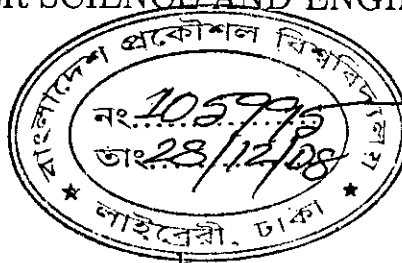


Straight-Line Grid Drawings of Planar Graphs with Sub-Quadratic Area

A thesis submitted to the Department of Computer Science
and Engineering in partial fulfillment of the requirement for the degree of
DOCTOR OF PHILOSOPHY

IN
COMPUTER SCIENCE AND ENGINEERING



by

Md. Rezaul Karim


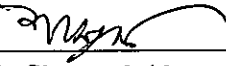

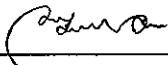

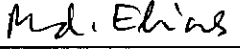
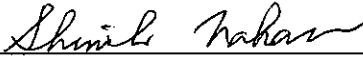
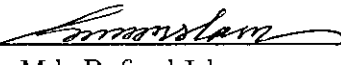


DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DHAKA 1000, BANGLADESH

November 2008

This thesis titled “Straight-Line Grid Drawings of Planar Graphs with Sub-Quadratic Area” submitted by Md. Rezaul Karim, Roll No. P10050501P, Session October 2005, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh, has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Doctor of Philosophy on November 25, 2008.

Board of Examiners

1. 
Dr. Md. Saidur Rahman
Professor & Head
Department of CSE, BUET, Dhaka
Chairman
(Supervisor & Ex-officio)
2. 
Dr. Md. Shamsul Alam
Professor
Department of CSE, BUET, Dhaka (on leave)
CS Department, Stamford University, Bangladesh
Member
3. 
Dr. Md. Abul Kashem Mia
Professor
Department of CSE, BUET, Dhaka &
Associate Director, ICT, BUET, Dhaka
Member
4. 
Dr. Md. Mostofa Akbar
Associate Professor, Department of CSE, BUET, Dhaka
Member
5. 
Dr. Masud Hasan
Assistant Professor, Department of CSE, BUET, Dhaka
Member
6. 
Dr. Md. Elias
Professor, Department of Mathematics, BUET, Dhaka
Member
7. 
Dr. Shin-ichi Nakano
Professor
Department of Computer Science, Faculty of Engineering
Gunma University, Gunma 376-8515, Japan
Member
(External)
8. 
Dr. Md. Rafiqul Islam
Professor
CSE Discipline, Khulna University, Khulna- 9208
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.

Signature of the candidate



(Md. Rezaul Karim)

Dedication

To my parents

Contents

| | |
|---|----------|
| Board of Examiners | i |
| Candidate's Declaration | ii |
| Dedication | iii |
| Acknowledgments | xi |
| Abstract | xiii |
| 1 Introduction | 1 |
| 1.1 Graph Drawing Conventions | 2 |
| 1.1.1 Planar Drawing | 3 |
| 1.1.2 Straight Line Drawing | 4 |
| 1.1.3 Grid Drawing | 5 |
| 1.1.4 Straight-Line Grid Drawing | 6 |
| 1.2 Drawing Aesthetics | 6 |
| 1.3 Motivation | 8 |
| 1.4 Scope of this Thesis | 9 |
| 1.4.1 Straight-line Grid Drawings | 10 |
| 1.4.2 Graph Partitioning | 14 |
| 1.4.3 Interconnection Networks | 15 |
| 1.5 Summary | 15 |

| | |
|---|-----------|
| 2 Preliminaries | 19 |
| 2.1 Basic Terminology | 19 |
| 2.1.1 Graphs | 19 |
| 2.1.2 Subgraphs | 20 |
| 2.1.3 Connectivity | 21 |
| 2.1.4 Paths and Cycles | 22 |
| 2.1.5 Trees | 23 |
| 2.2 Planar Graphs | 25 |
| 2.2.1 Planar Graphs and Plane Graphs | 25 |
| 2.2.2 Euler's Formula | 26 |
| 2.2.3 Outerplanar Graphs | 26 |
| 2.2.4 Dual Graphs | 27 |
| 2.3 Algorithms and Complexity | 28 |
| 2.3.1 The Notations $O(n)$, $\Omega(n)$, $o(n)$ | 29 |
| 2.3.2 Polynomial Algorithms | 29 |
| 2.3.3 <i>NP</i> -Complete | 29 |
| 2.4 A Graph Traversal Algorithm | 31 |
| 3 Doughnut Graphs | 32 |
| 3.1 Preliminaries | 33 |
| 3.2 Properties of Doughnut Graphs | 36 |
| 3.3 Straight-Line Grid Drawings of Doughnut Graphs | 43 |
| 3.4 Conclusion | 49 |
| 4 Spanning Subgraphs of Doughnut Graphs | 50 |
| 4.1 Preliminaries | 52 |
| 4.2 A Characterization for Spanning Subgraphs | 53 |
| 4.3 A Sufficient Condition for Linear Area Drawings | 62 |
| 4.4 Conclusion | 64 |

| | | |
|----------|---|------------|
| 5 | Label-Constrained Outerplanar Graphs | 65 |
| 5.1 | Preliminaries | 66 |
| 5.2 | Drawings of Label-Constrained Outerplanar Graphs | 70 |
| 5.3 | Recognition of Label-Constrained Outerplanar Graphs | 75 |
| 5.4 | Conclusion | 82 |
| 6 | Partitioning of Doughnut Graphs | 84 |
| 6.1 | Preliminaries | 84 |
| 6.2 | Finding Hamiltonian Path in Doughnut Graphs | 86 |
| 6.3 | k -Partition of a Doughnut Graph | 92 |
| 6.4 | Conclusion | 95 |
| 7 | Doughnut Graphs for Interconnection Networks | 96 |
| 7.1 | Parameters of Interconnection Networks | 97 |
| 7.2 | A Simple Routing Scheme | 98 |
| 7.3 | Recursive Structure of Doughnut Graphs | 105 |
| 7.4 | Topological Properties of Doughnut Graphs | 108 |
| 7.5 | Conclusion | 110 |
| 8 | Conclusion | 111 |
| | References | 114 |
| | List of Publications | 121 |
| | Index | 122 |

List of Figures

| | | |
|-----|--|----|
| 1.1 | A social network or activity graph. | 2 |
| 1.2 | A threatening activity (pattern graph). | 3 |
| 1.3 | Planar and non-planar drawings | 4 |
| 1.4 | A straight line drawing | 5 |
| 1.5 | A straight line grid drawing | 6 |
| 1.6 | A linear-area drawing of a doughnut graph | 12 |
| 1.7 | An $O(n \log n)$ area drawing of a label-constrained outerplanar graph | 13 |
| 1.8 | k -partition of a k -connected planar graph | 14 |
| 1.9 | A 5-partition of a 5-connected planar graph G with basis 5. | 14 |
| 2.1 | A graph with six vertices and eight edges. | 20 |
| 2.2 | (a) A graph G , (b) an induced subgraph of G and (c) a spanning subgraph of G | 21 |
| 2.3 | (a) A connected graph and (b) a disconnected graph. | 22 |
| 2.4 | A tree. | 23 |
| 2.5 | Three planar embeddings of the same graph. | 25 |
| 2.6 | A plane graph G | 26 |
| 2.7 | (a) A plane graph G and its dual graph G^* and (b) a plane graph G and its weak dual graph G' | 27 |
| 2.8 | An outerplanar graph G and its weak dual graph G' | 28 |

| | | |
|-----|---|----|
| 3.1 | A linear-area drawing of a doughnut graph | 34 |
| 3.2 | (a) Input graph G and (b) a triangulated plane graph G' of graph G where extra edges are drawn by dotted lines | 35 |
| 3.3 | Illustration for the construction of a planar embedding Γ of a p -doughnut graph G | 38 |
| 3.4 | Illustration for drawing algorithm of a doughnut graph | 46 |
| 3.5 | Illustration of case where z_p has two neighbors on C_1 | 47 |
| 4.1 | A linear-area drawing of a spanning subgraph of a doughnut graph | 51 |
| 4.2 | Illustration for two different types of triangulation of a quadrangle face | 57 |
| 4.3 | Examples of an α -face and a β -face | 57 |
| 4.4 | Illustration for valid triangulations of α -faces | 58 |
| 4.5 | Examples of a β_1 -face and a β_2 -face | 59 |
| 4.6 | Illustration for valid triangulations of β_1 -faces | 60 |
| 4.7 | Illustration for valid triangulations of β_2 -faces | 61 |
| 5.1 | Vertex labeling of a binary tree T | 67 |
| 5.2 | Illustration for different types of paths in a ordered binary tree T rooted at v_0 | 69 |
| 5.3 | Two rooted ordered binary dual trees of a maximal outerplanar graph | 71 |
| 5.4 | (a) A maximal outerplanar graph G , (b) the dual rooted ordered tree T_r of G and (c) a spanning-tree T' of $G - \{u, v\}$ is drawn by the solid lines. | 72 |
| 5.5 | Different steps of drawing of a label constrained outerplanar graph | 73 |
| 5.6 | (a) A maximal outerplanar graph G , (b) the rooted ordered binary dual tree T_r of G and (c) the rooted ordered binary dual tree T_p of G | 77 |
| 5.7 | Illustration for labeling of vertices of two rooted ordered binary dual tree of a maximal outerplanar graph | 78 |
| 5.8 | Illustration of different cases of computing $L_x(p)$ from given $L_r(T_r)$ | 79 |

| | | |
|------|---|-----|
| 5.9 | (a) A trivial outerplanar graph G with maximum degree $n - 1$ and (b) a straight-line grid drawing of G with $O(n)$ area. | 83 |
| 5.10 | An example of an outerplanar graph G which has maximum degree $O(n^{0.5})$ | 83 |
| 6.1 | (a) G is a p -doughnut graph where $p = 4$ and (b) doughnut embedding of G | 85 |
| 6.2 | Illustration for Hamiltonian path between two vertices on cycle C_1 | 88 |
| 6.3 | Illustration for Hamiltonian path between two vertices on C_2 where one of the vertex index is odd. | 89 |
| 6.4 | Illustration for Hamiltonian path between two vertices on C_2 where one of the vertex index is even. | 90 |
| 6.5 | Illustration for Hamiltonian path between a vertex on C_1 and a vertex on C_2 | 91 |
| 6.6 | Illustration for Hamiltonian path between a vertex on C_1 and a vertex on C_3 | 91 |
| 6.7 | A 5-partition of a 5-connected planar graph G | 92 |
| 6.8 | A 5-partition of a 5-connected planar graph G with basis 5. | 92 |
| 6.9 | (a) Hamiltonian path HP_{x_2, z_6} of G , (b) a 7-partition of G and (c) a 4-partition of G | 94 |
| 7.1 | Illustration for shortest path between two vertices on C_2 of a doughnut graph. | 101 |
| 7.2 | Illustration for shortest path between two vertices on Cycle C_1 | 102 |
| 7.3 | Illustration for shortest path between a vertex on cycle C_2 and a vertex on cycle C_1 | 104 |
| 7.4 | Illustration for shortest path between a vertex on C_1 and a vertex on C_3 | 105 |
| 7.5 | (a) A doughnut embedding of a p -doughnut graph of G and (b) straight-line grid drawing of G with linear area. | 106 |
| 7.6 | (a) A p -doughnut graph G where $p=4$ and (b) illustration for four partitions of edges of G | 107 |
| 7.7 | Illustration for recursive structure of a doughnut graph | 109 |

List of Tables

| | | |
|-----|---|----|
| 1.1 | Classes of planar graphs and their area requirement. | 16 |
| 1.2 | Results on graph partitioning | 17 |
| 1.3 | Topological comparison of various Cayley graphs with doughnut graphs. . . | 18 |

Acknowledgments

First of all, I would like to express my profound gratitude to Allah, the most gracious and the most merciful, for giving me adequate physical and mental strength for doing this research work. I am grateful to my supervisor Prof. Dr. Md. Saidur Rahman for introducing me in this beautiful and fascinating field of graph drawing, and for teaching me the fundamentals for doing research work. I express my gratitude for his patience in reading my inferior drafts and for making them publishable. I again express my heart-felt and most sincere gratitude to him for his constant supervision, valuable advice, and continual encouragement, without which this thesis would have not been possible.

