

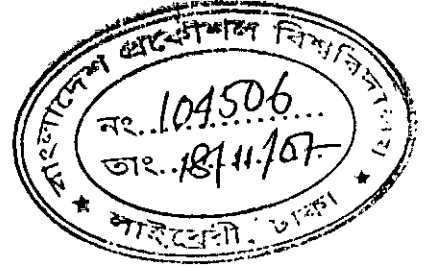
M.Sc. Engg. Thesis

1-Bend Orthogonal Drawings of  
Triconnected Planar 4-Graphs

by

Abu Hasnat Mohammad Ashfak Habib

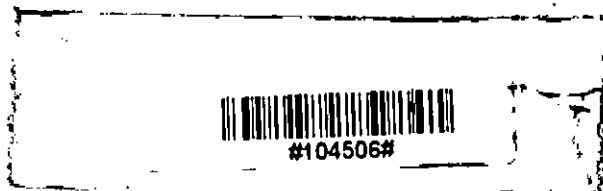
Submitted to



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

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

October 2007



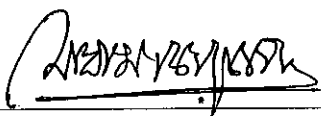
This thesis titled “1-Bend Orthogonal Drawings of Triconnected Planar 4-Graphs”, submitted by Abu Hasnat Mohammad Ashfak Habib, Roll No. 040505057F, Session April 2005, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory 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 October 31, 2007.

## Board of Examiners

1.   
\_\_\_\_\_

Dr. Md. Saidur Rahman  
Professor & Head  
Department of CSE  
BUET, Dhaka 1000

Chairman  
(Supervisor & Ex-officio)

2.   
\_\_\_\_\_

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

Member

3.   
\_\_\_\_\_

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

Member

4.   
\_\_\_\_\_

Dr. Md. Lutfar Rahman  
Professor  
Department of CSE  
Dhaka University, Dhaka 1000

Member  
(External)

# 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.



---

Abu Hasnat Mohammad Ashfaq Habib  
Candidate

# Contents

Board of Examiners	i
Candidate's Declaration	ii
Acknowledgments	vii
Abstract	viii
<b>1 Introduction</b>	<b>1</b>
1.1 Graph and Graph Drawing . . . . .	1
1.2 Graph Drawing Conventions . . . . .	2
1.2.1 Planar Drawing . . . . .	2
1.2.2 Straight Line Drawing . . . . .	3
1.2.3 Orthogonal Drawing . . . . .	3
1.2.4 Rectangular Drawing . . . . .	4
1.2.5 Grid Drawing . . . . .	5
1.3 Drawing Aesthetics . . . . .	6
1.4 Motivation and Literature Review . . . . .	7
1.5 Objectives of this Thesis . . . . .	11
1.6 Summary . . . . .	11
<b>2 Preliminaries</b>	<b>13</b>
2.1 Basic Terminology . . . . .	13

2.1.1	Graphs . . . . .	13
2.1.2	Subgraphs . . . . .	14
2.1.3	Paths and Cycles . . . . .	15
2.1.4	Connectivity . . . . .	15
2.1.5	Trees . . . . .	16
2.1.6	Planar Graphs . . . . .	16
2.2	Review of the Previous Results . . . . .	18
2.2.1	Rectangular Drawing . . . . .	19
2.2.2	Orthogonal Drawing . . . . .	20
<b>3</b>	<b>1-Bend Orthogonal Drawing</b>	<b>25</b>
3.1	Definitions . . . . .	25
3.1.1	$\alpha$ -Face . . . . .	25
3.1.2	Critical-Cycle . . . . .	26
3.1.3	Good String and Bad String . . . . .	26
3.1.4	Good Edge and Bad Edge . . . . .	27
3.1.5	Valid 4-Component . . . . .	28
3.2	Sufficient Condition . . . . .	29
3.2.1	An Outline of the Constructive Proof . . . . .	30
3.2.2	Inserting Dummy Vertices . . . . .	32
3.2.3	Inserting Dummy Edges . . . . .	36
3.2.4	Finding a Rectangular Drawing . . . . .	37
3.2.5	Patching the Feasible Orthogonal Drawings . . . . .	38
3.2.6	Contracting the Dummy Edges . . . . .	45
3.2.7	Proof for Sufficiency of Theorem 3.2.1 . . . . .	54
3.3	The Algorithm . . . . .	54
<b>4</b>	<b>Conclusion</b>	<b>58</b>

# List of Figures

1.1	Planar and non-planar drawings . . . . .	3
1.2	A straight line drawing . . . . .	3
1.3	A planar graph $G$ and an orthogonal drawing of $G$ . . . . .	4
1.4	A rectangular drawing . . . . .	5
1.5	A straight line grid drawing . . . . .	5
1.6	Two different diagrams of the same Computer Network . . . . .	8
1.7	Basic steps for finding a clear representation of a clumsy diagram . . . . .	9
1.8	Triconnected Planar 4-graphs having no 1-bend orthogonal drawing . . . . .	10
2.1	A graph with six vertices and eight edges . . . . .	14
2.2	A subgraph of the graph in Figure 2.1 . . . . .	14
2.3	Connected and disconnected graphs . . . . .	15
2.4	A tree . . . . .	16
2.5	Three planar embeddings of the same graph . . . . .	17
2.6	A plane graph $G$ . . . . .	18
2.7	3-legged and 4-legged cycles . . . . .	18
2.8	A plane graph $G$ and the rectangular drawing of $G$ . . . . .	19
2.9	Bad cycles, non bad cycles and maximal bad cycles . . . . .	20
2.10	Green paths . . . . .	21
2.11	A feasible drawing . . . . .	23
2.12	Cycles in $C_G$ and the corresponding genealogical tree $T_G$ . . . . .	24

3.1	An $\alpha$ -face $F$ . . . . .	26
3.2	Examples of critical-cycle . . . . .	26
3.3	Good strings and bad strings . . . . .	27
3.4	A triconnected plane 4-graph $G$ and the 4-components of $G$ . . . . .	29
3.5	An overview of the constructive proof of Theorem 3.2.1 . . . . .	31
3.6	Labeling the vertices and edges of the strings and 4-components . . . . .	35
3.7	Replacing the outer vertices of degree four with the dummy edges . . . . .	37
3.8	A three legged cycle obtained from a cycle having more than three legs . . . . .	40
3.9	Desired-paths . . . . .	41
3.10	$F$ , $H$ , $D(H)$ and $D(G(C))$ for case 3 . . . . .	44
3.11	Nine possible shapes of a dummy edge in the orthogonal drawing $D$ . . . . .	46
3.12	Contracted forms of the arrangements in Figure 3.11 . . . . .	47
3.13	Effect of dummy edge contraction for Case 1 . . . . .	49
3.14	Effect of dummy edge contraction for Case 2 . . . . .	51
3.15	The cycle of the non-tree component in Figure 3.6(d) is broken at $e_{n0}$ . . . . .	53

# Acknowledgments

At first, I would like to express my profound gratitude to Allah, the almighty and the merciful, for giving me adequate physical and mental strength for doing this research work.

I am grateful to my supervisor Professor Dr. Md. Saidur Rahman for allowing me to carry out this thesis under his supervision. In the field of graph theory Professor Dr. Rahman is a well known researcher at both home and abroad. I would like to thank him for teaching me the fundamentals for doing research work. I express my gratitude for his patience in reviewing my inferior drafts, for correcting my linguistic errors and for showing me the new ways of thinking.

I would also like to thank the members of the board of examiners, Professor Dr. M. Kaykobad, Dr. Masud Hasan and Professor Dr. Lutfar Rahman, for their valuable suggestions.

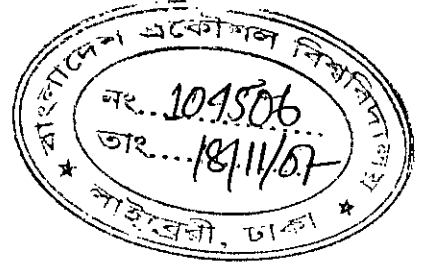
Moreover, my special thanks goes to the members of our research group for their valuable comments and suggestions. I thank my friends and colleagues who have given me their time, companionship, professional and personal help.

I am especially indebted to my parents and my family members for their continuous support and encouragement. Without their cooperation this work would have been hardly possible.



# Abstract

An orthogonal drawing of a planar graph is a drawing of the graph in which each vertex is mapped to a point, each edge is drawn as a sequence of alternate horizontal and vertical line segments, and any two edges do not cross except at their common end. In an orthogonal drawing, a bend is a point at which the drawing of an edge changes its direction. Every planar 4-graph i.e. a graph with vertices of the maximum degree at most four, has an orthogonal drawing, but may need bends. If the bend per edge of an orthogonal drawing is at most  $k$ , then we call the drawing a  $k$ -bend orthogonal drawing. It is already known that every planar 4-graph except an octahedron has a 2-bend orthogonal drawing. Moreover, it is an NP-complete problem to decide whether a given planar 4-graph has a 0-bend orthogonal drawing or not. In this thesis we consider 1-bend orthogonal drawings for planar 4-graphs. But unfortunately not every planar 4-graph has a 1-bend orthogonal drawing. That is why it is also interesting to find which class of planar 4-graphs has 1-bend orthogonal drawings. In this thesis we give a sufficient condition for a triconnected planar 4-graph to have a 1-bend orthogonal drawing. Furthermore, we give a linear-time algorithm for finding 1-bend orthogonal drawings of those graphs which satisfy the sufficient condition.



# Chapter 1

## Introduction

In this chapter we give an overview on the field of graph and graph drawing. Moreover, we discuss the drawing conventions and drawing aesthetics. At the end of this chapter we give an outline of this thesis.

### 1.1 Graph and Graph Drawing

By the word *Graph* people usually think about statistical graphs containing bars, lines, circles, etc. or a graph of a function drawn with respect to two or more axes. But in graph theory a graph is a mathematical model consists of a set of vertices and set of edges that can illustrate a real life problem or a situation.

Most of the problems of various research areas and most of the situations around us are actually some relations between the objects involved. A graph represents these relations between objects and thus makes complex systems easily understandable and describable. In comparison with other representations graphs are often not only easier but also shorter and more exact to define relations between objects. A *drawing of a graph* can be thought of as a diagram where the objects, called nodes or vertices, are normally drawn as points, circles, squares, or other simple geometric figures and the relations between objects, called

edges, are normally drawn as line segments, each connecting two vertices.

Graphs can be used not only in computer science but also in other sciences, business, management, and many more domains. A scientist can use graphs to model the molecular structures of atoms. A network administrator can use graphs to represent computer networks. A business man can use graphs to keep track of warehouses and retail outlets. Road transport authority can model the road networks with the help of graphs. And the list goes on. A graph can be drawn in various ways. In the next section we describe the styles of graph drawings.

## 1.2 Graph Drawing Conventions

In this section we describe the common conventions or styles of graph drawings which are largely studied in the various literatures due to their huge applications. In graph drawing vertices can be represented by various symbols but here we assume that vertices are represented by points and edges are represented by line segments. We now introduce the following drawing conventions.

### 1.2.1 Planar Drawing

A *planar drawing* is a drawing of a graph in which any two edges do not intersect at any point except at their common end vertex. Figures 1.1(a) and 1.1(b) illustrate a planar drawing and a non-planar drawing of the same graph respectively. If a graph has a planar drawing, then it is preferable to find it. Because planar drawings are relative easy to understand in comparison with the non-planar drawings. Unfortunately not every graph has a planar drawing. If a graph has a planar drawing, then it is called a *planar graph*.

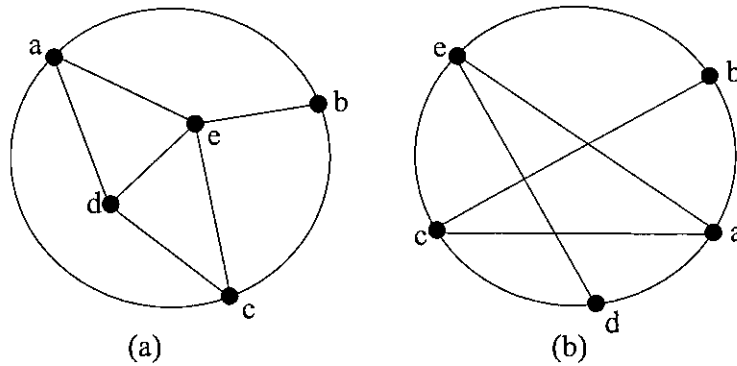


Figure 1.1: (a) A planar drawing, and (b) a non-planar drawing of the same graph.

### 1.2.2 Straight Line Drawing

Straight line drawing is one of the earliest graph drawing conventions. It is natural to draw each edge of a graph as a straight line between its end vertices and a drawing of a graph in which each edge is drawn as a straight line segment is called a *straight line drawing*. Figure 1.2 depicts a straight line drawing of a graph.

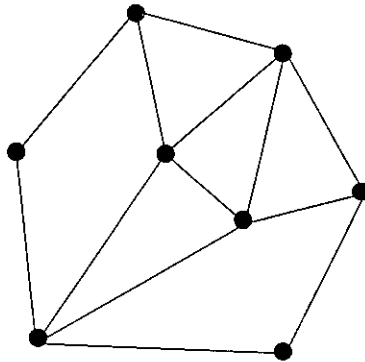


Figure 1.2: A straight line drawing.

### 1.2.3 Orthogonal Drawing

An *orthogonal drawing* of a planar graph is a drawing of that graph in which each vertex is mapped to a point, each edge is drawn as a sequence of alternate horizontal and vertical

line segments and any two edges do not cross except at their common end. Figure 1.3(b) illustrates the orthogonal drawing of the planar graph shown in Figure 1.3(a). In case of orthogonal drawing a *bend* is a point where the drawing of an edge changes its direction.

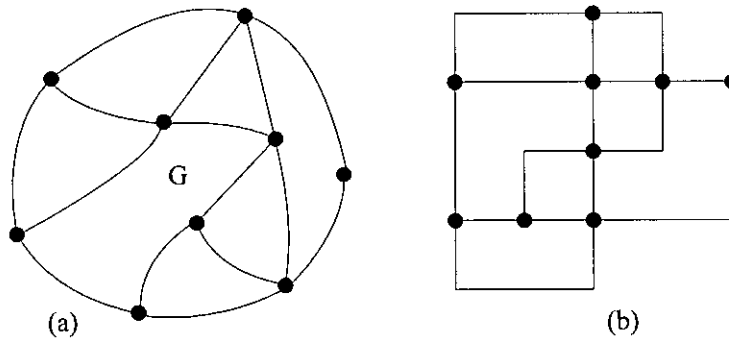


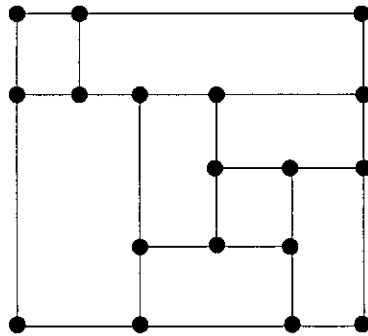
Figure 1.3: (a) A planar graph  $G$ , and (b) an orthogonal drawing of  $G$ .

Orthogonal drawings have numerous applications in various fields like VLSI circuit layouts, database management systems, software engineering, cartography etc. By virtue of their multifarious applications orthogonal drawings have drawn much attention of the researchers. Researchers are working on both two dimensional (2D) and three dimensional (3D) orthogonal drawings. But in this thesis we have worked on 2D orthogonal drawings and we have used the term *orthogonal drawing* to represent 2D orthogonal drawings only.

#### 1.2.4 Rectangular Drawing

A *rectangular drawing* of a planar graph  $G$  is a drawing of  $G$  such that each vertex is drawn as a point, each edge is drawn as a horizontal or vertical line segment without edge-crossings and each face is drawn as a rectangle as shown in Figure 1.4. Moreover, a rectangular drawing is an orthogonal drawing in which there is no bend and each face is drawn as a rectangle.

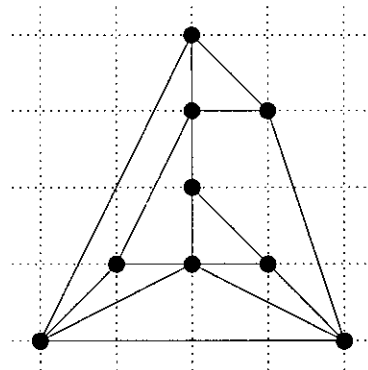
Not every planar graph has a rectangular drawing. A planar graph  $G$  has a rectangular drawing if at least one embedding of  $G$  has a rectangular drawing. Rahman et al. [RNN98]

Figure 1.4: *A rectangular drawing.*

derived a necessary and sufficient condition for a class of planar graphs to have rectangular drawings.

### 1.2.5 Grid Drawing

A drawing of a graph in which vertices and bends are located at grid points of an integer grid as illustrated in Figure 1.5 is called a *grid drawing*. Grid drawing is useful for drawing the embeddings on the raster devices. Moreover, area requirement for a grid drawing can be calculated and can be compared with the other grid drawings.

Figure 1.5: *A straight line grid drawing.*

## 1.3 Drawing Aesthetics

A graph has an infinite number of drawings. Some drawings are better than others in conveying information on the graph. Various criteria have been proposed in the literature to evaluate the aesthetic quality of a drawing. In this section we introduce some aesthetic criteria of graph drawings which we generally consider [NR04].

