M.Sc. Engg. Thesis

Minimum Segment Drawings of Series-Parallel Graphs with the Maximum Degree Three

by

Md. Abul Hassan Samee

Submitted to

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

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

July 2008

The thesis titled "**Minimum Segment Drawings of Series-Parallel Graphs with the Maximum Degree Three**", submitted by Md. Abul Hassan Samee, Roll No. 100605035P, Session October 2006, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering and approved as to its style and contents. Examination held on July 5, 2008.

Board of Examiners

Candidate's Declaration

It is hereby declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

Md. Abul Hassan Samee

Md. Abul Hassan Samee
Candidate

Contents

List of Figures

List of Tables

Acknowledgements

First of all, I would like to thank my supervisor Prof. Dr. Md. Saidur Rahman for introducing me in the beautiful field of graph drawing long before this thesis had begun. It is his supervision that has laid down the ground for me to build a research career in the academia. I acknowledge to him for teaching me how to carry on a research work, for providing me with numerous resources and giving useful directions in preparing my presentations. I express my deep gratitude for his patience in reading my numerous inferior drafts, constant supervision, valuable advice and continual encouragement, without which this thesis would have not been possible.

My special thanks goes to all other members of the our Graph Drawing Research Group; in particular, to Md. Jawaherul Alam and Muhammad Abdullah Adnan. My parents have given their best support to me throughout my work, I convey my deep reverence to them also.

I would also like to thank Prof. Dr. M. Kaykobad for his inspirations throughout my career, and as an examiner of this thesis. My heartfelt gratitude goes to all other respected members of the board of examiners: Dr. Mahmuda Naznin, Dr. M. Sohel Rahman, and Prof. Dr. Md. Rafiqul Islam, for their valuable suggestions, advice and corrections.

Finally, every honor and every victory on earth is due to Allah, descended from Him and must be ascribed to Him. He has endowed me with good health and with the capability to complete this work. I convey my utmost praise to Him for letting me the opportunity to submit this thesis.

Abstract

This thesis deals with minimum segment drawings of planar graphs. A minimum segment drawing of a planar graph G is a planar straight line drawing of G that has the minimum number of line segments among all possible planar straight line drawings of G. Computation of minimum segment drawings has recently drawn much attention among the graph drawing community. The problem is not only important for visualization applications, but has also given rise to a number of interesting theoretical problems. Unfortunately, very few significant results regarding this problem are known to date. For example, although there is an algorithm for computing minimum segment drawings of trees, no such algorithm is known for biconnected and triconnected planar graphs. The problem has also been studied for plane graphs. Although most graph drawing problems are known to be easy for plane graphs, the minimum segment drawing problem does not exhibit this trait. Even for degree-restricted cases of plane graphs, e.g., for plane triconnected cubic graphs, it has not been possible yet to give an algorithm for computing minimum segment drawings. Series-parallel graphs are well-known planar graphs that often arise in various problems involving scheduling, electrical networks, data-flow analysis, and database logic programs. In this thesis, we study the minimum segment drawing problem for biconnected series-parallel graphs with the maximum degree three. For such a graph G, we give a linear-time algorithm for choosing a suitable embedding of G that admits a minimum segment drawing, and a linear-time algorithm for computing a minimum segment drawing of G.

Chapter 1

Introduction

Identifying a set of related entities and capturing the structure of their relationship form the core of a large number of practical problems. Whenever we are working with a network of interconnected computers, an electrical circuit with the components and their interconnections, a network of roads and highways or an organic molecule, we first have to obtain a suitable model of the involved entities and their inherent relationship. Such relational structures among various entities are also encountered in less tangible application domains like ecosystems and sociological relationships. When such problems arising in different application domains are studied in computer science, the involved entities and their relationships are modeled by a combinatorial structure known as *graph.* A graph consists of two sets, namely the set of *vertices* and the set of *edges.* An entity is modeled as a vertex of the graph and the relationship among two entities is modeled as an edge of the graph. A graph can also capture various attributes of the vertices and the edges in a model. These attributes are often used in optimization algorithms applied on the model. But the most significant information that a graph represents is the relationship between different entities. Apart from using a graph model as an input to some sort of optimization algorithm, we are often interested to visualize these existing relationships between the entities, or to make different interpretations of the relationships and produce visualizations according to those interpretations.

Graph Drawing is a relatively new area in computer science. The focus of graph drawing is to produce a desired visualization of a graph such that the graph and its contents (*i.e.*, the entities and their relationships) are easily understandable. More precisely, graph drawing addresses the problem of constructing geometric representations of graphs, and in most of the cases, these geometric representations insist on some predefined geometric or aesthetic properties. There is no unique way of drawing a graph. In Fig. 1.1 we show four drawings of the same graph. Depending on the application, different drawings of the

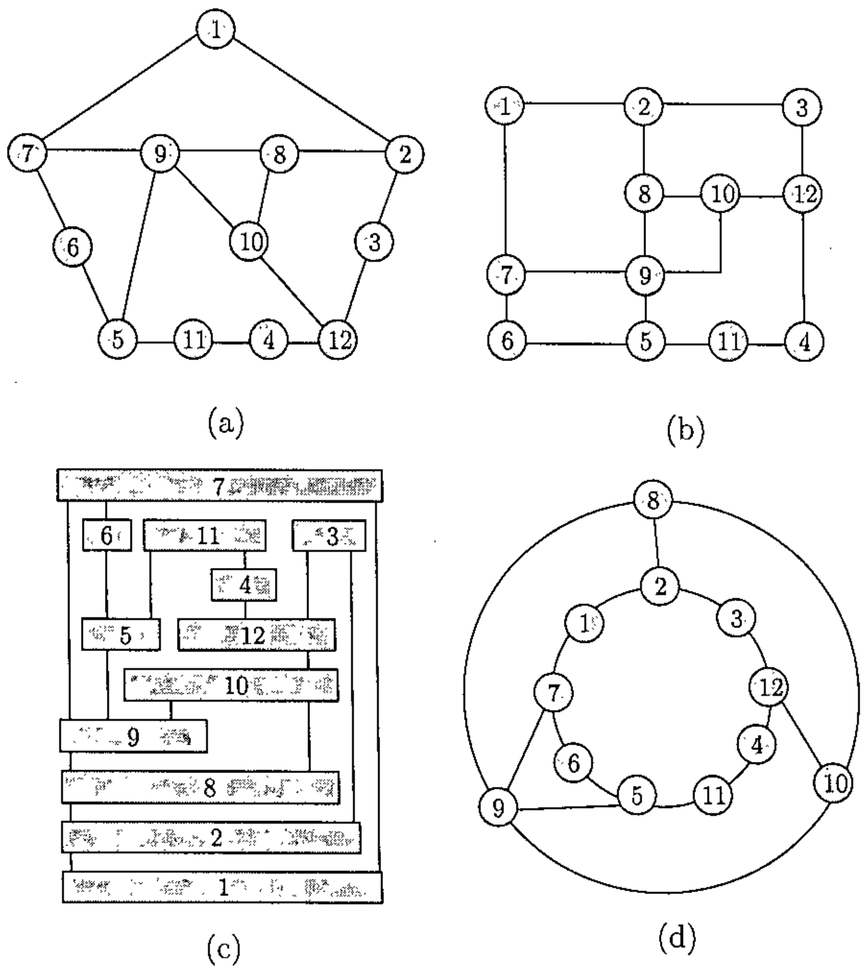

Figure 1.1: Different drawings of a graph.

same graph may be desirable. Unfortunately, not all drawings of a graph can be produced efficiently. Graph drawing researchers have to deal with enormous challenges regarding both the time complexity of their drawing algorithms and the area requirement of the drawings that they produce.

Although it is a relatively new area of research, the blend of graph theory and computational geometry has attracted much attention for graph drawing. Today graph drawing has become a key component of support tools for complex applications in science and engineering. In the following we list some interesting applications of graph drawing.

Graphs are often used to model the hierarchical relationship between objects. Such models are usually visualized by drawing every edge as a simple curve monotone in the upward (*i.e.*, vertical) direction. In Fig. 1.2 we show such a drawing of a graph that

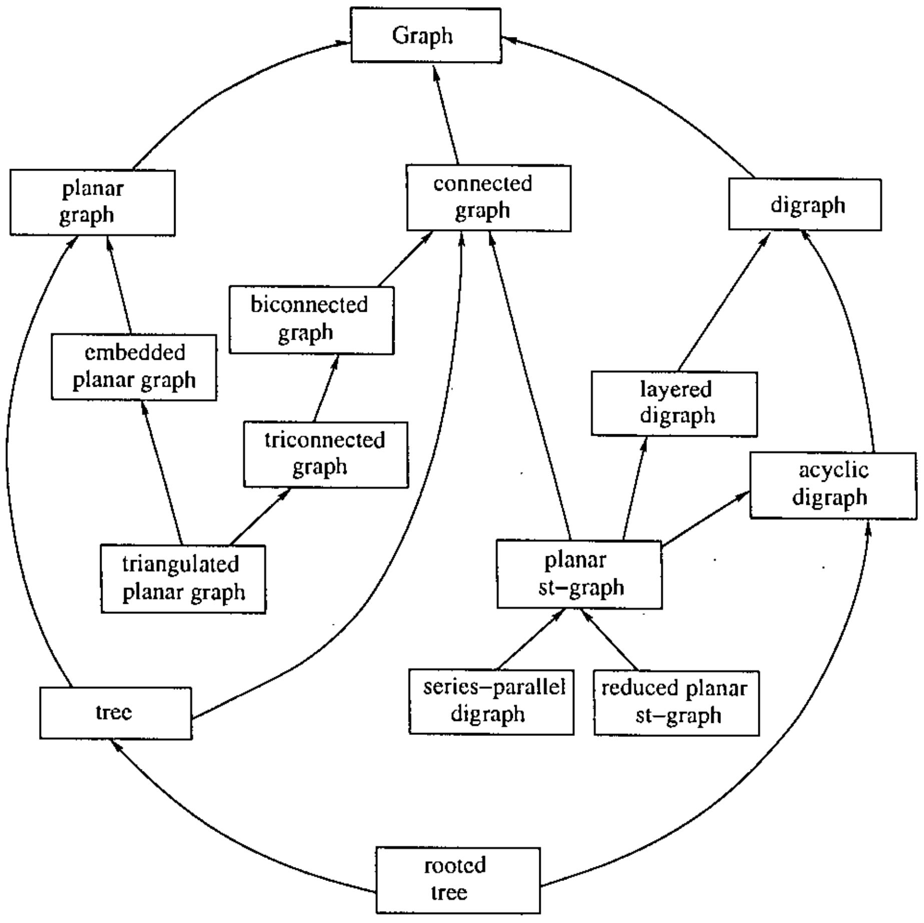

Figure 1.2: A drawing of the hierarchy of graph classes.

models the hierarchy of different graph classes. The model is significant because these are the classes of graphs that one mostly encounters in graph drawing.

Graph drawing also aids in understanding the working principles of different protocols and security models in computer systems and networks. For example, the Bell-La Padula multilevel security model imposes that a process running at security level k can read from objects at its own level or lower and can write to objects at its own level or higher. Viewed as a graph drawing problem, we simply need that no edge is drawn downward. The model can be understood from its drawing in Fig. 1.3.

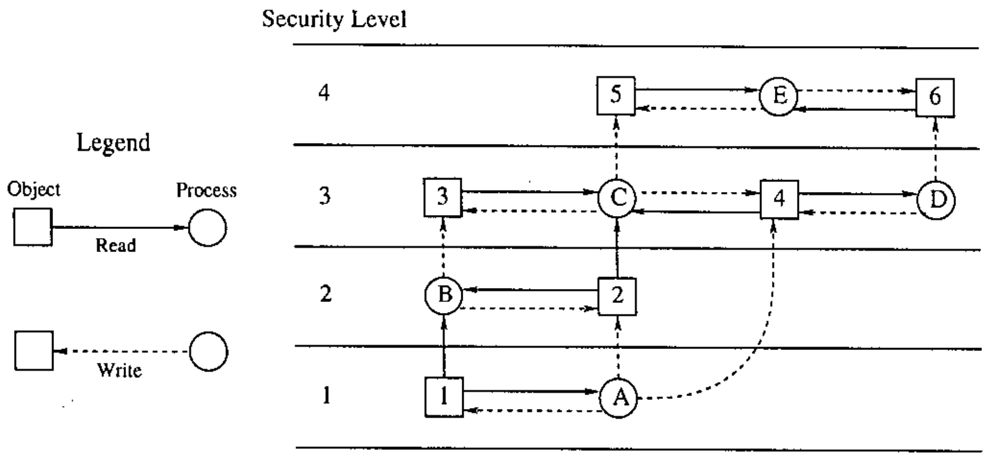

Figure 1.3: The Bell-La Padula multilevel security model.

Another important application of graph drawing algorithms is in designing layouts of electronic circuits. In Fig. 1.4(a) we show the components and their interconnections in a circuit. The representation in Fig. 1.4(a) is cumbersome. Moreover, this representation cannot be converted into a PCB layout because of the existing edge crossings. Figure 1.4(b) shows the same circuit but contains no edge crossings and is obviously more easy to trace. Above all, the representation in Fig. 1.4(b) can be easily used for producing a PCB layout.

Automatic graph drawings have numerous applications other than the above mentioned three examples. Graph drawing algorithms have important applications to key computer technologies such as computer networks (depicting the structure of the physical

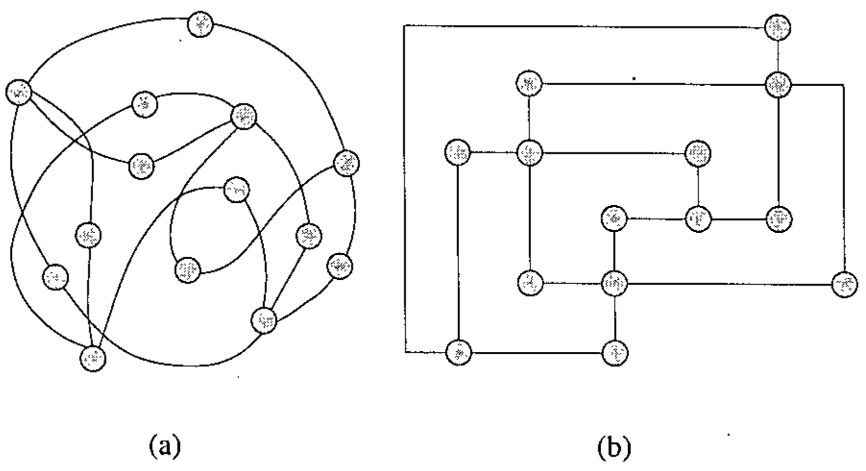
Figure 1.4: (a) An electronic circuit and (b) a possible layout of the circuit in (a).

network), software engineering (data-flow diagrams, subroutine-call graphs, program nesting trees), databases (entity-relationship diagrams), information systems (organization charts), real-time systems (state transition diagrams), artificial intelligence (knowledge-representation diagrams) etc. Further applications can be found in other science and engineering disciplines, such as medical science (concept lattices), chemistry (molecular drawings), civil engineering (floorplan maps) and cartography (map schematics) [Rah99].

In the rest of this chapter, we provide the necessary background and objectives for this study on graph drawing. In Section 1.1 we describe some of the most important graph drawing styles that have drawn much attention in the graph drawing community. As stated earlier, graph drawing algorithms are often motivated by different aesthetic criteria imposed by the application. We discuss some such aesthetics in Section 1.2. In Section 1.3 we discuss the applications of the problem addressed in this thesis, namely the minimum segment drawing problem. We give the problem statement in Section 1.4 and detail the objective of this thesis in Section 1.5. Finally, Section 1.6 is a summary of this work and Section 1.7 is the description of the organization of this thesis.

1.1 Drawing Conventions

In this section we introduce some well-known graph drawing conventions. Different drawing conventions are found suitable in different application domains. Depending on the purpose and objective, the vertices are typically represented with points or boxes and the edges are represented with simple Jordan curves. Some of the most important drawing conventions are introduced below. Interested readers are referred to [NR04].

