(B_u))* to v;

if v has no more blue child in T **then** remove v from B_σ;

Set *Type* according to the status of R_σ, B_σ and H_σ;

end

end

if *Type* is III **then** {c.f. case 3.}

begin

Add an element to the end of M_σ; {This corresponds a blue hole.}

Add an element to the end of H_σ that stores a pointer to *tail(M_σ)*;

Let v be the vertex stored in the first element of R_σ;

{Hence the rightmost R-live point represents v.}

Let u be the first red child of v in T;

Add u to the end of M_σ;

Remove u from the list of red children of v in T;

if v has no more red child in T **then** remove v from R_σ;

if u has at least one red child in T **then** store u at the end of R_σ;

if u has at least one blue child in T **then** store u at the end of B_σ;

Set *Type* according to the status of R_σ, B_σ and B_σ;

end

if *Type* is IV **then**

{There is no vertex of G left unmapped; c.f. case 4.

Hence this instance of the procedure terminates and returns a drawing.}

begin

Return R_σ, B_σ, H_σ, M_σ and *Type*; {Here the lists R_σ, B_σ are empty.}

{We do not differentiate between the two cases as described previously,

c.f case 4.1 and 4.2 since we are only interested in the sequence of vertices of G

that represents a mapping on an alternating point-set.}

end

if *Type* is V **then** {c.f. case 5}

begin

Let v be the vertex stored in the last element of R_σ;

{Hence the rightmost R-live point represents v.}

Let u be the first red child of v in T;

Add u to the end of M_σ;

Remove u from the list of red children of v in T;

if v has no more red child in T **then** remove v from R_σ;

if u has at least one red child in T **then** store u at the end of R_σ;

if u has at least one blue child in T **then** store u at the end of B_σ;

Set *Type* according to $c(v_0)$ the status of R_σ and B_σ;

end

if *Type* is VI **then** {c.f. case 6.}

begin

if *level*= 0 **then** {c.f. case 6.1.}

begin

$T := A_G^T$;

{Hence in subsequent steps the graph inversion of G is used by the procedure.}

$R_\sigma \leftrightarrow B_\sigma$; {Since the set of R-live points become B-live points

after inversion operation and vice versa. }

{We need not to invert M_σ since the color of any point can be identified

by the color of the vertex mapped to it.}

Type:= I;

end

else {i.e. *level*= 0; c.f. case 6.2}

then Return R_σ, B_σ, H_σ, M_σ and *Type*;

{In this case, the lists B_σ and H_σ are empty.}

end

if *Type* is VII **then** {There is no vertex of G left unmapped; c.f. case 7.

Hence this instance of the procedure terminates and control goes back to caller.}

begin

Return R_σ, B_σ, H_σ, M_σ and *Type*; { Here the lists R_σ, B_σ and H_σ are empty.}

{We do not differentiate between the two cases as described previously,

c.f case 7.1 and 7.2 since we are only interested in the sequence of vertices of G

that represents a mapping on an alternating point-set.}

end

end

end.

4.1.2 Algorithm Alternating-Embedding

Given a 2-colored tree G, Algorithm **Alternating-Embedding** computes a planar drawing Γ of G such that it satisfies the following two conditions: (i) each edge of G is drawn with at most one bend, and (ii) the set of points representing the vertices of G in Γ is an alternating RB-sequence. Let σ denotes the set of points in Γ. We say that Γ represents a bichromatic point-set embedding of G on an alternating RB-sequence σ with at most one bend per edge. We assume G contains equal number of red and blue vertices. We now present Algorithm **Alternating-Embedding**.

Algorithm Alternating-Embedding(G)
{G is a 2-colored tree.}
begin
Designate any red vertex v_0 of G as root of G;
Tree-Embed(G, v_0, 0);
Let Γ be drawing computed by the procedure;
Then Γ represents the desired drawing;
end.

4.1.3 Correctness and Time Complexity

In this section, we verify the correctness and time complexity of the Algorithm **Alternating-Embedding**. We first prove the following lemma on the correctness of the Algorithm **Alternating-Embedding**.