**Edge Crossings** An edge crossing is a point where two edges intersect each other. Each edge crossing of a graph is a source of confusion. Therefore, it is better to keep the number of edge crossings of a graph minimum. Moreover, in case of VLSI circuit design less number of edge crossings can minimize the number of layers of semiconductors.

**Area** Area of a drawing means the area of the convex hull of that drawing. If the area of a drawing is too small, then the drawing becomes unreadable. On the contrary if the area becomes too large, then the drawing cannot be accommodated in a single page or in the monitor. Both of these situations should also be avoided. Therefore, one major objective is to minimize the area of a drawing up to a certain limit. For VLSI floorplanning smaller area drawing is preferred because it helps us to avoid wasting of valuable space in the chip.

**Bends** In a polyline drawing a bend is a point where an edge changes its direction. It implies that the increased number of bends on an edge increases the difficulties of following the course of the edge. Therefore, the number bends per edge should be kept small. Moreover, bends increase the manufacturing cost of a VLSI chip. That is why the total number of bends of a drawing should also be kept small.

**Aspect Ratio** The aspect ratio of a drawing is the ratio of the length of the longest side to the length of the shortest side of the smallest rectangle which encloses that drawing. A drawing with high aspect ratio may not be conveniently placed on a

workstation screen, even if it has modest area. Hence, it is important to keep the aspect ratio small.

**Angular Resolution** The angular resolution of a polyline drawing is the smallest angle formed by two adjacent edges or two successive segments of an edge of that drawing. It is desirable to maximize the angular resolution for displaying a drawing on a raster device.

**Shape of Faces** A drawing in which every face has a regular shape looks better than a drawing having faces of irregular shape. For VLSI floorplanning it is desirable that each face is drawn as a rectangle.

**Symmetry** Symmetry is an important aesthetic criteria in graph drawing. Symmetry of a two-dimensional figure is an isometry of the plane that fixes the figure. Where possible, a symmetric view of the graph should be displayed. Because, increasing the local symmetry displayed in a graph drawing increases the understandability of the graph.

## 1.4 Motivation and Literature Review

In this section we give the motivation of the problem considered in this thesis and mention some related results.

Assume that a Network Administrator is trying to establish a computer network and he has drawn the diagram of the proposed network by hand as shown in Figure 1.6(a). The same computer network can also be represented by a diagram as shown in Figure 1.6(b). Among these two diagrams the second diagram is obviously easier to understand. Because in the second diagram the relations between the network components are relatively clear. When the number of components and interconnections are small then it is feasible for a person to draw a clear diagram as shown in Figure 1.6(b), from a clumsy diagram which



is shown in Figure 1.6(a). For a computer network having thousands of components and interconnections and hundreds of overlappings of the interconnections the task becomes almost impossible for a person to carry out. We can use machine (computer) to solve such type of problems. But the challenging issue is, how machines can solve such a problem.

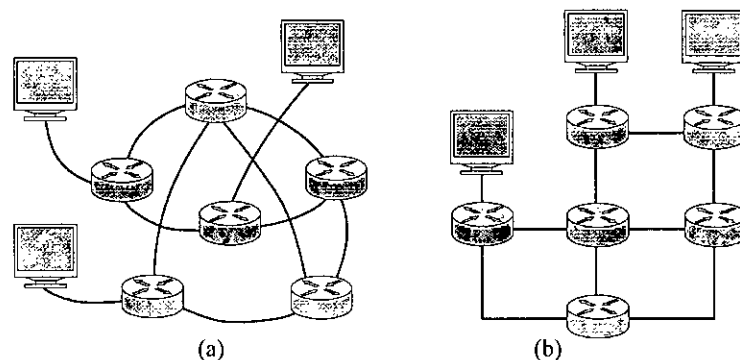


Figure 1.6: *Two different diagrams of the same Computer Network.*

The basic methodology for finding a clear diagram from a given clumsy diagram is already known. Suppose the clumsy diagram in Figure 1.7(a) is given. At first we can represent the given diagram by a graph as shown in Figure 1.7(b) by representing each of the components by a vertex and each of the interconnections by an edge. For identification we can number the network components arbitrarily and then put similar numbers to the corresponding vertices of the obtained graph. Secondly, we can find an orthogonal drawing for the obtained graph as illustrated in Figure 1.7(c), provided the obtained graph is a planar graph. Finally, we can get the resultant clear diagram as shown in Figure 1.7(d) by replacing each of all the vertices and edges by their corresponding network components and interconnections, respectively. Therefore, we can solve the problem through these three major steps, as shown in Figure 1.7. The first and the last steps are simple transformations. The most critical step is the second step i.e. to find an orthogonal drawing of the obtained graph. This is just an example of a problem domain but lot of similar problems can be solved in a similar fashion. For solving those problems the main difficulty is to find an orthogonal drawing of a given graph. This is the general problem

we have studied in this thesis.

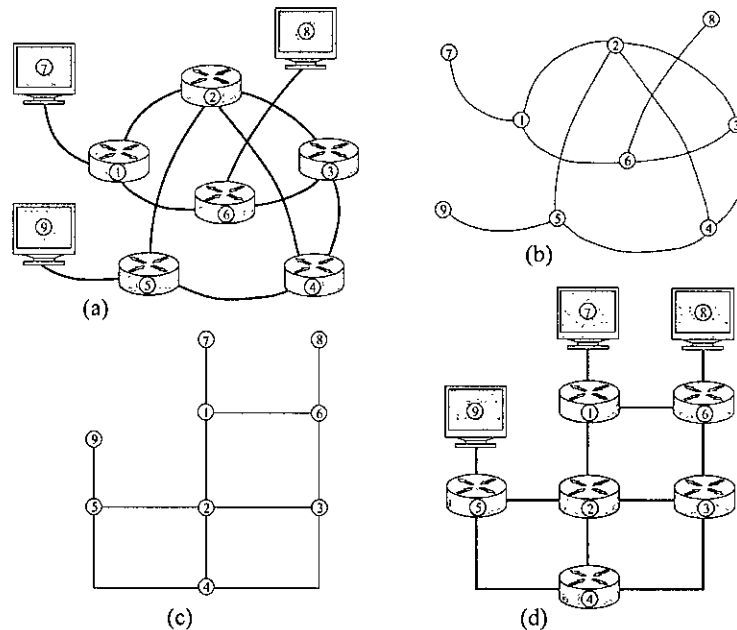


Figure 1.7: *Basic steps for finding a clear diagram of a given clumsy diagram.*

We now investigate the relevant works which have already been done on this topic. Due to the huge applications of orthogonal drawings in various fields like VLSI circuit design, database systems, software engineering, cartography, etc. [NR04, She95] numerous algorithms for finding orthogonal drawings have been studied in the literature. Many relevant works have been done on 3-graphs. In the year 1996 Kant [Kan96] proved that every planar 3-graph, except  $K_4$ , has a 1-bend orthogonal drawing. In 1999 Rahman et al. [RNN99] developed an algorithm for finding a bend-optimal orthogonal drawing for a triconnected cubic graph. In 2002 Rahman and Nishizeki [RN02] introduced an algorithm for finding a bend-minimum orthogonal drawing for a plane 3-graph. In 2003 Rahman et al. [RNN03] developed another algorithm for finding a 0-bend orthogonal drawing for a plane 3-graph if 0-bend orthogonal drawing exists. Here we have mentioned only some of the selected good results regarding 3-graphs where all the mentioned algorithms are of linear-time.

This thesis deals with 4-graphs and the results regarding 4-graphs are as follows. In 1998 Biedl and Kant [BK98] proved that every planar 4-graph, except an octahedron, has a 2-bend orthogonal drawing. In the same year Liu et al. [LMS98] developed same result with the only difference that they considered biconnected planar 4-graphs. In 2001 Garg and Tamassia [GT01] found that it is an NP-complete problem to decide whether a given planar 4-graph has a 0-bend orthogonal drawing or not. Recently in 2006 Tayu et al. [TNU06] developed an  $O(n^2)$  time algorithm for finding 1-bend orthogonal drawing for every series parallel 4-graph. In this thesis we work on triconnected planar 4-graph which is a larger class of graphs than series parallel 4-graphs. The target of this thesis is to develop a linear-time algorithm for finding a 1-bend orthogonal drawing for a triconnected planar 4-graph. But unfortunately not every triconnected planar 4-graph has a 1-bend orthogonal drawing. Figure 1.8 illustrates two examples of triconnected planar 4-graphs which do not have any 1-bend orthogonal drawing. Therefore, it is interesting to derive a condition for identifying the triconnected planar 4-graphs to have 1-bend orthogonal drawings.

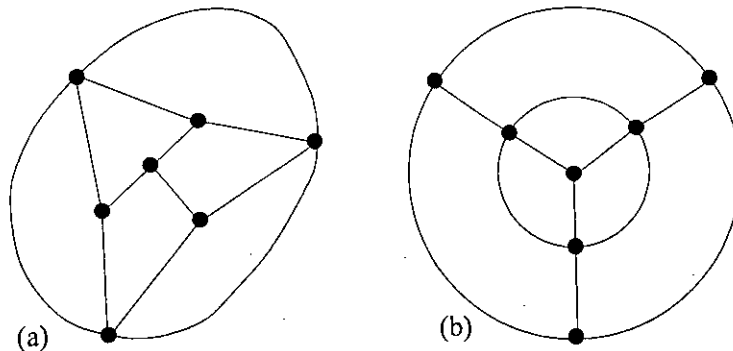


Figure 1.8: *Two examples of triconnected Planar 4-graphs which do not have 1-bend orthogonal drawings.*

## 1.5 Objectives of this Thesis

In this thesis we introduce the objectives of this thesis.

The problem we work on is an orthogonal drawing generating problem. We are concerned about finding 1-bend orthogonal drawings of triconnected planar 4-graphs. The specific objectives of this thesis are as follows:

- a) We have to derive a sufficient condition for a triconnected planar 4-graph to have a 1-bend orthogonal drawing.
- b) We have to develop a linear-time algorithm for finding one bend orthogonal drawings of those triconnected planar 4-graphs which satisfy the sufficient condition mentioned above.

## 1.6 Summary

In this section we summarize the results obtained in this thesis. Moreover, at the end of this section we give an outline of this thesis.

The results of this thesis are summarized as follows:

- a) We are concerned about the triconnected planar 4-graphs. By extensively examining the characteristics of triconnected planar 4-graphs we have derived a sufficient condition for triconnected planar 4-graphs to have 1-bend orthogonal drawings.
- b) We have given a constructive proof for the sufficient condition and from that constructive proof we have developed a linear-time algorithm for finding one bend orthogonal drawings of those triconnected planar 4-graphs which satisfy the sufficient condition.

The remainder of this thesis is organized as follows. In chapter 2 we give the preliminary definitions and mention two results that we use for finding our desired algorithm.

In chapter 3 we introduce the sufficient condition and give a proof for the sufficiency of the condition. We also give an algorithm at the end of that chapter. In chapter 5 we conclude the thesis with an outline of future research works.

# Chapter 2

## Preliminaries

### 2.1 Basic Terminology

In this section we give the definitions of those graph-theoretical terms which we have used throughout the remainder of this thesis.

#### 2.1.1 Graphs

A *graph*  $G$  is represented by  $G = (V, E)$  where  $V$  be the finite set of vertices of  $G$  and  $E$  be the finite set of edges of  $G$ . The finite set of vertices and edges of  $G$  are often denoted by  $V(G)$  and  $E(G)$ , respectively. Moreover, we denote the number of vertices of  $G$  by  $n(G)$  and the number of edges of  $G$  by  $m(G)$ , i.e.  $n(G) = |V(G)|$  and  $m(G) = |E(G)|$ . Figure 2.1 depicts a graph  $G$  where  $V(G) = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ ,  $E(G) = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$ ,  $n(G) = 6$  and  $m(G) = 8$ .

If more than one edges join the same pair of vertices, then the edges are called *multiple edges*. Besides a *loop* is an edge which joins a vertex itself. If a graph  $G$  has no multiple edges or loops, then  $G$  is called a *simple graph*. All the graphs, which we consider in this thesis, are simple graphs.

We denote by  $d_G(v)$  the degree of a vertex  $v$  in  $G$ , which represents the number of

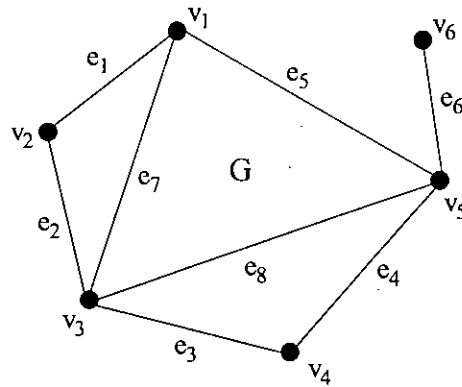


Figure 2.1: A graph with six vertices and eight edges.

edges incident to  $v$  in  $G$ . We also denote by  $\Delta(G)$  the maximum degree of vertices of  $G$ . Moreover,  $G$  is called a  $k$ -graph if  $\Delta(G) \leq k$ . The graphs of our interest are 4-graphs where the maximum degree of the vertices is at most 4. The graph in Figure 2.1 is an example of 4-graph.

### 2.1.2 Subgraphs

A *subgraph* of a graph  $G$  is a graph  $G'$  in which the vertex and edge sets are the subsets of those of  $G$ . Figure 2.2 illustrates a subgraph  $G'$  of the graph  $G$  in Figure 2.1.

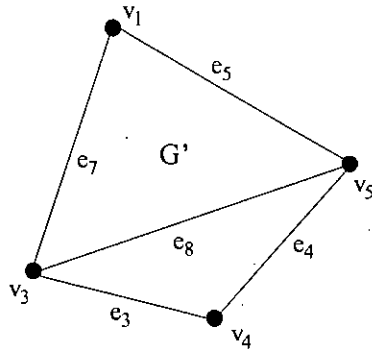


Figure 2.2: A subgraph of the graph in Figure 2.1.

### 2.1.3 Paths and Cycles

A *walk*,  $v_0, e_1, v_1, \dots, v_{l-1}, e_l$ , and  $v_l$  in a graph  $G$  is an alternating sequence of vertices and edges of  $G$ , beginning and ending with a vertex, in which each edge is incident to two vertices immediately preceding and following it. If the vertices  $v_0, v_1, \dots, v_l$  are distinct (except possibly  $v_0, v_l$ ), then the walk is called a *path* and usually denoted either by the sequence of vertices  $v_0, v_1, \dots, v_l$  or by the sequence of edges  $e_0, e_1, \dots, e_l$ . 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*. If a closed path contains exactly one edge, then it is called a loop.

### 2.1.4 Connectivity

A graph  $G$  is *connected* if for any two distinct vertices  $u$  and  $v$  there is a path between  $u$  and  $v$  in  $G$ . The graph in Figure 2.3(a) is a connected graph. A graph which is not connected is called a *disconnected* graph. The graph in Figure 2.3(b) is a disconnected graph because there exist no path in between  $v_1$  and  $v_9$ .

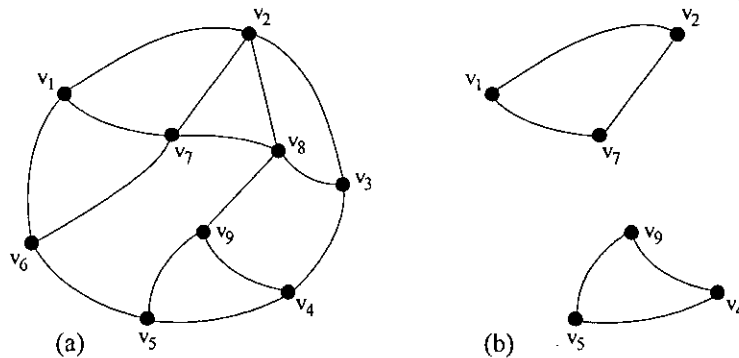


Figure 2.3: (a) A connected graph, (b) a disconnected graph.

The *connectivity*  $\kappa(G)$  of a graph  $G$  is the minimum number of vertices whose removal results in a disconnected graph or a single vertex graph.  $G$  is called *k-connected* if  $\kappa(G) \geq k$ . The graph in Figure 2.3(a) is a 3-connected graph because at least three vertices should



be removed to make this graph disconnected. We can make this graph disconnected by removing  $v_3$ ,  $v_6$  and  $v_8$  as shown in Figure 2.3(b). Note that 3-connected graphs are also known as *triconnected* graphs.

### 2.1.5 Trees

A *tree* is a simple connected graph which contains no cycle. The vertices in a tree are usually called *nodes*. Every vertex of degree one of a tree is called a *leaf*. The nodes other than the leaf nodes of a tree are the *internal nodes*. If a node of a tree is designated, then the tree is called a *rooted tree* and the designated node is called the *root* of the tree. Usually, the root of a tree is drawn at the top. Figure 2.4 illustrates a tree in which  $v_1$  is the root,  $v_5$ ,  $v_6$  and  $v_7$  are the leaf nodes and rest of the vertices are the internal nodes.

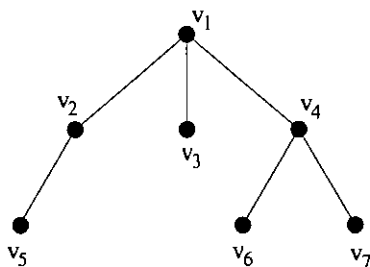


Figure 2.4: A tree.