I would like to express my utmost gratitude to the members of my doctoral committee, Prof. Dr. Md. Shamsul Alam, Prof. Dr. Md. Abul Kashem Mia, Dr. Md. Mostofa Akbar, Dr. Masud Hasan and Prof. Dr. Md. Elias for their rigorous monitoring my research progress and valuable suggestions. I would like to express my utmost gratitude to the external members of the board of examiners, Prof. Dr. Shin-ichi Nakano and Prof. Dr. Md. Rafiqul Islam, for their valuable suggestions.

My special thanks goes to all other members of the our Graph Drawing Research Group; in particular, to Md. Jawaherul Alam and Kaiser Md. Nahiduzzaman. I also thankful to Mr. Md. Abdus Sattar, Assistant Prof., Dept. of CSE, for introducing me with Prof. Dr. Md. Saidur Rahman and provided continual emotional support and understanding.

I am very much thankful to the authority of University of Dhaka for allowing me study

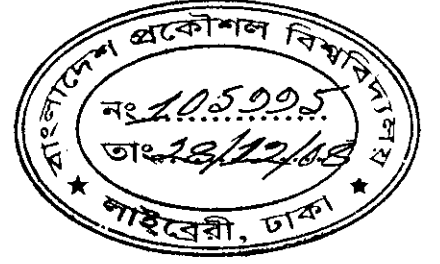
leave to complete my research work. My special thanks goes to my friends and colleagues who have given me their time, companionship, professional and personal help. I would like to acknowledge with sincere thanks the all-out cooperation and services rendered by the faculty members and staff of the Dept. of Computer Science and Engineering, BUET.

My full-hearted thanks should go to my family members for their unconditional love and faith in me. I will forever remain indebted to them. My full-hearted thanks should also go to my parents-in-law for their understanding, love and faith in me.

I would like to thank my wife Masuma Sultana for her love, mental support, constant encouragement and utmost patience. Finally I would like to express my endless love to my only son Reza Abdullah who missed me a lot during this work.

Abstract

This thesis deals with area efficient straight-line drawings of planar graphs. It is well known that a planar graph of n vertices admits a straight-line grid drawing on a grid of area $O(n^2)$. A lower bound of $\Omega(n^2)$ on the area-requirement for straight-line grid drawings of certain planar graphs are also known. In this thesis, we introduce some classes of planar graphs that admit straight-line grid drawing with sub-quadratic area. We introduce “doughnut graphs”, a subclass of 5-connected planar graphs as well as 3-outerplanar graphs, which admits a straight-line grid drawing on a grid of area $O(n)$. We introduce a subclass of 4-connected planar graphs that admits straight-line grid drawing with linear area. We also introduce a subclass of outerplanar graphs, which we call “label-constrained outerplanar graphs,” that admits straight-line grid drawings with $O(n \log n)$ area. We give linear-time algorithms to find such drawings. We also give linear-time algorithms for recognition of these classes of graphs. We have studied the k -partition problem for newly introduced classes of graphs, and found some interesting results for “doughnut graphs.” We give a linear-time algorithm for finding k -partition of a “doughnut graph.” We also study the topological properties of these classes of planar graphs for finding their suitable applications. We propose the class of “doughnut graphs” as a promising class of interconnection networks due to its regularity, smaller diameter, maximal fault tolerance, recursive structure and a very simple efficient routing scheme.



Chapter 1

Introduction

A graph is an abstract structure that is used to model information. Many real-world situations can conveniently be described by means of graphs. Graphs may be used to represent any information that can be modeled as objects and connections between those objects. Thus graph drawing addresses the problem of constructing geometric representations of conceptual structures that are modeled by graphs. For example, graphs are used to represent social network where each node represents an actor (generally a person or an object or an organization, etc.) and each edge represents a relationship or a communication between the actors. These graphs are typically drawn as diagrams with texts at the vertices and the line segments joining the vertices. Fig. 1.1 represents a social network. Fig. 1.2 illustrates a pattern graph which represent a threatening pattern. To understanding, monitoring and controlling different activity like terrorism, drug trafficking etc., a social network analyst try to find a pattern graph as illustrated in Fig. 1.2 as a subgraph in the social network as illustrated in Fig. 1.1.

Other than social networks, automatic graph drawings have important applications to key computer technologies such as software engineering (data flow diagrams, subroutine-call graphs, program nesting trees), real-time systems (state-transition diagrams), artificial intelligence (knowledge-representation diagrams) etc. Further application can be

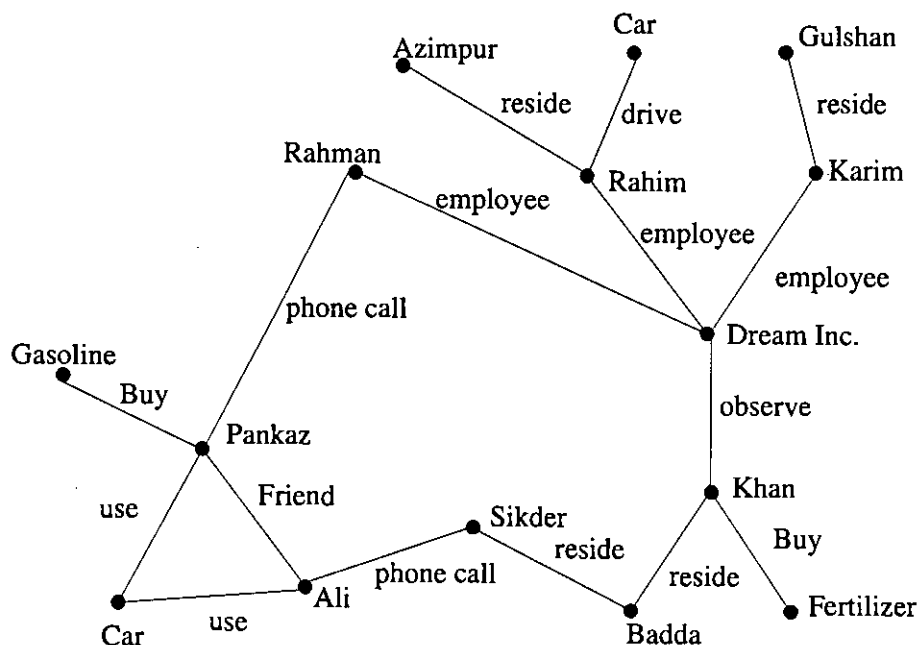


Figure 1.1: A social network or activity graph.

found in other science and engineering disciplines, such as medical science (concept lattices), chemistry (molecular drawings), civil engineering (floorplan maps), cartography (map schematics).

In this chapter we provide the necessary background and motivation for this study on graph drawings. In Section 1.1 we discuss about some important drawing conventions that have been studied in this thesis. We discuss different drawing aesthetics in Section 1.2. In Section 1.4 we describe the scope of this thesis. We devote Section 1.5 to summarize our new results together with known ones.

1.1 Graph Drawing Conventions

There are different graph drawing styles. The drawing styles are - planar drawing, straight-line drawing, grid drawing, orthogonal drawing, convex drawing, Polyline drawing, rectangular drawing, box-rectangular drawing, visibility drawing, etc. In the different drawing

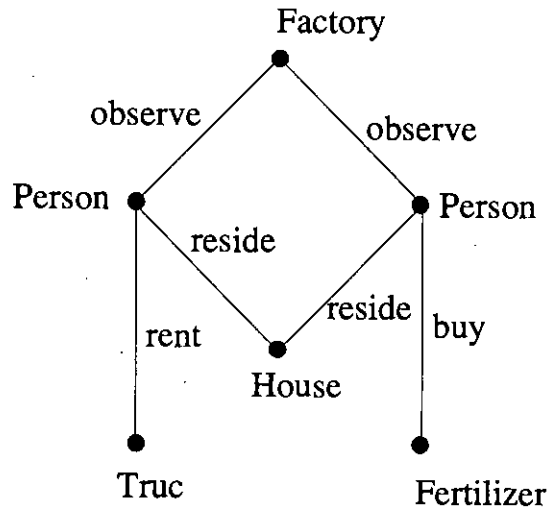


Figure 1.2: A threatening activity (pattern graph).

styles, vertices are represented by small circles, or points, or rectangular boxes, or horizontal line segments; and the edges are represented by straight-line segments, or chain of horizontal and vertical line segments, or horizontal line segments, or vertical line segments etc. In this thesis, we address area efficient drawings of planar graphs. Planar drawing style is very much relevant to the drawings of planar graphs which improves the readability of the drawing. It is natural to draw each edge of a graph as a straight-line segment between its end vertices. We need to draw the vertices on the integer coordinates when we are going to compare the area requirement for two or more different drawings. Hence grid drawing style is essential when the area of a drawing is concerned. We now discuss the following drawing styles which we study in this thesis.

1.1.1 Planar Drawing

A *planar drawing* is a drawing of a graph in which no two edges intersect in the drawing except at their common end vertex. Fig. 1.3(a) and Fig. 1.3(b) illustrate a planar drawing and a non-planar drawing of the same graph respectively. It is always preferable to find planar drawing if a graph has a planar drawing. Because planar drawings are relative

easy to understand in comparison with the non-planar drawings. Unfortunately not every graph has a planar drawing. A graph is called *planar graph* if it admits planar drawing.

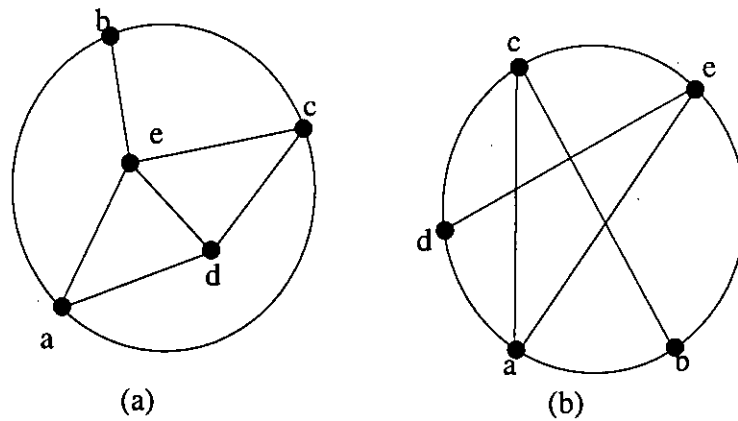


Figure 1.3: (a) A planar drawing, and (b) a non-planar drawing of the same graph.

To find a planar drawings of a given graph, it is needed to test whether the given graph is planar or not. If the graph is planar, then he/she needs to find a planar representation of the graph which is a data structure representing adjacency lists: lists in which the edges incident to a vertex are ordered, all clockwise or all counterclockwise, according to the planar representation. Kuratowski [Kur30] gave the first complete characterization of planar graphs. Unfortunately Kuratowski's characterization does not lead to an efficient algorithm for planarity testing. Linear-time algorithms for this problem have been developed by Hopcroft and Tarjan [HT74], and Booth and Lucker [BL76]. Chiba *et al.* [CNAO85], Mutzel [Mut92] and Shin and Hsu [SH99] gave linear-time algorithms for finding a planar representation of a planar graph. A planar graph with a fixed planar representation is called a *plane graph*.

1.1.2 Straight Line Drawing

Straight line drawing is the most typical and widely studied drawing styles. It is natural to draw each edge of a graph as a straight line segment 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*. Fig. 1.4 depicts a straight line drawing of a graph.