1.1.1 Planar Drawing

In a *planar drawing* of a graph, the simple curves representing the edges do not intersect one another. It has been shown empirically in [Pur97] that the presence of edge-crossings in a drawing of a graph make it more difficult for a person to understand the information being modeled. In Fig. 1.5(a) and (b) we show a planar and a non-planar drawing of the same graph. Unfortunately, not all graphs have a planar drawing. Figure 1.5(c) is

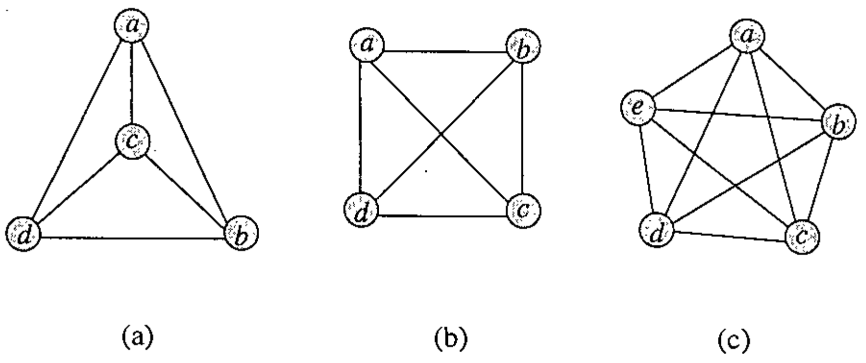

Figure 1.5: (a) A planar drawing of a graph, (b) a non-planar drawing of the graph drawn in (a), and (c) a graph that does not have a planar drawing.

an example of one such graph. A graph which has a planar drawing is called a *planar graph*. A planar graph has a planar embedding, which is a data structure representing the adjacency lists: in each list, the edges incident to a vertex are ordered, all clockwise or all counter-clockwise. A *plane graph* is a planar graph with a given planar embedding.

If one wants to find a planar drawing of a given graph, he/she first needs to test the planarity of the graph. Kuratowski [Kur30] gave the first complete characterization of planar graphs. Unfortunately, his characterization did not yield an efficient algorithm. Later, linear-time algorithms were presented by Hopcroft and Tarjan [HT74], and Booth and Lueker [BL76]. Chiba *et al.* [CNAO85] and Mehlhorn and Mutzel [MM96] gave linear-time algorithms for finding a planar embedding of a planar graph. Shih and Hsu [SH99] gave a simple linear-time algorithm which performs planarity testing and finds a planar embedding of a planar graph simultaneously.

1.1.2 Polyline Drawing

In a *polyline drawing* of a graph each edge is represented by a polygonal chain. An example of polyline drawing is shown in Fig. 1.6. A point on the polyline where the edge changes

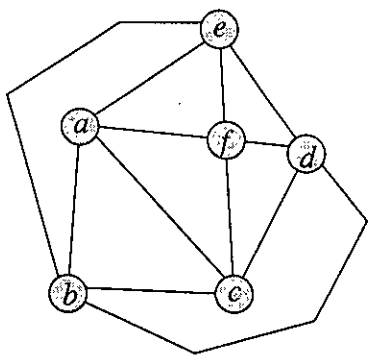

Figure 1.6: A polyline drawing.

direction is called a *bend*. Although polyline drawing has the flexibility of approximating a drawing with curved edges, too many bends on the edges (more than two or three) make it difficult to follow the edges by the eye [NR04].

1.1.3 Straight Line Drawing

A *straight line drawing* is a drawing of a graph in which each edge is drawn as a straight line segment, as illustrated in Fig. 1.7. Clearly, straight line drawing is a special case of polyline drawing where the edges are drawn without any bend. Wagner [Wag36], Fary

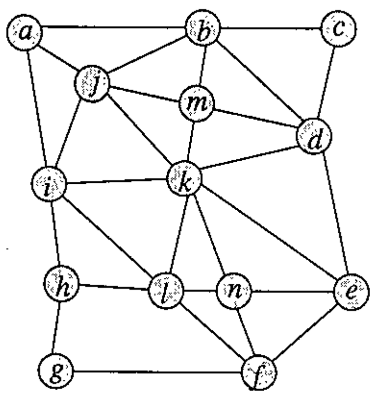

Figure 1.7: A straight line drawing.

[Far48] and Stein [Ste51] independently proved that every planar graph has a straight line drawing. A significant number of works done on straight line drawings are mentioned in [DETT94].

1.1.4 Minimum Segment Drawing

As discussed above, a straight line drawing Γ of a planar graph G is a planar drawing where each vertex u of G is mapped to a point $p(u)$ in the plane and each edge $e = (u, v)$ of G is drawn as a straight line segment $l(e)$ closed between the points $p(u)$ and $p(v)$. A line segment L in a straight line drawing Γ of G is said to be a *maximal line segment* in Γ if L is formed by a maximal set of line segments $l_1, l_2, \ldots, l_k$ where each pair of line segments l_i and l_{i+1} have a common end point in Γ $(0 < i < k)$. Typically, the term *segment* is used in the literature [DESW07] to refer to a maximal line segment in a straight line drawing of a planar graph. A straight line drawing Γ of a planar graph G is called a *minimum segment drawing* of G if Γ has the minimum number of segments among all possible straight line drawings of G. For example, the graph G shown in Fig. 1.8(a) can be drawn with seven segments as shown in Fig. 1.8(b). Another drawing of G with five segments is shown in Fig. 1.8(c). One can easily verify that if we do not change

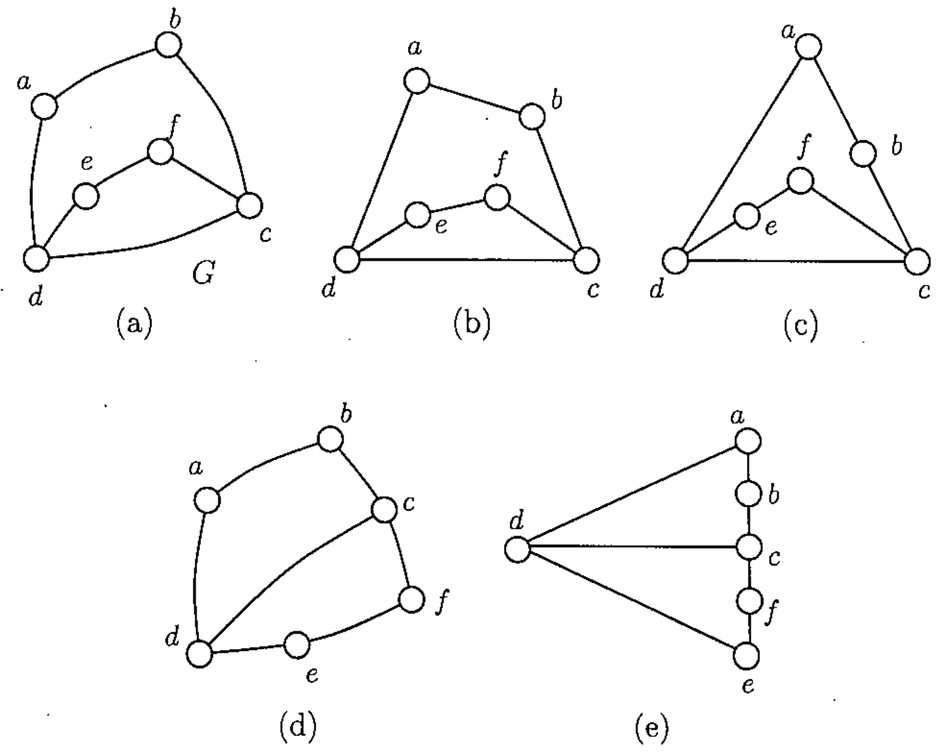

Figure 1.8: (a) The graph G, (b) a drawing of G on seven segments, (c) a drawing of G on five segments, (d) another embedding of G, and (e) a minimum segment drawing of G.

the embedding of G in Fig. 1.8(a), then it is not possible to draw G with less than five segments. However, if we alternate the embedding of G and consider the one shown in Fig. 1.8(d), then we can draw G with four segments as shown in Fig. 1.8(e). One can also verify that, it is not possible to draw any embedding of G with less than four segments. Thus, the drawing of G as shown in Fig. 1.8(e) is a minimum segment drawing of G.

1.1.5 Grid Drawing

A drawing of a graph is called a *grid drawing* if the vertices and bends are all located at grid points of an integer grid as illustrated in Fig. 1.9. This drawing approach overcomes

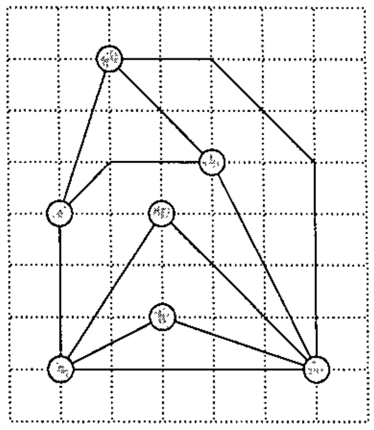

Figure 1.9: A grid drawing.

the following problems in graph drawing with real number arithmetic [NR04].

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

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

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

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

It is a very challenging problem to draw a plane graph on a grid of the minimum size. In recent years, several works are devoted to this field [Cha08, CN98, DF05, FPP90, KR07, Sch90]; for example, it has been shown that, every plane graph of n vertices has a straight line grid drawing on a grid size $W \times H \leq (n-2) \times (n-2)$ [Sch90]; also, special graph classes have been identified that admits drawing in $O(n)$ area [DF05, KR07].

1.2 Drawing Aesthetics

Aesthetics specifies graphic properties of the drawing that one wants to be applied *as much as possible* to achieve better readability [DETT99]. There have been a number of studies on some commonly adopted aesthetics [BFN85, DESW07, PCJ96, STT81]. We describe here some of the most important aesthetics [DETT99, NR04].

Crossings: Minimization of the total number of edge crossings is desired. Ideally, one expects to obtain a crossing-free drawing, but not every graph admits one.

Area: Minimization of the area of the drawing is desired. In practical visualization applications it is important to construct area-efficient drawings because saving screen-space is a major concern.

Segment count: It is often desired that a straight line drawing should use as few number of segments as possible. This aesthetic has twofold importance in practical applications. Firstly, it increases the readability of the drawing and secondly, it makes the rendering of the drawing faster since in this case, the scan conversion algorithm for raster devices is applied on fewer number of line segments.

Bends: Minimization of the total number of bends along the edges is desired. This aesthetic is especially important for polyline drawings, while it is trivially satisfied by straight line drawings.

Angular Resolution: Maximization of the smallest angle between two edges incident on the same vertex is desired. This aesthetic is especially meaningful for straight line drawings.

Aspect Ratio: Aspect ratio is defined as the ratio of the length of the longest side to the length of the shortest side of the smallest rectangle with horizontal and vertical sides covering the drawing. Minimization of aspect ratio is a desired aesthetic because a drawing with a moderate area but high aspect ratio can still remain problematic.

Symmetry: It is desired to display the symmetries of the graph in the drawing. This aesthetic can be further formalized by introducing a mathematical model of symmetries in graphs and their drawings (see, e.g., [Ead88, LNS85, MACLP95]).

Achieving a drawing with desired aesthetics is often associated with optimization problems and many of these problems are computationally hard. For example, Garey and Johnson showed that the minimization of the number of crossings in a graph is NP-complete [GJ83]. To determine whether a graph can be embedded in a grid of a given size is also NP-complete [KL84]. Garg and Tamassia proved the problem of minimizing the number of bends of orthogonal drawings (a special class of polyline drawings) to be NP-complete [GT01]. Computing minimum segment drawings has also been found to be quite challenging [DESW07]. The only known algorithm for computing minimum segment drawings works for trees [DESW07], but no such algorithm has been given so far for other important graph classes. For these reasons, finding useful approximation strategies and heuristics for achieving various aesthetics [DETT99] is also an active area of research.

1.3 Applications of Minimum Segment Drawings

The problem of computing minimum segment drawings has mostly attracted the graph drawing community due to its theoretical elegance [DSW05, KPPT06, DESW07]. Apart from theoretical interests, the problem has also several practical applications as discussed below. Although planar straight line drawings are considered as the best means for visualizing planar graphs [Pur97], minimization of the number of segments in these drawings can greatly enhance the overall readability [DESW07]. On the other hand, fewer number of segments in the drawing often implies fewer number of slopes in the drawing [DESW07]. Both these characteristics have important effects on scan conversion algorithms for lines in raster devices. In raster devices, the grid location of each pixel has to be computed separately. Moreover, this computation is largely dependent on the slope of the segment [FDFH03]. If both the number of segments and the number of slopes in the drawing are small, then these computations can be performed faster yielding a faster rendering of the drawing.

1.4 Problem Statement

The problem of computing straight line drawings of planar graphs has been studied for long with various application specific objectives in the focus [Far48, FPP90, Pur97, Sch90, Ste51, Wag36]. Recently, Dujmović *et al.* have studied this problem with the new objective of minimizing the number of segments in a drawing [DESW07], and the insightful results presented in their work have established a new line of research henceforth. However, as their results suggest, this problem is quite difficult for most of the non-trivial graph classes. For most of the cases, bounds have been given on the number of segments in a drawing, but no algorithm is known for computing a minimum segment drawing. For example, although Dujmović *et al.* have provided an algorithm for computing minimum segment drawings of trees, no such algorithm is known for biconnected and triconnected planar graphs. The problem has also been studied for plane graphs, *i.e.*, for the case where one is given a fixed embedding of a graph and is not allowed to change it [DESW07]. Although dealing with plane graphs is typically easier than dealing with planar graphs, no algorithm is known for computing minimum segment drawings of biconnected and triconnected plane graphs as well. Even for degree restricted cases of plane graphs, e.g., for plane triconnected cubic graphs, no algorithm has yet been devised for computing minimum segment drawings.

1.5 Objective of this thesis

In this thesis, we study the minimum segment drawing problem for series-parallel graphs with the maximum degree three. Even in this degree restricted setting, the work is quite challenging due to the following facts. Firstly, we have dealt here with planar graphs, *i.e.*, the case where one has to choose a suitable embedding among all possible planar embeddings of the graph. Since we have dealt with planar graphs, the number of possible planar embeddings of such graphs is exponential in the number of vertices of the graph. Therefore, the first challenging issue in this thesis is to compute a suitable embedding of the graph out of the exponential number of possibilities. Secondly, as earlier results suggest, it is not so easy to compute minimum segment drawings of plane graphs even in the degree restricted case. After computing a suitable embedding of the graph, our second challenging issue is to compute a minimum segment drawing of that embedding - a task which was not possible for the graph classes studied earlier in the literature.

1.6 Summary of Results

In this thesis, we study the minimum segment drawing problem for series-parallel graphs with the maximum degree three. For a biconnected series-parallel graph G with the maximum degree three, we give linear-time algorithms for choosing such an embedding of G that admits a planar straight line drawing on the minimum number of segments, and for computing a minimum segment drawing of G. We also give a tight lower bound on the number of segments in planar straight line drawings of G. Our result together with some previous results are listed in Table 1.1. Meanings of the notations used in this table are as follows. The symbol η denotes the number of odd degree vertices in a tree. The symbol n denotes the number of vertices in a graph. For a biconnected series-parallel graph G with the maximum degree $\Delta = 3$, the symbols P and N respectively denote the number of P-nodes and the number of primitive P-nodes in an SPQ-tree of G, and $k \in \{1, 2\}$

Graph class	Bound on segments		Minimum Segment Drawing Algorithm	Reference
	Lower	Upper		
Tree	$\frac{\eta}{2}$	$\frac{\eta}{2}$	Yes	[DESW07]
Plane 2-connected	$\frac{5}{2}n$	–	No	
Planar 2-connected	$2n$	–	No	
Plane 3-connected	$2n$	$\frac{5}{2}n$	No	
Planar 3-connected	$2n$	$\frac{5}{2}n$	No	
Plane 3-connected cubic	–	$n+2$	No	
Biconnected Series-parallel ($\Delta = 3$)	$P+N+k$	$P+N+k$	Yes	**Ours**

Table 1.1: Known results for the minimum segment drawing problem

based on a characterization of the SPQ-tree of G. An SPQ-tree is a data structure for decomposing G. We have given a detail discussion on SPQ-trees in Chapter 2. In short, an SPQ-tree consists of three types of nodes, namely P-nodes, S-nodes and Q-nodes. A P-node in an SPQ-tree can be further classified as a "primitive" P-node or a "non-primitive" P-node. The lower bound presented in this work on the number of segments in any planar straight line drawing of G is given in terms of the total number of P-nodes and the total number of primitive P-nodes in an SPQ-tree of G.