### 2.1.6 Planar Graphs

A graph is *planar* if it has at least one embedding in the plane such that no two edges intersect at any point except at their common end vertex. A planar graph may have an exponential number of planar embeddings. Figure 2.5 illustrates three different planar embeddings of the same planar graph.

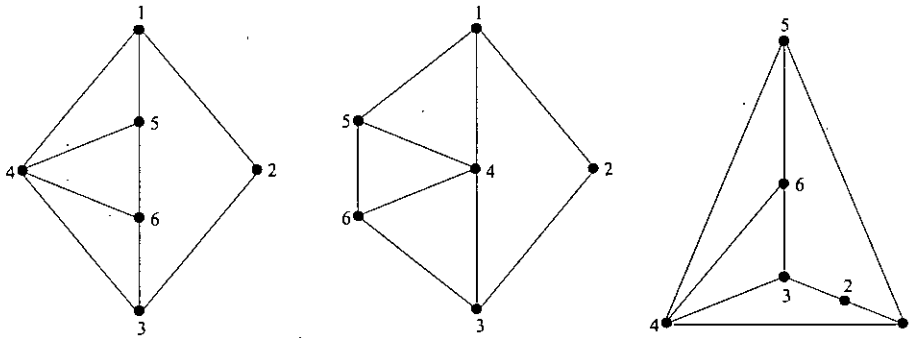


Figure 2.5: Three planar embeddings of the same graph.

### Plane Graphs

A *plane graph*  $G$  is a planar graph with a fixed planar embedding in the plane as shown in Figure 2.6. A planar embedding of a graph divide the plane into some connected regions called *faces*. The shaded region of the graph  $G$  in Figure 2.6 is an example of a face. The outermost unbounded region of a plane graph is called the *outer face* of the graph. If a graph is 2-connected and if it has at least three vertices, then the boundary of each face of the graph is a cycle [NR04]. The boundary of the outer face of  $G$  is called the *outer boundary* of  $G$  and denoted by  $C_o(G)$ . If  $C_o(G)$  is a cycle, then  $C_o(G)$  is called the *outer cycle* of  $G$ . Any vertex  $v$  on  $C_o(G)$  is called an *outer vertex* of  $G$ ; otherwise  $v$  is called an *inner vertex* of  $G$ . The vertices  $v_7, v_8,$  and  $v_9$  of the graph  $G$  in Figure 2.6 are the inner vertices and rest of the vertices are the outer vertices of  $G$ . Similarly, any edge  $e$  on  $C_o(G)$  is called an *outer edge* of  $G$ ; otherwise  $e$  is called an *inner edge* of  $G$ .

For a cycle  $C$  in a simple graph  $G$  we denote by  $G(C)$  the plane subgraph of  $G$  inside  $C$ , including  $C$ . An edge of  $G$  which is incident to exactly one vertex of a cycle  $C$  and located outside  $C$  is called a *leg* of the cycle  $C$ . The vertex of  $C$  to which a leg is incident is called a *leg-vertex* of  $C$ . A cycle  $C$  in  $G$  is called a *k-legged cycle* of  $G$  if  $C$  has exactly  $k$  legs in  $G$ .

We assume that  $G$  is a simple triconnected planar 4-graph. Therefore, in case of  $G$  no 1-legged cycle or 2-legged cycle can exist. Any cycle  $C$  in  $G$  must have at least three legs

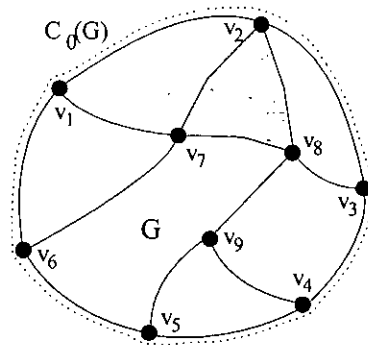


Figure 2.6: A plane graph  $G$ .

which are incident to three different leg vertices. Note that since  $G$  is a 4-graph, more than one and at most two legs can be incident to the same leg vertex of  $C$ . The cycles  $C_1$ ,  $C_2$ , and  $C_3$  in Figure 2.7, which are indicated by the dotted lines, are 3-legged cycles. But the cycle  $C_4$  in Figure 2.7 is a 4-legged cycle. In Figure 2.7 the leg vertices are colored with gray color.

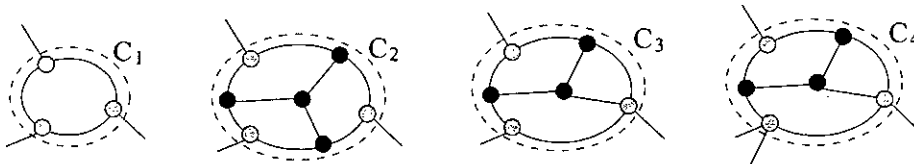


Figure 2.7: 3-legged and 4-legged cycles.

For a simple cycle  $C$  in a plane graph  $G$  we denote by  $G(C)$  the plane subgraph of  $G$  inside  $C$  (including  $C$ ). We say that cycles  $C$  and  $C'$  in a plane graph  $G$  are *independent* if  $G(C)$  and  $G(C')$  have no common vertex.

## 2.2 Review of the Previous Results

In this section we present two known algorithms which we have used in our proposed algorithm. The first one is a rectangular drawing algorithm and the second one is an algorithm for finding bend minimal orthogonal drawings.

### 2.2.1 Rectangular Drawing

A *rectangular drawing* of a plane graph is a drawing of that graph such that each vertex is drawn as a point, each edge is drawn as a horizontal or vertical line segment without edge-crossings, and each face is drawn as a rectangle. The graph in Figure 2.8(a) is a plane graph  $G$  and Figure 2.8(b) illustrates the rectangular drawing of  $G$ .

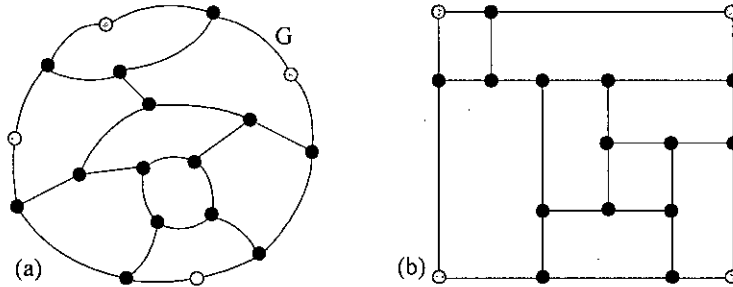


Figure 2.8: (a) A plane graph  $G$ , (b) rectangular drawing of  $G$ .

Not every planar graph has a rectangular drawing. A planar graph  $G$  has a rectangular drawing if at least one embedding of  $G$  has a rectangular drawing. Rahman et al. [RNN98] derived a necessary and sufficient condition for a class of plane graphs to have rectangular drawings. They proved the following lemma.

**Lemma 2.2.1** [RNN98] *Let  $G$  be a connected plane graph such that all vertices are of degree three except four vertices of degree two on the outer cycle  $C_o(G)$  of  $G$ . Then  $G$  has a rectangular drawing if and only if  $G$  has none of the following three types of simple cycles [Tho84]:*

- (r1) 1-legged cycles;
- (r2) 2-legged cycles which contain at most one vertex of degree two; and
- (r3) 3-legged cycles which contain no vertex of degree two.

Furthermore one can check in linear-time whether  $G$  satisfies the condition above and if  $G$  does, then one can find a rectangular drawing of  $G$  in linear-time.

Linear-time algorithms for finding a rectangular drawing of a plane graph satisfying the condition in Lemma 2.2.1 have been obtained in [KH97] and [RNN98]. Our proposed drawing algorithm uses the algorithm in [RNN98]. In this thesis we call the algorithm of [RNN98] as **Rectangular-Draw**.

A cycle of type (r1) or (r2) or (r3) of Lemma 2.2.1 is called a *bad cycle*. Since in this thesis we consider triconnected planar graphs, no 1-legged or 2-legged cycle can exist in the given graph. Figure 2.9 illustrates some 3-legged cycles where the cycles  $C_1$ ,  $C_2$  and  $C_5$  contain no vertex of degree two. Therefore, these three cycles are bad cycles. Whereas the cycles  $C_3$  and  $C_4$  contain one and two vertices of degree two respectively. That is why these two cycles are not the bad cycles. Now a bad cycle which is not contained in another bad cycle is called a *maximal bad cycle*. The cycles  $C_1$  and  $C_5$  in Figure 2.9 are maximal bad cycles. Whereas the cycle  $C_2$  is not a maximal bad cycle because it is contained in another bad cycle  $C_5$ .

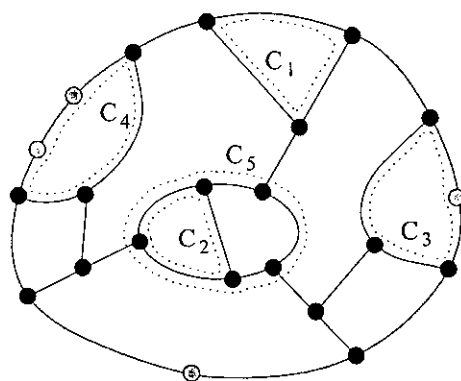


Figure 2.9: *Bad cycles ( $C_1$ ;  $C_2$ ;  $C_5$ ), non bad cycles ( $C_3$ ;  $C_4$ ) and maximal bad cycle ( $C_1$ ;  $C_5$ ).*

## 2.2.2 Orthogonal Drawing

In the year 1999 Rahman et al. [RNN99] developed a linear time algorithm for finding bend-optimal orthogonal drawings of triconnected cubic plane graphs. A part of that

algorithm is another algorithm which can find an orthogonal drawing of a maximal bad cycle such that the drawing can be patched into another orthogonal drawing. This part is termed as **Feasible-Draw** in [RNN99] and in our proposed algorithm we use this algorithm. Before discussing the algorithm **Feasible-Draw** we need to present following definitions.

### Green Path and Red Path

Let  $G$  be a triconnected cubic plane graph and  $C$  be a 3-legged cycle in  $G$ . Each 3-legged cycle like  $C$  in  $G$  is divided into three paths  $P_1$ ,  $P_2$  and  $P_3$  by the three leg-vertices  $x$ ,  $y$  and  $z$  of  $C$  as illustrated in Figure 2.10. These three paths  $P_1$ ,  $P_2$  and  $P_3$  are called the *contour paths* of  $C$ . Each contour path of  $C$  is classified as either a *green path* or a *red path*. Green paths and red paths are defined [RNN99] from the following three cases.

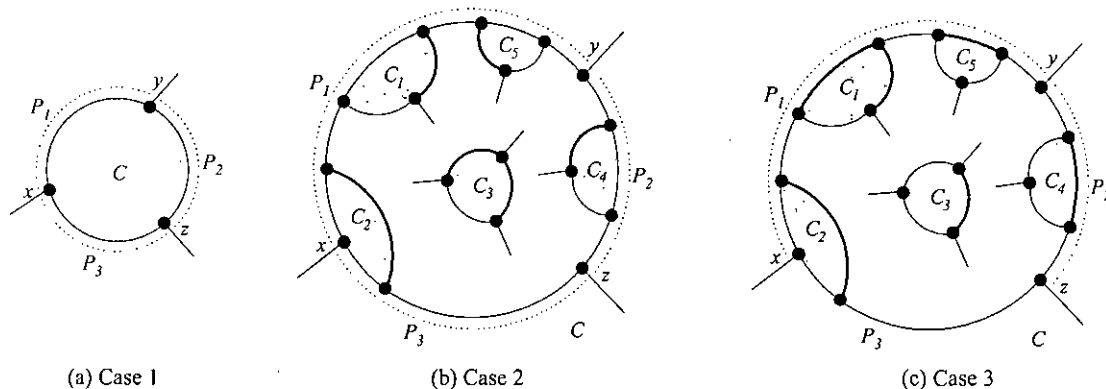


Figure 2.10: *Green paths.*

**Case 1:**  $C$  has no child cycle as shown in Figure 2.10(a). In this case all the three contour paths of  $C$  are classified as green paths. In Figure 2.10(a) green paths are indicated by dotted lines.

**Case 2:** Let  $C_1, C_2, \dots, C_l$  be the child-cycles of  $C$  and none of the child-cycles of  $C$  has a green path on  $C$ . In this case all the three contour paths of  $C$  are classified as green paths. In Figure 2.10(b) the child-cycles of  $C$  are  $C_1, C_2, \dots, C_5$ , and all green paths of

them, drawn by thick lines, do not lie on  $C$ . Therefore, all the three contour paths of  $C$  are the green paths and those are indicated by dotted lines.

**Case 3:** Let  $C_1, C_2, \dots, C_l$  be the child-cycles of  $C$  and at least one of the child-cycles of  $C$  has a green path on  $C$ . In this case a contour path  $P_i$ ,  $1 \leq i \leq 3$ , is classified as a green path if a child-cycle of  $C$  has its green path on  $P_i$ . Otherwise,  $P_i$  is classified as a red path. In Figure 2.10(c)  $P_1$  and  $P_2$  are green paths but  $P_3$  is a red path.

### Feasible Orthogonal Drawing

Let  $C$  be a 3-legged cycle in a triconnected cubic plane graph  $G$  and  $x, y$  and  $z$  be the three leg vertices of  $C$  in  $G$ . One may assume that  $x, y$  and  $z$  appear on  $C$  in clockwise order. For a green path  $P$ , with ends  $x$  and  $y$  on  $C$ , an orthogonal drawing of  $G(C)$  is defined [RNN99] to be *feasible* for  $P$  if the drawing satisfies the following properties (p1)-(p2):

(p1) At least one bend appears on the green path  $P$ .

(p2) The drawing of  $G(C)$  intersects none of the following six open halflines.

- the vertical open halfline with the upper end at  $x$ .
- the horizontal open halfline with the right end at  $x$ .
- the vertical open halfline with the lower end at  $y$ .
- the horizontal open halfline with the left end at  $y$ .
- the vertical open halfline with the upper end at  $z$ .
- the horizontal open halfline with the left end at  $z$ .

The property (p2) implies that in the drawing of  $G$  any vertex of  $G(C)$ , except  $x, y$  and  $z$ , is located in none of the following three areas (shaded in Figure 2.11): the third quadrant with the origin  $x$ , the first quadrant with the origin  $y$ , and the fourth quadrant

with the origin  $z$ . It should be noted that each leg of  $C$  must start with a line segment on one of the six open halflines above if an orthogonal drawing of  $G$  is extended from an orthogonal drawing of  $G(C)$  feasible for  $P$ . Figure 2.11 illustrates an orthogonal drawing feasible for a green path  $P$ . An orthogonal drawing of  $G(C)$  feasible for a green path of  $C$  is often simply called a *feasible orthogonal drawing* of  $G(C)$ .

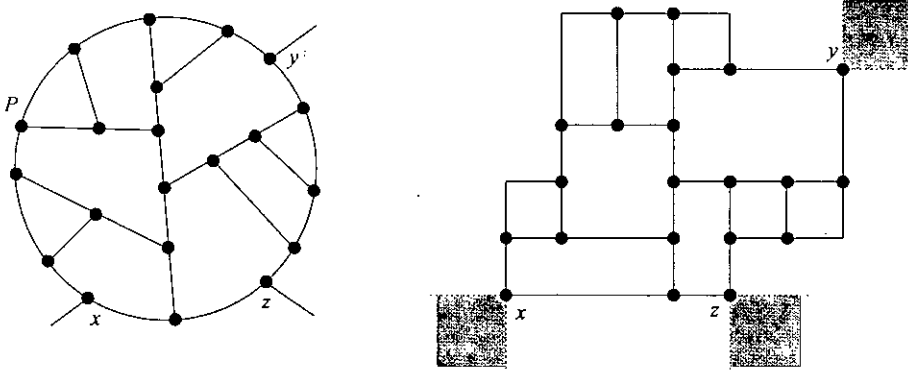


Figure 2.11: A feasible drawing.

Let us present the results regarding the feasible orthogonal drawing which were actually developed and proved by Rahman et al. [RNN99]. They proved the following lemma.

**Lemma 2.2.2** [RNN99] *Let  $G$  be a triconnected cubic plane graph. For any 3-legged cycle  $C$  of  $G$  and any green path  $P$  of  $C$ ,  $G(C)$  has an orthogonal drawing feasible for  $P$ .*

The algorithm for finding a feasible orthogonal drawing of  $G(C)$  is described in the proof of Lemma 2.2.2 and that algorithm is called **Feasible-Draw**. We also have the following lemma regarding **Feasible-Draw**.

**Lemma 2.2.3** [RNN99] *Algorithm Feasible-Draw finds a feasible orthogonal drawing of  $G(C)$  for a 3-legged cycle  $C$  in linear-time, that is, in time  $O(n(G(C)))$ .*

Rahman et al. [RNN99] also defined the following term which we also use for describing our proposed algorithm.