Lemma 4.1.4 *Algorithm* **Alternating-Embedding** *computes a bichromatic point-set embedding of a 2-colored tree G on an alternating RB-sequence with at most one bend per edge. Moreover, number of points in the point-set equals $|V(G)|$.*

Proof. We first show that the Algorithm **Alternating-Embedding** terminates. Since **Alternating-Embedding** invokes Procedure **Tree-Embed**, therefore, we need to prove that **Tree-Embed** terminates. Consider the operations at some intermediate step $k(k > 0)$ inside **Tree-Embed**. The output drawing from the previous step denoted by γ_{k-1} satisfies the step invariant properties. Hence by Property 4, type of γ_{k-1} is either of types I-VII. Since we specify the next operations for each of these seven types, it implies that our case analysis is complete. When γ_{k-1} is of Type IV or Type VII i.e. there are no live points in σ_{k-1}, the procedure terminates. Now consider the cases when σ_{k-1} contains at least one live point i.e. γ_{k-1} is of Type I, II, III, V or VI. If γ_{k-1} is of Type I, III or IV, the operations specified for each of these cases embed an unmapped vertex of the input graph (c.f. case 1, case 3 and case 4). In case γ_{k-1} is of Type II, we invoke **Tree-Embed** on some connected subgraph of unmapped vertices (c.f case 2). This operation reduces the number of unmapped vertex by at least one since each invocation of **Tree-Embed** embeds at least one unmapped vertex i.e. the root vertex. When γ_k is of Type V, there are two subcases as determined by the value of *level*. If *level* $= 1$, the procedure terminates(refer to case 6.2). Otherwise we specify operations such that the resulting drawing is of Type III, which ensures that a new unmapped vertex will be mapped in the immediate next iteration. Thus after $O(V(G))$ steps, there remains no live points and the resulting drawing γ_G reduces to Type IV or VII.

Now let γ_G be output drawing when the invocation **Tree-Embed**$(G, v_0, 0)$ inside the Algorithm **Alternating-Embedding** returns. Let σ_G be the output RB-sequence. Since γ_G satisfies the step invariant properties as defined in the Procedure **Tree-Embed**, it follows that σ_G is an alternating RB-sequence(by Property 4). Moreover, γ_G represents a bichromatic point-set embedding of a connected graph G_S that contains the root v_0 of G (by Property 5). Now we need to show that G_S is the graph G i.e. the set $V(G)\backslash V(G_S) = \phi$. In other words, we need to ensure that no vertex in G is left unmapped in γ_G. Now for contradiction, assume $V(G)\backslash V(G_S) \neq \phi$. Let v be a vertex of $V(G)\backslash V(G_S)$. Since σ_G is of Type either IV or VII (when *l*evel$= 0$, these are the only two cases where the procedure terminates), if follows that σ_G has no live points. Hence there is no vertex

$u \in V(G_S)$ such that v is a neighbor of u in G; otherwise u would have been a live vertex and the point representing u would be a live point. It follows that G has more than one component which is a contradiction since G is connected. Therefore, vertices such as v cannot exist and $V(G)\backslash V(G_S) = \phi$.

Now it remains to prove that $|\sigma_G| = |V(G)|$ i.e. σ_G contains no hole (a point where no vertex of G has been mapped). There may be the following cases.

(i) Rightmost point of σ_G is red and σ_G contains no red holes: This is according to case 4.1 and when no inversion of the output drawing is required. We assume that G has equal number of red and blue vertices. Let n_{rb} denotes the number of red (blue) vertices in G. As shown previously, each vertex of G is mapped on some point of σ_G. Since σ_G contains no red holes, it follows that there are exactly n_{rb} red points in σ_G. Also σ_G must contain at least n_{rb} blue points. Therefore, σ_G may contain at most one blue hole since σ_G is an alternating RB-sequence. However in that case both the leftmost and rightmost points of σ_G must be blue. But the rightmost point of σ_G is red. Hence σ_G contains no blue hole.

(ii) Rightmost point of σ_G is blue and σ_G contains no blue holes: This is according to case 4.1 and when the output drawing is inverted. Using the same reasoning described above, it can be shown that σ_G contains no red hole either.

(iii) Rightmost point of σ_G is red and σ_G contains no holes: This is according to case 7.1.