1.7 Thesis Organization

The rest of this thesis is organized as follows. In Chapter 2 we discuss the relevant ideas and necessary definitions from graph theory and algorithm theory. We also give an overview of the literature on the minimum segment drawing problem in Chapter 2. In Chapter 3 we present a linear-time algorithm for computing minimum segment drawings of biconnected series-parallel graphs with the maximum degree three. Finally, Chapter 4 is a conclusion.

Chapter 2

Preliminaries

In this chapter we define some basic terminology of graph theory and algorithm theory. Definitions that are not included in this chapter will be introduced as they are needed. We start, in Section 2.1, by giving some definitions of standard graph-theoretical terms used throughout the remainder of this thesis. We devote Section 2.2 to define terms related to planar graphs. In Section 2.3 we discuss a particular class of graphs known as series-parallel graphs. Our algorithms introduced in Chapter 3 are designed to work on biconnected series-parallel graphs. Section 2.4 discusses SPQ-trees, the data structure that we have used for representing a decomposition of a biconnected series-parallel graph. The notion of time complexity is discussed in Section 2.5. Finally we give a review of the literature on the minimum segment drawing problem in Section 2.6.

2.1 Basic Terminology

In this section we give some definitions of standard graph-theoretical terms used throughout this thesis. For readers interested in graph theory we refer to [Wes01].

2.1.1 Graphs and Multigraphs

A *graph* G is a structure (V, E) which consists of a finite set of *vertices* V and a finite set of *edges* E; each edge is an unordered pair of vertices. The sets of vertices and edges of G are denoted by $V(G)$ and $E(G)$ respectively. Fig. 2.1 depicts a graph G where

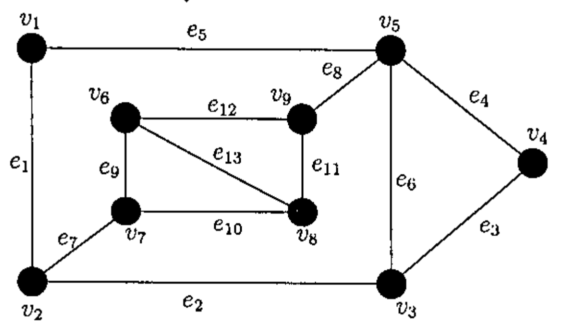

Figure 2.1: A graph with nine vertices and thirteen edges.

each vertex in $V(G) = \{v_1, v_2, \ldots, v_9\}$ is drawn as a small dark circle and each edge in $E(G) = \{e_1, e_2, \ldots, e_{13}\}$ is drawn by a line segment.

If a graph G has no "multiple edges" or "loops", then G is said to be a *simple graph. Multiple edges* join the same pair of vertices, while a *loop* joins a vertex with itself. A graph in which loops and multiple edges are allowed is called a *multigraph.* Often it is clear from the context that the graph is simple. In such cases, a simple graph is called a *graph.* In the remainder of thesis we assume that G has no loop.

We denote an edge between two vertices u and v of G by (u, v). If $(u, v) \in E$ then two vertices u and v of graph G are said to be *adjacent*; edge (u, v) is then said to be *incident* to vertices u and v; u is a neighbor of v. The *degree* of a vertex v in G, denoted by $d(v)$ is the number of edges incident to v. In the graph shown in Fig. 2.1 vertices v_1 and v_2 are adjacent, and $d(v_5) = 4$, since four of the edges, namely e_4, e_5, e_6 and e_8 are incident to v_5.

2.1.2 Subgraphs

A *subgraph* of a graph $G = (V, E)$ is a graph $G' = (V', E')$ such that $V' \subseteq V$ and $E' \subseteq E$; we then write $G' \subseteq G$. If G' contains all the edges of G that join two vertices in V', then G' is said to be the *subgraph induced by* V', and is denoted by $G[V']$. Fig. 2.2 depicts a subgraph of G in Fig. 2.1 induced by $\{v_3, v_4, v_5, v_6, v_8, v_9\}$.

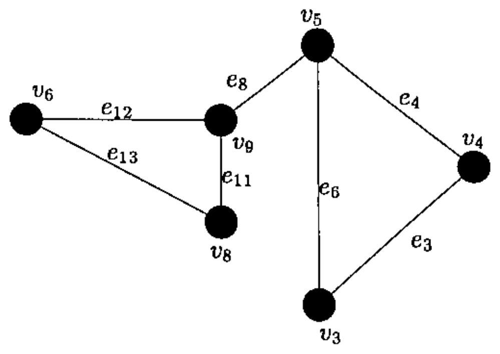

Figure 2.2: A vertex-induced subgraph.

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

2.1.3 Connectivity

A graph G is a *connected graph* if for every pair $\{u, v\}$ of distinct vertices there is a path between u and v. A graph which is not connected is called a *disconnected graph.* A *(connected) component* of a graph is a maximal connected subgraph. The graph in

Fig. 2.3(a) is a connected graph since there is a path for every pair of distinct vertices of the graph. On the other hand the graph in Fig. 2.3(b) is a disconnected graph since there is no path between v_1 and v_{10}. The graph in Fig. 2.3(b) has three connected components G_1, G_2 and G_3 indicated by dotted lines.

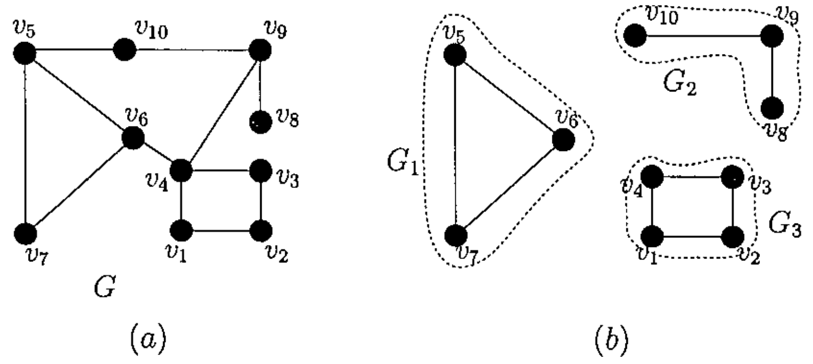

Figure 2.3: (a) A connected graph and (b) a disconnected graph with three connected components.

The *connectivity* $\kappa(G)$ of a graph G is the minimum number of vertices whose removal results in a disconnected graph or a single-vertex graph K_1. We say that G is *k-connected* if $\kappa(G) \geq k$. We call a set of vertices in a connected graph G a *separator* or a *vertex cut* if the removal of the vertices in the set results in a disconnected or single-vertex graph. If a vertex-cut contains exactly one vertex then we call the vertex a *cut vertex*. A *block* is a maximal biconnected subgraph of G.

2.1.4 Paths and Cycles

A v_0, v_l walk, $v_0, e_1, v_1, \ldots, v_{l-1}, e_l, v_l$, in G is an alternating sequence of vertices and edges of G, beginning and ending with a vertex, in which each edge is incident to the two vertices immediately preceding and following it. If the vertices $v_0, v_1, \ldots, v_l$ are distinct (except possibly v_0, v_l), then the walk is called a *path* and usually denoted either by the sequence of vertices $v_0, v_1, \ldots, v_l$ or by the sequence of edges $e_1, e_2, \ldots, e_l$. The length of the path is l, one less than the number of vertices on the path. A path or walk is *closed* if $v_0 = v_l$. A closed path containing at least one edge is called a *cycle.*

2.1.5 Trees

A *tree* is a connected graph without any cycle. Fig. 2.4 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 the other nodes. The distinguished node is called the *root* of the tree. The root of a tree is generally drawn at the top. In Fig. 2.4, the root is v_1. Every node u other than the root is connected by an edge to some other node p called the *parent*

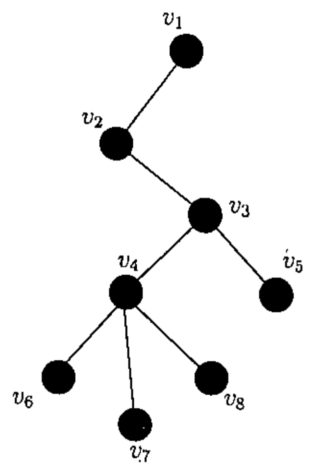

Figure 2.4: Example of a tree.

of u. We also call u a *child* of p. We draw the parent of a node above that node. For example, in Fig. 2.4, v_1 is the parent of v_2, while v_4 is the parent of v_6, v_7 and v_8; v_4 and v_5 are children of v_3. A *leaf* is a node of a tree that has no children. An *internal node* is a node that has one or more children. Thus every node of a tree is either a leaf or an internal node. In Fig. 2.4, the leaves are v_5, v_6, v_7 and v_8, and the nodes v_1, v_2, v_3 and v_4 are internal nodes.

The parent-child relationship can be extended naturally to ancestors and descendants. Suppose that $u_1, u_2, \ldots, u_l$ is a sequence of nodes in a tree such that u_1 is the parent of

u_2, which is a parent of u_3, and so on. Then node u_1 is called an *ancestor* of u_l and node u_l a *descendant* of u_1. The root is an ancestor of every node in a tree and every node is a descendant of the root. In Fig. 2.4, all eight nodes are descendants of v_1, and v_1 is an ancestor of all nodes. The *height* of a node u in a tree is the length of a longest path from u to a leaf. The *height of a tree* is the height of the root.

2.2 Planar Graphs and Plane Graphs

In this section we give some definitions related to planar graphs used in the remainder of the thesis. For readers interested in planar graphs we refer to [NC88].

A graph is a *planar graph* if it can be embedded in the plane so that no two edges intersect geometrically except at a vertex to which they are both incident. Note that a planar graph may have an exponential number of embeddings. Fig. 2.5 shows four planar embeddings of the same planar graph.

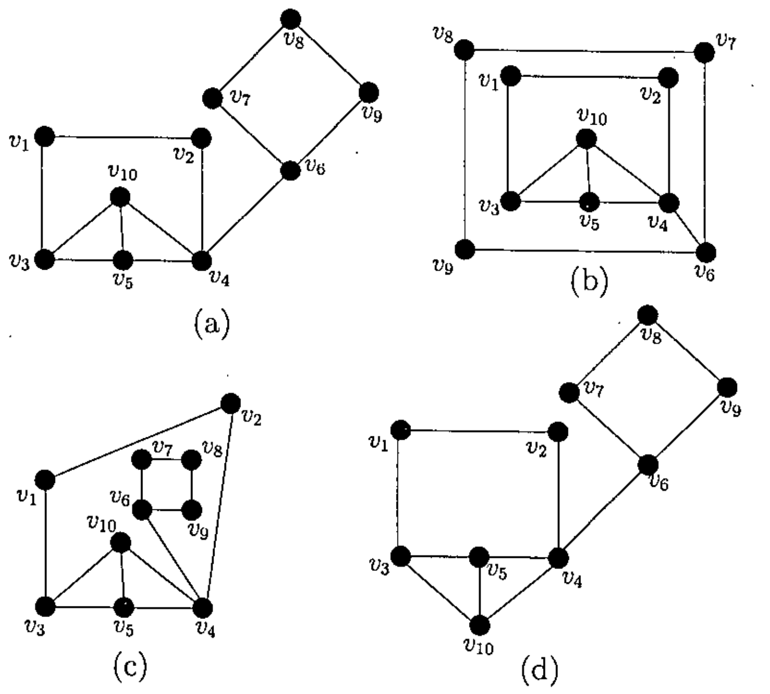

Figure 2.5: Four planar embeddings of the same planar graph.

A *plane graph* is a planar graph with a fixed embedding. A plane graph divides the plane into connected regions called *faces*. We regard the contour of a face as a clockwise cycle formed by the edges on the boundary of the face. We denote the contour of the outer face of graph G by $C_o(G)$. A cycle of a plane graph is called a *facial cycle* if it is the boundary of a face f and denoted by C_f.

2.3 Series-Parallel Graphs

A graph $G = (V, E)$ is called a *series-parallel graph* (with source s and sink t) if either G consist of a pair of vertices connected by a single edge as illustrated in Fig. 2.6(a), or there exist two series-parallel graphs $G_i = (V_i, E_i), i = 1, 2$, with source s_i and sink t_i such that $V = V_1 \cup V_2, E = E_1 \cup E_2$, and either $s = s_1$, $t_1 = s_2$ and $t = t_2$ as illustrated in Fig. 2.6(b) or $s = s_1 = s_2$ and $t = t_1 = t_2$ as illustrated in Fig. 2.6(c) [REN05]. Fig. 2.6(d) illustrates a series-parallel graph.

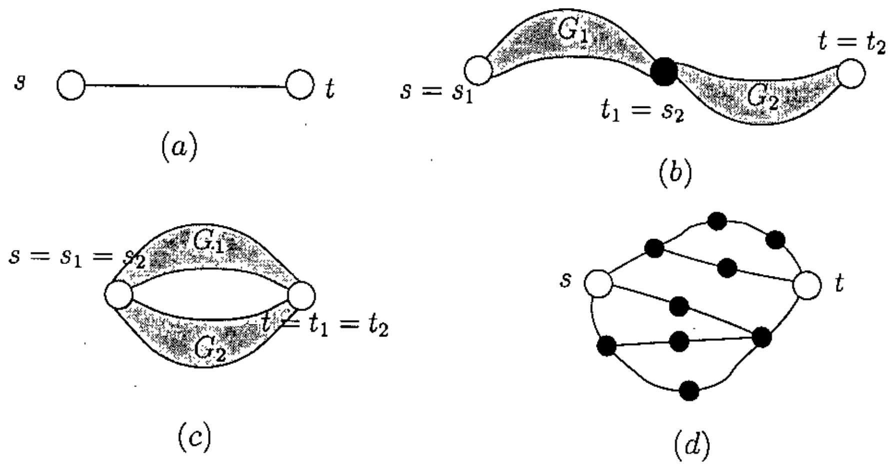

Figure 2.6: (a) A basic series-parallel graph, (b) series connection, (c) parallel connection, and (d) a series-parallel graph.

A pair $\{u, v\}$ of vertices of a connected graph G is a *split pair* if there exist two subgraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ satisfying the following two conditions:

1. $V = V_1 \cup V_2$, $V_1 \cap V_2 = \{u, v\}$; and

2. $E = E_1 \cup E_2$, $E_1 \cap E_2 = \emptyset$, $|E_1| \geq 1$, $|E_2| \geq 1$.

Thus every pair of adjacent vertices is a split pair. For the graph in Fig. 2.7, the split pairs are $\{c, f\}$, $\{c, g\}$, $\{d, f\}$, $\{d, g\}$ and the pairs of adjacent vertices. A *split component* of a split pair $\{u, v\}$ is either an edge (u, v) or a maximal connected subgraph H of G such that $\{u, v\}$ is not a split pair of H. There are two split components of split pair $\{c, g\}$ for the graph G in Fig. 2.7; one is the subgraph of G induced by vertices a, b, c, g, h and i; and the other by c, d, e, f, g and j. A split pair $\{u, v\}$ of G is called a *maximal split pair* with respect to a *reference split pair* $\{s, t\}$ if, for any other split pair $\{u', v'\}$, vertices s, t, u and v are in the same split component of $\{u', v'\}$. In Fig. 2.7, the maximal split pairs with respect to reference edge (a, b) are the pair $\{c, g\}$ and the pairs $\{a, i\}$, $\{a, g\}$, $\{b, i\}$, $\{b, c\}$, $\{c, h\}$, $\{h, i\}$, $\{g, h\}$ of adjacent vertices. The split pair $\{d, f\}$ is not a maximal split pair, because vertices a, b, d and f are not in the same split component of $\{c, g\}$. Similarly neither the split pair $\{c, f\}$ nor the split pair $\{d, g\}$ is a maximal split pair.

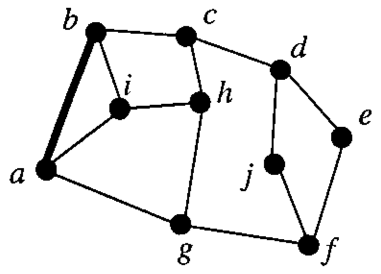

Figure 2.7: A biconnected planar graph G.