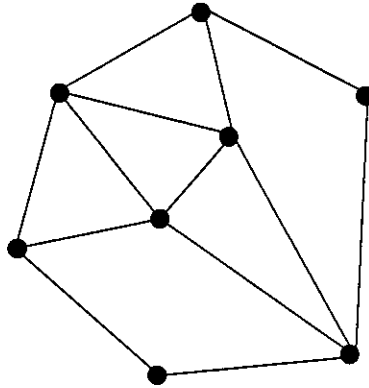


Figure 1.4: A straight line drawing.

Wagner [Wag36], Fary [Far48] and Stein [Ste51] independently proved that every planar graph has a straight line representation. Their proofs immediately yield polynomial-time algorithms to find a straight-line drawing of a given plane graph. Many works have been published on straight line drawings of planar graphs [DETT94].

1.1.3 Grid Drawing

Grid drawing is a drawing of a graph in which vertices as well as bends are located at integer coordinates. In this drawing style, the minimum distance between two vertices is at least unit distance, and the integer coordinates of vertices allow such drawings to be rendered on displays, such as computer screen, without any distortions due to truncation and round-off errors. The *size* of an integer grid required for a grid drawing is measured by the size of the smallest rectangle on the grid which encloses the drawing. The *width* W of the grid is the width of the rectangle and the *height* H of the grid is the height of the rectangle. The grid size is usually described as $W \times H$. The grid size is sometimes described by the *half-perimeter* $W + H$ or the *area* $W.H$ of the grid. Fig 1.5 illustrates a straight line grid drawing.

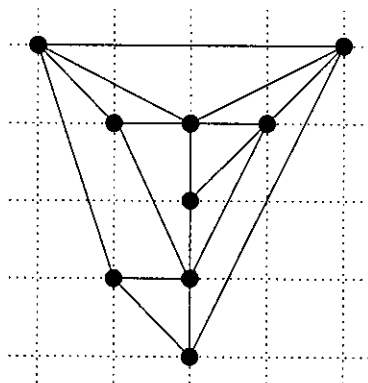


Figure 1.5: A straight line grid drawing.

1.1.4 Straight-Line Grid Drawing

A *straight-line grid drawing* of a planar graph G is a straight-line drawing of G on an integer grid such that each vertex is drawn as a grid point. Wagner [Wag36], Fary [Far48] and Stein [Ste51] independently proved that every planar graph has a straight line representation. Their proofs immediately yield polynomial-time algorithms to find a straight-line drawing of a given plane graph. However, the area of a rectangle enclosing a drawing on an integer grid obtained by these algorithms is not bounded by any polynomial of the number n of vertices in G . In fact, to obtain a drawing of area bounded by a polynomial remained as an open problem for long time. In 1990, de Fraysseix *et al.* [FPP90] and Schnyder [Sch90] showed by two different methods that every planar graph of $n \geq 3$ vertices has a straight-line drawing on an integer grid of size $(2n - 4) \times (n - 2)$ and $(n - 2) \times (n - 2)$, respectively.

1.2 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]. For example, we may be interested in a area efficient drawing of planar graphs. The smaller area of drawing improves the readability as well as reduces the cost of VLSI chip by proper utilization of valuable space in the chip. It is difficult to define a nice drawing precisely, rather we can specify some criteria of a drawing. The criteria of a drawing can be as follows.

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 smallest rectangle that encloses the drawing. If the area becomes too large, then we have to use many pages, or we must decrease resolution, so either way the drawing becomes unreadable. Therefore one major objective is to ensure a small area. For VLSI floorplanning smaller area drawing is preferred because it helps us to avoid wasting of valuable space in the chip.

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 poly line 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.

For most of the above cases, it is hard to achieve optimum. Garey and Johnson showed that minimizing the number of crossings is NP-complete [GJ83]. Kramer and van Leeuwen [KvL84] proved that to test whether a graph can be embedded in a grid of prescribed size is NP-complete, and Formann and Wagner pointed out some corrections to the proof [FW91]. Garg and Tamassia prove the NP-completeness of determining the minimum number of bends for orthogonal drawing [GT95].

1.3 Motivation

In this thesis, we address the area efficient straight-line grid drawings of planar graphs. Smaller area of a drawing increases the readability of the drawing. Compact drawing of a circuit is preferable for VLSI fabrication since a compact drawing helps us to avoid wasting of valuable wafer space. Rather than this practical importance, the motivation of our work primarily comes from the theoretical point of view. Wagner [Wag36], Fary [Far48] and Stein [Ste51] independently proved that every planar graph G has a straight-line drawing. Their proofs immediately yield polynomial-time algorithms to find a straight-line drawing of a given plane graph. However, the area of a rectangle enclosing a drawing on an integer grid obtained by these algorithms is not bounded by any polynomial of the number n of vertices in G . In fact, to obtain a drawing of area bounded by a polynomial remained as

an open problem for long time. In 1990, de Fraysseix *et al.* [FPP90] and Schnyder [Sch90] showed by two different methods that every planar graph of $n \geq 3$ vertices has a straight-line drawing on an integer grid of size $(2n - 4) \times (n - 2)$ and $(n - 2) \times (n - 2)$, respectively. A natural question arises: what is the minimum size of a grid required for a straight-line drawing? de Fraysseix *et al.* showed that, for each $n \geq 3$, there exists a plane graph of n vertices, for example nested triangles, which needs a grid size of at least $\lfloor 2(n - 1)/3 \rfloor \times \lfloor 2(n - 1)/3 \rfloor$ for any grid drawing [CN98, FPP90]. It has been conjectured that every plane graph of n vertices has a grid drawing on a $\lceil 2n/3 \rceil \times \lceil 2n/3 \rceil$ grid, but it is still an open problem. For some restricted classes of graphs, more compact straight-line grid drawings are known. For example, a 4-connected plane graph G having at least four vertices on the outer face has a straight-line grid drawing with area $(\lceil n/2 \rceil - 1) \times (\lfloor n/2 \rfloor)$ [MNN01]. Garg and Rusu showed that an n -node binary tree has a planar straight-line grid drawing with area $O(n)$ [GR02, GR04b]. Although trees admit straight-line grid drawings with linear area, it is generally thought that triangulations may require a grid of quadratic size. Hence finding nontrivial classes of planar graphs of n vertices richer than trees that admit straight-line grid drawings with area $o(n^2)$ is posted as an open problem in [BEGKLM04]. We address this open problem in this thesis.

1.4 Scope of this Thesis

In this section we introduce the topics of graph drawings, partitioning problems and interconnection networks which fall in the scope of this thesis, and mention the previous results and the results obtained by this thesis.

One of the objectives of graph drawings is to obtain a nice drawing of a given graph. We thus assume that if a graph can be drawn without any pair of crossing edges, then we draw accordingly. In this thesis we concentrate our attention to the drawings of planar graphs. Our work is devoted to straight-line drawings of planar graphs. Area of drawing

is one of the important aesthetics criteria. More specifically, our work is devoted to straight-line grid drawings of planar graphs since our works address the area requirement of drawings. In our works, we have identified some classes of planar graphs those admit straight-line grid drawings with sub-quadratic area. Apart from the characterization as well as drawing algorithm for the newly introduced classes of graphs, we also study their k -partitioning problem and the topological properties for finding their other applications.

1.4.1 Straight-line Grid Drawings

The most typical and widely studied drawing style is the straight-line drawing of a planar graph. It is well known that a planar graph of n vertices admits a straight-line grid drawing on a grid of area $O(n^2)$ [FPP90, Sch90]. A lower bound on $\Omega(n^2)$ on the area-requirement for straight-line grid drawings of certain planar graphs are also known [CN98, FPP90]. It has been conjectured that every plane graph of n vertices has a grid drawing on a $\lceil 2n/3 \rceil \times \lceil 2n/3 \rceil$ grid, but it is still an open problem. For some restricted classes of graphs, more compact straight-line grid drawings are known. For example, a 4-connected plane graph G having at least four vertices on the outer face has a straight-line grid drawing with area $(\lceil n/2 \rceil - 1) \times (\lfloor n/2 \rfloor)$ [MNN01]. Garg and Rusu showed that an n -node binary tree has a planar straight-line grid drawing with area $O(n)$ [GR02, GR04b]. Although trees admit straight-line grid drawings with linear area, it is generally thought that triangulations may require a grid of quadratic size. Hence finding nontrivial classes of planar graphs of n vertices richer than trees that admit straight-line grid drawings with area $o(n^2)$ is posted as an open problem in [BEGKLM04]. The problem of finding straight-line grid drawings of outerplanar graphs with $o(n^2)$ area was first posed by Biedl in [Bie02], and Garg and Rusu showed that an outerplanar graph with n vertices and the maximum degree d has a planar straight-line drawing with area $O(dn^{1.48})$ [GR04a]. Di Battista and Frati showed that a “balanced” outerplanar graph of n vertices has a straight-line grid drawing with area $O(n)$ and a general outerplanar graph of n vertices

has a straight-line grid drawing with area $O(n^{1.48})$ [DF06]. Recently Frati showed that a general outerplanar graphs with n vertices admits a straight-line grid drawing with area $O(dn \log n)$, where d is the maximum degree of the graph [Fra07].

In this thesis, we introduce a new class of planar graphs which has a straight-line grid drawing on a grid of area $O(n)$. We give a linear-time algorithm to find such a drawing. We call the class “doughnut graph” since a graph in this class has a doughnut-like embedding. The definition of the class is as follows. Let G be a 5-connected planar graph, let Γ be any planar embedding of G and let p be an integer such that $p \geq 4$. We call G a p -doughnut graph if the following Conditions (d_1) and (d_2) hold:

- (d_1) Γ has two vertex-disjoint faces each of which has exactly p vertices, and all the other faces of Γ has exactly three vertices; and
- (d_2) G has the minimum number of vertices satisfying Condition (d_1).

In general, we call a p -doughnut graph for $p \geq 4$ a *doughnut graph*. Fig. 1.6(d) illustrates a straight-line grid drawing of a doughnut graph G with linear area in Fig. 1.6(a). We describe our results on doughnut graphs in Chapter 3.

One can easily observe that any spanning subgraph of a doughnut graph also admits straight-line grid drawing with linear area. But recognition of a spanning subgraph of a given graph is an NP-complete problem in general [GJ79]. We establish a necessary and sufficient condition for a 4-connected planar graph G to be a spanning subgraph of a doughnut graph. We provide a linear-time algorithm to augment a 4-connected spanning subgraph of a doughnut graph to a doughnut graph. Thus we have identified a subclass of 4-connected planar graphs that admits straight-line grid drawing with linear area. We deal with the spanning subgraphs of doughnut graphs in Chapter 4.