**Genealogical Tree**

Let  $C$  be a 3-legged cycle in a triconnected cubic plane graph  $G$ . Then the three leg-vertices of  $C$  are distinct with each other since  $G$  is cubic. We denote by  $C_C$  the set of all 3-legged cycles of  $G$  in  $G(C)$  including  $C$  itself. For the cycle  $C$  in Figure 2.12(a)  $C_C = C, C_1, C_2, \dots, C_7$ , where all cycles in  $C_C$  are drawn by thick lines. For any two 3-legged cycles  $C'$  and  $C''$  in  $C_C$ , we say that  $C''$  is a *descendant cycle* of  $C'$  and  $C'$  is an *ancestor cycle* of  $C''$  if  $C''$  is contained in  $G(C')$ . We also say that a descendant cycle  $C''$  of  $C'$  is a *child-cycle* of  $C'$  if  $C''$  is not a descendant cycle of any other descendant cycle of  $C'$ . In Figure 2.12(a) cycles  $C_1, C_2, \dots, C_7$  are the descendant cycles of  $C$ , cycles  $C_1, C_2, \dots, C_5$  are the child-cycles of  $C$ , and cycles  $C_6$  and  $C_7$  are the child-cycles of  $C_4$ . It is shown in [RNN99] that if  $C$  is a 3-legged cycle in a 3-connected cubic plane graph  $G$ , then the child-cycles of  $C$  are independent of each other. It implies that the containment relation among cycles in  $C_C$  can be represented by a tree as illustrated in Figure 2.12(b); the tree is called the *genealogical tree* of  $C$  and denoted by  $T_C$ .

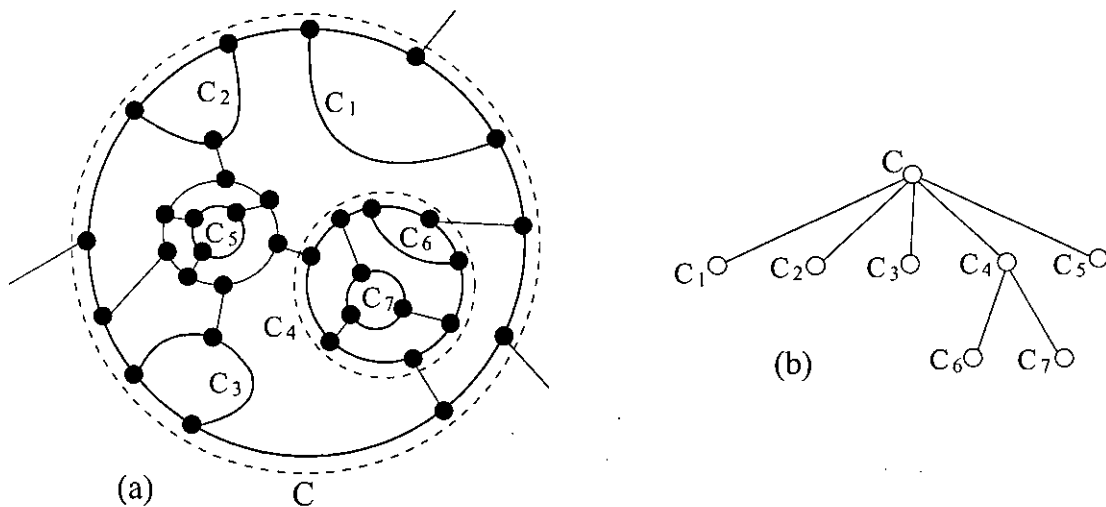


Figure 2.12: (a) Cycles in  $C_C$  and (b) Genealogical tree  $T_C$ .

In the next chapter we use these definitions and results for describing our proposed algorithm.

# Chapter 3

## 1-Bend Orthogonal Drawing

In this chapter we first give a sufficient condition for triconnected planar 4-graphs to have 1-bend orthogonal drawings. We next give a constructive proof for the sufficient condition. We finally give an algorithm for finding 1-bend orthogonal drawings of those graphs which satisfy the sufficient condition based on the constructive proof.

### 3.1 Definitions

By extensively examining the characteristics of triconnected planar 4-graphs we derive a sufficient condition. Before introducing the condition we need to define some terms.

#### 3.1.1 $\alpha$ -Face

Let  $F$  be a face of a triconnected planar 4-graph  $G$ . We call the face  $F$  in  $G$  an  $\alpha$ -face if the boundary of  $F$  contains at least four vertices of degree three. Figure 3.1 illustrates a graph  $G$  in which the shaded face  $F$  is an  $\alpha$ -face.

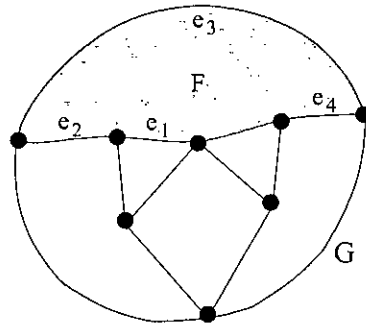


Figure 3.1: An  $\alpha$ -face  $F$ .

### 3.1.2 Critical-Cycle

Let  $C$  be a cycle in a simple 4-graph  $G$  and let  $C$  have exactly three distinct leg vertices, each of which has one or two legs incident to it. We call the cycle  $C$  a *critical-cycle* if two of the three contour paths of  $C$  do not contain any edge  $e$  such that  $e$  connects two vertices of degree three. Three examples of critical-cycles are illustrated in Figure 3.2, where  $x$ ,  $y$  and  $z$  are the leg vertices.

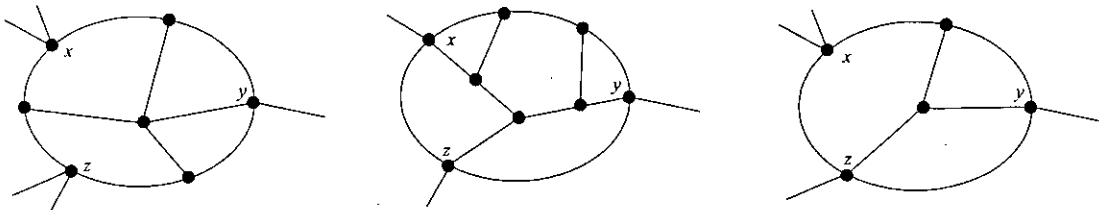


Figure 3.2: *Examples of critical-cycle.*

### 3.1.3 Good String and Bad String

Let  $G$  be a triconnected plane 4-graph in which the outer face is an  $\alpha$ -face. It implies that the outer cycle  $C_o(G)$ , which is also the boundary of an  $\alpha$ -face, contains at least four vertices of degree three. These four vertices divide the outer cycle  $C_o(G)$  into four paths. If we arbitrarily choose one edge from each path, then we find four edges on  $C_o(G)$  and we call each of these four edges a *safe-edge* of  $G$ .

The outer cycle  $C_o(G)$  may have some vertices of degree four. We call a maximal path  $P$  on  $C_o(G)$  a *string* of  $G$  if  $P$  contains only vertices of degree four and does not contain any safe-edge of  $G$ .

Figure 3.3 illustrates an outer cycle  $C_o(G)$  of a graph  $G$  in which the outer face is an  $\alpha$ -face. In Figure 3.3  $e_1, e_2, e_3$  and  $e_4$  are the four safe-edges of  $G$  and the strings ( $S_1, S_2, S_3, S_4$  and  $S_5$ ) of  $G$  are marked with the dotted lines.

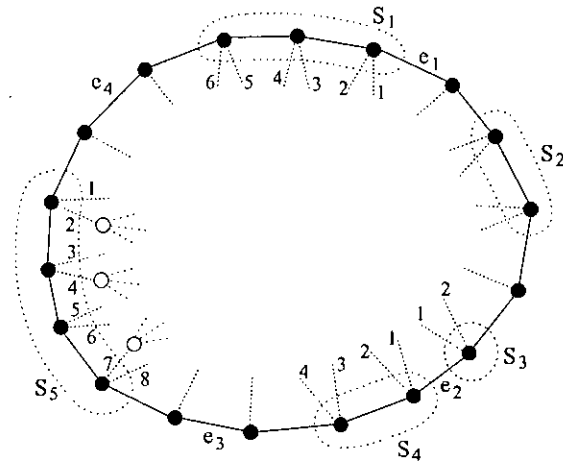


Figure 3.3: *Good strings and bad strings.*

We call a string  $S$  of  $G$  a *good string* of  $G$  if no safe-edge of  $G$  is incident to any vertex of  $S$ . On the contrary we call a string  $S$  of  $G$  a *bad string* of  $G$  if exactly one safe-edge of  $G$  is incident to a vertex of  $S$ . Note that by the definition of safe-edge a string  $S$  can have at most one safe-edge incident to a vertex of  $S$ . Among the strings in Figure 3.3  $S_2$  and  $S_5$  are the good strings of  $G$ , whereas  $S_1, S_3$  and  $S_4$  are the bad strings of  $G$ .

### 3.1.4 Good Edge and Bad Edge

Let  $S$  be a string of a triconnected plane 4-graph  $G$  and the outer face of  $G$  is an  $\alpha$ -face. Since every vertex of a string is of degree four, every vertex of  $S$  has four edges incident to it. Among these four edges two edges are outer edges and the other two edges are inner edges.

Suppose  $S$  is a bad string of  $G$  and a safe-edge  $e$  of  $G$  is incident to an end vertex  $v$  of  $S$ . Let  $e'$  be the inner edge which is incident to  $v$  such that  $e$  and  $e'$  are contained in the same inner face. We assign number 1 to  $e'$  and assign numbers to the rest of the inner edges incident to the vertices of  $S$  sequentially. In this way we assign numbers to the inner edges of  $G$  which are incident to the vertices of all bad strings of  $G$ . We now call the odd numbered edges as *good edges* of  $G$  and the even numbered edges as *bad edges* of  $G$ . In Figure 3.3 the odd numbered and the even numbered inner edges which are incident to the bad strings  $S_1$ ,  $S_3$  and  $S_4$  of  $G$  are the good edges and bad edges, respectively.

If  $S$  is a good string of  $G$ , then no safe-edge is incident to any vertex of  $S$ . Let  $e''$  be an outer edge of  $G$ , other than the edges of  $S$ , such that  $e''$  is incident to an end vertex  $v$  of  $S$ . Let  $e'$  be the inner edge which is incident to  $v$  such that  $e'$  and  $e''$  are contained in the same inner face. We assign number 1 to  $e'$  and assign numbers to the rest of the inner edges incident to the vertices of  $S$  sequentially. After numbering the inner edges of  $G$  if we find that the odd numbered edges are mostly connected with inner vertices of degree four, then we call the odd numbered edges as good edges of  $G$  and the even numbered edges as bad edges of  $G$ . Otherwise we call the even numbered edges as good edges of  $G$  and the odd numbered edges as bad edges of  $G$ . The good string  $S_5$  in Figure 3.3 has eight inner edges which are numbered sequentially and the 2<sup>nd</sup>, 4<sup>th</sup> and 7<sup>th</sup> inner edges are incident to inner vertices of degree four. Therefore, for this case the even numbered edges are good edges and the odd numbered edges are bad edges.

### 3.1.5 Valid 4-Component

Let  $G$  be a triconnected plane 4-graph in which the outer face  $F$  is an  $\alpha$ -face. If we delete from  $G$  the outer cycle  $C_o(G)$  and all the inner vertices of degree three of  $G$ , then we may get some connected subgraphs of  $G$  in which all the vertices are of degree four. We call each of these connected subgraphs of  $G$  a *4-component* of  $G$ . Figure 3.4(a) illustrates a triconnected plane 4-graph  $G$  in which the outer face is an  $\alpha$ -face. After removing

$C_o(G)$  and all the inner vertices of degree three from  $G$  we get the 4-components of  $G$  as illustrated in Figures 3.4(b)-(d). The 4-components of  $G$  can be classified into two types: *tree components* and *non-tree components*. We call a 4-component  $H$  of  $G$  a tree component of  $G$  if  $H$  does not contain a cycle, otherwise we call  $H$  a non-tree component of  $G$ . Figures 3.4(b) and 3.4(d) illustrate the tree components and Figure 3.4(c) illustrates a non-tree component of the graph  $G$  in Figure 3.4(a).

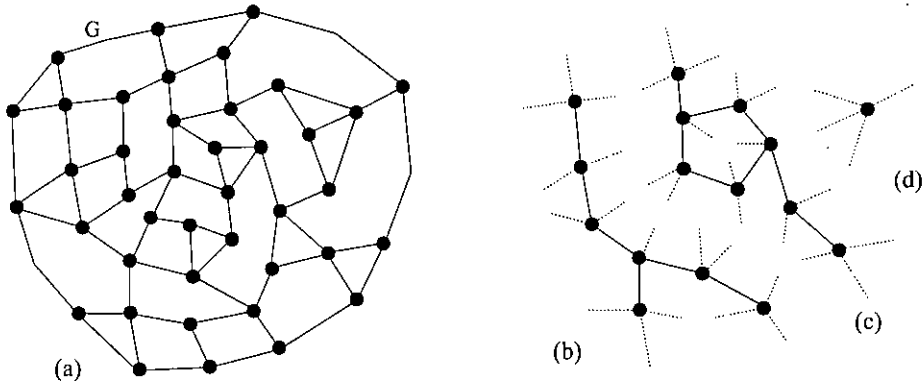


Figure 3.4: (a) A triconnected plane 4-graph  $G$ , (b-d) 4-Components of  $G$ .

We call a tree component  $H$  of  $G$  *valid* if at most one bad edge of  $G$  is incident to a vertex of  $H$ . Besides we call a non-tree component  $H$  of  $G$  *valid* if  $H$  has at most one cycle and if no bad edge of  $G$  is incident to any vertex of  $H$ .

## 3.2 Sufficient Condition

We are now ready to introduce the sufficient condition for a triconnected planar 4-graph to have a 1-bend orthogonal drawing. The sufficient condition is as follows.

**Theorem 3.2.1** *A triconnected planar 4-graph  $G$  has a 1-bend orthogonal drawing if  $G$  has at least one  $\alpha$ -face  $F$  such that the planar embedding of  $G$  having  $F$  as the outer face contains only valid 4-components and contains no critical-cycle.*

We give a constructive proof for the sufficiency of Theorem 3.2.1. Let us assume that  $G$  is the given simple triconnected planar 4-graph and  $G$  satisfies the condition of Theorem 3.2.1. It means  $G$  has at least one  $\alpha$ -face  $F$  and the planar embedding of  $G$  in which  $F$  is the outer face contains only valid 4-components and contains no critical-cycle. More than one  $\alpha$ -faces like  $F$  may exist in  $G$ . We therefore arbitrarily choose one  $\alpha$ -face like  $F$  of  $G$  and call it a *desired  $\alpha$ -face* of  $G$ . In a triconnected graph minimum degree of the vertices is three and in a 4-graph maximum degree of the vertices is four. Therefore, the minimum degree of the vertices of  $G$  is three and maximum degree of the vertices of  $G$  is four. We next give an outline of the constructive proof.

### 3.2.1 An Outline of the Constructive Proof

An outline of the constructive proof of Theorem 3.2.1 is illustrated in Figure 3.5. The given graph  $G$  is illustrated in Figure 3.5(a). At first we find a planar embedding  $G_1$  of  $G$  in which the outer face is a desired  $\alpha$ -face. The shaded face, illustrated in Figure 3.5(a), is a desired  $\alpha$ -face of  $G$ . This  $\alpha$ -face is transformed to the outer face in the planar embedding  $G_1$  of  $G$  as shown in Figure 3.5(b). For this particular embedding of  $G$  we find a 1-bend orthogonal drawing. After this step we can consider that  $G_1$  is a plane graph. Since the outer face of  $G_1$  is an  $\alpha$ -face, we can find four safe-edges of  $G_1$  on the outer cycle of  $G_1$ . We next put four dummy vertices of degree two on four safe-edges of  $G_1$  and obtain a graph  $G_2$  as shown in Figure 3.5(c), where the gray colored vertices represent the dummy vertices. We then replace each vertex of degree four of  $G_2$  by a dummy edge end points of which are incident to two vertices of degree three. In Figure 3.5(c) the vertex  $v$  of  $G_2$  is of degree four. We replace  $v$  by a dummy edge  $e$  and obtain the graph  $G_3$  as shown in Figure 3.5(d). This step converts the given triconnected 4-graph to a triconnected cubic graph, except the four dummy vertices.