2.4 SPQ-tree

Let G be a biconnected series-parallel graph. Let (s, t) be an edge of G. The SPQ-tree $\mathcal{T}$ of G with respect to a *reference edge* $e = (s, t)$ describes a recursive decomposition of

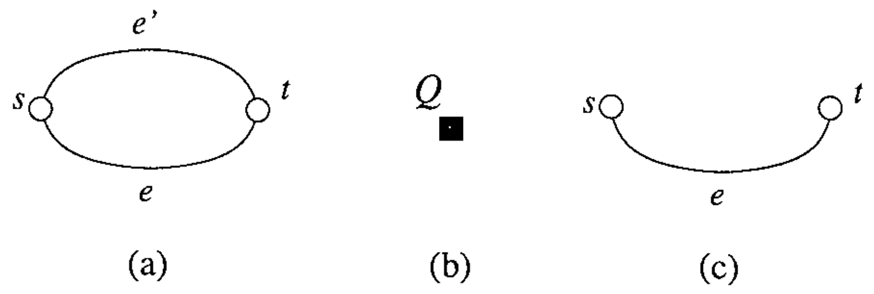

Figure 2.8: (a) $G = skeleton(x)$, (b) $\mathcal{T}$, and (c) $G(x)$ for trivial case.

G induced by its split pairs [GL99]. The SPQ-tree $\mathcal{T}$ defined below is the one used in [REN05] and is a special case of an "$SPQR$-tree" [DT96, GL99] with the exception that an SPQ-tree contains no R-node and the root of the tree is a Q-node corresponding to the reference edge e.

2.4.1 Nodes in an SPQ-tree

An SPQ tree $\mathcal{T}$ is a rooted ordered tree whose nodes are of three types: S, P and Q. Each node x of $\mathcal{T}$ corresponds to a subgraph of G, called its *pertinent graph* $G(x)$. Each node x of $\mathcal{T}$ has an associated biconnected multigraph, called the *skeleton* of x and denoted by *skeleton(x)*. Tree $\mathcal{T}$ is recursively defined as follows.

• *Trivial Case*: In this case, G consists of exactly two parallel edges e and e' joining s and t as illustrated in Fig. 2.8(a). $\mathcal{T}$ consists of a single Q-node x as illustrated in Fig. 2.8(b) where a Q-node is drawn by a black square. The skeleton of x is G itself, as illustrated in Fig. 2.8(a). The pertinent graph $G(x)$ consists of only the edge e' as illustrated in Fig. 2.8(c).

• *Parallel Case*: In this case, the split pair $\{s, t\}$ has three or more split components $G_0, G_1, \cdots, G_k, k \geq 2$, and G_0 consists of only a reference edge $e = (s, t)$, as illustrated in Fig. 2.9(a). The root of $\mathcal{T}$ is a P-node x, as illustrated in Fig. 2.9(b). The *skeleton*(x) consists of $k+1$ parallel edges $e_0, e_1, \cdots, e_k$ joining s and t, as illustrated in Fig. 2.9(c), where $e_0 = e = (s, t)$ and $e_i, 1 \leq i \leq k$, corresponds to G_i. The pertinent graph $G(x) =$

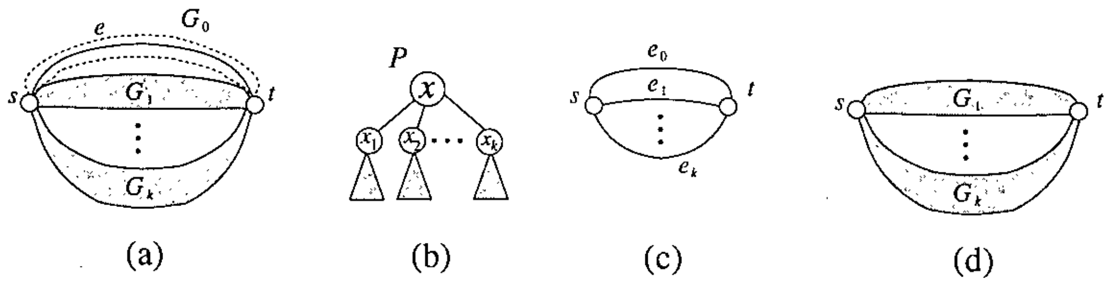

Figure 2.9: (a) G, (b) $\mathcal{T}$, (c) *skeleton(x)* and (d) $G(x)$ for parallel case.

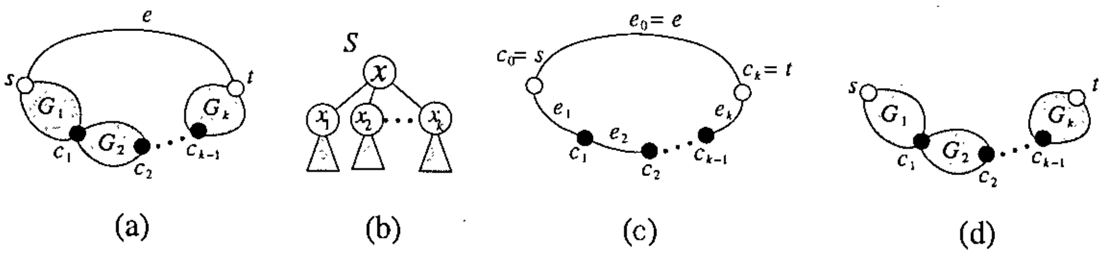

Figure 2.10: (a) G, (b) $\mathcal{T}$, (c) *skeleton(x)* and (d) $G(x)$ for series case.

$G_1 \cup G_2 \cup \cdots \cup G_k$ is a union of $G_1, G_2, \cdots, G_k$ as illustrated in Fig. 2.9(d). (The *skeleton* of P-node P_2 in Fig. 2.11 consists of three parallel edges joining vertices a and c. Figure 2.11(d) depicts the pertinent graph of P_2.)

• *Series Case*: In this case the split pair $\{s, t\}$ has exactly two split components, and one of them consists of the reference edge e. One may assume that the other split component has cut-vertices $c_1, c_2, \cdots, c_{k-1}$, $k \geq 2$, that partition the component into its blocks $G_1, G_2, \cdots, G_k$ in this order from s to t, as illustrated in Fig. 2.10(d). Then the root of $\mathcal{T}$ is an S-node x, as illustrated in Fig. 2.10(b). The skeleton of x is a cycle $e_0, e_1, \cdots, e_k$ where $e_0 = e$, $c_0 = s$, $c_k = t$, and e_i joins c_{i-1} and c_i, $1 \leq i \leq k$, as illustrated in Fig. 2.10(c). The pertinent graph $G(x)$ of node x is a union of $G_1, G_2, \cdots, G_k$ as illustrated in Fig. 2.10(d). (The *skeleton* of S-node S_3 in Fig. 2.11 is the cycle a, i, l, c, a. Figure 2.11(c) depicts the pertinent graph $G(S_3)$ of S_3.)

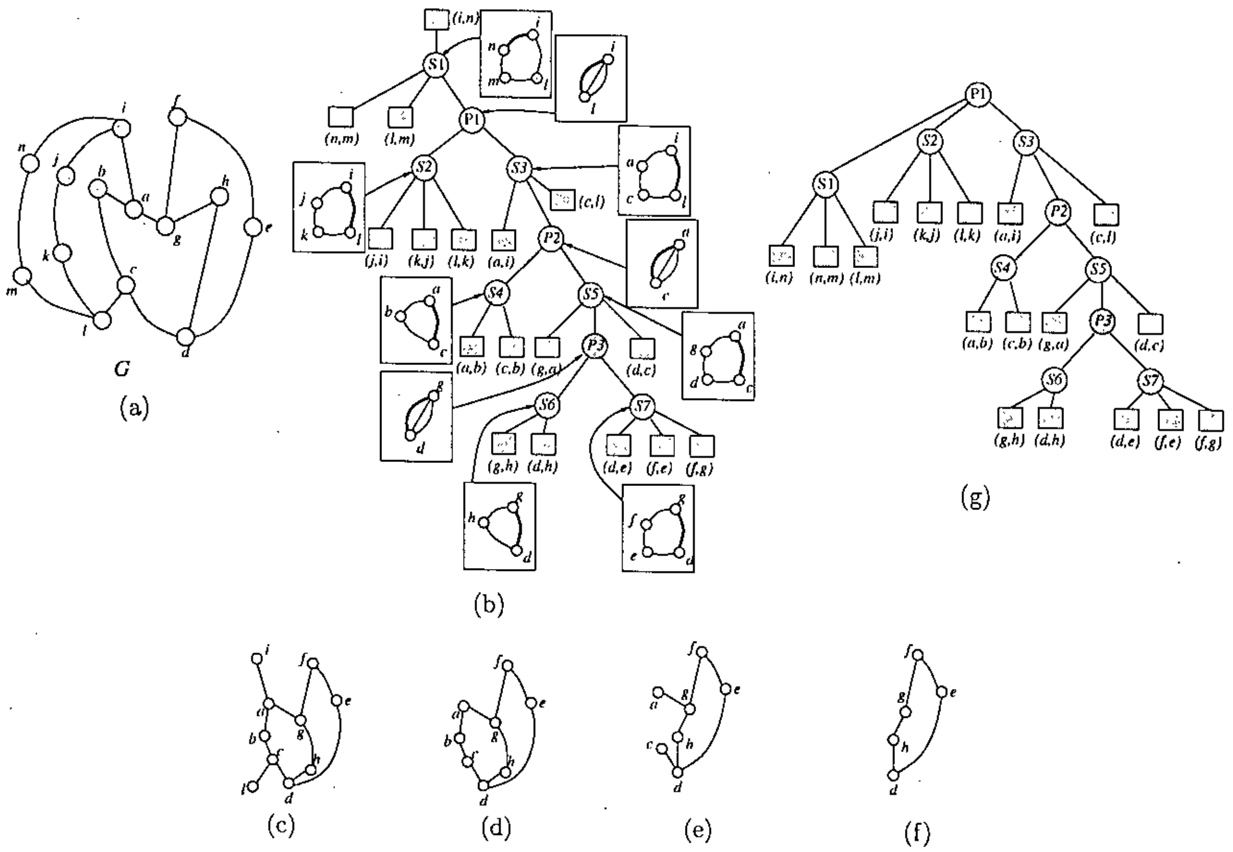

Figure 2.11: (a)A biconnected series-parallel graph G with $\Delta = 3$, (b) SPQ-tree $\mathcal{T}$ of G with respect to reference edge (i, n), and skeletons of P- and S-nodes, (c) the pertinent graph $G(S_3)$ of S-node S_3, (d) the pertinent graph $G(P_2)$ of P-node P_2, (e) the pertinent graph $G(S_5)$ of S-node S_5, (f) the pertinent graph $G(P_3)$ of P-node P_3, and (g) SPQ-tree $\mathcal{T}$ of G with P-node P_1 as the root.

2.4.2 Reference edge and Pole

In all the cases discussed above, we call the edge e the *reference edge* of node x. Except for the trivial case, node x of $\mathcal{T}$ has children $x_1, x_2, \cdots, x_k$ in this order; x_i is the root of the SPQ-tree of graph $G_i \cup e_i$ with respect to the reference edge e_i, $1 \leq i \leq k$. We call edge e_i *the reference edge of node* x_i, and call the endpoints of edge e_i the *poles* of node x_i. The tree obtained so far has a Q-node associated with each edge of G, except the reference edge e. We complete the SPQ-tree $\mathcal{T}$ by adding a Q-node, representing the reference edge e, and making it the parent of x so that it becomes the root of $\mathcal{T}$. An example of the SPQ-tree of a biconnected series-parallel graph in Fig. 2.11(a) is illustrated in Fig. 2.11(b), where the edge drawn by a thick line in each skeleton is the reference edge of the skeleton. One can easily modify $\mathcal{T}$ to an SPQ-tree $\mathcal{T}'$ with an arbitrary P-node as the root as illustrated in Fig. 2.11(f).

2.5 Algorithms and Complexity

In this section we briefly introduce some terminologies related to complexity of algorithms. For interested readers, we refer the book of Garey and Johnson [GJ79].

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

The Notation $O(n)$

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

Polynomial Algorithms

An algorithm is said to be *polynomially bounded* (or simply *polynomial*) if its complexity is bounded by a polynomial of the size of a problem instance. Examples of such complexities are $O(n)$, $O(nlogn)$, $O(n^{100})$, etc. The remaining algorithms are usually referred as *exponential* or *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.

NP-complete

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

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

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

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

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

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

2.6 Drawing Graphs with Few Segments

The problem of computing minimum segment drawings of planar graphs is a relatively new one, and was originated from the seminal work of Dujmović *et al.* [DESW07]. In this section we give an overview of some of the most important results presented in [DESW07]. It is worth mentioning that, although Dujmović *et al.* have given both lower bounds and upper bounds on the number of segments in drawings of several important graph classes, algorithm for computing minimum segment drawings was given only for trees.

More interestingly, for some non-trivial graph classes, like plane biconnected and planar biconnected graphs, even no upper bound was given. Similarly, for plane triconnected cubic graphs, no lower bound was given. Nevertheless, each of these results is quite insightful and is necessary for subsequent research on this problem.

2.6.1 Trees

Let T be a tree. Let η denote the number of odd degree vertices of T. It was shown in [DESW07] that any planar straight-line drawing Γ of T requires at least $\frac{\eta}{2}$ number of segments. The claim holds since each odd degree vertex u of T is an endpoint of some segment in Γ. It is notable that, the number of odd degree vertices in a graph is even and hence, $\frac{\eta}{2}$ is an integer.

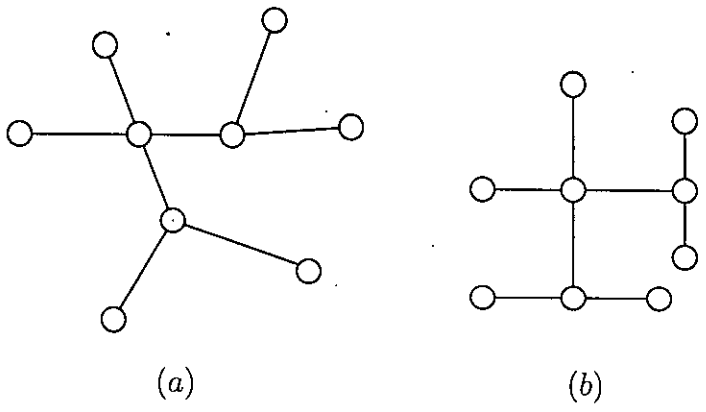

Figure 2.12: (a) A tree T and (b) a minimum segment drawing of T.

It has also been proved in [DESW07] that T admits a planar straight-line drawing on exactly $\frac{\eta}{2}$ number of segments. The proof of this claim is constructive. To prove this claim, a drawing Γ of T has been computed in [DESW07] such that every odd degree vertex of T is an endpoint of exactly one segment in Γ and no even degree vertex is an endpoint of a segment in Γ. Such a drawing of a tree T in Fig. 2.12(a) is illustrated in Fig. 2.12(b).

2.6.2 2-Connected Graphs

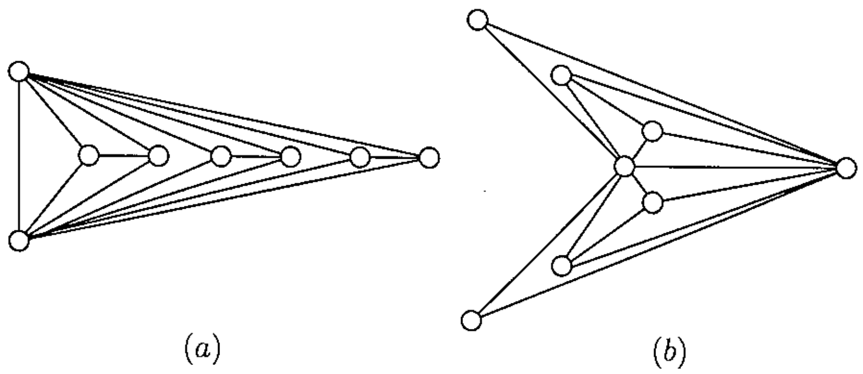

Figure 2.13: (a) A 2-connected plane graph G that requires $\frac{5}{2}n - 4$ segments in any drawing and (b) a drawing of G on $2n - 1$ segments.