We introduce a subclass of outerplanar graphs which has a straight-line grid drawing on a grid of area $O(n \log n)$. We give a linear-time algorithm to find such a drawing. We call this class “label-constrained outerplanar graphs” since a “vertex labeling” of the

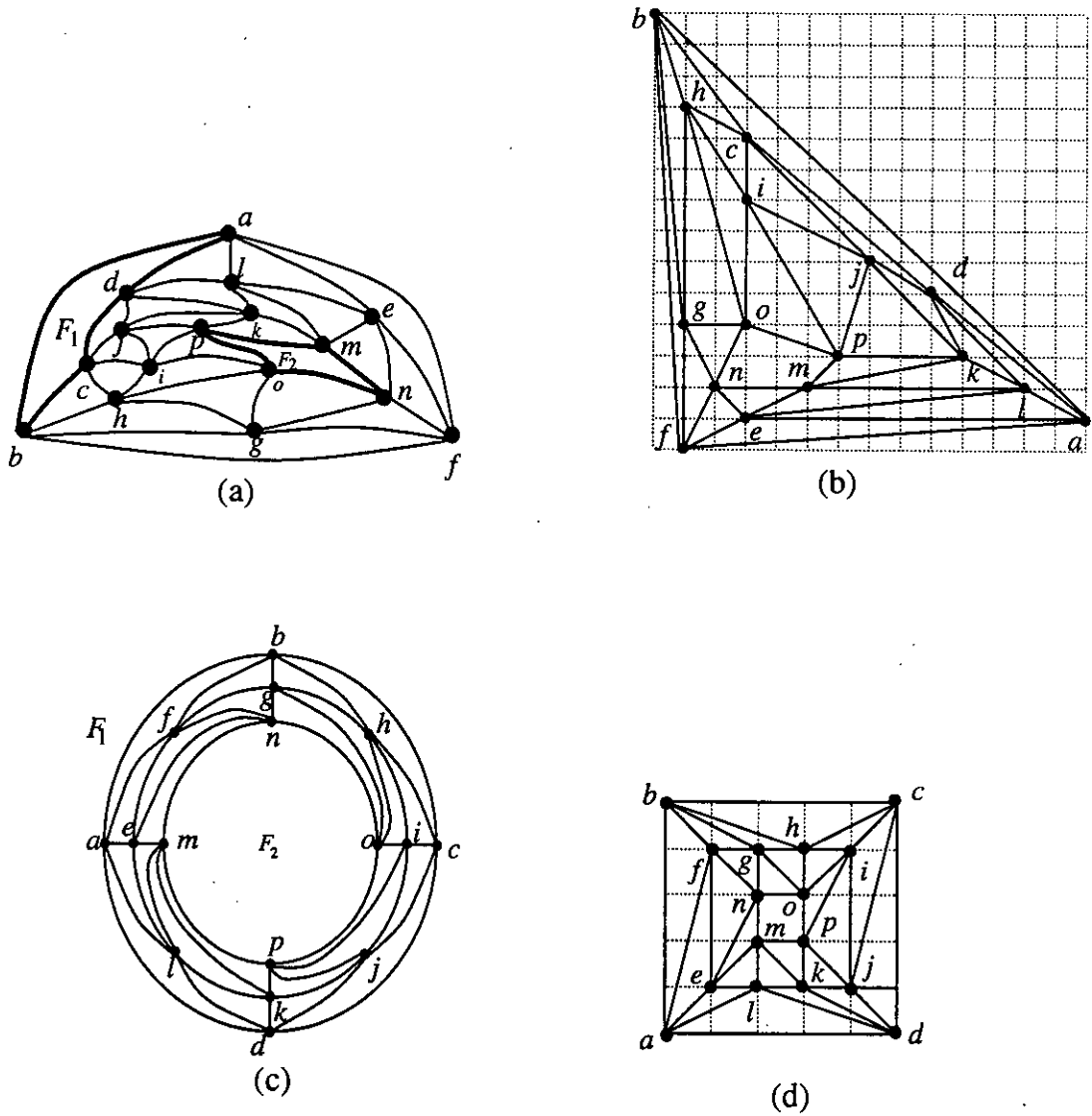


Figure 1.6: (a) A planar graph G , (b) a straight-line grid drawing of G with area $O(n^2)$, (c) a doughnut embedding of G and (d) a straight-line grid drawing of G with area $O(n)$.

8.

dual tree of this graph satisfies certain constraints. The “label-constrained outerplanar graphs” are richer than “balanced” outerplanar graphs. The definition of the class is as follows. Let G be a maximal outerplanar graph and let T be the dual tree of G . We call G a *label-constrained outerplanar graph* if T can be converted to a rooted ordered binary dual tree T_r such that $L_r(T_r)$ is a “flat labeling.” Fig. 1.7(d) illustrates a straight-line grid drawing of a label-constrained outerplanar graph G in Fig. 1.7(a). We also give a

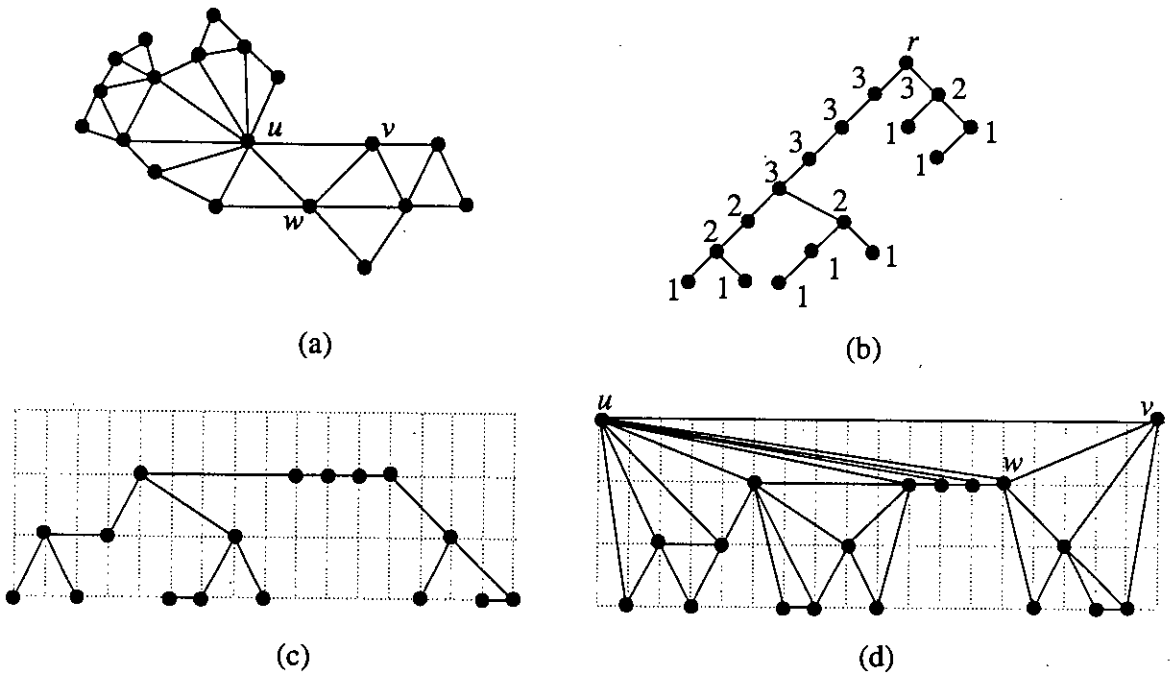


Figure 1.7: (a) A label-constrained outerplanar graph G , (b) the dual rooted ordered tree T_r of G , (c) a straight-line grid drawing of T_r and (d) a straight-line grid drawing of G .

linear-time algorithm for recognition of a “label-constrained outerplanar graph.” Chapter 5 covers the results on label-constrained outerplanar graphs.

1.4.2 Graph Partitioning

Given a graph $G = (V, E)$, k natural numbers n_1, n_2, \dots, n_k such that $\sum_{i=1}^k n_i = |V|$, we wish to find a k -partition V_1, V_2, \dots, V_k of the vertex set V such that $|V_i| = n_i$ and V_i induces a connected subgraph of G for each $i, 1 \leq i \leq k$. A k -partition of a graph G is illustrated in Fig. 1.8 for $k = 5$ where the edges of five connected subgraphs are drawn by solid lines, and the remaining edges of G are drawn by dotted lines. Let $B = u_1, u_2, \dots, u_m$

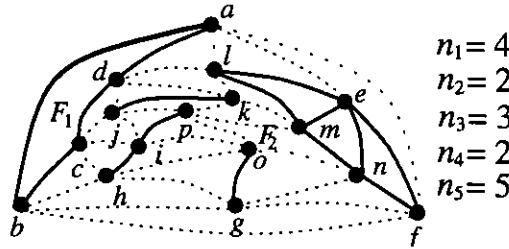


Figure 1.8: A 5-partition of a 5-connected planar graph G .

be a sequence of distinct vertices of G with $m \leq k$. A k -partition of G with basis B is a k -partition with the additional restriction that $u_i \in V_i$, for $1 \leq i \leq m$. A k -partition of a graph G with basis m is illustrated in Fig. 6.8 for $k = 5$ and $m = 5$.

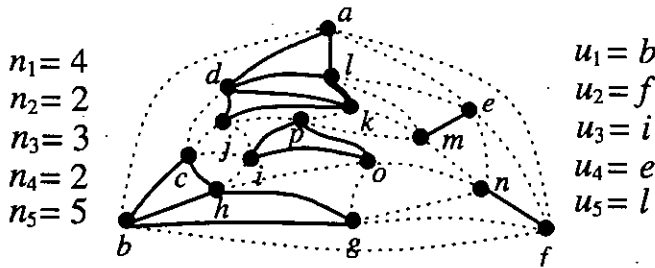


Figure 1.9: A 5-partition of a 5-connected planar graph G with basis 5.

The problem of finding a k -partition of a given graph often appears in the load distribution among different power plants and the fault-tolerant routing of communication networks [NRN97, NN01]. The problem is NP-hard in general even k is limited to 2 [DF85], and hence it is very unlikely that there is a polynomial-time algorithm to solve the problem. Although not every graph has a k -partition, Györi and Lovász independently proved

that every k -connected graph has a k -partition [Gyr78, Lov77]. However, their proofs do not yield any polynomial-time algorithm for finding a k -partition of a k -connected graph. A linear-time algorithm is known for 4-partitioning of a 4-connected plane graph if the four basis vertices are all on the boundary of one face [NRN97]. A linear-time algorithm is also known for 5-partitioning of a 5-connected internally triangulated plane graph if the five basis vertices are all on the boundary of one face [NN01]

We give a linear-time algorithm for k -partitioning of doughnut graphs in Chapter 6.

1.4.3 Interconnection Networks

An interconnection network is an important integral part of any parallel processing or distributed systems. Performance of the distributed systems is significantly depends on the choice of the network topology. An *interconnection network* usually modeled as an undirected graph G where each vertex corresponds to a processor and an edge corresponds to the communication channel between the two processors corresponding to the end vertices of the edge. The popular interconnection networks are hypercube, mesh, butterfly etc. We propose the class of doughnut graphs as a promising class of interconnection networks due to their regularity, smaller diameter, maximal fault tolerance and recursive structure. We also propose a simple routing algorithm in doughnut graphs. The topological properties as well as an efficient routing scheme in the doughnut graphs are covered in Chapter 7.

1.5 Summary

In this thesis we introduce some new classes of planar graphs which have beautiful area efficient drawing properties. We have studied the k -partitioning problem for newly introduced classes of graphs. We also study the topological properties of these classes of planar graphs for finding suitable applications. In this section we summarize our main

results. Our main results can be divided into three parts.

The first part of our result is to identify some new classes of planar graphs those admit straight line grid drawings with sub-quadratic area. We characterize the new classes of planar graphs and provide linear-time algorithms for finding such drawings. Our new results together with the known ones are listed in Table 1.1.

Table 1.1: Classes of planar graphs and their area requirement.

| Classes of planar graphs | Area | Reference |
|--|----------------|-----------|
| Binary Tree | $O(n)$ | [GR04b] |
| Outerplanar graphs with the maximum degree d | $O(dn^{1.48})$ | [GR04a] |
| Outerplanar graphs | $O(n^{1.48})$ | [DF06] |
| “Balanced” outerplanar graphs | $O(n)$ | [DF06] |
| Outerplanar graphs with the maximum degree d | $O(dn \log n)$ | [Fra07] |
| Doughnut graphs | $O(n)$ | Ours |
| A subclass of 4-connected planar graphs | $O(n)$ | Ours |
| Label-Constrained outerplanar graphs | $O(n \log n)$ | Ours |

The second part of the result is about the study of k -partitioning problem for our newly introduced classes of planar graphs. We have found some interesting results for doughnut graphs. We give a linear-time algorithm for k -partitioning of a doughnut graph. This new result together with the known ones are listed in Table 1.2.