Therefore, there are no holes in σ_G and thus $|\sigma_G| = |V(G)|$. □

We now have the following lemma on the time complexity of the Algorithm **Alternating-Embedding**

Lemma 4.1.5 *Algorithm* **Alternating-Embedding** *runs in linear time.*

Proof. Since the Algorithm **Alternating-Embedding** invokes Procedure **Tree-Embed**, we need to show that each instance of this procedure runs in $O(|V(G)|)$ time where G denotes the input graph for that instance. From the description of **Tree-Embed**, one can see that each step performs either of the following tasks. (i) Maps an unmapped vertex (c.f. case 1, case 3 and case 5); (ii) Maps a connected subgraph of G by recursive invocation (refer to case 2); (iii) Transforms the resulting drawing in such a way that ensures mapping of a new unmapped vertex in the immediate next step (refer to case 6). Thus **Tree-Embed** requires $O(|V(G)|)$ steps to compute a bichromatic point-set embedding of G. From the formal description of Procedure **Tree-Embed** in Section 4.1.1, one can readily find that for all possible cases, operations in each of the steps of **Tree-Embed** take constant time. Therefore, the Algorithm **Alternating-Embedding** runs in linear time. □

4.2 Bichromatic Point-Set embedding on Alternating Point-Set

In this section, we prove the existence of bichromatic point-set embedding of trees on alternating point-sets with at most one bend per edge. We in fact prove the following theorem.

Theorem 4.2.1 *Let $G = (V, E)$ be a 2-colored tree. Let S be a 2-colored alternating point-set compatible with G. G has a bichromatic point-set embedding on S with at most one bend per edge. Moreover, such a drawing can be computed in linear time.*

Proof. The proof is constructive. We assume the leftmost point of S be red. Using the Algorithm **Alternating-Embedding** we compute a bichromatic point-set embedding of G on some alternating RB-sequence σ; by Lemma 3.1.2 this takes linear time. Let γ denotes the drawing. It follows from the Algorithm **Alternating-Embedding** that the leftmost point of σ is red. Moreover, $|\sigma| = |V(G)|$. Since $|S|$ is compatible with G, $|S| = |V(G)|$. It follows that $|\sigma| = |S|$. Since both S and σ are alternating point-sets and the leftmost points of both S and σ are red, it follows from Observation 2.2.2 that σ is chromatic equivalent to S. Now using the technique used in the proof of Lemma 2.2.3, we compute a bichromatic point-set embedding of G on S with at most one bend per edge from bichromatic point-set embedding of G on σ and this also takes linear amount of time. Thus it requires linear time to construct a bichromatic point-set embedding of G on S. □

4.3 Summary

In this chapter, we have proved the existence of bichromatic point-set embeddings of trees on alternating point-sets with at most one bend per edge. We have described a linear-time algorithm which finds a bichromatic point-set embedding of a 2-colored tree on an alternating RB-sequence with at most one bend per edge. Then using such drawing we have shown how to construct a bichromatic point-set embedding of the given tree on any alternating point-set with at most one bend per edge in linear time.

Chapter 5

Conclusion

In this thesis, we have dealt with the problem of computing bichromatic point-set embedding of trees on consecutive and alternating point-sets with at most one bend per edge on some special configurations of point-set. Even though linear number of bends per edge is required to compute bichromatic point-set embedding of planar graphs, there are results for restricted classes of planar graphs namely path and caterpillars that allow bichromatic point-set embedding with at most one bend per edge. On the other hand, outer planar graphs admit k-chromatic point-set embedding with at most $4k + 1$ bends on a consecutive point-set. These results have motivated us to a explore a combination of these two directions i.e. to look for other larger classes of planar graphs that admit bichromatic point-set embeddings on special configurations of point-sets with at most one bend per edge. The class of planar graphs we have considered here is 'tree' which is a larger class than path and caterpillars. The contributions of this research work are listed below.

(1) We have given a linear-time algorithm for computing bichromatic point-set embedding of trees on consecutive point-sets with at most one bend per edge.

(2) We have given a linear-time algorithm for computing bichromatic point-set embedding of trees on alternating point-sets with at most one bend per edge.