It was shown in [DESW07] that there is an n-vertex 2-connected plane graph G with $\frac{5}{2}n - 4$ edges such that any straight line drawing of G requires $\frac{5}{2}n - 4$ number of segments. Such a graph G is shown in Fig. 2.13(a). However, it was also shown in [DESW07] that the same graph requires at least $2n - 1$ segments in every planar drawing as shown in Fig. 2.13(b). In summary, the known result on minimum segment drawing problem states that there is an n-vertex plane 2-connected graph that can be drawn using at most $\frac{5}{2}n$ number of segments, and an n-vertex planar 2-connected graph that requires at least $2n + O(1)$ number of segments in any planar drawing.

2.6.3 3-Connected Graphs

Let G be a 3-connected graph. Based on a canonical decomposition [Kan96] of G, it was shown in [DESW07] that every 3-connected graph G has a plane drawing with at most $\frac{5}{2}n$ line segments. Although it was not shown whether $\frac{5}{2}n$ line segments are necessary for every drawing of G, it was shown that there is a 3-connected plane graph G with $n = 3k$ ($k \in \mathbb{N}$) vertices that requires at least $2n$ number of segments in any planar straight-line drawing. Such a graph G with 12 vertices is shown in Fig. 2.14.

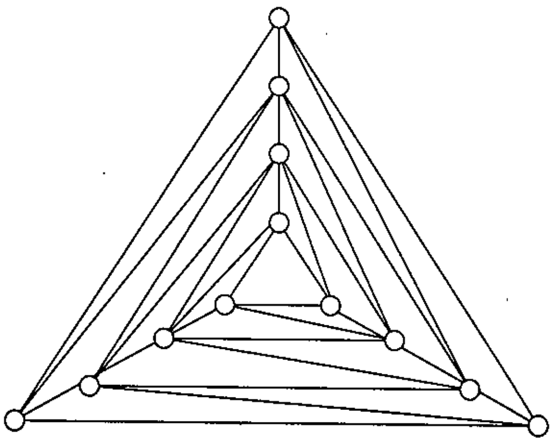

Figure 2.14: A 3-connected graph G that requires at least $2n$ segments in any drawing.

2.6.4 3-Connected Cubic Plane Graph

Let G be a 3-connected cubic plane graph. Based on a canonical decomposition of G, it was shown in [DESW07] that G can always be drawn using at most $n+2$ number of segments. Although this establishes an upper bound of the number of segments required for any drawing of G, no lower bound of the number of segments required for any drawing of G is known as yet. An example of a drawing of a 3-connected cubic plane graph G using exactly $n+2$ segments is shown in Fig. 2.15.

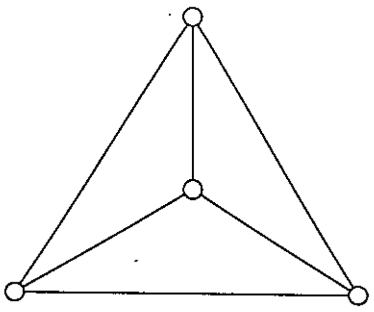

Figure 2.15: Drawing of a 3-connected cubic graph G using $n+2$ segments.

Chapter 3

Minimum Segment Drawings

In this chapter we give an algorithm for computing a minimum segment drawing of a biconnected series-parallel graph G with the maximum degree three. We first present some relevant definitions and our preliminary results in Section 3.1. In Section 3.2 we give a lower bound on the number of segments in any planar straight-line drawing of G. We give our linear-time algorithm for computing a minimum segment drawing of G in Section 3.3. Finally Section 3.5 is a conclusion.

3.1 Preliminaries

Let G be a biconnected series-parallel graph with $\Delta(G) = 3$. Let $\mathcal{T}$ be an SPQ-tree of G. As defined in Chapter 2, the root of $\mathcal{T}$ is a Q-node corresponding to the reference edge e of $\mathcal{T}$. One can easily modify $\mathcal{T}$ to an SPQ-tree $\mathcal{T}'$ with an arbitrary P-node as the root as illustrated in Fig. 3.1(e). In the remainder of this chapter, we thus consider an SPQ-tree $\mathcal{T}$ of a biconnected series-parallel graph G with a P-node as the root. Based on the assumption that $\Delta(G) = 3$, the following facts were mentioned in [REN05].

Fact 1. *Let (s,t) be the reference edge of an S-node x of $\mathcal{T}$, and let $x_1, x_2, \cdots, x_k$ be the children of x in this order from s to t. Then the following (i)–(iii) hold. (i) Each child*

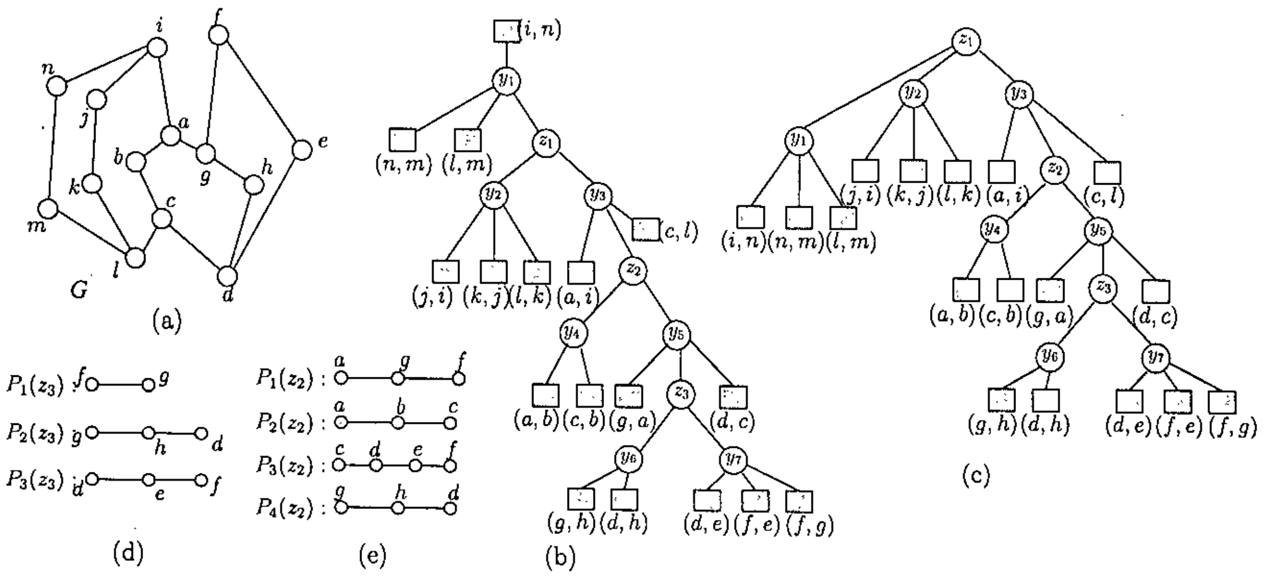

Figure 3.1: (a) A biconnected series-parallel graph G with $\Delta(G) = 3$, (b) SPQ-tree $\mathcal{T}$ of G with respect to reference edge (i, n), (c) SPQ-tree $\mathcal{T}$ of G with P-node z_1 as the root, (d) the three core paths of node z_3, and (e) the four core paths of node z_2.

x_i of x is either a P-node or a Q-node; (ii) both x_1 and x_k are Q-nodes; and (iii) x_{i-1} and x_{i+1} must be Q-nodes if x_i is a P-node where $2 \leq i \leq k-1$.

Fact 2. *The root P-node of $\mathcal{T}$ has exactly three children and each non-root P-node of $\mathcal{T}$ has exactly two children. For a non-root P-node x in $\mathcal{T}$, either both the children of x are S-nodes in $\mathcal{T}$, or one child of x is an S-node and the other child of x is a Q-node in $\mathcal{T}$.*

A node x in $\mathcal{T}$ is *primitive* if x does not have any descendant P-node in $\mathcal{T}$. We define the *height* of a primitive P-node to be zero. The *height* of any other P-node is $(i+1)$ if the maximum of the heights of its descendant P-nodes is i. For two given P-nodes x and z in $\mathcal{T}$, we say that z is a *child P-node* of x if there is an S-node y in $\mathcal{T}$ such that y is a child of x and z is a child of y in $\mathcal{T}$.

Let Γ be a planar straight line drawing of G and G' be the plane graph corresponding to Γ. Let $\mathcal{T}'$ be an SPQ-tree of G'. Let r be such a P-node in $\mathcal{T}'$ that the poles of r appear on the outerface of Γ. An *SPQ-tree of G corresponding to* Γ is the SPQ-tree obtained by considering $\mathcal{T}'$ rooted at r. We use the notation $\mathcal{T}_\Gamma$ to denote an SPQ-tree of G corresponding to a drawing Γ of G. For a node x in $\mathcal{T}_\Gamma$, let P_x and N_x denote the number of P-nodes and primitive P-nodes in the subtree of $\mathcal{T}_\Gamma$ rooted at x. If x is a non-root P-node, then let y and y' denote the two children of x in $\mathcal{T}_\Gamma$. Let p and q denote the number of child P- and Q-nodes respectively of the node y in $\mathcal{T}_\Gamma$. Similarly, let p' and q' denote the number of child P- and Q-nodes respectively of y' in $\mathcal{T}_\Gamma$. Let z_i denote the i-th child P-node of y in $\mathcal{T}_\Gamma$ and e_i denote the edge corresponding to the i-th child Q-node of y in $\mathcal{T}_\Gamma$. Similarly, let z'_i denote the i-th child P-node of y' in $\mathcal{T}_\Gamma$ and e'_i denote the edge corresponding to the i-th child Q-node of y' in $\mathcal{T}_\Gamma$.

For each non-root P-node x of $\mathcal{T}_\Gamma$, we now define the *core paths* of $G(x)$ as follows. If x is a primitive P-node, then let $q \geq q'$. We then define three core paths $P_i(x)$ $(1 \leq i \leq 3)$ of $G(x)$ as $P_1(x) = e_1$, $P_2(x) = G(y')$ and $P_3(x) = \bigcup_{i=2}^{q} e_i$, as shown in Fig. 3.1(d). Otherwise, x is not primitive. In this case, we assume that y has at least two child P-nodes in $\mathcal{T}$. If y does not have at least two child P-nodes in $\mathcal{T}$, but y' has at least two child P-nodes in $\mathcal{T}$, then we rename y as y', y' as y and proceed as follows. On the other hand, if neither y nor y' has at least two child P-nodes in $\mathcal{T}$, then we assume that $p \geq p'$ and proceed as follows. Let e_j and e_k denote the edges corresponding to the Q-nodes immediately preceding z_1 and z_p, respectively in $\mathcal{T}_\Gamma$. We then define four core paths $P_i(x)$ $(1 \leq i \leq 4)$ of $G(x)$ as $P_1(x) = P_1(z_1) \cup \bigcup_{i=1}^{j} e_i$, $P_2(x) = \bigcup_{i=1}^{p'} P_2(z'_i) \cup \bigcup_{i=1}^{q'} e'_i$, $P_3(x) = P_3(z_p) \cup \bigcup_{i=k+1}^{q} e_i$, and $P_4(x) = \bigcup_{i=1}^{p} P_2(z_i) \cup \bigcup_{i=j+1}^{k} e_i$, as shown in Fig. 3.1(e).

We define a straight line drawing Γ of G to be a *canonical drawing of G* if the following (a) and (b) hold for Γ. *(a)* For each non-root P-node x in $\mathcal{T}_\Gamma$, each core path $P_i(x)$ of $G(x)$ is drawn on a different line segment $L_i(x)$ closed between the points corresponding to the two end vertices of $P_i(x)$; and *(b)* there is a primitive P-node w in $\mathcal{T}_\Gamma$ such that the poles of w appear on the outerface of Γ.

Let $L(\Gamma)$ denote the number of segments in the drawing Γ of G. We call a line segment l_1 in Γ to be *collinear* with another line segment l_2 in Γ if l_1 and l_2 have the same slope, and the perpendicular distance between l_1 and l_2 is zero. For a node x in $\mathcal{T}_\Gamma$, we use the notation $\Gamma(x)$ to denote the drawing of $G(x)$ in Γ, and the notation $\Gamma \setminus \Gamma(x)$ to denote the drawing obtained by deleting $\Gamma(x)$ from Γ. If Γ is a canonical drawing of G, then we say that $\Gamma(x)$ is a *canonical drawing of* $G(x)$. We say that $\Gamma(x)$ *shares* a line segment with $\Gamma \setminus \Gamma(x)$ if there is a line segment l_1 in $\Gamma(x)$ and a line segment l_2 in $\Gamma \setminus \Gamma(x)$ such that l_1 and l_2 are collinear and have a common end point. We now have the following lemma.

Lemma 3.1.1 *Let G be a biconnected series-parallel graph with $\Delta(G) = 3$. Then for any planar straight-line drawing Γ of G, a canonical drawing Γ_c of G can be computed such that $L(\Gamma_c) \leq L(\Gamma)$.*

Proof. Let x be a non-root P-node having poles u and v in $\mathcal{T}_\Gamma$. By Fact 1, there is a sibling Q-node of x preceding it and a sibling Q-node of x following it in $\mathcal{T}_\Gamma$. Let $e(x) = (u', u)$ and $e'(x) = (v', v)$ denote the two edges corresponding to these two Q-nodes respectively. Let $h(x)$ denote the height of x in $\mathcal{T}_\Gamma$. Using induction on $h(x)$ we now prove that for each non-root P-node x of $\mathcal{T}_\Gamma$, we can compute a canonical drawing $\Gamma_c(x)$ of $G(x)$ such that replacing $\Gamma(x)$ with $\Gamma_c(x)$ in Γ does not increase $L(\Gamma)$.

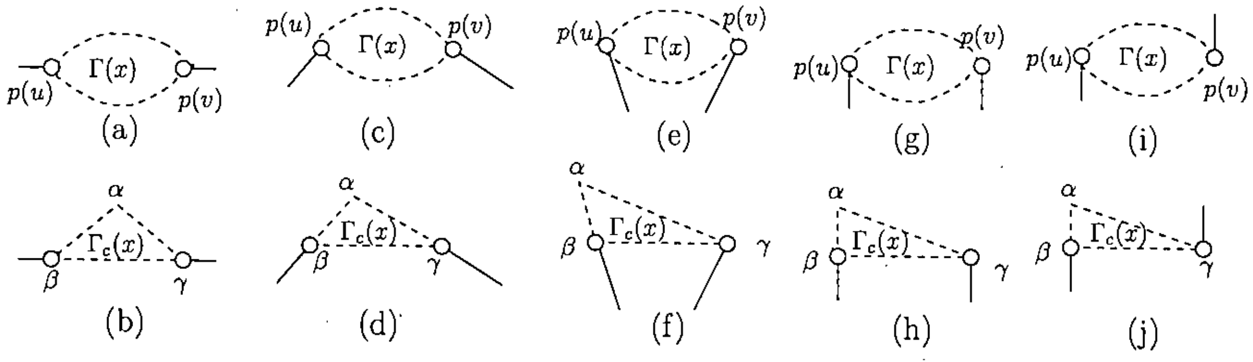

Figure 3.2: All possible cases for computing $\Gamma_c(x)$ when x is a primitive P-node.