The obtained graph  $G_3$  contains several cycles and from those cycles the maximal bad cycles are contracted along with their inner subgraphs. Figure 3.5(e) illustrates how the

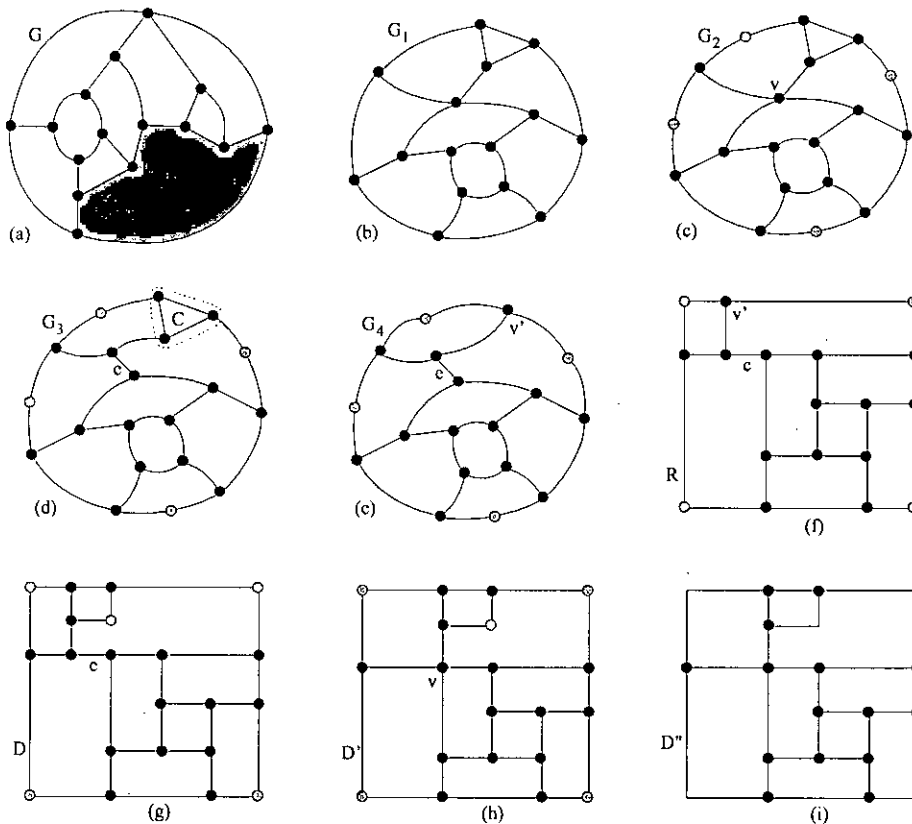


Figure 3.5: An overview of the constructive proof of Theorem 3.2.1.

maximal bad cycle  $C$  (marked with a dotted line) of  $G_3$  is contracted to a vertex  $v'$  of degree three. Let  $G_4$  be the resulting graph. The graph  $G_4$  satisfies the necessary and sufficient condition of Lemma 2.2.1. Therefore,  $G_4$  has a rectangular drawing. We next apply an efficient linear-time algorithm of [RNN98] on  $G_4$  in order to get a rectangular drawing  $R$  as shown in Figure 3.5(f). The rectangular drawing  $R$  contains some vertices of degree three like  $v'$  which represent the location of previously contracted maximal bad cycles. We now find feasible drawing for each of the contracted maximal bad cycles and patch those feasible drawings to their corresponding locations. After patching the feasible drawing of  $C$  at the location of  $v'$  in  $R$  we get the drawing  $D$  as shown in Figure 3.5(g). Next the dummy edges like  $e$  of the resulting drawing are contracted and thus the original vertices of degree four are retrieved. After contracting the dummy edge  $e$  from  $D$  we find



the drawing  $D'$  as shown in Figure 3.5(h). Finally we replace each of all the dummy vertices of degree two with a bend and thus get the final drawing. By replacing all the dummy vertices of  $D'$  with bends we get the final 1-bend orthogonal drawing  $D''$  of the planar embedding  $G_1$  of the given graph  $G$  as shown in Figure 3.5(i).

In this section we give the outline of the constructive proof. In the following sections we give the detail proof. In section 3.2.2 and section 3.2.3 we present the technique for transforming the given graph so that it satisfies the necessary and sufficient condition of Lemma 2.2.1. How to find rectangular drawing is discussed in section 3.2.4. In section 3.2.5 we explain the patching operation. How the dummy edges are contracted is discussed in section 3.2.6. In section 3.2.7 we show how we get the resulting 1-bend orthogonal drawing by replacing the dummy vertices with bends and we give a proof for sufficiency of Theorem 3.2.1.

### 3.2.2 Inserting Dummy Vertices

Every planar graph has a planar embedding [NR04]. Therefore,  $G$  must have a planar embedding. Many algorithms have been developed for finding planar embeddings of planar graphs. 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. Using any one of these algorithms we can find a planar embedding  $G'$  of  $G$ . Since  $G$  contains a desired  $\alpha$ -face, we can find that desired  $\alpha$ -face in  $G'$  by checking the faces of  $G'$ . After finding the desired  $\alpha$ -face of  $G$  in  $G'$  we modify the embedding  $G'$  such that in the modified planar embedding  $G_1$  the desired  $\alpha$ -face of  $G$  becomes the outer face.  $G_1$  is the planar embedding of  $G$  for which we find the final 1-bend orthogonal drawing. Therefore, from now on we thus assume that  $G_1$  is a plane graph such that  $C_o(G_1)$  is an  $\alpha$  face, all 4-components in  $G_1$  are valid 4-components and  $G_1$  contains no critical-cycle.

Since  $G_1$  is a triconnected plane 4-graph, the degree of a vertex of  $G_1$  is either three or four. But according to Lemma 2.2.1, a graph must have at least four vertices of degree two on its outer cycle to have a rectangular drawing. It implies that for getting a rectangular drawing of  $G_1$  the outer cycle  $C_o(G_1)$  of  $G_1$  must have at least four vertices of degree two. We therefore put four dummy vertices of degree two on  $C_o(G_1)$ . We now show that  $C_o(G_1)$  has enough edges to accommodate four dummy vertices of degree two.

Each of the four dummy vertices is finally replaced by a convex bend, (i.e., bend of inner angle  $90^\circ$ ). If we place more than one dummy vertex on a single edge, then after replacing those dummy vertices by bends more than one bends appear on a single edge, which violates our claim that our algorithm gives 1-bend orthogonal drawings. Therefore, at most one dummy vertex of degree two can be placed on a single edge on  $C_o(G_1)$ .

Four dummy vertices of degree two should be placed on  $C_o(G_1)$  for having a rectangular drawing and at most one dummy vertex can be placed on a single edge on  $C_o(G_1)$ . Therefore,  $C_o(G_1)$  must contain at least four edges. Since the outer face of  $G_1$  is an  $\alpha$ -face,  $C_o(G_1)$  contains at least four safe-edges on it. We now put four dummy vertices of degree two on four safe-edges of  $C_o(G_1)$ . Let  $G_2$  be the resulting graph as shown in Figure 3.5(c) where dummy vertices are colored with gray color. In this thesis we often call this operation of inserting four dummy vertices of degree two on four safe-edges of  $C_o(G_1)$  as *vertex-insertion* operation.

We next find the strings and 4-components of  $G_2$  and label their vertices and edges. We use this labeling later for the ease of explanation. Suppose  $p, q, r,$  and  $s$  are the four dummy vertices on  $C_o(G_2)$  and these vertices appear on  $C_o(G_2)$  in clockwise order. The dummy vertices  $p, q, r,$  and  $s$  divide  $C_o(G_2)$  into four contour paths. Since  $C_o(G_2)$  is also the boundary of an  $\alpha$ -face and  $p, q, r,$  and  $s$  are placed on four different safe-edges on  $C_o(G_2)$ , each contour path contains at least one vertex of degree three.

Let us first investigate the method of labeling the edges and vertices of the strings in between  $p$  and  $q$ . Same method can be followed for the rest of the contour paths. If the

contour path  $pq$  contains no string, then we label none of the edges and vertices of  $pq$ . If the path  $pq$  contains any string (either good or bad), then we label all the vertices and edges of that string. If a string  $S$  on  $pq$  contains  $n'$  vertices, then in  $G_2$  there are  $n' + 1$  outer edges and  $2n'$  inner edges incident to the  $n'$  vertices of  $S$ .

If  $S$  is a bad string, then one end vertex of  $S$  has a safe-edge incident to it. We label the safe-edge of  $S$  with  $e_{b1}$ . Other outer edges of  $G_2$  which are incident to the vertices of  $S$  are labeled sequentially with  $e_{b2}, e_{b3}, \dots$ , and  $e_{b(n'+1)}$ , respectively. We label the vertices of  $S$  sequentially starting from the vertex to which  $e_{b1}$  and  $e_{b2}$  are incident. We label these vertices with  $b_1, b_2, \dots$ , and  $b_{n'}$ , respectively. Each vertex of  $S$  has two inner edges incident to it. The inner edge which is incident to  $b_1$  and which share a common face with  $e_{b1}$  is labeled with  $e_{b1o}$ . The other inner edge which is also incident to  $b1$  is labeled with  $e_{b1e}$ . Similarly the inner edge, which is incident to  $b_2$  and share a common face with  $e_{b2}$ , is labeled with  $e_{b2o}$ . The other inner edge which is incident to  $b2$  is labeled with  $e_{b2e}$ . The other inner edges which are incident to the vertices of  $S$  are labeled similarly. We follow this technique of labeling for all the bad strings on  $pq$  as well as on  $C_o(G_2)$ .

If  $S$  is a good string, then we label the vertices and edges of  $S$  starting from the end which is found first, while traversing  $C_o(G_2)$  to the clockwise direction. We label the vertices of  $S$  with  $g_1, g_2, \dots, g_{n'}$ , respectively. The edges of  $S$  including the outer edges which are incident to  $g_1$  and  $g_{n'}$  are sequentially labeled with  $e_{g1}, e_{g2}, \dots, e_{g(n'+1)}$  and  $g_1, g_2, \dots, g_{n'}$ , respectively. The inner edges are labeled following the similar technique as discussed for bad strings with the exception that the alphabet  $b$  is replaced with  $g$ . After labeling the inner edges if we find that the inner edges, having the suffix  $e$ , are mostly connected with the inner vertices of degree four in comparison with the inner edges having the suffix  $o$ , then we replace all  $o$  with  $e$  and all  $e$  with  $o$  of that string. We follow this technique of labeling for all the good strings on  $pq$  as well as on  $C_o(G_2)$ . Figure 3.6(a) illustrates an outer cycle of a graph like  $G_2$ , where the vertices and edges are labeled.

We now explain the technique of labeling for a 4-component  $H$  of  $G_2$ . A tree compo-

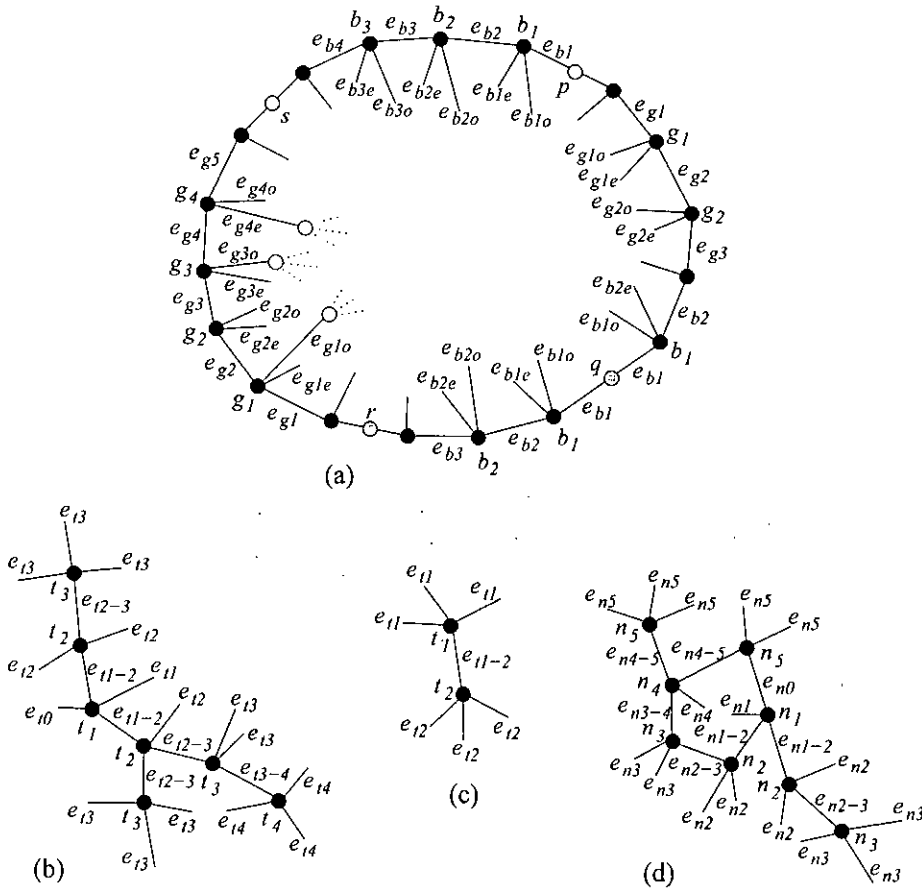


Figure 3.6: Labeling the vertices and edges of the strings and 4-components.

ment can have at most one bad edge incident to a vertex of it. If  $H$  is a tree component and  $H$  has a bad edge incident to a vertex of  $H$ , then we start labeling from the bad edge and we label it with  $e_{i0}$ . We next label with  $t_1$  the vertex to which the bad edge is incident. If the vertex  $t_1$  has one or more neighbor vertices in  $H$ , then we label all of those neighbor vertices by  $t_2$ . If the vertices of  $H$  which are labeled by  $t_2$  have any number of unlabeled neighbor vertices in  $H$ , then we label all of those neighbor vertices by  $t_3$ . Similarly we label all the vertices of  $H$ . If  $H$  has no bad edge, then we start labeling from any vertex of  $H$  by labeling it with  $t_1$ . Rest of the vertices of  $H$  are labeled in the same way as explained earlier. After labeling the vertices of  $H$  we label the edges in between  $t_1$  and  $t_2$  by  $e_{t_1-2}$ . Similarly the edges in between  $t_2$  and  $t_3$  are labeled with  $e_{t_2-3}$ . In this

way we label all the edges in between any two vertices of  $H$ . We next label the unlabeled edges of  $H$ . If an edge is incident to  $t_1$ , then we label it with  $e_{t_1}$ . If this type of edges are incident to  $t_2$  we denote them by  $e_{t_2}$  and so on. In this way we label all the vertices and edges of all the tree components of  $G_2$ . Figures 3.6(b) and 3.6(c) illustrate two tree components in which the vertices and edges are labeled.

A non-tree component has exactly one cycle. Let  $H$  be a non-tree component. We first label any edge on the cycle in  $H$  with  $e_{n_0}$ . The edge  $e_{n_0}$  is incident to two vertices on the cycle of  $H$  and we denote any one of those two vertices by  $n_1$ . We next cut the edge  $e_{n_0}$  and thus the non-tree component becomes a tree component. We now label all the vertices and edges of this 4-component following the technique explained for tree components. The only difference is that we consider  $n$  in the place of  $t$ . In this way we can label all the vertices and edges of all the non tree components of  $G_2$ . Figure 3.6(d) illustrates a non tree component in which the vertices and edges are labeled.

### 3.2.3 Inserting Dummy Edges

In this section we show that the graph  $G_2$  can be converted to a triconnected cubic plane graph, except four dummy vertices of degree two on the outer cycle.

The graph  $G_2$  is a plane graph in which all the vertices are of degree three or of degree four, except the four vertices of degree two on the outer cycle  $C_o(G_2)$ . We next replace each of all the vertices of degree four of  $G_2$  with a dummy edge end points of which are incident to two vertices of degree three. This step converts the graph  $G_2$  to a triconnected cubic plane graph in which all the vertices are of degree three, except four vertices of degree two on the outer boundary. Let  $G_3$  be the newly obtained graph. Figures 3.5(c) and 3.5(d) illustrate how a vertex of degree four is replaced by a dummy edge. In this thesis we often call this operation of replacing all the vertices of degree four of  $G_2$  with dummy edges as *edge-insertion* operation.

$G_2$  has two types of vertices of degree four: some are inner vertices and some are outer

vertices. Each of these vertices has four edges incident to it. In case of an outer vertex of degree four two incident edges are outer edges and the other two incident edges are the inner edges. Figure 3.7(a) partially illustrates the outer cycle of  $G_2$  which contains an outer vertex  $v_4$  of degree four. Such an outer vertex of degree four can be replaced by a dummy edge in two ways. We can either connect the two outer edges to the alternate end points of the same dummy edge or we can connect them to the same end point of a dummy edge. For the first type of replacement the dummy edge becomes an outer edge and for the second type the dummy edge becomes an inner edge. The vertex  $v_4$  in Figure 3.7(a) is replaced in two different ways by the dummy edges  $e$  and  $e'$  as shown in Figures 3.7(b) and 3.7(c), respectively. Among these two ways of replacements we prefer the first one (see section 3.2.6 for the reason). It means the outer vertices of degree four of  $G_2$  are replaced by dummy edges such that the dummy edges become the outer edges. In case of an inner vertex of degree four all of the four incident edges are inner edges. Therefore, during the replacement of the inner vertices of degree four with the dummy edges there is no such restriction.

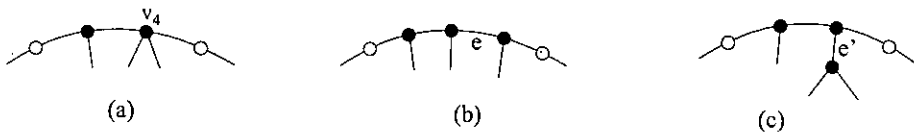


Figure 3.7: (a) A vertex  $v_4$  of degree four on the outer cycle, (b) dummy edge as an outer edge, (c) dummy edge as an inner edge.

### 3.2.4 Finding a Rectangular Drawing

In this section we show that the graph  $G_3$  can be modified such that the modified graph has a rectangular drawing.