The third part of the result is about application of doughnut graph as an interconnection network. We have identified a set of topological properties of a doughnut graph those can be exploited to design an efficient interconnection network. The class of doughnut graphs is a maximal fault tolerant graphs since the members of this class is degree regular. The degree of a vertex of a doughnut graph does not change with the increase the size of the graph. This is the property which makes an interconnection network scalable. This property is also important from the view point of VLSI implementation. We also give an

Table 1.2: Results on graph partitioning

| Classes of planar graphs | Partitions | Time | Reference |
|------------------------------|-------------------|-------------------|-----------|
| General graphs | k -partitioning | NP-hard | [DF85] |
| k -connected graphs | k -partitioning | Existential proof | [Gyr78] |
| Biconnected graphs | Bipartitioning | $O(n)$ | [STN90] |
| Triconnected planar graphs | tripartitioning | $O(n)$ | [JSN94] |
| Four-connected planar graphs | Four-partitioning | $O(n)$ | [NRN97] |
| Five-connected plane graphs | Five-partitioning | $O(n)$ | [NN01] |
| Doughnut graphs | k -partitioning | $O(n)$ | Ours |

efficient routing algorithm for a doughnut graph through exploiting its simple structure. A topological comparison of different Cayley graphs, which are used as interconnection networks, with doughnut graphs is given in the Table 1.3.

The rest of the thesis is organized as follows. In Chapter 2, we define some basic terms of graph theory and algorithm theory. Chapter 3 provides the results on doughnut graphs that admit straight-line grid drawing with linear area. Chapter 4 deals with the spanning subgraphs of doughnut graphs. In Chapter 5, we introduce a subclass of outerplanar graphs that admit straight-line grid drawing with $O(n \log n)$ area. We give a linear-time algorithm to find such a drawing. We present a linear-time algorithm for k -partitioning of doughnut graphs in Chapter 6. Chapter 7 deals with the application of doughnut graphs. In this chapter, we present the topological properties of doughnut graphs and an efficient routing scheme in doughnut graphs. We propose doughnut graphs as a promising interconnection networks for their topological properties and efficient routing scheme. Chapter 8 concludes the thesis.

Table 1.3: Topological comparison of various Cayley graphs with doughnut graphs.

| Topology | number of nodes | diameter | degree | connectivity | maximal fault tolerance |
|---|-----------------|----------------------------|--------|--------------|-------------------------|
| n -cycle | n | $\lfloor n/2 \rfloor$ | 2 | 2 | yes |
| Cube-connected -cycle [PV81] | $d2^d$ | $\lfloor 5d/2 \rfloor - 2$ | 3 | 3 | yes |
| Wrapped around butterfly graph [Lei92] | $d2^d$ | $\lfloor 3d/2 \rfloor$ | 4 | 4 | yes |
| d -Dimensional hypercube [BA84] | 2^d | d | d | d | yes |
| k -degree Cayley graph [HH06] | $d(k-1)^d$ | $\lfloor 5d/2 \rfloor - 2$ | k | k | yes |
| p -doughnut graphs [Ours] | $4p$ | $\lfloor p/2 \rfloor + 2$ | 5 | 5 | yes |

Chapter 2

Preliminaries

In this chapter we define some basic terms of graph theory and algorithm theory. Definitions which are not included in this chapter will be introduced as they are needed. We start, in Section 2.1, by giving some definitions of standard graph-theoretical terms used throughout the remainder of this thesis. We devote Section 2.2 to define terms related to planar graphs. Finally in Section 2.3 we introduce the notion of time complexity

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. For readers interested in graph theory, we refer to [Wes01].

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. Fig. 2.1 depicts a graph G where $V(G) = \{v_1, v_2, v_3, v_4, v_5, v_6\}$, and $E(G) = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8\}$. Throughout this thesis the

number of vertices of G is denoted by n , i.e., $n = |V|$, and the number of edges of G is denoted by m , i.e., $m = |E|$. Thus for the graph G in Fig. 2.1 $n = 6$ and $m = 8$.

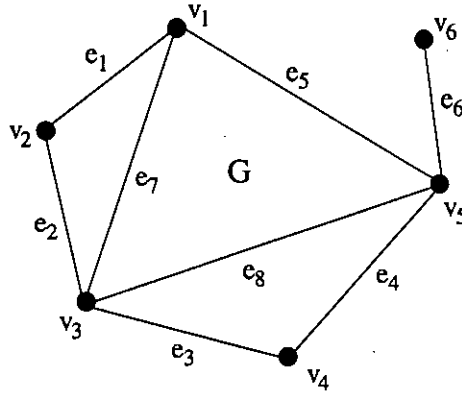


Figure 2.1: A graph with six vertices and eight edges.

If more than one edges of a graph G join the same pair of vertices, then the edges are called *multiple edges*. 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 an edge between two vertices u and v of G by (u, v) . If $(u, v) \in E$, then two vertices u and v of graph G are said to be adjacent; edge (u, v) is then said to be incident to vertices u and v ; u is a neighbor of v . The *degree* of a vertex v in G is the number of edges incident to v , which we denote by $d(v)$. In the graph in Fig. 2.1, vertices v_3 and v_5 are adjacent, and $d(v_3) = 4$, since 4 edges e_2, e_7, e_8 and e_3 are incident to v_3 . The *maximum degree* of G is denoted by $\Delta(G) = \max_{v \in V(G)} \{d(v)\}$, and the *minimum degree* of G is denoted by $\delta(G) = \min_{v \in V(G)} \{d(v)\}$.

2.1.2 Subgraphs

A *subgraph* of a graph $G = (V, E)$ is a graph $G' = (V', E')$ such that $V' \subseteq V$ and $E' \subseteq E$; We then write $G' \subseteq G$. If G' contains all the edges of G that join two vertices in V' , then G' is said to be the *subgraph induced by V'* , and is denoted by $G[V']$. If G' contains

all the vertices of G , i.e., $V'(G') = V(G)$ and $E'(G') \subseteq E(G)$, G' is said to be *spanning subgraph* of G . A *spanning tree* is a spanning subgraph that is tree. Fig. 2.2(b) illustrates a subgraph induced by $V' = \{v_1, v_3, v_4, v_5\}$ of G in Fig. 2.2(a), and Fig. 2.2(c) illustrates a spanning subgraph of G in Fig. 2.2(a).

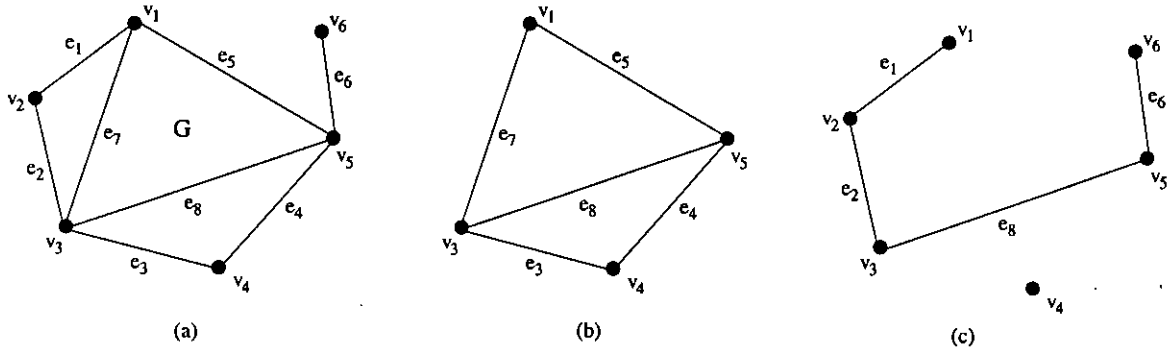


Figure 2.2: (a) A graph G , (b) an induced subgraph of G and (c) a spanning subgraph of G

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

2.1.3 Connectivity

A graph G is *connected* if for any two distinct vertices u and v there is a path between u and v in G . A graph which is not connected is called a *disconnected* graph. A (*connected*) *component* of a graph is a maximal connected subgraph of the graph. The graph in

Fig. 2.3(a) is a connected graph since there is a path for every pair of distinct vertices of the graph. On the other hand the graph in Fig. 2.3(b) is a disconnected graph because there exists no path between v_1 and v_9 . The graph in Fig. 2.3(b) has two connected components G_1 and G_2 indicated by dotted lines.

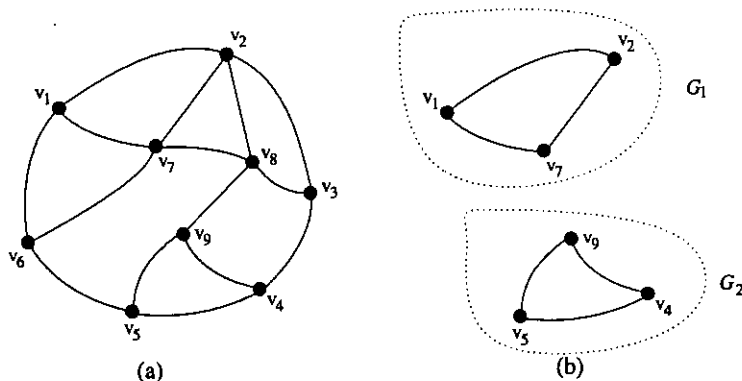


Figure 2.3: (a) A connected graph and (b) a disconnected graph.

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

2.1.4 Paths and Cycles

A $v_0 - v_l$ *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_1, e_2, \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.

2.1.5 Trees

A *tree* is a simple connected graph which contains no cycle. The vertices in a tree are usually called *nodes*. A *rooted tree* is a tree in which one of the nodes is distinguished from others. The distinguished node is called the *root* of the tree. The root of a tree is generally drawn at the top. The node v_1 in Fig. 2.4 is the root. If a rooted tree is regarded as a directed graph in which each edge is directed from top to bottom, then every node u other than root is connected by an edge from some other node p is called the *parent* of u . We also call u a *child* of p . We draw the parent of a node above that node. For example, in Fig. 2.4, v_1 is the parent of v_2 , v_3 and v_4 , while v_2 is the parent of v_5 , and v_4 is the parent of v_6 and v_7 ; v_2 , v_3 and v_4 are children of v_1 , while v_6 and v_7 are the children of v_4 . A *leaf* is a node of a tree that has no children. An *internal node* is a node that has one or more children. Thus every node of a tree is either a leaf or an internal node. Fig. 2.4 illustrates a tree in which v_5 , v_3 , v_6 and v_7 are the leaf nodes, and the nodes v_1 , v_2 , v_4 are the internal nodes.

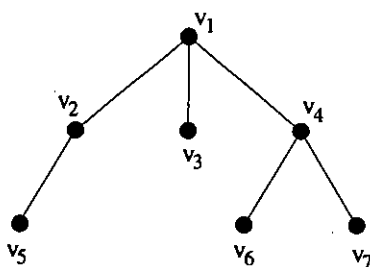


Figure 2.4: A tree.

The parent-child relationship can be extended naturally to ancestors and descendants. Suppose that u_1, u_2, \dots, u_l is a sequence of nodes in a tree such that u_1 is the parent of u_2 , which is a parent of u_3 , and so on. Then node u_1 is called an *ancestor* of u_l and node u_l a *descendant* of u_1 . The root is an ancestor of every node in a tree and every node is

a descendant of the root. In Fig. 2.4, all the nodes except v_1 is the descendant of v_1 , and v_1 is the ancestor of all nodes.

The *height* of a node u in a tree is the length of a longest path from u to a leaf. The *height* of a tree is the height of the root. The *depth* of a node u in a tree is the length of a path from the root to u . The *level* of a node u in a tree is the height of the tree minus the depth of u . In Fig. 2.4, for example, node v_2 is of height 1, depth 1 and level 1. The tree in Fig. 2.4 has height 2.