We take $h(x) = 0$ as the basis. In this case, x is a primitive P-node. We compute $\Gamma_c(x)$ by first drawing a triangle with three segments $L_i(x)$ $(1 \leq i \leq 3)$, and then drawing the core path $P_i(x)$ on $L_i(x)$ $(1 \leq i \leq 3)$. Considering all possible orientations of $l(e(x))$ and $l(e'(x))$, computation of the drawing $\Gamma_c(x)$ is shown in Fig. 3.2. In each of the cases, we choose the line segment closed between α and β as $L_1(x)$, the one closed between β and γ as $L_2(x)$, and the one closed between α and γ as $L_3(x)$. We now show that, $L(\Gamma)$ will not increase if we replace the drawing $\Gamma(x)$ with $\Gamma_c(x)$ in Γ. Since $G(x)$ is a simple cycle, any straight line drawing of $G(x)$ would require at least three line segments. Again, in any straight line drawing of G, $\Gamma(x)$ may share at most two line segments with $\Gamma \setminus \Gamma(x)$. Except for the case where $l(e(x))$ and $l(e'(x))$ are parallel (as in Fig. 3.2(g) and (i)) or diverging (as in Fig. 3.2(e)), we have not reduced the number of line segments that might have been shared between $\Gamma(x)$ and $\Gamma \setminus \Gamma(x)$ as shown in Fig. 3.2(a)–(d). If $l(e(x))$ and $l(e'(x))$ are parallel or diverging as illustrated in Fig. 3.2(e)–(j), our drawing might have reduced this number by at most one if $\Gamma(x)$ had shared both the line-segments $l(e(x))$ and $l(e'(x))$. On the other hand, if $\Gamma(x)$ had shared both the line segments $l(e(x))$ and $l(e'(x))$, then in each of these cases, any straight line drawing of $G(x)$ would require at least four segments, and we have reduced this number by at least one. Hence, replacing $\Gamma(x)$ with $\Gamma_c(x)$ in Γ would not increase $L(\Gamma)$ in any of the cases.

We now assume that $h(x) > 0$ and for all the descendant P-nodes w of x, a canonical drawing $\Gamma_c(w)$ has been computed such that replacing $\Gamma(w)$ with $\Gamma_c(w)$ in Γ does not increase $L(\Gamma)$. We now compute the drawing $\Gamma_c(x)$ as follows. We first draw a quadrangle with four line segments $L_i(x)$ $(1 \leq i \leq 4)$, in such a way that $L_2(x)$ is the line segment closed between $p(u)$ and $p(v)$. Based on different orientation of $l(e(x)))$ and $l(e'(x))$, the drawing of the quadrangle is illustrated in Fig. 3.3. In each of the cases, we choose the line segment closed between α and β as $L_1(x)$, the one closed between β and γ as $L_2(x)$, the one between γ and δ as L_3 and the one between α and δ as $L_4(x)$. We then draw the core path $P_i(x)$ along $L_i(x)$ $(1 \leq i \leq 4)$. Finally, for each child P-node w of y, we add $\Gamma_c(w)$ by making $L_2(w)$ and $L_4(x)$ collinear. Similarly, for each child P-node w of y', we draw $\Gamma_c(w)$ by making $L_2(w)$ and $L_2(x)$ collinear.

The fact that replacing $\Gamma(x)$ with $\Gamma_c(x)$ does not increase $L(\Gamma)$ can be understood as follows. Let $\Gamma'(x)$ denote the drawing obtained by considering $\Gamma_c(x)$ and the two

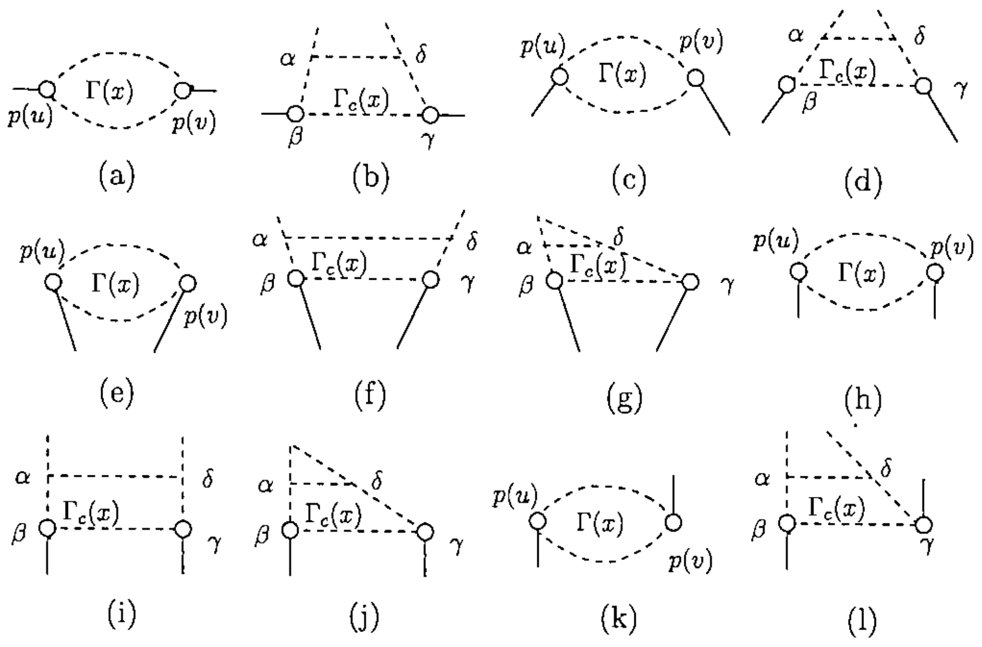

Figure 3.3: All possible cases for computing $\Gamma_c(x)$ when $h(x) > 0$.

line segments $l(e(x))$ and $l(e'(x))$. Let $G'(x)$ denote the underlying graph of $\Gamma'(x)$. Hence $G'(x) = G(x) \cup e(x) \cup e'(x)$. If $l(e(x))$ and $l(e'(x))$ are collinear or converging as illustrated in Fig. 3.3(a)–(d), then for each degree two vertex v' of $G'(x)$, the two incident edges of v' are collinear in $\Gamma'(x)$ with the exception that for each primitive P-node, the incident edges of exactly one degree two vertex are non-collinear. Again for each degree three vertex v' of $G'(x)$, exactly two of the three incident edges are collinear in $\Gamma'(x)$. Thus, $\Gamma'(x)$ has the maximum possible sharing between the drawings of the edges of $G'(x)$, and replacing $\Gamma(x)$ with $\Gamma_c(x)$ in Γ will not cause $L(\Gamma)$ to increase. Similarly, if $l(e(x))$ and $l(e'(x))$ are parallel with the angle between them being 0° as shown in Fig. 3.3(h)–(j), and if x has a child S-node with at least two child P-nodes, then $\Gamma'(x)$ will have the maximum possible sharing between the drawings of the edges of $G(x)$, and replacing $\Gamma(x)$ with $\Gamma_c(x)$ in Γ will not cause $L(\Gamma)$ to increase. On the other hand, if each child S-node of x has at most one child P-node, then the three incident edges of v are pairwise non-collinear in $\Gamma'(x)$ as shown in Fig. 3.3(j) and Fig. 3.4(c). However, $L(\Gamma)$ will not increase even in this case. If both the line segments $l(e(x))$ and $l(e'(x))$ were shared by some line-segment in $\Gamma(x)$, then either of the following (a) and (b) will hold. *(a)* There is at least one descendant non-primitive P-node x' of x such that at either of the two poles of x', all the three incident edges are pairwise non-collinear as shown in Fig. 3.4(a);, and *(b)* there is at least one primitive P-node x'' of x such that $\Gamma(x'')$ uses four line-segments as shown in Fig. 3.4(b). In both the cases, replacing $\Gamma(x)$ with $\Gamma_c(x)$ does not increase $L(\Gamma)$. The reasoning for the cases where line segments $l(e(x))$ and $l(e'(x))$ are parallel with the angle between them being 180° (as shown in Fig. 3.3(k) and (l)) or where $l(e(x))$ and $l(e'(x))$ are diverging (as shown in Fig. 3.3(e)–(g)) follows from similar arguments.

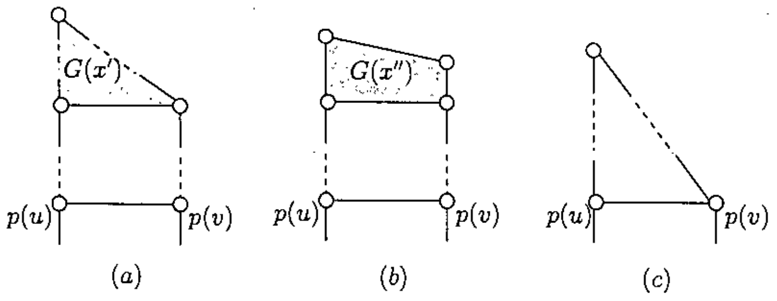

Figure 3.4: (a) and (b) Two drawings of $G(x)$ that shares both the line segments $l(e(x))$ and $l(e'(x))$ with the rest of the drawing, and (c) a canonical drawing $\Gamma_c(x)$ of $G(x)$.

It now remains for us to show that we can obtain a primitive P-node w in $\mathcal{T}_\Gamma$ such that the poles of w appear on the outerface of Γ_c. One can observe that for each non-root P-node x of $\mathcal{T}_\Gamma$, there is a primitive P-node w in the subtree of $\mathcal{T}_\Gamma$ rooted at x such that the poles of w appear on the outerface of $\Gamma_c(x)$. Let r denote the root of $\mathcal{T}$. Let y_1, y_2 and y_3 denote the three children of r in $\mathcal{T}$. We first consider the nodes y_1 and y_2 as the two children of a temporary P-node x' and compute $\Gamma_c(x')$ in the same way as described in the inductive step above. We then replace $\Gamma(x')$ with $\Gamma_c(x')$, and this does not increase $L(\Gamma)$. We next redraw $G(y_3)$ as follows. We first take a single line segment and draw on it all the edges corresponding to the child Q-nodes of y_3 along with all the paths $P_2(z)$ for each child P-node z of y_3. Let $\Gamma'(y_3)$ denote this drawing of $G(y_3)$. We then compute Γ_c by merging $\Gamma_c(x')$ with $\Gamma'(y_3)$. As shown in the induction step above, this merging can be performed by sharing two line segments between $\Gamma_c(x')$ and $\Gamma'(y_3)$ if there is an S-node in $\mathcal{T}_\Gamma$ with at least two child P-nodes. Otherwise, this merging can be performed by sharing one line segment between $\Gamma_c(x')$ and $\Gamma'(y_3)$, and by drawing the three edges incident on u on three different line segments. The details of the proof that the merging of $\Gamma_c(x')$ and $\Gamma'(y_3)$ does not increase $L(\Gamma)$ is omitted in this extended abstract since the arguments are similar to those given in the induction step above. One can also observe that after performing the merging of $\Gamma_c(x')$ and $\Gamma'(y_3)$, we will obtain the poles of a primitive P-node in the outerface of Γ_c. Thus we have transformed Γ into a canonical drawing Γ_c of G without increasing the number of segments in Γ and this completes our proof. □

3.2 Lower Bound

Let G be a biconnected series-parallel graph with $\Delta(G) = 3$. Let Γ be a planar straight line drawing of G. In this section, we give a lower bound of $L(\Gamma)$. Lemma 3.1.1 implies that any planar straight line drawing Γ of G requires at least $L(\Gamma_c)$ number of line segments where Γ_c is a canonical drawing of G obtained by transforming Γ. In this section, we therefore focus on giving a lower bound of $L(\Gamma_c)$. For the clarity of notations, we use $\mathcal{T}$ instead of $\mathcal{T}_{\Gamma_c}$ to denote an SPQ-tree corresponding to the drawing Γ_c. Since there is always a primitive P-node w in $\mathcal{T}$ such that the poles of w appear on the outerface of Γ_c, we assume that the root of $\mathcal{T}$ has two child S-nodes that are primitive in $\mathcal{T}$. We first have the following lemma.

Lemma 3.2.1 *Let G be a biconnected series-parallel graph with $\Delta(G) = 3$. Let Γ_c be a canonical drawing of G. Let $\mathcal{T}$ be an SPQ-tree of G corresponding to Γ_c. Then for a non-root P-node x in $\mathcal{T}$, $L(\Gamma_c(x)) \geq P_x + N_x + 1$.*

Proof. We use induction on P_x. In the basis case we take $P_x = 1$, *i.e.*, x is a primitive P-node in $\mathcal{T}$. Hence $N_x = 1$ and $P_x + N_x + 1 = 3$. Since $G(x)$ is a simple cycle when x is primitive and any planar straight line drawing of a cycle requires at least three segments, the claim holds.

We now assume that $P_x > 0$ and the claim holds for every P-node w in $\mathcal{T}$ having $P_w < P_x$. Hence $L(\Gamma_c(w)) \geq P_w + N_w + 1$ for every descendant P-node w of x in $\mathcal{T}$. We now take a child P-node w of x and delete the drawing $\Gamma_c(w)$ from Γ_c. Let G' denote the underlying graph of this drawing $\Gamma_c \backslash \Gamma_c(w)$. The graph G' is not necessarily a biconnected series-parallel graph. Let u and v be the two poles of w in $\mathcal{T}$. In order to ensure that we are working with a biconnected series-parallel graph, we now add an edge (u, v) to G', and a new line segment between the points $p(u)$ and $p(v)$ in Γ_c. We then replace the node

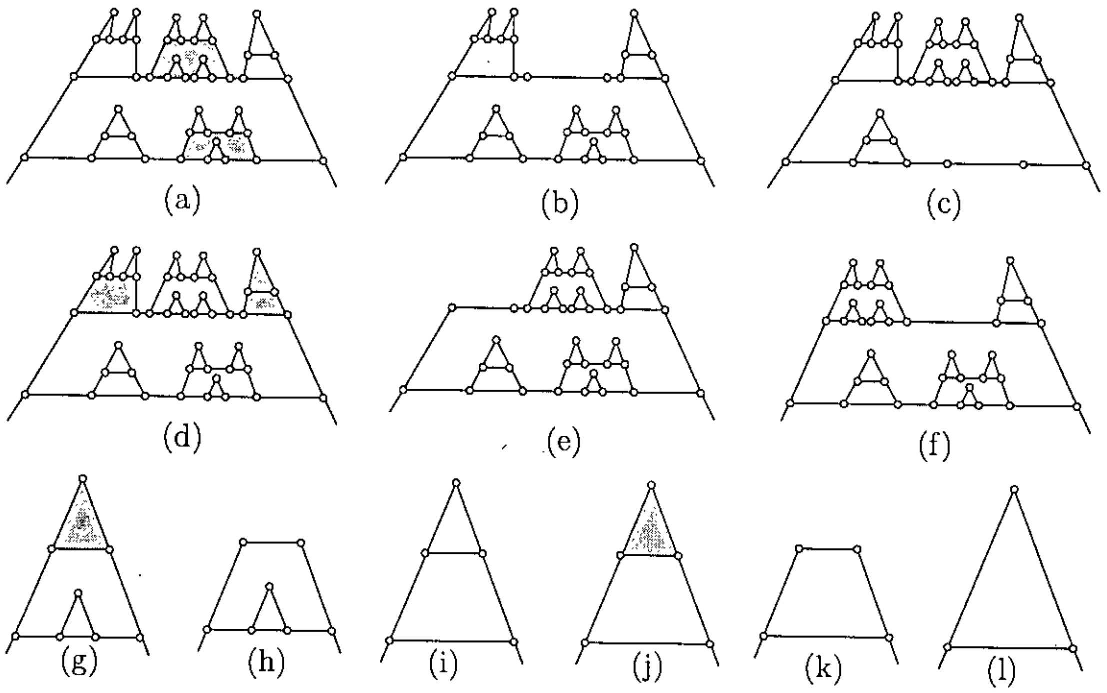

Figure 3.5: (a)–(l) Cases in the induction step of the proof of Lemma 3.2.1. $\Gamma_c(w)$ shown highlighted in each case.

w in $\mathcal{T}$ with a Q-node representing the edge (u, v), and rename the node x as x'. Let Γ' denote this newly computed drawing of $G(x')$. Since $P_{x'} < P_x$, by induction hypothesis we have $L(\Gamma_c(x') \geq P_{x'} + N_{x'} + 1$. We now have the following two cases to consider.

Case 1. Γ' *is canonical.* This case may occur in either of the following two subcases. *(i)* $p > 2$ and $w = z_i$ $(2 \le i \le p-1)$, as illustrated in Fig. 3.5(a) and (b); and *(ii)* $p' \ge 1$ and $w = z'_i$ $(1 \le i \le p')$, as illustrated in Fig. 3.5(a) and (c). In both of these subcases, $\Gamma_c(w)$ had exactly one line segment shared with $\Gamma_c(x)$. Thus, $L(\Gamma') = L(\Gamma_c(x)) - L(\Gamma_c(w)) + 1$. Again, since Γ' is canonical, $L(\Gamma')) = L(\Gamma_c(x'))$. By induction hypothesis we then have $L(\Gamma_c(x)) \ge P_{x'} + N_{x'} + 1 + P_w + N_w + 1 - 1 = P_x + N_x + 1$.