In the graph  $G_3$  all the vertices are of degree three, except four vertices of degree two on  $C_o(G_3)$ . According to Lemma 2.2.1  $G_3$  has a rectangular drawing if and only if  $G_3$

contains no bad cycle. The construction of  $G_3$  implies that  $G_3$  may have some bad cycles. Since  $G_3$  is a triconnected cubic plane graph (except four dummy vertices of degree two on the outer cycle),  $G_3$  has no 1- or 2-legged cycle. It implies that any bad cycle in  $G_3$  is a 3-legged cycle. Moreover, all maximal bad cycles in  $G_3$  are independent of each other [RNN99]. We now contract all the maximal bad cycles  $C_i$  of  $G_3$  along with their inner subgraphs  $G(C_i)$  in  $G_3$  into their corresponding vertices  $v_i$  of degree three, where  $1 \leq i \leq l$  and  $l$  is the total number of maximal bad cycles in  $G_3$ . Let  $G_4$  be the graph obtained from  $G_3$  by contracting all the maximal bad cycles in  $G_3$ . Figures 3.5(d) and 3.5(e) illustrate how a maximal bad cycle is contracted to a vertex of degree three. It implies that the graph  $G_4$  is a triconnected cubic plane graph and it contains no bad cycle i.e.  $G_4$  satisfies the necessary and sufficient condition of Lemma 2.2.1. Therefore, the graph  $G_4$  has a rectangular drawing. If we apply the rectangular drawing algorithm **Rectangular-Draw** of [RNN98] on the obtained graph  $G_4$ , then we get a rectangular drawing  $R$  of  $G_4$  as shown in Figure 3.5(f).

### 3.2.5 Patching the Feasible Orthogonal Drawings

In this section we show that all the contracted maximal bad cycles of  $G_3$  has a feasible orthogonal drawing which can be patched to their corresponding locations in the rectangular drawing  $R$ .

$G_3$  is a triconnected cubic plane graph (except four dummy vertices on  $C_o(G_3)$ ) and contracted maximal bad cycles of  $G_3$  are the 3-legged cycles of  $G_3$ . Therefore, according to Lemma 2.2.2 every contracted maximal bad cycles of  $G_3$  has a feasible orthogonal drawing. If we use the algorithm **Feasible-Draw** of [RNN99] we can find the feasible orthogonal drawings of all the contracted maximal bad cycles of  $G_3$ . But unfortunately we cannot use those drawings because the maximal bad cycles of  $G_3$  contain dummy edges and those dummy edges should be contracted later. Therefore, we should modify the algorithm **Feasible-Draw** according to our need.

A 3-legged cycle  $C$  of  $G_3$  may have more than one green paths  $P_i$  where  $1 \leq i \leq 3$  and the algorithm **Feasible-Draw** can find orthogonal drawing of  $C$  feasible for any green path  $P_i$  of  $C$ . Since a 3-legged cycle  $C$  of  $G_3$  may contain dummy edges and those edges should be contracted later, we cannot choose any green path  $P_i$  for finding orthogonal drawing feasible for  $P_i$ . We need to define which path of  $C$  should be selected. Before defining it we need to define another term called *desired-edge* for the simplicity of explanation. We call any edge  $e$  on a three legged cycle  $C$  of  $G_3$  a desired-edge if  $e$  is not a dummy edge and if the end vertices of  $e$  are of degree three having no dummy edge incident to them. We have the following lemma.

**Lemma 3.2.2** *Let  $C$  be a three legged cycle of a triconnected cubic plane graph  $G$ .  $G$  is obtained from a triconnected plane 4-graph  $G'$  by the edge-insertion operation. If  $G'$  contains no critical-cycle, then at least two contour paths of every three legged cycle  $C$  of  $G$  contain desired-edges.*

**Proof.** Every three legged cycle  $C'$  of  $G'$  is also a three legged cycle of  $G$  but may have some dummy edges. Moreover,  $G$  has some extra three legged cycles which are created due to the edge-insertion operation. Each extra cycle of  $G$  is generated from a cycle  $C''$  of  $G'$  such that  $C''$  has three leg vertices but more than three legs in  $G'$ . We have the following two cases.

**Case 1:**  $C$  of  $G$  is obtained from a three legged cycle  $C'$  of  $G'$ .

$C'$  is a three legged cycle of  $G'$ . Therefore,  $C'$  has three leg vertices in  $G'$ . Since  $G'$  has no critical-cycle, at least two contour paths of  $C'$  of  $G'$  contain desired-edges.  $C$  of  $G$  is obtained from  $C'$  of  $G'$  by the edge-insertion operation. Moreover, edge-insertion operation creates no new face and doesn't change the edges and the vertices of degree three of  $C'$ . Therefore, the desired-edges of  $C'$  of  $G'$  remain unchanged on the contour paths of  $C$  of  $G$ .

**Case 2:**  $C$  of  $G$  is obtained from a cycle  $C''$  having three leg vertices and more than three legs in  $G'$ .



$C''$  of  $G'$  has three leg vertices and more than three legs. Therefore, at least one leg vertex of  $C''$  of  $G'$  is of degree four and has two legs incident to it. Since  $C$  of  $G$  is a three legged cycle, during the edge-insertion operation each leg vertex of degree four of  $C''$  of  $G'$  is replaced with a dummy edge such that the dummy edge becomes a leg of  $C$  of  $G$ . Figure 3.8 illustrates an example of  $C$  and  $C''$  where the dummy edges are drawn with the dotted lines. The other vertices of degree four of  $C''$  are replaced by normal edge-insertion operation. Similar to case 1 for this case the edge-insertion operation does not create any new face and does not change any existing edges and vertices of degree three of  $C''$ . Therefore, we can say that the desired-edges exist on any two contour paths of  $C$  of  $G$ .

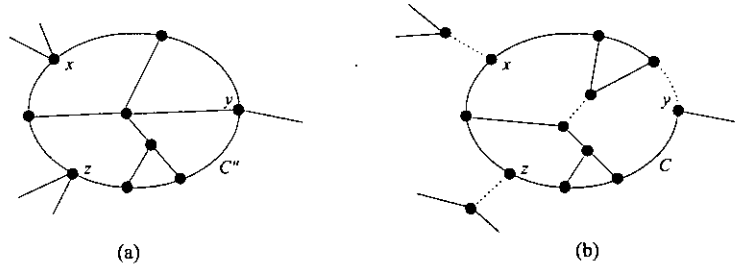


Figure 3.8: (a) A cycle  $C''$  having three leg vertices but more than three legs, (b) a three legged cycle  $C$  obtained from  $C''$ .

□

Lemma 3.2.2 implies that every three legged cycle  $C$  of  $G_3$  has three contour paths and at least two of them contain desired-edges. We have the following lemma.

**Lemma 3.2.3** *Let  $C$  be a three legged cycle of a triconnected cubic plane graph  $G$ .  $G$  is obtained from a triconnected plane 4-graph  $G'$  by the edge-insertion operation. If  $G'$  contains no critical-cycle, then at least one contour path of  $C$  of  $G$  contains a desired-edge and doesn't contain any edge common with the outer cycle  $C_o(G)$  of  $G$ .*

**Proof.** Suppose for a contradiction that all of the three contour paths of  $C$  completely lay on  $C_o(G)$ . Therefore, the three leg vertices of  $C$  of  $G$  also lay on  $C_o(G)$ . It implies

that the total cycle  $C$  completely coincide with  $C_o(G)$ , which is contradictory with the definition of three legged cycle of  $G$ . Again for contradiction we assume that three leg vertices and two contour paths of  $C$  completely lay on  $C_o(G)$ . Then we find that one leg vertex of  $C$  on  $C_o(G)$  has no leg incident to it because  $C_o(G)$  is the outer cycle of  $G$ . This is contradictory with the definition of three legged cycle of  $G$ . Therefore, at least two contour paths of  $C$  of  $G$  have no edge common with  $C_o(G)$ . We call a contour path of  $C$  having no edge common with  $C_o(G)$  an *inner contour paths* of  $C$ . According to Lemma 3.2.2 at least two contour paths of  $C$  contain desired-edges. Therefore, among the two inner contour paths of  $C$  at least one has desired-edge on it.  $\square$

According to Lemma 3.2.3 every three legged cycle  $C$  of  $G_3$  has at least one contour path  $P$  such that  $P$  contains a desired-edge but contains no edge common with the outer cycle  $C_o(G_3)$  of  $G_3$ . For every three legged cycle  $C$  of  $G_3$  we need to define one contour paths  $P$  of  $C$  as *desired-path*. We find the definition of desired-path from the following cases.

**Case 1:**  $C$  has no edge common with  $C_o(G_3)$  or with any other ancestor three legged cycle of  $C$  in  $G_3$  as shown in Figure 3.9(a). In this case any contour path of  $C$  having desired-edge can be termed as desired-path of  $C$ .  $C$  may or may not have any child cycle and  $C$  may or may not be a child cycle of any other three legged cycle of  $G_3$ . In Figure 3.9(a) the desired paths of  $C$  are indicated by the dotted lines.

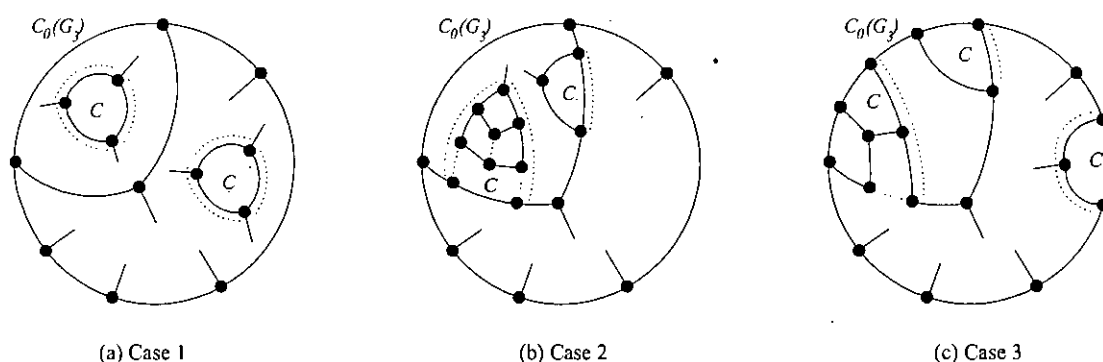


Figure 3.9: *Desired-paths*.

**Case 2:**  $C$  has no edge common with  $C_o(G_3)$  but at least one contour path of  $C$  lays on any ancestor three legged cycle of  $C$  in  $G_3$  as shown in Figure 3.9(b). In this case if the contour path of  $C$ , which lays on an ancestor three legged cycle of  $C$ , contains any desired-edge, then that contour path is termed as desired-path of  $C$ . Otherwise the contour paths which do not lay on any ancestor three legged cycle but contain desired-edges are termed as desired-paths of  $C$ . In Figure 3.9(b) desired paths of  $C$  are indicated by dotted lines beside the paths whereas the edges drawn with dotted lines represent the dummy edges.

**Case 3:** At least one edge of  $C$  is common with  $C_o(G_3)$  as shown in Figure 3.9(c). If  $C$  has one edge common with  $C_o(G_3)$ , then the total contour path of  $C$  having that common edge lays on  $C_o(G_3)$ . According to Lemma 3.2.3 at least one inner contour path of  $C$  contains desired-edge and for this case we call that inner contour path as a desired-path.

We now have the following lemmas.

**Lemma 3.2.4** *At least one of the three contour paths of every 3-legged cycle in  $G$  is a desired-path under the classification above.*

**Proof.** Immediate. □

**Lemma 3.2.5** *Let  $G$  be a triconnected cubic plane graph. For any 3-legged cycle  $C$  of  $G$  and any contour path  $P$  of  $C$ ,  $G(C)$  has an orthogonal drawing feasible for  $P$ .*

**Proof.** This lemma is a slight modification of Lemma 2.2.2. We have the following cases to consider.

**Case 1:**  $C$  has no child cycle as shown in Figure 2.10(a). According to Lemma 2.2.2,  $G(C)$  has an orthogonal drawing feasible for any green path of  $C$ . In this case all of the three contour paths of  $C$  are green paths. Therefore, we can say  $G(C)$  has an orthogonal drawing feasible for any contour path  $P$  of  $C$ .

**Case 2:** None of the child-cycles of  $C$  has a green path on  $C$  as shown in Figure 2.10(b). All of the three contour paths of  $C$  are green paths for this case also. Therefore,

similar to Case 1 we can say that  $G(C)$  has an orthogonal drawing feasible for any contour path  $P$  of  $C$ .

**Case 3:** Otherwise, (see Figure 2.10(c)).

In this case  $C$  has at least one and at-most two red paths. Other path(s) of  $C$  is(are) green path(s). According to Lemma 2.2.2  $G(C)$  has an orthogonal drawing feasible for any green path of  $C$ . But this lemma claims that  $G(C)$  has an orthogonal drawing feasible for red path also.

Let  $C_1, C_2, \dots, C_l$  be the the child cycles of  $C$ , where  $l \geq 1$ . At first for each  $i$ ,  $1 \leq i \leq l$ , we choose an arbitrary green path of  $C_i$  and find an orthogonal drawing  $D(G(C_i))$  of  $G(C_i)$  feasible for the green path in a recursive manner.

We next construct a plane graph  $F$  from  $G(C)$  by contracting each  $G(C_i)$ ,  $1 \leq i \leq l$ , to a single vertex  $v_i$ . Figure 3.10(a) illustrates  $F$  for  $G(C)$  in Figure 2.10(c) where the red path  $P$  is assumed to be  $P_3$ . We add a dummy vertex  $t$  on any of the edges of  $P$  as shown in Figure 3.10(b) and let  $H$  be the resulting plane graph. All vertices of  $H$  have degree three except the four vertices  $x, y, z$ , and  $t$  of degree two on  $C_o(H)$ . Moreover,  $H$  has no bad cycle. Therefore, by **Rectangular-Draw** we can find a rectangular drawing  $D(H)$  of  $H$  with four corners on  $x, y, z$ , and  $t$ . Figure 3.10c illustrates a rectangular drawing of  $H$  for  $C$  and  $P = P_3$  in Figure 2.10(c).

Finally, patching the drawings  $D(G(C_1)), D(G(C_2)), \dots, D(G(C_l))$  into  $D(H)$ , as explained in **Feasible-Draw**, we can construct an orthogonal drawing  $D(G(C))$  of  $G(C)$  as shown in Figure 3.10(d). Clearly  $t$  is a bend on  $P$  and patching operation doesn't produce new bend. Thus the orthogonal drawing of  $G(C)$  is feasible for  $P$ .  $\square$

For any 3-legged cycle  $C$  of  $G_3$  the algorithm **Feasible-Draw** can find orthogonal drawing of  $G(C)$  feasible for any green path of  $C$ . Since  $C$  may contain dummy edges and those edges should be contracted later, we should not use **Feasible-Draw** as it is. We should slightly modify **Feasible-Draw** before use. Let **Modified-Feasible-Draw** be the algorithm which we are going to use for finding feasible orthogonal drawing of

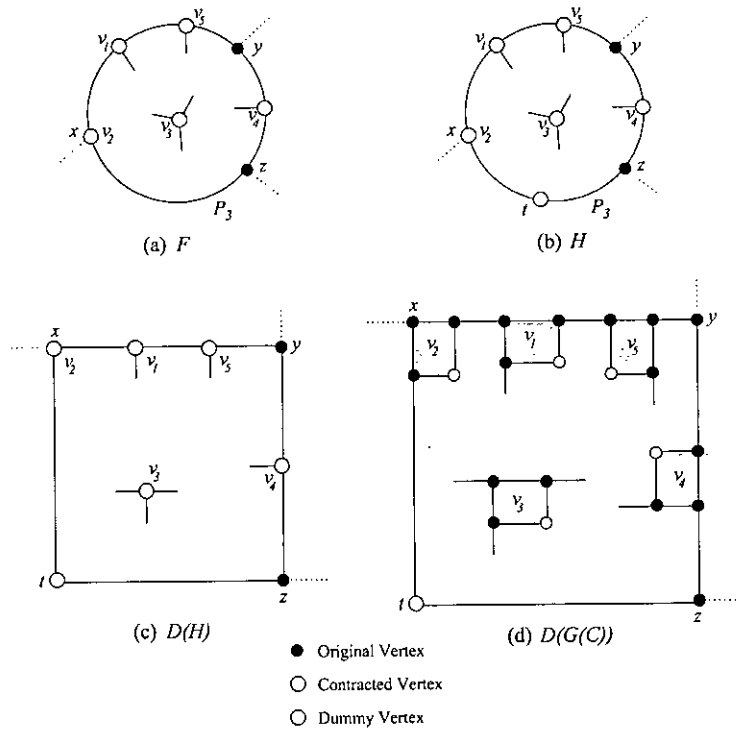


Figure 3.10:  $F$ ,  $H$ ,  $D(H)$  and  $D(G(C))$  for case 3.

$G(C)$ . Before introducing the algorithm **Modified-Feasible-Draw** we have the following lemma.

**Lemma 3.2.6** *Let  $G$  be a triconnected cubic plane graph and  $G$  is obtained from a triconnected plane 4-graph  $G'$  by edge-insertion operation. If  $G'$  has no critical-cycle, then for any 3-legged cycle  $C$  of  $G$  and any desired-path  $P$  of  $C$ ,  $G(C)$  has an orthogonal drawing feasible for  $P$ .*

**Proof.** Lemma 2.2.2 states that  $G(C)$  has an orthogonal drawing feasible for any green path of  $C$ , whereas Lemma 3.2.5 states that  $G(C)$  has an orthogonal drawing feasible for red path of  $C$  also. Therefore,  $G(C)$  has an orthogonal drawing feasible for any contour path of  $C$ . Since  $G'$  has no critical-cycle,  $G$  also has no critical-cycle. Moreover, according to Lemma 3.2.4, every three legged cycle  $C$  of  $G$  has at least one desired-path. Therefore, we can say that  $G(C)$  has an orthogonal drawing feasible for any desired-path  $P$  of  $C$ .  $\square$

Let  $C$  be a three legged cycle of a triconnected cubic plane graph and  $G$ . The algorithm **Feasible-Draw** can find orthogonal drawing of  $G(C)$  feasible for any green path of  $C$ . For finding a feasible orthogonal drawing of  $G(C)$  **Feasible-Draw** always inserts a dummy vertex  $t$  of degree two on an arbitrary green path of  $C$ . The algorithm **Modified-Feasible-Draw** inserts  $t$  on a desired-edge of an arbitrarily chosen desired-path  $P$  of  $C$  for finding orthogonal drawing feasible for any desired-path  $P$  of  $C$ . This is the only difference between **Feasible-Draw** and **Modified-Feasible-Draw**.