Below we summarize each chapter and its contribution.

In chapter 1, we have defined point-set embedding problem and described some applications of point-set embedding. Then we have focused on the previous works in this field and justified the motivation of our work. We have then described the scope of this thesis work and the main achievements of this research work.

In chapter 2, we have given the definitions of some basic graph theoretical terminologies and terminologies regarding bichromatic point-set embedding problem. We have also discussed complexity of algorithms.

In chapter 3, we have constructively proved the existence of bichromatic point-set embeddings of trees on consecutive point-sets with at most one bend per edge. This constructive proof leads to an algorithm that computes a bichromatic point-set embedding of a 2-colored tree on a 2-colored consecutive point-set in linear time.

In chapter 4, we have constructively proved the existence of bichromatic point-set embeddings of trees on alternating point-sets with at most one bend per edge. This constructive proof leads to an algorithm that computes a bichromatic point-set embedding of a 2-colored tree on a 2-colored alternating point-set in linear time.

Due to the practical applications, the attentions of many researchers have been drawn on point-set embedding problems. But the following problems are still open relating to bichromatic as well as k-chromatic point-set embedding problem.

(1) Proving or disproving the existence of bichromatic point-set embedding of trees on general 2-colored point-set with at most one bend per edge.

(2) Finding other larger classes of outer planar graphs as well as special configurations of point-set that admit bichromatic point-set embeddings with at most one bend per edge.

(3) Exploring 3-chromtic point-set embedding problem with constant number of bends per edge for outer planar graphs.

Bibliography

[B02] P. Bose, *On embedding an Outer-Planar Graph in a Point Set*, Computational Geometry: Theory and Applications, 23(3), pp. 303-312, 2002.

[BETT99] G. D. Battista, P. Eades, R. Tamassia and I. G. Tollis, *Graph Drawing: Algorithms for the visualization of graphs*, Prentice Hall, New Jersey, 1999.

[C03] S. Cabello, *Planar embeddability of the vertices of a graph using a fixed point set is NP-Hard*, Journal of Graph Algorithms and Applications, 10(2), pp. 353-363, 2006.

[CLRS04] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms*, 2nd ed., McGraw-Hill Book Company, London, 2004-05.

[GDL06] D. Giacomo, W. Didimo, G. Liotta, H. Meijer, F. Trotta and S. K. Wismath, *k-Colored Point-Set Embeddability of Outerplanar Graphs*, Proceedings of 14th International Symposium on Graph Drawing (GD 2006), Lecture Notes in Computer Science, Springer, 4372, pp. 318-329, 2006.

[GDL07] D. Giacomo, W. Didimo, G. Liotta, H. Meijer, F. Trotta and S. K. Wismath, *Drawing Colored Graphs with Constrained Vertex Positions and Few Bends per Edge*, Proceedings of 15th International Symposium on Graph Drawing (GD 2007), Lecture Notes in Computer Science, Springer, 4875, pp. 315-326, 2007.

[GLT06] D. Giacomo, G. Liotta and F. Trotta, *On Embedding a Graph on Two Sets of Points*, International Journal of Foundations of Computer Science, 17(5), pp. 1071-1094, 2006.

[KW02] M. Kaufmann and R. Wiese, *Embedding vertices at Points: Few Bends suffice for Planar Graphs*, Journal of Graph Algorithms and Applications, 6(1), pp. 115-129, 2002.

[NR04] T. Nishizeki and M. S. Rahman, *Planar Graph Drawing*, World Scientific, Singapore, 2004.

[PW01] J. Pach and R. Wenger, *Embedding Planar Graphs at Fixed Vertex Locations*, Graphs and Combinatorics, 17(4), pp. 717-728, 2001.

[S02] K. Suhiyama, *Graph Drawing and Applications for Software and Knowledge Engineers*, World Scientific Publisher, 2002.

[SY02] S. M. Sait and H. Youssef, *VLSI Physical Design and Automation, Theory and Practice*, World Scientific Publisher, 2002.

[We01] D. B. West, *Introduction to Graph Theory*, Prentice-Hall, Inc., New Jersey, 2001.

Index