Case 2. Γ' *is not canonical.* Here we have the following three subcases.
(i) $p > 2$ and either $w = z_1$ or $w = z_p$, as illustrated in Fig. 3.5(d) and (e). In this subcase, $\Gamma_c(w)$ had exactly two line segments shared with $\Gamma_c(x)$. Thus, $L(\Gamma') = L(\Gamma_c(x)) - L(\Gamma_c(w)) + 2$. Since Γ' is not canonical, we now make it canonical by making $L_1(z_2)$ collinear with $L_1(x')$ if $w = z_1$ or, by making $L_3(z_{p-1})$ collinear with $L_3(x')$ if $w = z_p$ as illustrated in Fig. 3.5(f). One can observe that, in both the cases, the number of line segments decreases by exactly one in $\Gamma_c(x')$. Thus, $L(\Gamma_c(x')) = L(\Gamma') - 1$. By induction hypothesis we then have $L(\Gamma_c(x)) \ge P_{x'} + N_{x'} + 1 + P_w + N_w + 1 - 1 = P_x + N_x + 1$.
(ii) $p = p' = 1$ and $w = z_1$, as illustrated in Fig. 3.5(g) and (h). In this subcase, $\Gamma_c(w)$ had three line segments shared with $\Gamma_c(x)$. Thus $L(\Gamma') = L(\Gamma_c(x)) - L(\Gamma_c(w)) + 3$. Since Γ' is not canonical, we now make it canonical by renaming y as y', y' as y and then by computing the canonical drawing of $G(x')$ as illustrated in Fig. 3.5(i). One can observe that the number of line segments decreases by exactly two in $\Gamma_c(x')$. Thus, $L(\Gamma_c(x')) = L(\Gamma') - 2$. By induction hypothesis we then have $L(\Gamma_c(x)) \ge P_{x'} + N_{x'} + 1 + P_w + N_w + 1 - 1 = P_x + N_x + 1$.
(iii) $p = 1, p' = 0$ and $w = z_1$, as illustrated in Fig. 3.5(j) and (k). In this subcase, $\Gamma_c(w)$ had three line segments shared with $\Gamma_c(x)$. Thus $L(\Gamma') = L(\Gamma_c(x)) - L(\Gamma_c(w)) + 3$. Since x' has become a primitive P-node a canonical drawing of $G(x')$ would be drawn on three line segments. Since Γ' is not canonical, we now make it canonical by drawing it on three line segments as illustrated in Fig. 3.5(l). We observe that the number of line segments decreases by exactly one in $\Gamma_c(x')$. Thus, $L(\Gamma_c(x')) = L(\Gamma') - 1$. By induction hypothesis

we then have $L(\Gamma_c(x)) \geq P_{x'} + N_{x'} + 1 + P_w + N_w + 1 - 2 = P_{x'} + N_{x'} + P_w + N_w$. Since x' is primitive, $N_{x'} = 1$. Hence $L(\Gamma_c(x)) \geq P_x + N_x + 1$. □

We now have the following theorem.

Theorem 3.2.2 *Let G be a biconnected series-parallel graph with $\Delta(G) = 3$. Let Γ_c be a canonical drawing of G. Let $\mathcal{T}$ be an SPQ-tree of G corresponding to Γ_c. Let $P_\mathcal{T}$ and $N_\mathcal{T}$ denote the number of P-nodes and the number of primitive P-nodes respectively in $\mathcal{T}$. Then the following (a) and (b) hold. (a) $L(\Gamma_c) \geq P_\mathcal{T} + N_\mathcal{T} + 2$, if every S-node in $\mathcal{T}$ has at most one child P-node; and (b) $L(\Gamma_c) \geq P_\mathcal{T} + N_\mathcal{T} + 1$, otherwise.*

Proof. Let r be the root of $\mathcal{T}$. Let y_1, y_2 and y_3 be the three children of r in $\mathcal{T}$. Since Γ_c is a canonical drawing, we assume that y_1 and y_2 are primitive in $\mathcal{T}$. Since G is a simple graph, exactly one of y_1, y_2 and y_3 can be a Q-node in $\mathcal{T}$. Thus, if there is a Q-node among y_1, y_2 and y_3, then we assume that y_2 is the Q-node. The proofs of the claims (a) and (b) are given below.

(a) We have the following two cases to consider here.

Case 1. *y_3 is primitive in $\mathcal{T}$.* Since $G(y_1) \cup G(y_2)$ is a cycle, at least three line segments are required to draw $G(y_1) \cup G(y_2)$ in Γ_c as illustrated in Fig. 3.6(a). Since $G(y_3)$ is a path, at least one new line segment is required to draw $G(y_3)$ along with $G(y_1) \cup G(y_2)$ in Γ_c as illustrated through the thick line segment in Fig. 3.6(b). Since $P_\Gamma = 1, N_\Gamma = 1$, and $P_\Gamma + N_\Gamma + 2 = 4$, we thus have $L(\Gamma_c) = 4 \geq P_\Gamma + N_\Gamma + 2$.

Case 2. *y_3 is not primitive in $\mathcal{T}$.* Let z denote the child P-node of y_3 in $\mathcal{T}$. By Lemma 3.2.1, $L(\Gamma_c(z)) \geq P_z + N_z + 1$. One can observe that $G(y_1) \cup G(y_2)$ is connected with $G(y_3)$ through exactly two edges, namely the two edges incident to the two poles of $G(y_3)$. Hence, any drawing of $G(y_1) \cup G(y_2)$ can share at most two segments with the drawing $\Gamma_c(z)$. However, as shown in the proof of Lemma 3.1.1, since each S-node in $\mathcal{T}$ has at most one child P-node, we cannot draw the line segments $L_1(z)$ and $L_3(z)$ as converging in the exterior of $\Gamma_c(z)$ without increasing $L(\Gamma_c(z))$. Since $L_1(z)$

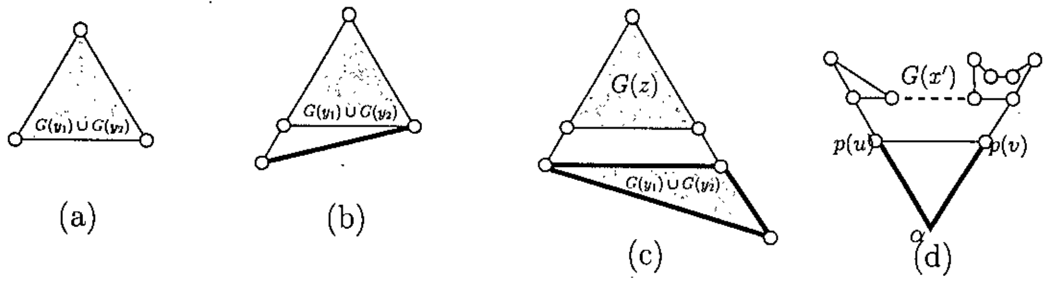

Figure 3.6: (a) The drawing of $G(y_1) \cup G(y_2)$, (b) Γ_c if y_3 is primitive, (c) Γ_c if y_3 has exactly one child P-node, and (d) Γ_c if y_3 has at least two child P-nodes.

and $L_3(z)$ are converging in the interior of $\Gamma_c(z)$, at least two new segments are required to draw $G(y_1) \cup G(y_2)$ along with $G(y_3)$ as shown through the thick line segments in Fig. 3.6(c). Thus, $L(\Gamma_c) \geq P_z + N_z + 1 + 2$. Since $P_\Gamma = P_z + 1, N_\Gamma = N_z$, we thus have $L(\Gamma_c) \geq P_\Gamma + N_\Gamma + 1$ in this case.

(*b*) In this case y_3 has at least two child P-nodes in $\mathcal{T}$. We first consider y_2 and y_3 as the two S-nodes of a temporary P-node x' in $\mathcal{T}$. Then we compute $\Gamma_c(x')$ without increasing $L(\Gamma_c)$ in the same way as described in the proof of Lemma 3.1.1. By Lemma 3.2.1, $L(\Gamma_c(x')) \geq P_{x'} + N_{x'} + 1$. As shown in the proof of Lemma 3.1.1, since at least one S-node in $\mathcal{T}$ has two child P-nodes, we can draw the line segments $L_1(x')$ and $L_3(x')$ as converging in the exterior of $\Gamma_c(x')$ as illustrated in Fig. 3.6(d). Let α denote the point where $L_1(x')$ and $L_3(x')$ converges. Let u and v denote the poles of r. Since y_3 is not a Q-node, we can now complete the drawing of $G(y_3)$ on the two line segments closed between $p(u), \alpha$ and $\alpha, p(v)$ without requiring any new line segment. Since $P_{x'} = P_\Gamma, N_{x'} = N_\Gamma$, we finally have $L(\Gamma_c) \geq P_{x'} + N_{x'} + 1 = P_\Gamma + N_\Gamma + 1$. □

3.3 Drawings of Biconnected Graphs

We now present the main result of this chapter in the following theorem.

Theorem 3.3.1 *Let G be a biconnected series-parallel graph with $\Delta(G) = 3$. Then a minimum segment drawing of G can be computed in linear time.*

Proof. We give our proof of the above claim in the following three steps. (a) We first compute a straight line drawing Γ of G, (b) we then show that Γ can be computed in linear time, and (c) we finally show that Γ is a minimum segment drawing of G.

(a) Let $\mathcal{T}$ be an SPQ-tree of G rooted at an arbitrary P-node r. Let y_1, y_2 and y_3 denote the three children of r in $\mathcal{T}$. Since G is a simple graph, at most one of y_1, y_2 and y_3 can be a Q-node. Thus, if there is a Q-node among y_1, y_2 and y_3, then we assume that y_2 is the Q-node. In order to compute a minimum segment drawing of G, we want the following two conditions to hold for y_1, y_2 and y_3 in $\mathcal{T}$. *(a)* y_1 and y_2 are primitive in $\mathcal{T}$; and *(b)* y_3 has at least two child P-nodes in $\mathcal{T}$. We have illustrated these two conditions

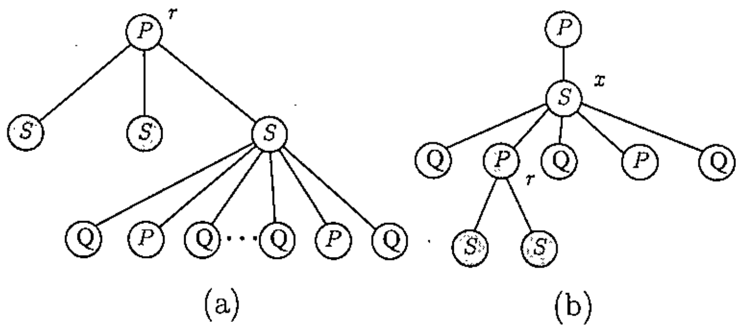

Figure 3.7: (a) A desired root r of $\mathcal{T}$ and (b) an alternate scenario where r can be found by traversing $\mathcal{T}$. Primitive nodes are shown shaded.

for the root r of the SPQ-tree shown in Fig. 3.7(a). If these conditions hold for the three children of our arbitrarily chosen root r, then we are done. Otherwise, we search for such a P-node r in $\mathcal{T}$. If there is such a P-node r in $\mathcal{T}$, then there is an S-node x in $\mathcal{T}$ such that x has at least two child P-nodes, one of which is primitive. We can search for such an S-node x in $\mathcal{T}$ in linear time. If we find such an S-node x in $\mathcal{T}$, then the child primitive P-node r of x will be our desired root of $\mathcal{T}$. If we fail to find any such S-node x in $\mathcal{T}$, then each S-node in $\mathcal{T}$ has at most one child P-node in $\mathcal{T}$. We then choose any primitive P-node r in $\mathcal{T}$ as the root of $\mathcal{T}$.

We now compute Γ in a bottom up traversal of $\mathcal{T}$. At first, in each non-root P-node x of $\mathcal{T}$, we compute a canonical drawing $\Gamma(x)$ of $G(x)$ from the previously computed canonical drawings $\Gamma(w)$ of $G(w)$ for each child P-node w of x. Then we compute $\Gamma = \Gamma(r)$ for the root r of $\mathcal{T}$. We describe this construction inductively. For a primitive P-node x in $\mathcal{T}$, we draw the three core paths $P_i(x)$ $(1 \leq i \leq 3)$ of $G(x)$ on three line segments $L_i(x)$ $(1 \leq i \leq 3)$ such that each line segment $L_i(x)$ is closed between the end vertices of $P_i(x)$ $(1 \leq i \leq 3)$, as illustrated in Fig. 3.8(a). Now let x be a non-root and non-primitive P-node in $\mathcal{T}$. We assume inductively that for each descendant child P-node w of x, we have computed a canonical drawing $\Gamma(w)$ of $G(w)$. We then draw the four core paths $P_i(x)$

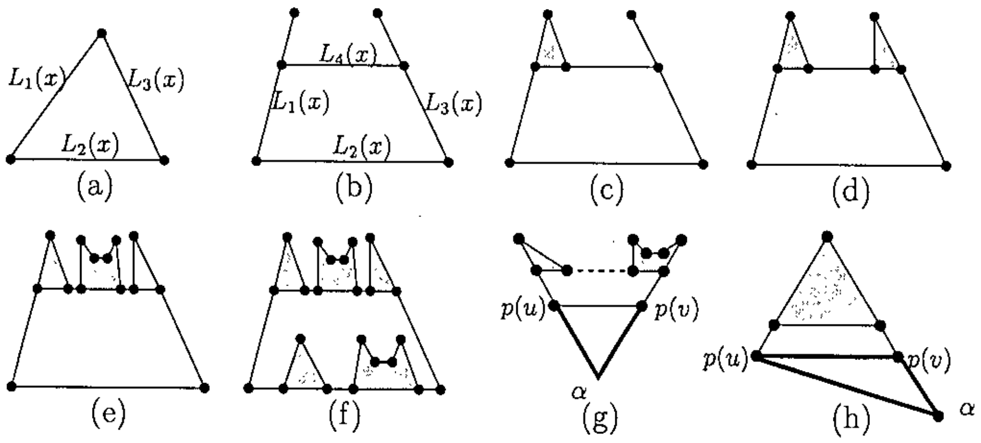

Figure 3.8: (a) Drawing of $G(x)$ for a primitive P-node x, (b) the quadrangle for a non-root and non-primitive P-node x, (c)–(f) completing the drawing $\Gamma(x)$ for a non-root and non-primitive P-node x, (g) Γ when there is a suitable root in $\mathcal{T}$, and (h) Γ when there is no suitable root in $\mathcal{T}$.

$(1 \leq i \leq 4)$ on four line segments $L_i(x)$ $(1 \leq i \leq 4)$ such that each line segment $L_i(x)$ is closed between the end vertices of $P_i(x)$ $(1 \leq i \leq 4)$, as illustrated in Fig. 3.8(b). For each child P-node w of x, we now add $\Gamma(w)$ to this quadrangle and complete the drawing $\Gamma(x)$ as follows. At first, we draw $\Gamma(z_1)$ by making $L_1(z_1)$ collinear with $L_1(x)$, and $L_2(z_1)$ collinear with $L_4(x)$ as shown in Fig. 3.8(c). Next we draw $\Gamma(z_p)$ by making $L_3(z_p)$ collinear with $L_3(x)$ and $L_2(z_p)$ collinear with $L_4(x)$ as shown in Fig. 3.8(d). Finally, for each $w = z_i$ $(2 \leq i \leq p-1)$, we draw $\Gamma(w)$ by making $L_2(w)$ collinear with $L_4(x)$ as shown in Fig. 3.8(e), and for each $w = z'_i$ $(1 \leq i \leq p')$, we draw $\Gamma(w)$ by making $L_2(w)$

collinear with $L_2(x)$ as shown in Fig. 3.8(f).