Lemma 3.2.6 implies that the algorithm **Modified-Feasible-Draw** can find the orthogonal drawings of the previously contracted maximal bad cycles of  $G_3$  feasible for their corresponding desired-paths. After finding those feasible orthogonal drawings we patch those drawings to their corresponding locations in the rectangular drawing  $R$  following the technique explained in **Modified-Feasible-Draw**. We often call the operation of finding and patching the feasible orthogonal drawings of the previously contracted three legged cycles to their corresponding location in  $R$  as *patching operation*. This patching operation converts the rectangular drawing  $R$  to an orthogonal drawing  $D$  as shown in Figure 3.5(g).

The feasible orthogonal drawing of every three legged cycle  $C$  of  $G_3$  contains a bend on a desired-path of  $C$  at the location of the dummy vertex  $t$ . Since none of the three legged cycles of  $G_3$  has a desired-path on the outer cycle  $C_o(G_3)$ , no new bend appears on the drawing of  $C_o(G_3)$  after the patching operation. It means the rectangular outer boundary of  $R$  remains rectangular in  $D$ .

### 3.2.6 Contracting the Dummy Edges

In this section we show that after contracting the dummy edges of the orthogonal drawing  $D$  we get a 1-bend orthogonal drawing.

Since patching operation doesn't introduce any bend [RNN99] on the existing edges of  $R$ , the orthogonal drawing  $D$  has no bend. Therefore, the dummy edges in  $D$  can take

any one of the nine possible shapes, illustrated in Figures 3.11(a)-(i), or their rotated forms. In Figure 3.11 the dummy edges are drawn with the dotted lines and the other edges which are incident to the end vertices of the dummy edges are drawn with the solid lines.

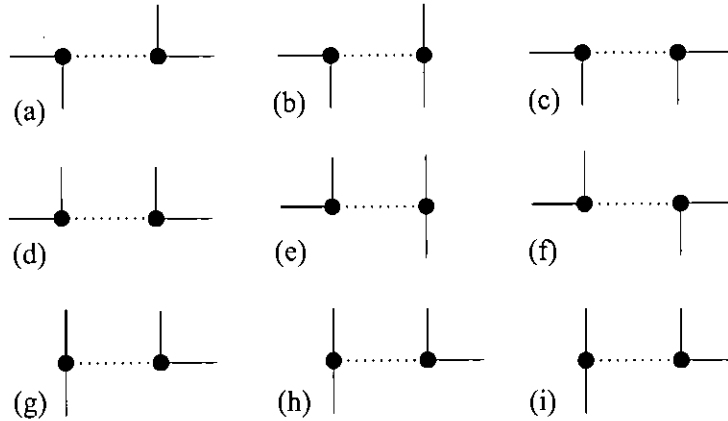


Figure 3.11: *Nine possible shapes of a dummy edge in the orthogonal drawing  $D$ .*

If we contract the dummy edges, then bend appears on the edges which are incident to the end vertices of the dummy edges. Each of the nine shapes in Figure 3.11 can be contracted in two ways. Figure 3.12 illustrates the contracted forms of the shapes in Figure 3.11, where  $a_1$  and  $a_2$  are the two contracted forms of the shape in Figure 3.11(a). Other forms are labeled similarly. For the ease of identification, thick incident edges in Figure 3.11 are also represented by thick edges in Figure 3.12. Other edges can be identified by their corresponding angular position with respect to the thick edge. From the contracted forms in Figure 3.12 we have the following three observations.

Firstly, in the contracted forms some incident edges are safe (i.e. do not hold any bend) and the others are not (i.e. hold a bend). The incident edges which are safe in a contracted form of a shape are not safe in the other contracted form of that shape. Similarly, the incident edges which are not safe in the first contracted form are safe in the second form of contraction.

The second observation is: there are two groups of incident edges. Before character-

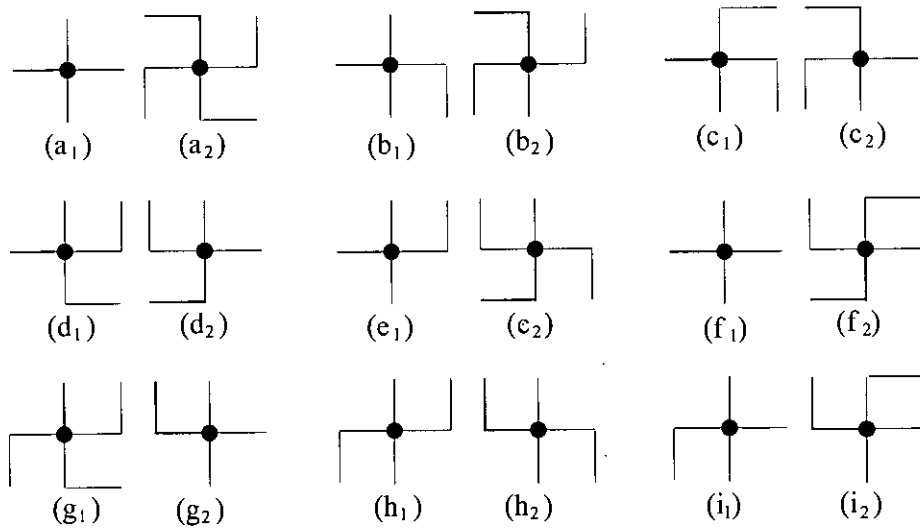


Figure 3.12: Contracted forms of the arrangements in Figure 3.11.

izing these groups let us define two terms. If two edges, which are incident to the end vertices of the same dummy edge, share a common face, then we call these two incident edges as the *adjacent incident edges*; otherwise, we call these edges as the *opposite incident edges*. Two adjacent incident edges are of same group if they are  $90^\circ$  apart from each other. Moreover, two opposite incident edges are of same group if they are  $180^\circ$  apart from each other. After contraction if a member of a group contains bend, then all other members of that group contain bend.

The third observation is: after contraction of a dummy edge any one of its four incident edges can always be kept safe (i.e. we can always contract a dummy edge without putting bend on a particular incident edge). We cannot always keep more than one incident edge safe. Because, two or more incident edges do not always exist in the same group. For example the thick edge and its opposite incident edge do not reside in the same group in all possible shapes in Figure 3.11. In the shapes in Figures 3.11(a), 3.11(f), 3.11(h) and 3.11(i) those edges reside in the same group because they are  $180^\circ$  out of phase with each other. But in the other shapes in Figure 3.11 those edges reside in the alternate groups.

The orthogonal drawing  $D$  contains no bend but contains the dummy vertices only.



Before replacing these dummy vertices with bends  $D$  is a zero bend orthogonal drawing of  $G_3$ . Moreover,  $D$  contains dummy edges and the outer boundary of  $D$  is a rectangle. From the contracted forms in Figure 3.12 it is clear that if we contract the dummy edges of  $D$ , then bend may appear on those edges which are incident to the end vertices of the dummy edges. Therefore, during the contraction of the dummy edges we should follow some rules so that more than one bend cannot appear on a single edge. From the proof of the following lemma we get an algorithm for the contraction of the dummy edges of  $D$ .

**Lemma 3.2.7** *Let  $G$  be a triconnected plane 4-graph such that the outer face of  $G$  is an  $\alpha$ -face, all the 4-components of  $G$  are valid 4-components and  $G$  has no critical-cycle. Let  $G'$  be the graph obtained from  $G$  after vertex-insertion and edge-insertion operation and the contraction of the bad cycles of  $G$  and let  $R$  be a rectangular drawing of  $G'$ . Let  $D$  be a zero bend orthogonal drawing obtained from  $R$  after the patching operation. If we contract all the dummy edges of  $D$ , then the resulting drawing is a one bend orthogonal drawing of  $G$ .*

**Proof.** We give a constructive proof for this lemma which gives an algorithm for contracting the dummy edges of  $D$ . For the ease of explanation we use the labeling convention discussed in section 3.2.2. Since  $D$  is a zero bend orthogonal drawing, to prove the claim it is sufficient to show that the contraction of dummy edges are possible without introducing more than one bend on a single edge in the resulting drawing. We have the following cases to consider.

**Case 1:** Dummy edges of  $D$  are created from the bad strings of  $G$ .

Let  $b_1, b_2, \dots, b_n$  be the vertices of a bad string  $S_b$  of  $G$  as shown in Figure 3.13(a), where  $n$  is the number of vertices of degree four in  $S_b$  and  $1 \leq n$ . The gray colored vertices represent the dummy vertices of degree two. After edge-insertion operation  $S_b$  in Figure 3.13(a) is transformed to a string as in Figure 3.13(b), where the dummy edges are represented by the dotted lines. In the zero bend orthogonal drawing  $D$  the part of

the outer cycle of  $G'$  in between two successive dummy vertices of degree two becomes a straight path as illustrated in Figure 3.13(c). The inner edges which are incident to the end vertices of the dummy edges on  $S_b$  become perpendicular to this straight path.

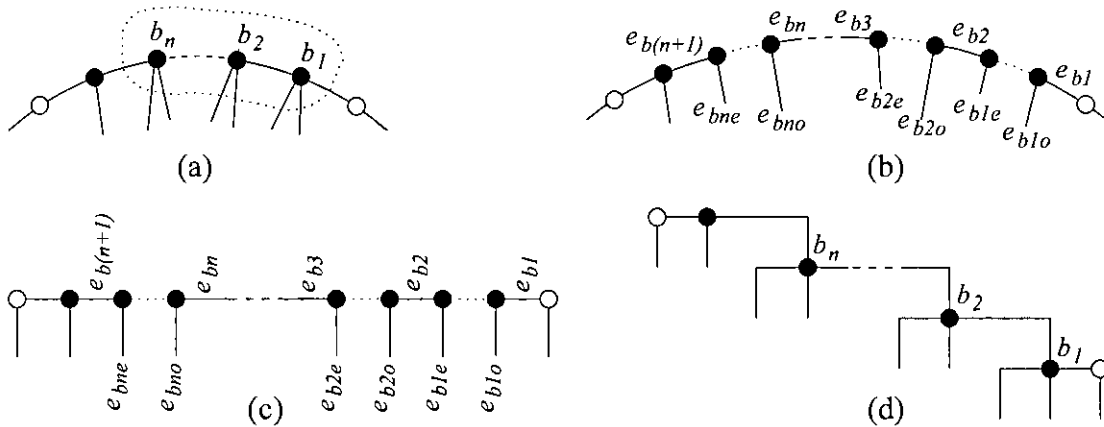


Figure 3.13: *Effect of dummy edge contraction for Case 1.*

The corresponding dummy edge for any vertex  $b_k$  on  $S_b$ ,  $1 \leq k \leq n$ , and the other four edges incident to the end vertices of the dummy edge in  $D$  take the shapes shown in Figure 3.11(c). This dummy edge can be contracted in two ways as shown in Figures 3.12(c<sub>1</sub>) and 3.12(c<sub>2</sub>). If we choose the first form of contraction, then a bend appears on the outer edge  $e_{bk}$ , while the other outer edge  $e_{b(k+1)}$  remains safe (i.e. no bend appears on it). If we choose the second form of contraction, then the alternate situation occurs. But in this algorithm we contract the corresponding dummy edge for  $b_k$  such that the outer edge  $e_{bk}$  remains safe and bend appears on  $e_{b(k+1)}$ . At the same time the inner edge  $e_{bko}$  remains safe and bend appears on  $e_{bke}$ . It means bend appears on every bad edge whereas the good edges remain safe. Figure 3.13(d) illustrates the drawing obtained after contracting the dummy edges in Figure 3.13(c).

In this case the contraction of the dummy edges cannot produce more than one bend on a single inner edge, because each inner edge is connected with at-most one end vertex of a dummy edge. We are now going to show that for this case the contraction operation

cannot produce more than one bend on the outer edges also.

Suppose for contradiction let us assume that contraction operation produced two bends on the outer edge  $e_{bk}$ . This outer edge  $e_{bk}$  interconnects two outer vertices of degree four and those vertices are  $b_{(k-1)}$  and  $b_k$ . Since the contraction of a dummy edge can produce at-most one bend on an incident edge, two bends of  $e_{bk}$  are produced by the contraction of the corresponding dummy edges of  $b_{(k-1)}$  and  $b_k$ . It means during the contraction of dummy edge for  $b_k$ ,  $e_{b(k+1)}$  is kept safe and a bend appears on  $e_{bk}$ . This contradicts the technique we follow for contraction. Moreover, the outer edge  $e_{b1}$  is kept safe after contraction. Therefore, if we replace the dummy vertex of degree two on  $e_{b1}$  with bend, then the drawing will remain one bend drawing.

**Case 2:** Dummy edges of  $D$  are created from the good strings of  $G$ .

Let  $g_1, g_2, \dots, g_n$  be the vertices of a good string  $S_g$  in  $G$  as shown in Figure 3.14(a), where  $n$  be the number of vertices of degree four in  $S_g$  and  $1 \leq n$ . In the graph  $G'$  the string  $S_g$  is transformed as shown in Figure 3.14(b), where the dummy edges are represented by the dotted lines. In  $D$  the part of the outer cycle of  $G'$  in between two successive dummy vertices of degree two is a straight path as illustrated in Figure 3.14(c). Moreover, the dummy edges become the outer edges. The inner edges which are incident on the end vertices of the dummy edges become perpendicular to this straight path.

The shape of the dummy edges and the other incident edges are same as case 1. For this case we contract the corresponding dummy edge for  $g_k$  such that the inner edge  $e_{gko}$  remains safe and bend appears on the inner edge  $e_{gke}$ . If we keep  $e_{gko}$  safe, then the outer edge which share a common face with  $e_{gko}$  and which is incident to a common vertex with  $e_{gko}$  remains safe. Therefore, bend appears on the other outer edge which is incident to an end vertex of the dummy edge for  $g_k$ . Bend appears on every bad edge for this case also, whereas the good edges remain safe. Figure 3.14(d) illustrates the drawing obtained after contracting the dummy edges in Figure 3.14(c). Similar to case 1 it can be easily shown that the contraction of the dummy edges cannot produce more than one bend on

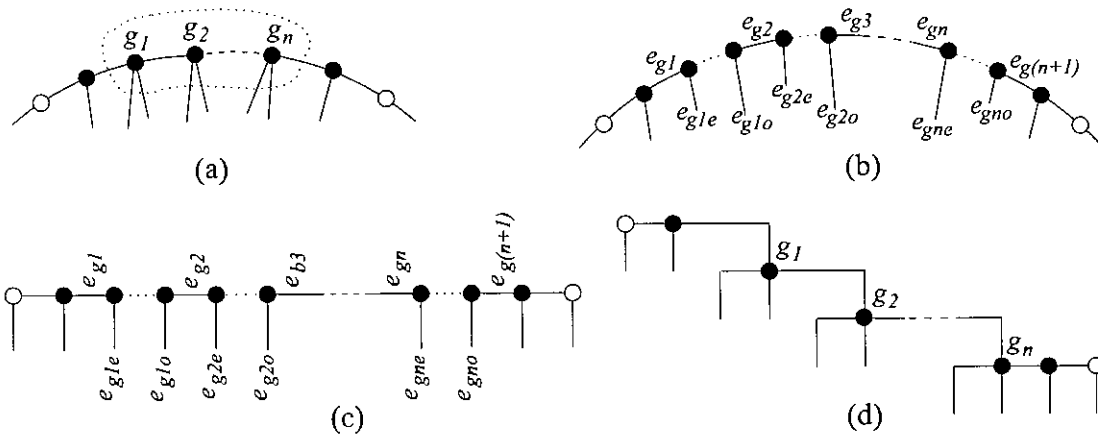


Figure 3.14: *Effect of dummy edge contraction for Case 2.*

a single edge for this case also.

**Case 3:** Dummy edges of  $D$  are created from the tree components of  $G$ .

In this case the vertices of a tree component are always inner vertices. Therefore, the dummy edges, which are created from the vertices of tree components, and the other edges, which are incident to the end vertices of those dummy edges, are always the inner edges. In the orthogonal drawing  $D$  these inner dummy edges can take any one of the nine shapes in Figure 3.11 or their rotated forms. According to the third observation of contraction we can say that after contraction of a dummy edge at most one edge, which is incident to an end vertex of that dummy edge, can always be kept safe.

Since  $G$  contains only valid 4-components, a tree component  $H$  of  $G$  can have at most one bad edge incident to a vertex of  $H$ . Figures 3.6(b) and 3.6(c) illustrate two tree components in which the first one has a bad edge which is labeled by  $e_{t_0}$ . We start our contraction from the corresponding dummy edge of the vertex  $t_1$ . If  $t_1$  has any bad edge incident to it, then after contraction we keep that bad edge safe and bend appears on the other three incident edges. If  $t_1$  has no bad edge incident to it, then we keep any of the four incident edges safe and create bend on the other three incident edges. Next after the contraction of the corresponding dummy edge of  $t_k$ ,  $k > 1$ , we keep the incident edge

$e_{t_{(k-1)-k}}$  safe and create bend on the other three incident edges.