An *ordered rooted tree* is a rooted tree where the children of each internal vertex are ordered from left to right. If v is a node in a tree T , then the *subtree* with v as its root is the subgraph of T consisting of v and its descendants and all edges incident to these descendants. By *tree traversal*, we mean a method for systematically visiting every node of an ordered rooted tree. Three most commonly used such methods are - preorder traversal, inorder traversal and postorder traversal. We define the *preorder* traversal of a ordered rooted tree T as follows. If T consists only of r , then r is the preorder traversal of T . Otherwise, suppose that T_1, T_2, \dots, T_n are the subtrees rooted at the children of r from left to right. Then the preorder traversal begins by visiting r . It continues by traversing T_1 in preorder, then T_2 in preorder, and so on, until T_n is traversed in preorder. In case of *inorder* traversal, if T consists only of r , then r is the inorder traversal of T . Otherwise, the inorder traversal of T begins by traversing T_1 in inorder, then visiting r . It continues by traversing T_2 inorder, then T_3 in inorder, ..., and finally T_n in inorder. In case of *postorder* traversal, if T consists only of r , then r is the postorder traversal of T . Otherwise, the postorder traversal of T begins by traversing T_1 in postorder, then T_2 in postorder, ..., then T_n in postorder, and ends by visiting r . For tree T in Fig. 2.4, preorder traversal of T gives $v_1, v_2, v_5, v_3, v_4, v_6, v_7$; inorder traversal of T gives $v_5, v_2, v_1, v_3, v_6, v_4, v_7$; and postorder traversal of T gives $v_5, v_2, v_3, v_6, v_7, v_4, v_1$.

2.2 Planar Graphs

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

2.2.1 Planar Graphs and Plane 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. Fig. 2.5 illustrates three different planar embeddings of the same planar graph.

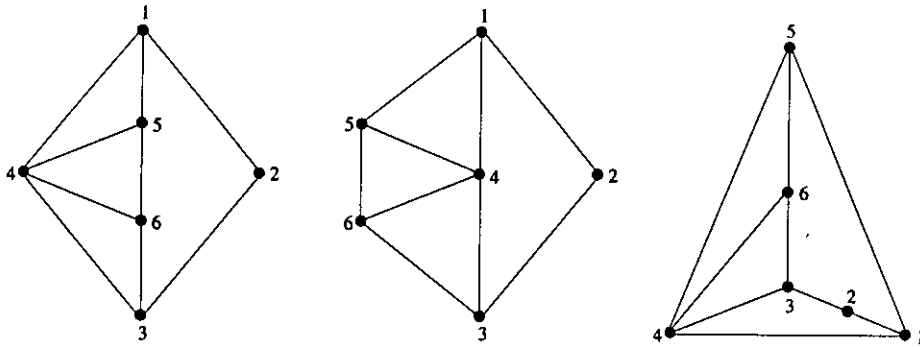


Figure 2.5: Three planar embeddings of the same graph.

A *plane graph* G is a planar graph with a fixed embedding in the plane as shown in Fig. 2.6. A plane graph divides the plane into some connected regions called *faces*. The shaded region of the graph G in Fig. 2.6 is an example of a face. A bounded region is called an *inner face* and the 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 Fig. 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 .

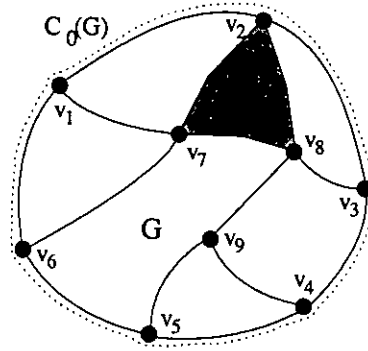


Figure 2.6: A plane graph G .

2.2.2 Euler's Formula

There is a simple formula relating the number of vertices, edges and faces in a connected plane graph. It is known as Euler's formula because Euler established it in 1750 for those plane graph defined by the vertices and edges of polyhedra. We now present Euler's formula as the following lemma.

Lemma 2.2.1 *Let G be a connected plane graph, let n , m , and f denote respectively the number of vertices, edges and faces of G . Then $n - f + m = 2$*

2.2.3 Outerplanar Graphs

A plane graph G is an *outerplanar* (*1-outerplanar*) graph if its all vertices of G lie on the outer face. A plane graph G is *k -outerplanar* ($k > 1$) if the graph G' obtained by removing all the vertices of the outer face of G is $(k - 1)$ -outerplanar graph. A graph is *k -outerplanar* if it admits a k -outerplanar embedding. A planar graph G has *outerplanarity* k ($k > 0$) if it is k -outerplanar and it is not j -outerplanar for $0 < j < k$. A *maximal*

outerplanar graph is an outerplanar graph in which no edge can be added without losing outerplanarity. (Note that all the maximal outerplanar graphs are biconnected.) Clearly, each inner face of a maximal outerplanar graph has three edges. It is easy to see that any outerplanar graph can be augmented in linear time to a maximal outerplanar graph by adding only linear number of extra edges.

2.2.4 Dual Graphs

For a plane graph G , we often construct another graph G^* called the (*geometric dual*) of G as follows. A vertex v_i^* is placed in each face of F_i of G ; these are the vertices of G^* . Corresponding to each edge e of G we draw an edge e^* which crosses e (but no other edge of G) and joins the vertices v_i^* which lie in the faces F_i adjoining e ; these are the edges of G^* . The construction is illustrated in Fig 2.7(a); the vertices v_i^* are represented by small white circle, and the edges e^* of G^* by dotted lines. G^* is not necessarily a simple graph even if G is simple. Clearly the dual of G^* of a plane graph G is also plane. One can

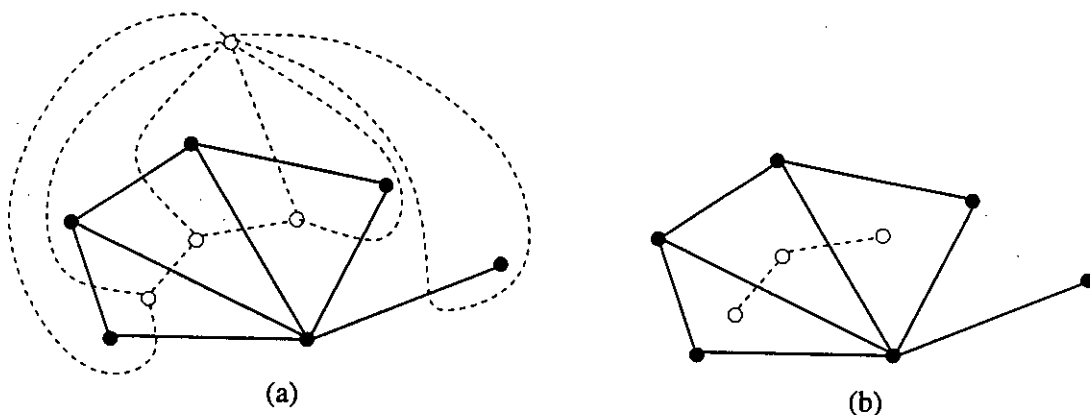


Figure 2.7: (a) A plane graph G and its dual graph G^* and (b) a plane graph G and its weak dual graph G' .

easily observe the following lemma.

Lemma 2.2.2 *Let G be a connected plane graph with n vertices m edges and f faces, and*

let the dual G^* have n^* vertices, m^* edges and f^* faces; then $n^* = f$, $m^* = m$, and $f^* = n$.

Clearly the dual of the dual of the plane graph G is the original graph G . However a planar graph may give rise to two or more geometric duals since the plane embedding is not necessarily unique. If we ignore the outerface, we call the graph G' as a *weak dual* of G . Fig. 2.8(b) illustrates the weak dual graph G' of a graph G where the vertices of G' are represented by small white circle and edges of G' are represented by dotted lines. Clearly the weak dual of an outerplanar graph is a tree as illustrated in Fig. 2.8. Hence we call the weak dual of an outerplanar graph is a *dual tree*.

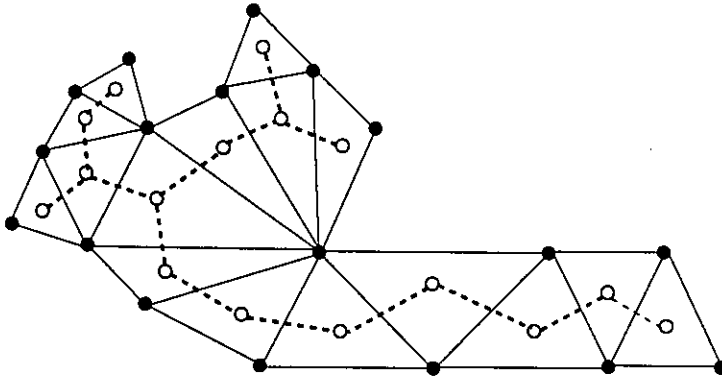


Figure 2.8: An outerplanar graph G and its weak dual graph G' .

2.3 Algorithms and Complexity

In this section we briefly introduce some terminologies related to complexity of algorithms. For interested readers, we refer the book of Garey and Johnson [GJ79]. The most widely accepted complexity measure for an algorithm is the *running time* which is expressed by the number of operations it performs before producing the final answer. The number of operations required by an algorithm is not the same for all problem instances. Thus, we consider all inputs of a given size together, and we define the complexity of the algorithm

for that input size to be the worst case behavior of the algorithm on any of these inputs. Then the running time is a function of size n of the input.

2.3.1 The Notations $O(n)$, $\Omega(n)$, $o(n)$

In analyzing the complexity of an algorithm, we are often interested only in the “asymptotic behavior,” that is, the behavior of the algorithm when applied to very large inputs. To deal with such a property of functions we shall use the following notations for asymptotic running time. Let $f(n)$ and $g(n)$ are the functions from the positive integers to the positive reals, then we write (i) $f(n) = O(g(n))$ if there exists positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$, (ii) $f(n) = \Omega(g(n))$ if there exists positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$, and (iii) $f(n) = o(n)$ for any positive constant $c > 0$, there exists a $n_0 > 0$ such that $0 \leq f(n) < cg(n)$ for all $n \geq n_0$. Thus the running time of an algorithm may be bounded from above by phrasing like “takes time $O(n^2)$ or $\Omega(n^2)$ or $o(n^2)$.”

2.3.2 Polynomial Algorithms

An algorithm is said to be *polynomially bounded* (or simply *polynomial*) if its complexity is bounded by a polynomial of the size of a problem instance. Examples of such complexities are $O(n)$, $O(n \log n)$, $O(n^{100})$, etc. The remaining algorithms are usually referred as *exponential* or *nonpolynomial*. Example of such complexity are $O(2^n)$, $O(n!)$, etc.

When the running time of an algorithm is bounded by $O(n)$, we call it a *linear-time* algorithm or simply a *linear* algorithm.

2.3.3 NP-Complete

There are number of interesting computational problems for which it has not been proved whether there is a polynomial time algorithm or not. Most of them are “*NP-complete*,”

which we will briefly explain in this section.

The state of algorithms consists of the current values of all the variables and the location of the current instruction to be executed. A *deterministic algorithm* is one for which each state, upon execution of the instruction, uniquely determines at most one of the following state (next state). All computers, which exist now, run deterministically. A problem Q is in the *class of P* if there exists a deterministic polynomial-time algorithm which solves Q .

In contrast, a *nondeterministic algorithm* is one for which a state may determine many next states simultaneously. We may regard a nondeterministic algorithm as having the capability of branching off into many copies of itself, one for the each next state. Thus, while a deterministic algorithm must explore a set of alternatives one at a time, a nondeterministic algorithm examines all alternatives at the same time. A problem Q is in the class NP if there exists a nondeterministic polynomial-time algorithm which solves Q . Clearly $P \subset NP$.

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

- (a) No NP -complete problem can be solved by any known polynomial algorithm.
- (b) If there is a polynomial algorithm for any NP -complete problem, then there are polynomial algorithm for all NP -complete problems.

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

2.4 A Graph Traversal Algorithm

When designing algorithms on graphs, we often need a method for exploring the vertices and edges of a graph. In this section we introduce such a method named *depth first search* (*DFS*). In DFS each edge is traversed exactly once in the forward and reverse directions and each vertex is visited. Thus DFS runs in linear time. We now describe the method.