We finally assume that x is the root P-node of $\mathcal{T}$. Let u and v denote the poles of x in $\mathcal{T}$. To compute $\Gamma = \Gamma(x)$, we first consider y_2 and y_3 as the children of a temporary P-node x' and compute the canonical drawing $\Gamma(x')$ in the same way as described in the inductive case above. We now have the following two cases to consider. We first consider the case where $\mathcal{T}$ has a suitable root as described earlier. By construction, we will have two line segments in $\Gamma(x')$ in this case, namely $L_1(x')$ and $L_3(x')$ that can be drawn as converging in the exterior of $\Gamma(x')$. Let α denote the point where $L_1(x')$ and $L_3(x')$ converge. We then draw the graph $G(y_3)$ on the two line segments closed between $p(u), \alpha$ and $\alpha, p(v)$, as shown in Fig. 3.8(g). We next consider the case where $\mathcal{T}$ does not have a suitable root r as described earlier. In this case, we take a point α in the exterior of $\Gamma(x')$ such that the points $p(u), p(v)$ and α form a triangle as shown in Fig. 3.8(h). We then draw the graph $G(y_3)$ on the two line segments closed between $p(u), \alpha$ and $\alpha, p(v)$.

(*b*) We now prove that the drawing Γ obtained above can be computed in linear time. For each primitive P-node x in $\mathcal{T}$, we merely compute a triangle with arbitrary apices. For each non-root P-node x in $\mathcal{T}$, including the temporary P-node x', we perform the following operations.

1. At first we compute the lengths of the line segments $L_1(x)$ and $L_4(x)$ from the lengths $L_2(w)$ for each child P-node w of x.

2. We next compute a quadrangle with four line segments $L_i(x)$ $(1 \leq i \leq 4)$.

3. Finally, for each child P-node w of x, we make the slope of $L_2(w)$ equal to the slope of $L_2(x)$ if $w = z_i$ $(1 \leq i \leq p)$ and to the slope of $L_4(x)$ if $w = z'_i$ $(1 \leq i \leq p')$. We also make the slope of $L_1(z_1)$ equal to $L_1(x)$ and the slope of $L_3(z_p)$ equal to $L_3(x)$.

Finally, at the root r of $\mathcal{T}$, we merely compute two line segments for drawing the graph $G(y_3)$. Clearly, each of the above steps in computing Γ can be performed in time linear in the number of P-nodes in $\mathcal{T}$. Since the number of P-nodes in $\mathcal{T}$ is linear in the number of vertices of G [DT96], the drawing Γ can be computed in linear time.

(c) We now prove that Γ has the minimum number of segments. We first prove that for each non-root P-node x of $\mathcal{T}$, we draw $G(x)$ on $P_x + N_x + 1$ segments, We give here an inductive proof by taking induction on the height $h(x)$ of x. For the base case, $h(x) = 0$. Here $P_x + N_x + 1 = 3$. We have drawn $G(x)$ on three line segments. Thus our claim holds for the basis case. We now consider $h(x) > 0$ and x is a non-root and non-primitive P-node. We inductively assume that, for each child P-node w of x, $G(w)$ has been drawn on $P_w + N_w + 1$ segments. While computing $\Gamma(x)$, we have drawn $G(y')$ in such a way that all the edges corresponding to the child Q-nodes of y' were drawn on a single segment, and $L_2(z'_i)$ for each $G(z'_i)$ was drawn on the same segment. Thus the number of segments in this drawing of $G(y')$ is $P'_y + N'_y + p' - (p' - 1) = P'_y + N'_y + 1$. Similarly, $G(y)$ was first drawn on $P_y + N_y + 1$ segments and then the path $\bigcup_{i=1}^{j} e_i$ was drawn on the same segment as $L_1(z_1)$ and the path $\bigcup_{i=k+1}^{q} e_i$ was drawn on the same segment as $L_3(z_p)$. Here e_j and e_k are the two edges corresponding to the two Q-nodes immediately preceding z_1 and z_p respectively in $\mathcal{T}$. Since we had reused an already drawn segment, this last operation did not increase the number of segments. We finally had merged these drawings of $G(y)$ and $G(y')$ together to get a drawing of $G(x)$ on $P_y + N_y + 1 + P'_y + N'_y + 1 = (P_y + P'_y + 1) + N_x + 1 = P_x + N_x + 1$ segments. Finally, in the root node, we did not draw any new line segment if a suitable root r was found for $\mathcal{T}$, otherwise we had drawn exactly one new line segment. Thus we had drawn Γ on $P'_x + N'_x + 1$ segments in the first case, and on $P'_x + N'_x + 2$ segments in the second case. Since $P_\Gamma = P_{x'}, N_\Gamma = N_{x'}$, we have ultimately drawn Γ on $P_\Gamma + N_\Gamma + 2$ segments if each S-node in $\mathcal{T}$ had at most one child P-node, and on $P_\Gamma + N_\Gamma + 1$ segments otherwise. Both these quantities matches the bound given on $L(\Gamma)$ in Theorem 3.2.2. This completes our proof that we have computed a minimum segment drawing of G in linear time. □

3.4 Drawing of Connected Graphs

So far in this chapter we have dealt with biconnected series-parallel graphs with the maximum degree three. However, the same approach can be adopted to compute minimum segment drawings of series-parallel graphs that are not biconnected. Let G be a series-

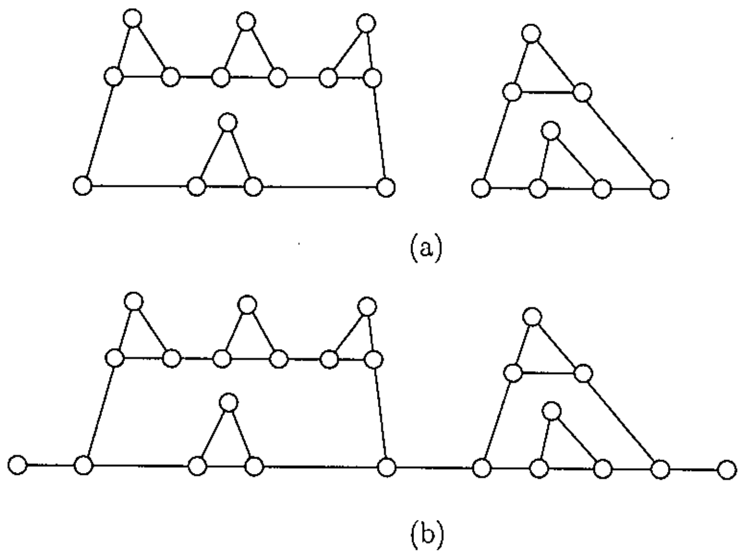
Figure 3.9: (a) Canonical drawings of the blocks of G, and (b) a minimum segment drawing of G.

parallel graph with the maximum degree three. In this section we assume that G is not biconnected and illustrate the computation of a minimum segment drawing of G. In this case, we first compute the blocks of G. Each block of G is either an edge or a series-parallel graph $G(x)$ whose SPQ-tree can be rooted at a P-node x having exactly two children. For each such graph $G(x)$, we then compute a canonical drawing of $G(x)$ as illustrated in Fig. 3.9(a). Next we add a single line segment aligned with the path $P_2(x)$ of each block $G(x)$ and complete the drawing of G as illustrated in Fig. 3.9(b).

3.5 Summary

In this chapter we have given a linear-time algorithm for computing minimum segment drawings of series-parallel graphs with the maximum degree three. We have also given a lower bound on the number of segments required for any planar straight line drawing of such graphs. We have first shown in Lemma 3.1.1 that any planar straight line drawing Γ of a series-parallel graph G can be transformed into a canonical drawing Γ_c of G such that $L(\Gamma_c) \leq L(\Gamma)$. We have next given a lower bound on $L(\Gamma)$ in Theorem 3.2.2; and finally, have shown in Theorem 3.3.1 that a minimum segment drawing Γ of G can be computed in linear time.

To the best of our knowledge, ours is the first such result in the minimum segment drawing problem focusing on an important subclass of planar graphs. It remains as our future work to achieve similar results for wider subclasses of planar graphs.

Chapter 4

Conclusion

In this thesis we have dealt with minimum segment drawings of series-parallel graphs. We have started with an introductory overview on graph drawing in Chapter 1. In that chapter we have discussed different graph drawing conventions relevant to the minimum segment drawing problem and have established our objective in this thesis.

In Chapter 2 we have introduced the preliminary ideas on graph theory and on minimum segment drawings. We have also discussed series-parallel graphs and SPQ-trees in detail in this chapter.

In Chapter 3 we have started with a discussion on the terminology pertinent to minimum segment drawings of biconnected series-parallel graphs with the maximum degree three. Next we have established a lower bound on the number of segments in any planar straight line drawing of a biconnected series-parallel graph G with the maximum degree three. Finally, we have presented a linear-time algorithm for computing a minimum segment drawing of G. To the best of our knowledge this has been the first such result in the minimum segment drawing problem for an important subclass of biconnected planar graphs. However, the following problems remained as future works.

1. To study the minimum segment drawing problem in conjunction with other aesthetic criteria like area requirement and symmetry of the drawing.

2. To obtain minimum segment drawing algorithm for any biconnected planar graph with the maximum degree three.

3. To obtain minimum segment drawing algorithm for series-parallel graphs without any restriction on the degree.

4. To obtain minimum segment drawing algorithms for larger subclass of planar graphs.

Bibliography

[BFN85] C. Batini, L. Furlani and E. Nardelli, *What is a good diagram? A parametric approach*, Proc. of 4th Internat. Conf. on Entity Relationship Approach, pp. 312-319, 1985.

[BL76] K. S. Booth and G. S. Lueker, *Testing the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms,* J. Comput. Syst. Sci., 13 (3), pp. 335-379, 1976.

[Cha08] T. Chan, *A near-linear area bound for drawing binary trees,* Algorithmica, 34 (1), pp. 1-13, 2008.

[CN98] M. Chrobak and S. Nakano, *Minimum-width grid drawings of plane graphs*, Comp. Geom. Theory and Appl., 11, pp. 29-54, 1998.

[CNAO85] N. Chiba, T. Nishizeki, S. Abe and T. Ozawa, *A linear algorithm for embedding planar graphs using PQ-trees,* J. Comput. Syst. Sci., 30, pp. 54-76, 1985.

[DESW07] V. Dujmović, D. Eppstein, M. Suderman, and D. R. Wood, *Drawings of planar graphs with few slopes and segments*, Computational Geometry, 38, pp. 194–212, 2007.

[DETT94] G. Di Battista, P. Eades, R. Tamassia and I. G. Tollis, *Algorithms for drawing graphs: an annotated bibliography*, Comp. Geom. Theory and Appl., 4, pp. 235-282, 1994.

[DETT99] G. Di Battista, P. Eades, R. Tamassia and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice-Hall Inc., Upper Saddle River, New Jersey, 1999.

[DF05] G. Di Battista and F. Frati, *Small area drawings of outerplanar graphs,* Proc. of GD'05, Lecture Notes in Computer Science, 3843, pp. 89-100, 2005.

[DSW05] V. Dujmović, M. Suderman, and D. R. Wood, *Really straight graph drawings*, Proc. of GD'05, Lecture Notes in Computer Science, 3843, pp. 122-132, 2005.

[DT96] G. Di Battista and R. Tamassia (editors), *Special Issue on Graph Drawing*, Algorithmica, 16 (1), 1996.

[Ead88] P. Eades, *Symmetry finding algorithms*, Computational Morphology, pp. 41-51, North-Holland, Amsterdam, Netherlands, 1988.

[Far48] I. Fary, *On straight line representations of planar graphs*, Acta Sci. Math. Szeged. 11, pp. 229-233, 1948.

[FDFH03] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics Principles and Practice*, Pearson Education Pte. Ltd., 2003.

[FPP90] H. de Fraysseix, J. Pach and R. Pollack, *How to draw a planar graph on a grid*, Combinatorica, 10, pp. 41-51, 1990.

[GJ79] M. R. Garey and D. S. Johnson, *Computers and Intractability a Guide to Theory of NP-Completeness*, W. H. Freeman and Company, New York, 1979.

[GJ83] M. R. Garey and D. S. Johnson, *Crossing number is NP-complete*, SIAM J. Alg. Disc. Methods, 4 (3), pp. 312-316, 1983.

[GL99] A. Garg and G. Liotta, *Almost bend-optimal planar orthogonal drawings of biconnected degree-3 planar graphs in quadratic time*, Proc. of GD'99, Lecture Notes in Computer Science, 1731, pp. 38-48, Springer, 1999.

[GT01] A. Garg and R. Tamassia, *On the computational complexity of upward and rectilinear planarity testing*, SIAM Journal on Computing, 31 (2), pp. 601-625, 2001.

[HT74] J. Hopcroft and R. E. Tarjan, *Efficient planarity testing*, J. ACM, 21 (4), pp. 549-568, 1974.

[Kan96] G. Kant, *Drawing planar graphs using the canonical ordering*, Algorithmica, 16, pp. 4-32, 1996.

[KL84] M. R. Kramer and J. van Leeuwen, *The complexity of wire routing and finding minimum area layouts for VLSI circuits*, (Editors) R. P. Preparata, Advances in Computing Research, 2, VLSI Theory, JAI press, Reading, MA, pp. 129-146, 1984.

[KPPT06] B. Keszegh, J. Pach, D. Pálvölgyi, and G. Tóth, *Drawing cubic graphs with at most five slopes*, Proc. of GD'06, Lecture Notes in Computer Science, 4372, pp. 114-125, 2006.

[KR07] M. R. Karim and M. S. Rahman, *Straight line grid drawings of planar graphs with linear area,* Proc. of APVIS07, pp. 109-112, 2007.

[Kur30] C. Kuratowski, *Sur le probleme des courbes gauches en topologie,* Fundamenta Math., 15, pp. 271-283, 1930

[LNS85] R. J. Lipton, S. C. North and J. S. Sandberg, *A method for drawing graphs*, Proc. of 1st Annu. ACM Sympos. Comput. Geom., pp. 153-160, 1985.

[MACLP95] J. Manning, M. Atallah, K. Cudjoe, J. Lozito and R. Pacheco, *A system for drawing graphs with geometric symmetry*, Proc. of GD'94, Lecture Notes in Computer Science, 894, pp. 262-265, Springer, 1995.

[MM96] K. Mehlhorn and P. Mutzel, *On the embedding phase of the Hopcroft and Tarjan planarity testing algorithm*, Algorithmica, 16, pp. 233-242, 1996.

[NC88] T. Nishizeki and N. Chiba, *Planar graphs: theory and algorithms*, North-Holland, Amsterdam, 1988.

[NR04] T. Nishizeki and M. S. Rahman, *Planar Graph Drawing*, Lecture Notes Series on Computing, 12, World Scientific Publishing Company, 2004.

[PCJ96] H. C. Purchase, R. F. Cohen and M. James, *Validating graph drawing aesthetics*, Proc. of GD'95, Lecture Notes in Computer Science, 1027, pp. 435-446, Springer, 1996.

[Pur97] H. C. Purchase, *Which aesthetic has the greatest effect on human understanding?* Proc. of GD'97, Lecture notes in computer science, 1353, pp. 248-261, Springer, 1997.

[Rah99] M. S. Rahman, *Efficient Algorithms for Drawing Planar Graphs*, Ph. D. Thesis, Graduate School of Information Sciences,

[REN05] M. S. Rahman, N. Egi, and T. Nishizeki, *No-bend orthogonal drawings of series-parallel graphs*, Proc. of GD'05, Lecture Notes in Computer Science, 3843, pp. 409-420. Springer, 2006.

[Sch90] W. Schnyder, *Embedding planar graphs on the grid*, Proc. First ACM-SIAM Symp. on Discrete Algorithms, San Francisco, pp. 138-148, 1990.

[SH99] W. K. Shih and W.-L. Hsu, *A new planarity test*, Theoretical Computer Science, 223, pp. 179-191, 1999.

[Ste51] K. S. Stein, *Convex maps*, Proc. Amer. Math. Soc., 2, pp. 464-466, 1951.

[STT81] K. Sugiyama, S. Tagawa and M. Toda, *Methods for visual understanding of hierarchical systems*, IEEE Trans. Syst. Man Cybern., SMC-11, 2, pp. 102-125, 1981.

[Wag36] K. Wagner, *Bemerkungen zum vierfarbenproblem*, Jahresber. Deutsch. Math-Verien., 46, pp. 26-32, 1936.

[Wes01] D. B. West, *Introduction to Graph Theory*, Prentice-Hall, Upper Saddle River, New Jersey, 2001.

Index