We now show that, for this case the contraction operation cannot produce more than one bend on a single edge. Suppose for contradiction that an edge  $e_{t_k-(k+1)}$  in between two vertices  $t_k$  and  $t_{(k+1)}$  of a tree component contains two bends on it. It is observed that one contraction operation can produce one bend on an edge. Therefore, one of the two bends on  $e_{t_k-(k+1)}$  is produced due to the contraction of the corresponding dummy edges of  $t_{(k+1)}$ . According to our algorithm, after the contraction of the corresponding dummy edge of  $t_{(k+1)}$  the edge  $e_{t_k-(k+1)}$  is kept safe. Therefore,  $e_{t_k-(k+1)}$  cannot contain more than one bend on it. Moreover, the bad edge is also kept safe during the contraction so that the edge also cannot contain more than one bend. The rest of the edges are either good edges or they are connected with the vertices of degree three. Therefore, the contraction operation cannot produce more than one bend on a single edge i.e. after contraction the drawing will remain one bend drawing.

**Case 4:** Dummy edges of  $D$  are created from the non-tree components of  $G$ .

In this case both the dummy edges and the other incident edges are the inner edges. In the orthogonal drawing  $D$  the inner dummy edges can take any one of the nine shapes in Figure 3.11 or their rotated forms. Therefore, after the contraction of each dummy edge at most one edge, which is incident to an end vertex of that dummy edge, can always be kept safe.

Since  $G$  contains only valid 4-components, a non-tree component of  $G$  can have at most one cycle in it and can have no bad edge incident to a vertex of it. Figure 3.6(d) illustrates a non-tree component. We start our contraction from the dummy edge created from the vertex  $n_1$ . According to the labeling technique,  $n_1$  is a vertex on the cycle of that 4-component and an edge  $e_{n_0}$  of that cycle is incident to  $n_1$ . During the contraction of the dummy edge for  $n_1$  we keep  $e_{n_0}$  safe and bend can appear on the other three incident edges. We next contract the dummy edge which is created from  $n_k$ ,  $k > 1$ . During the contraction we keep the incident edge  $e_{n_{(k-1)-k}}$  safe and create bend on the other three

incident edges.

We now show that, for this case the contraction operation cannot produce more than one bend on a single edge. If we break the cycle of the non-tree component in Figure 3.6(d) at the edge  $e_{n_0}$  as shown in Figure 3.15, then the non-tree component becomes a tree component. The contraction technique is the same as the technique for tree component, where the contraction is started by keeping  $e_{n_0}$  safe. Since the contraction technique for tree component doesn't produce more than one bend on a single edge, this technique will also do the same. Therefore, after contraction the drawing will remain one bend drawing.

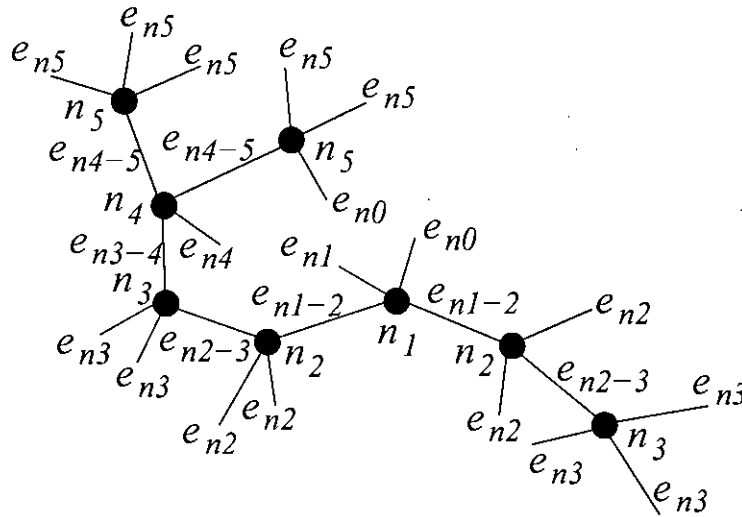


Figure 3.15: The cycle of the non-tree component in Figure 3.6(d) is broken at the edge  $e_{n_0}$ .

□

We call the algorithm for the contraction of the dummy edges of the zero bend orthogonal drawing  $D$ , which is described in the proof of Lemma 3.2.7, as **Dummy-Edge-Contract**. After contracting the dummy edges of  $D$ , using this algorithm, we get an orthogonal drawing  $D'$  as illustrated in Figure 3.5(h).

In the next section we give a proof for sufficiency of Theorem 3.2.1.

### 3.2.7 Proof for Sufficiency of Theorem 3.2.1

We transform the given triconnected planar 4-graph  $G$  into a triconnected cubic plane graph  $G_4$  such that  $G_4$  satisfies the necessary and sufficient condition for a graph to have a rectangular drawing. We can find  $G_4$  by contracting the maximal bad cycles of a graph  $G_3$  such that  $G_3$  is obtained from  $G$  by the vertex-insertion and the edge-insertion operations. Lemma 2.2.1 implies that  $G_4$  has a rectangular drawing. We can find a corresponding rectangular drawing  $R$  of  $G_4$  by using the algorithm **Rectangular-Draw**. According to Lemmas 3.2.4 and 3.2.6, every contracted maximal bad cycle of  $G_3$  has a feasible orthogonal drawing. We can find the feasible orthogonal drawings of the contracted maximal bad cycles by the algorithm **Modified-Feasible-Draw**. By patching these feasible orthogonal drawings into  $R$  we can get a no bend orthogonal drawing  $D$ . We can contract the dummy edges of  $D$  using the algorithm **Dummy-Edge-Contract**. According to Lemma 3.2.7 the resulting drawing  $D'$  is a 1-bend orthogonal drawing. We now replace each of all the dummy vertices of degree two of  $D'$  with a bend. The dummy vertices are placed either on the safe-edges or on the desired-edges. The algorithm **Dummy-Edge-Contract** produce bend neither on the safe-edges nor on the desired-edges. Therefore, the resulting drawing is a 1-bend orthogonal drawing of the given triconnected planar 4-graph  $G$ .  $\square$

In the next section we formally describe our algorithm.

## 3.3 The Algorithm

We are now ready to present our algorithm for finding one bend orthogonal drawings of the graphs which satisfy the sufficient condition of Theorem 3.2.1. As a pre-processing we find a planar embedding of the given graph by a simple linear-time algorithm (e.g. the algorithm of Shih and Hsu [SH99]) and let  $G$  be the resulting graph. The algorithm is as follows.

**Algorithm 1-Bend-Ortho-Draw ( $G$ )**

begin

- 1 find the desired  $\alpha$ -face and convert that face to the outer face;
- 2 let  $G_1$  be the resulting graph;
- 3 add four dummy vertices of degree two on four safe-edges of  $C_o(G_1)$ ;
- 4 let  $G_2$  be the resulting graph;
- 5 replace each of all the vertices of degree four of  $G_2$  with a dummy edge;
- 6 let  $G_3$  be the resulting graph in which  $C_1, C_2, \dots, C_l$  be the maximal bad cycles;
- 7 for each  $i, 1 \leq i \leq l$ , construct genealogical trees  $T_{C_i}$  and determine the desired-paths for every cycle in  $T_{C_i}$ ;
- 8 for each  $i, 1 \leq i \leq l$ , find an orthogonal drawing  $D(G(C_i))$  of  $G(C_i)$  feasible for any desired-path of  $C_i$  by **Modified-Feasible-Draw**;
- 9 let  $G_4$  be a plane graph obtained from  $G_3$  by contracting each  $G(C_i), 1 \leq i \leq l$ , to a single vertex  $v_i$ ;
- 10 find a rectangular drawing  $D(G_4)$  of  $G_4$  by **Rectangular-Draw**;
- 11 patch the drawings  $D(G(C_1)), D(G(C_2)), \dots, D(G(C_l))$  into  $D(G_4)$  to get a zero bend orthogonal drawing  $D$ ;
- 12 contract all the dummy edges of  $D$  by **Dummy-Edge-Contract** to get a 1-bend orthogonal drawing  $D'$ ;
- 13 replace each of all the dummy vertices with a bend to get a 1-bend orthogonal drawing of  $G$ ;

end.

We now have the following theorem regarding the time complexity of the algorithm **1-Bend-Ortho-Draw**.

**Theorem 3.3.1** *Let  $G$  be a triconnected planar 4-graph and  $G$  contains at least one  $\alpha$ -*



face  $F$  such that the planar embedding of  $G$  having  $F$  as the outer face contains only valid 4-components and contains no critical-cycle. Then the algorithm **1-Bend-Ortho-Draw** finds a 1-bend orthogonal drawing of  $G$  in linear time.

**Proof.** The input of this algorithm is a planar embedding of  $G$  which is obtained from  $G$  by using a linear time algorithm of Shih and Hsu [SH99].

We can find the desired  $\alpha$ -face  $F$  of  $G$  by traversing the contours of the faces of  $G$  and this can be done in linear time. After finding  $F$ , we can change the embedding of  $G$  such that  $F$  becomes the outer face of the new embedding. This new embedding of  $G$  can be found in linear time [NR04].

If we traverse the outer cycle of  $G_1$  once, then we can find the four safe-edges of  $C_o(G_1)$ . After finding the safe-edges the vertex-insertion operation takes constant time.

We can find all the vertices of degree four of  $G_2$  by checking all the vertices of  $G_2$  once. Therefore, the edge-insertion operation takes linear time.

According to the Lemma 3 of [RNN99] the genealogical tree  $T_{C_i}$  of a maximal bad cycle  $C_i$  of  $G_3$  can be found in linear time. Using  $T_{C_i}$ , we can determine the desired-paths for all the descendant cycles in  $T_{C_i}$  in linear time.

The algorithm **Feasible-Draw** finds an orthogonal drawing  $D(G(C_i))$  of  $G(C_i)$  feasible for any green path of  $C_i$  whereas the algorithm **Modified-Feasible-Draw** finds an orthogonal drawing  $D(G(C_i))$  of  $G(C_i)$  feasible for any desired-path of  $C_i$ . According to the Lemma 2.2.3 the algorithm **Feasible-Draw** finds  $D(G(C_i))$  in linear time. Therefore, the algorithm **Modified-Feasible-Draw** can find an orthogonal drawing  $D(G(C_i))$  of  $G(C_i)$  feasible for any desired-path of  $C_i$  in linear time.

It is also shown in Lemma 13 of [RNN99] that contracting all the maximal bad cycles of  $G_3$  and patching all the feasible orthogonal drawings in  $D$  both takes linear time.

For finding the rectangular drawing  $D(G_4)$  of  $G_4$  we use the linear time algorithm **Rectangular-Draw** of [RNN98].

If the labels of the four edges, which are incident to the end vertices of a dummy edge

$e$ , are given, then the dummy edge  $e$  can be contracted in constant time. Therefore, the algorithm **Dummy-Edge-Contract** takes linear time to contract all the dummy edges of  $D$ .

Finally, we can replace each of all the dummy vertices with a bend in linear time and thus get a 1-bend orthogonal drawing of  $G$ . The above discussion implies that the algorithm **1-Bend-Ortho-Draw** finds a 1-bend orthogonal drawing of  $G$  in linear time.

□

# Chapter 4

## Conclusion

In this thesis we have studied the orthogonal drawing generating problem for the triconnected planar 4-graphs. The results of this thesis are summarized as follows:

- We have derived a sufficient condition for the triconnected planar 4-graphs to have 1-bend orthogonal drawings.
- We have given a constructive proof for the sufficient condition and from that constructive proof we have developed a linear-time algorithm for finding the 1-bend orthogonal drawings of those triconnected planar 4-graphs which satisfy the sufficient condition.

Some interesting directions in which the future research works can be done are as follows.

- A necessary and sufficient condition can be derived for the triconnected planar 4-graphs to have 1-bend orthogonal drawings. It can be easily shown that a triconnected planar 4-graph must contain an  $\alpha$ -face to have a 1-bend orthogonal drawing. If the arrangements of the dummy edges in the rectangular drawing can be predicted, then the necessity of having valid 4-components and not having critical-cycles can be proved.

- Not every planar 4-graph has 1-bend orthogonal drawing. Therefore, it is interesting to find the classes of planar 4-graphs that have 1-bend orthogonal drawings. Condition can also be derived for every planar 4-graphs for having 1-bend orthogonal drawings.

# Bibliography

- [BK98] Biedl, T. C. and Kant, G., *A better heuristic for orthogonal graph drawings*, Computational Geometry Theo. Appl., Vol-9, pp 159-180, 1998.
- [CNAO85] Chiba, N., Nishizeki, T., Abe, S. and Ozawa, T., *A linear algorithm for embedding planar graphs using PQ-trees*, J. Comput. Syst. Sci., Vol-30, pp 54-76, 1985.
- [GT01] Garg, A. and Tamassia, R., *On the computational complexity of upward and rectilinear planarity testing*, SIAM J. Comput., Vol-31(2), pp 601-625, 2001.
- [Kan96] Kant, G., *Drawing planar graphs using the canonical ordering*, Algorithmica, Vol-16, pp 4-32, 1996.
- [KH97] Kant, G. and He, X., *Regular edge labeling of 4-connected plane graphs and its applications in graph drawing problems*, Theoretical Computer Science, Vol-172, pp 175-193, 1997.
- [LMS98] Liu, Y., Morgana, A. and Simeone, B., *A linear algorithm for 2-bend embeddings of planar graphs in the two-dimensional grid*, Discrete Applied Mathematics, Vol-81, pp 69-91, 1998.
- [MM96] Mehlhorn, K. and Mutzel, P., *On the embedding phase of the Hopcroft and Tarjan planarity testing algorithm*, Algorithmica, Vol-16, pp 233-242, 1996.
- [NR04] Nishizeki, T. and Rahman, M. S., *Planar Graph Drawing*, World Scientific, Singapore, 2004.

- [RN02] Rahman, M. S. and Nishizeki, T., *Bend-minimum orthogonal drawings of plane 3-graphs*, Proc. of WG '02, Lect. Notes in Computer Science, Springer, Vol-2573, pp 367-378, 2002.
- [RNN03] Rahman, M. S., Nishizeki, T. and Naznin, M., *Orthogonal drawings of plane graphs without bends*, Journal of Graph Algorithms and Applications, Vol-7, No. 4, pp 335-362, 2003.
- [RNN98] Rahman, M. S., Nakano, S. and Nishizeki, T., *Rectangular grid drawings of plane graphs*, Comp. Geom. Theo. Appl., Vol-10(3), pp 203-220, 1998.
- [RNN99] Rahman, M. S., Nakano, S. and Nishizeki, T., *A linear algorithm for bend-optimal orthogonal drawings of triconnected cubic plane graphs*, Journal of Graph Alg. and Appl., <http://jgaa.info>, Vol-3(4), pp 31-62, 1999.
- [SH99] Shih, W. K. and Hsu, W. L., *A new planarity test*, Theoretical Computer Science, Vol-223, pp 179-191, 1999.
- [She95] Sherwani, N., *Algorithms for VLSI Physical Design Automation*, Kluwer Academic Publishers, Boston, 1995.
- [Tho84] Thomassen, C., *Plane representations of graphs*, Progress in Graph Theory, Academic Press Canada, Ontario, pp 43-69, 1984.
- [TNU06] Tayu, S., Nomura, K. and Ueno, S., *On the Two-Dimensional Orthogonal Drawing of Series-Parallel Graphs (Extended Abstract)*, ISCAS, pp 1796-1799, 2006.

# Index

- $\alpha$ -face, 25
- $k$ -connected, 15
- $k$ -graph, 14
- 4-component, 28
  - valid, 29
- adjacent incident edge, 47
- ancestor cycle, 24
- bad cycle, 20
- bad edges, 28
- bend, 4
- child-cycle, 24
- connected, 15
- connectivity, 15
- contour path, 21
- critical-cycle, 26
- cycle, 15
- descendant cycle, 24
- desired  $\alpha$ -face, 30
- desired-edge, 39
- desired-path, 41
- disconnected, 15
- drawing, 1
  - grid drawing, 5
  - orthogonal drawing, 3, 4
  - rectangular drawing, 4, 19
  - straight line drawing, 3
- edge-insertion, 36
- feasible orthogonal drawing, 22, 23
- genealogical tree, 24
- good edges, 28
- graph, 1, 13
- green path, 21
- independent, 18
- inner contour path, 41
- inner edge, 17
- inner vertex, 17
- internal node, 16
- $k$ -legged cycle, 17
- leaf, 16
- leg, 17
- leg-vertex, 17

- loop, 13, 15
- maximal bad cycle, 20
- multiple edges, 13
- node, 16
- non-tree component, 29
- opposite incident edge, 47
- outer boundary, 17
- outer cycle, 17
- outer edge, 17
- outer face, 17
- outer vertex, 17
- patching operation, 45
- path, 15
- planar
  - drawing, 2
  - graph, 2, 16
- plane graph, 17
- red path, 21
- root, 16
- rooted tree, 16
- safe-edge, 26
- simple graph, 13
- string, 27
  - bad, 27
  - good, 27
- subgraph, 14
- tree, 16
- tree component, 29
- triconnected, 16
- vertex-insertion, 33
- walk, 15