Consider visiting the vertices of a graph in the following way. We select and visit a starting vertex v . Then we select any edge (v, w) incident on v and visit w . In general, suppose x is the most recent visited vertex. The search is continued by selecting some unexplored edge (x, y) incident on x . If y has been previously visited, we find another new edge incident on x . If y has not been previously visited, then we visit y and begin a new search starting at y . After completing the search through all paths beginning at y , the search returns to x , the vertex from which y was first reached. The process of selecting unexplored edge incident to x is continued until the list of these edges is exhausted. This method is called *depth-first search* since we continue searching in the deeper direction as long as possible.

Chapter 3

Doughnut Graphs

In this chapter, we introduce a new class of planar graphs, which we call *doughnut graphs*, has a straight-line grid drawing on a grid of area $O(n)$. It is well known that a planar graph of n vertices admits a straight-line grid drawing on a grid of area $O(n^2)$. A lower bound of $\Omega(n^2)$ on the area-requirement for straight-line grid drawings of certain planar graphs are also known. It has been conjectured that every plane graph of n vertices has a grid drawing on a $\lceil 2n/3 \rceil \times \lceil 2n/3 \rceil$ grid, but it is still an open problem. For some restricted classes of graphs, more compact straight-line grid drawings are known. For example, a 4-connected plane graph G having at least four vertices on the outer face has a straight-line grid drawing with area $(\lceil n/2 \rceil - 1) \times (\lfloor n/2 \rfloor)$ [MNN01]. Garg and Rusu showed that an n -node binary tree has a planar straight-line grid drawing with area $O(n)$ [GR02, GR04b]. Although trees admit straight-line grid drawings with linear area, it is generally thought that triangulations may require a grid of quadratic size. Hence finding nontrivial classes of planar graphs of n vertices richer than trees that admit straight-line grid drawings with area $o(n^2)$ is posted as an open problem in [BEGKLM04]. Garg and Rusu showed that an outerplanar graph with n vertices and the maximum degree d has a planar straight-line drawing with area $O(dn^{1.48})$ [GR04a]. Di Battista and Frati improved the result by showing that a “balanced” outerplanar graph of n vertices has a

straight-line grid drawing with area $O(n)$ and a general outerplanar graph of n vertices has a straight-line grid drawing with area $O(n^{1.48})$ [DF06]. Recently Frati showed that any outerplanar graph with the maximum degree d has a straight-line grid drawing with $O(dn \log n)$ area [Fra07]. In this context, we introduce this class of graphs. We also give a linear-time algorithm to find such a drawing. Our new class of planar graphs is a subclass of 5-connected planar graphs, and we call the class “doughnut graphs” since a graph in this class has a doughnut-like embedding as illustrated in Fig. 3.1(c). In an embedding of a doughnut graph of n vertices there are two vertex-disjoint faces each having exactly $n/4$ vertices and each of all the other faces has exactly three vertices. Fig. 3.1(a) illustrates a doughnut graph of 16 vertices where each of the two faces F_1 and F_2 contains four vertices and each of all other faces contains exactly three vertices. Fig. 3.1(c) illustrates a doughnut-like embedding of G where F_1 is embedded as the outer face and F_2 is embedded as an inner face.

This chapter is organized as follows. In Section 3.1, we give some definitions. Section 3.2 provides some properties of the class of doughnut graphs. Section 3.3 deals with straight-line grid drawings of doughnut graphs. Finally Section 3.4 concludes the chapter. Our results presented in this chapter are published in [KR07].

3.1 Preliminaries

In this section we give some definitions.

Let $G = (V, E)$ be a connected simple plane graph with vertex set V and edge set E . For a face F in G we denote by $V(F)$ the set of vertices of G on the boundary of face F . We call two faces F_1 and F_2 are *vertex-disjoint* if $V(F_1) \cap V(F_2) = \emptyset$. Let F be a face in a plane graph G with $n \geq 3$. If the boundary of F has exactly three edges then we call F a triangulated face. One can divide a face F of p ($p \geq 3$) vertices into $p - 2$ triangulated faces by adding $p - 3$ extra edges. The operation above is called *triangulating a face*. If

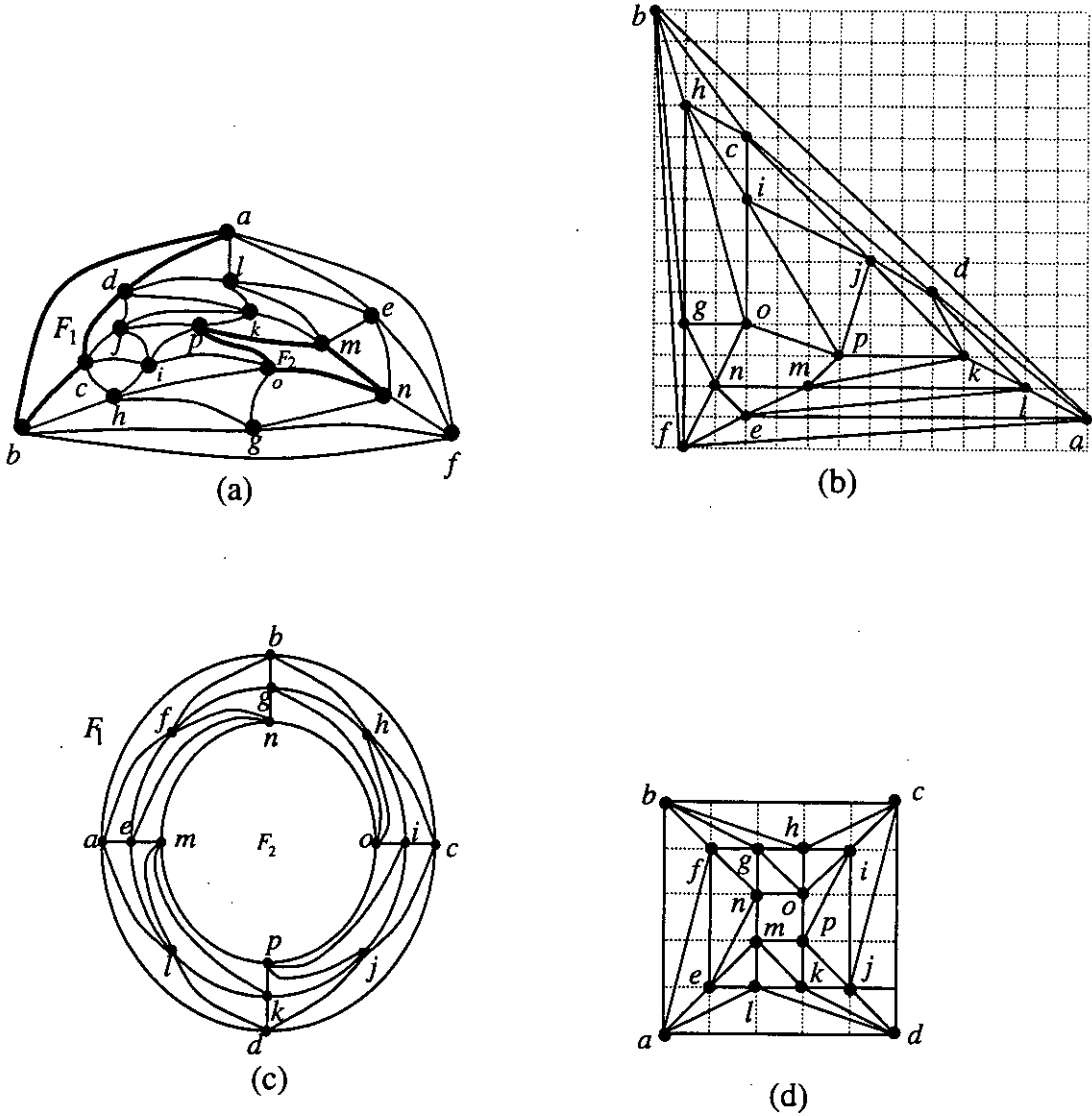


Figure 3.1: (a) A planar graph G , (b) a straight-line grid drawing of G with area $O(n^2)$, (c) a doughnut embedding of G and (d) a straight-line grid drawing of G with area $O(n)$.

every face of a graph is triangulated, then the graph is called a *triangulated plane graph*. We can obtain a triangulated plane graph G' from a non-triangulated plane graph by triangulating all of its faces. Fig. 3.2(b) illustrates a triangulated plane graph obtained from the plane graph in Fig. 3.2(a).

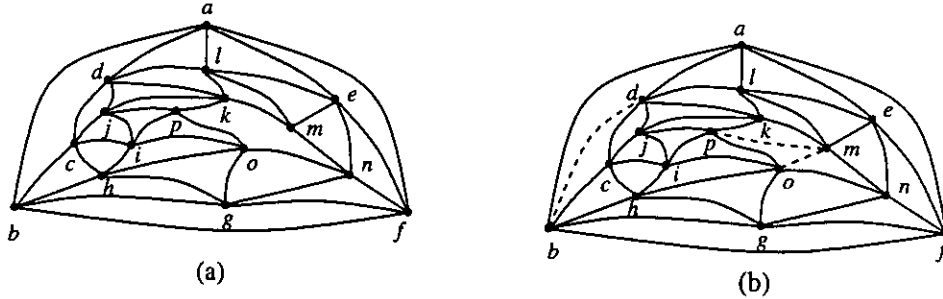


Figure 3.2: (a) Input graph G and (b) a triangulated plane graph G' of graph G where extra edges are drawn by dotted lines

A *maximal planar* graph is one to which no edge can be added without losing planarity. Thus in any embedding of a maximal planar graph G with $n \geq 3$, the boundary of every face of G is a triangle, and hence an embedding of a maximal planar graph is often called a triangulated plane graph. It can be derived from the Euler's formula for planar graphs that if G is a maximal planar graph with n vertices and m edges then $m = 3n - 6$, for more details see [NR04].

For any 3-connected planar graph the following fact holds [Whi33].

Fact 3.1.1 *Let G be a 3-connected planar graph and let Γ and Γ' be any two planar embeddings of G . Then any facial cycle of Γ is a facial cycle of Γ' and vice versa.*

Let G be a 5-connected planar graph, let Γ be any planar embedding of G and let p be an integer such that $p \geq 4$. We call G a *p-doughnut* graph if the following Conditions (d_1) and (d_2) hold:

- (d_1) Γ has two vertex-disjoint faces each of which has exactly p vertices, and all the other faces of Γ has exactly three vertices; and

(d_2) G has the minimum number of vertices satisfying Condition (d_1).

In general, we call a p -doughnut graph for $p \geq 4$ a *doughnut graph*. Since a doughnut graph is a 5-connected planar graph, Fact 3.1.1 implies that decomposition of a doughnut graph into its facial cycles is unique. Throughout the thesis we often mention faces of a doughnut graph G without mentioning its planar embedding where description of the faces are valid for any planar embedding of G .

3.2 Properties of Doughnut Graphs

In this section we will show some properties of a p -doughnut graph. We have the following lemma on the number of vertices of a graph satisfying Condition (d_1).

Lemma 3.2.1 *Let G be a 5-connected planar graph, let Γ be any planar embedding of G , and let p be an integer such that $p \geq 4$. Assume that Γ has two vertex-disjoint faces each of which has exactly p vertices, and all the other faces of Γ has exactly three vertices. Then G has at least $4p$ vertices.*

Proof. Let F_1 and F_2 be the two faces of Γ each of which contains exactly p vertices. Let x be the number of vertices in G which are neither on F_1 nor on F_2 . Then G has $x + 2p$ vertices.

We now calculate the number of edges in graph G . Faces F_1 and F_2 may not be triangulated. If we triangulate F_1 and F_2 of Γ then the resulting graph G' is a maximal planar graph. Using Euler formula, G' has exactly $3(x + 2p) - 6 = 3x + 6p - 6$ edges. To triangulate each of F_1 and F_2 , we need to add $p - 3$ edges and hence the number of edges in G is exactly

$$(3x + 6p - 6) - 2(p - 3) = 3x + 4p \tag{3.1}$$

Since G is 5-connected, using the degree-sum formula we get $2(3x + 4p) \geq 5(x + 2p)$. This relation implies

$$x \geq 2p \quad (3.2)$$

Therefore G has at least $4p$ vertices. \square

Lemma 3.2.1 implies that a p -doughnut graph has $4p$ or more vertices. We now show that $4p$ vertices are sufficient to construct a p -doughnut graph as in the following lemma.

Lemma 3.2.2 *For an integer p , ($p \geq 4$), one can construct a p -doughnut graph G with $4p$ vertices.*

To prove Lemma 3.2.2 we first construct a planar embedding Γ of G with $4p$ vertices by the construction **Construct-Doughnut** given below and then show that G is a p -doughnut graph.

Construct-Doughnut. Let C_1, C_2, C_3 be three cycles such that C_1 contains p vertices, C_2 contains $2p$ vertices and C_3 contains p vertices. Let x_1, x_2, \dots, x_p be the vertices on C_1 , y_1, y_2, \dots, y_p be the vertices on C_3 , and z_1, z_2, \dots, z_{2p} be the vertices on C_2 . Let R_1, R_2 and R_3 be three concentric circles on a plane with radius r_1, r_2 and r_3 , respectively, such that $r_1 > r_2 > r_3$. We embed C_1, C_2 and C_3 on R_1, R_2 and R_3 respectively, as follows. We put the vertices x_1, x_2, \dots, x_p of C_1 on R_1 in clockwise order such that x_1 is put on the leftmost position among the vertices x_1, x_2, \dots, x_p . Similarly, we put vertices z_1, z_2, \dots, z_{2p} of C_2 on R_2 and y_1, y_2, \dots, y_p of C_3 on R_3 . We now add edges between the vertices on C_1 and C_2 , and between the vertices on C_2 and C_3 . We have two cases to consider.

Case 1: k is even in z_k .

In this case, we add two edges $(z_k, x_{k/2}), (z_k, x_i)$ between C_2 and C_1 , and one edge (z_k, y_i) between C_2 and C_3 where $i = 1$ if $k = 2p$, $i = k/2 + 1$ otherwise.

Case 2: k is odd in z_k .

In this case we add two edges $(z_k, y_{\lceil k/2 \rceil}), (z_k, y_i)$ between C_2 and C_3 , and one edge $(z_k, x_{\lceil k/2 \rceil})$ between C_1 and C_2 where $i = 1$ if $k = 2p - 1$, $i = \lceil k/2 \rceil + 1$ otherwise.

We thus constructed a planar embedding Γ of G . Fig. 3.3 illustrates the construction above for the case of $p = 4$. □

We now have the following lemma.

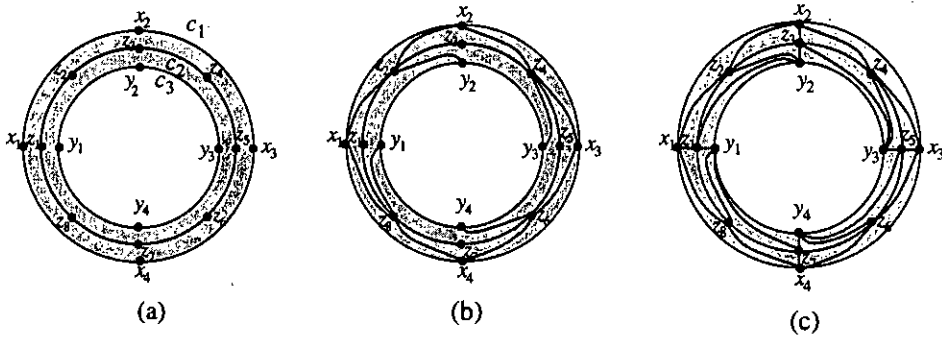


Figure 3.3: Illustration for the construction of a planar embedding Γ of a p -doughnut graph G for $p = 4$; (a) embedding of the three cycles C_1 , C_2 and C_3 on three concentric circles, (b) addition of edges for the case where k is even in z_k and (c) Γ .

Lemma 3.2.3 *Let Γ be the plane graph of $4p$ vertices obtained by the construction Construct-Doughnut. Then Γ has exactly two vertex-disjoint faces F_1 and F_2 each of which has exactly p vertices, and the rest of the faces are triangulated.*

Proof. The construction of Γ implies that Cycle C_1 is the boundary of the outer face F_1 of Γ and Cycle C_3 is the boundary of an inner face F_2 . Each of F_1 and F_2 has exactly p vertices. Clearly the two faces F_1 and F_2 of Γ are vertex-disjoint. Thus Γ has exactly two vertex-disjoint faces F_1 and F_2 each of which has exactly p vertices.

We now show that the rest of the faces of Γ are triangulated. The rest of the faces can be divided into two groups; (i) faces having vertices on both the cycles C_1 and C_2 and (ii) faces having the vertices on both the cycles C_2 and C_3 .

We only prove that each face in Group (i) is triangulated, since similarly we can prove that each face in Group (ii) is triangulated.

From our construction, each vertex z_i with even i has exactly two neighbors on C_1 and the two neighbors of z_i on C_1 are consecutive. Hence we get a triangulated face for

each z_i with even i which contains z_i and the two neighbors of z_i on C_1 .

We now show that the remaining faces in Group (i) are triangulated. Clearly each of the remaining faces in Group (i) must contain a vertex z_i with odd i since a vertex on a face in Group (i) is either on C_1 or on C_2 and a vertex on C_2 has at most two neighbors on C_1 . Let z_i, z_{i+1} and z_{i+2} be three consecutive vertices on C_2 with even i . Then z_i and z_{i+2} has a common neighbor x on C_1 . One can observe from our construction that x is also the only neighbor of z_{i+1} on C_1 . Then exactly two faces in Group (i) contains z_{i+1} and the two faces are triangulated. This implies that for each z_i on C_2 with odd i there are exactly two faces in Group (i) which contain z_i , and the two faces are triangulated. Therefore each face in Group (i) is triangulated.

Thus Γ has exactly two vertex-disjoint faces F_1 and F_2 each of which has exactly p vertices, and the rest of the faces are triangulated. \square

We are now ready to prove the Lemma 3.2.2.

Proof of Lemma 3.2.2

We first construct a planar embedding Γ of a graph G with $4p$ vertices by the construction **Construct-Doughnut**. We now show that G is a p -doughnut graph. To prove this claim we need to prove that G satisfies the following properties (a)–(c):

- (a) the graph G is a 5-connected planar graph;
- (b) any planar embedding Γ' of G has exactly two vertex-disjoint faces each of which has exactly p vertices, and all the other faces are triangulated; and
- (c) G has the minimum number of vertices satisfying (a) and (b).

(a) G is a planar graph since it has a planar embedding Γ as illustrated in Fig. 3.3(c). To prove that G is 5-connected, we show that the size of any cut-set of G is 5 or more. We first show that G is 5-regular. From the construction, one can easily see that each of the vertex of C_2 has exactly three neighbors in $V(C_1) \cup V(C_3)$. Hence the degree of each

vertex of C_2 is exactly 5. We now only prove that the degree of each vertex of C_1 is exactly 5 since the proof is similar for the vertices of C_3 . Each even index vertex v of C_2 has two neighbors on C_1 and the two neighbors of v are consecutive on C_1 by construction. Since C_2 has p even index vertices, C_1 has p vertices, and Γ is a planar embedding, each vertex u of C_1 has at most two even index neighbors on C_2 . Assume that a vertex u of C_1 has two even index neighbors y_i and y_{i+2} on C_2 . Since Γ is a planar embedding y_{i+1} can have only one neighbor on C_1 which is u . Thus a vertex u on C_1 has at most three neighbors on C_2 . Since there are exactly $3p$ edges each of which has one end point on C_1 and the other on C_2 , and a vertex on C_1 has at most three neighbors on C_2 , each vertex of C_1 has exactly three neighbors on C_2 . Hence the degree of a vertex on C_1 is 5. Therefore G is 5-regular. We now show that G is 5-connected. Assume for a contradiction that G has a cut-set of less than five vertices. In such a case, G would have a vertex of degree less than five, a contradiction. (Note that G is 5-regular, the vertices of G lie on three vertex disjoint cycles C_1 , C_2 and C_3 , none of the vertices of C_1 has a neighbor on C_3 , each of the faces of G is triangulated except faces F_1 and F_2 .) Hence G is 5-connected.

Therefore the graph G is a 5-connected planar graph.

(b) By Lemma 3.2.3, G has a planar embedding Γ such that Γ has exactly two vertex-disjoint faces F_1 and F_2 each of which has exactly p vertices, and the rest of the faces are triangulated. Since G is 5-connected, Fact 3.1.1 implies that any planar embedding Γ' of G has exactly two vertex-disjoint faces each of which has exactly p vertices, and all the other faces are triangulated.

(c) We have constructed the graph G with $4p$ vertices and proofs for (a) and (b) imply that G satisfies properties (a) and (b). G is a 5-connected planar graph and hence satisfies (d₁) of the definition of a p -doughnut graph. By Lemma 3.2.1, $4p$ is the minimum number of vertices of such a graph. Since we have constructed G with $4p$ vertices, G has the minimum number of vertices satisfying properties (a) and (b). \square

Lemma 3.2.1 and 3.2.2 imply that a p -doughnut graph G has exactly $4p$ vertices. Then the value of x in Eq. (3.2) is $2p$ in G . By Eq. (3.1), G has exactly $3x + 4p = 10p$ edges. Since G is 5-connected, every vertex has degree 5 or more. Then degree-sum formula implies that every vertex of G has degree exactly 5, since G has $4p$ vertices and $10p$ edges. Therefore the following theorem holds

Theorem 3.2.4 *Let G be a p -doughnut graph. Then G is 5-regular and has exactly $4p$ vertices.*

For a cycle C in a plane graph G , we denote by $G(C)$ the plane subgraph of G inside C excluding C . Let C_1 , C_2 and C_3 be three vertex-disjoint cycles in G such that $V(C_1) \cup V(C_2) \cup V(C_3) = V(G)$. Then we call an embedding Γ of G a *doughnut embedding* of G if C_1 is the outer face and C_3 is an inner face of Γ , $G(C_1)$ contains C_2 and $G(C_2)$ contains C_3 . We call C_1 the *outer cycle*, C_2 the *middle cycle* and C_3 the *inner cycle* of Γ . We next show that a p -doughnut graph has a doughnut embedding. To prove the claim we need the following lemma.

Lemma 3.2.5 *Let G be a p -doughnut graph. Let F_1 and F_2 be the two faces of G each of which contains exactly p vertices. Then $G - \{V(F_1) \cup V(F_2)\}$ is connected and contains a cycle.*

Proof. Since G is 5-connected, $G' = G - \{V(F_1) \cup V(F_2)\}$ is connected; otherwise, G would have a cut-set of 4 vertices - two of them are on F_1 and the other two are on F_2 , a contradiction. G' contains $2p$ vertices and at least $2p$ edges; if there is no edge between a vertex on F_1 and a vertex on F_2 in G then G' contains exactly $2p$ edges, otherwise G' contains more than $2p$ edges. Since G' is connected, has $2p$ vertices and has at least $2p$ edges, G' must have a cycle. \square

We now prove the following theorem.

Theorem 3.2.6 *A p -doughnut graph always has a doughnut embedding.*

Proof. Let F_1 and F_2 be two faces of G each of which contains exactly p vertices. Let Γ be a plane embedding of G such that F_1 is embedded as the outer face. By Lemma 3.2.5 $G - \{V(F_1) \cup V(F_2)\}$ is connected and contains a cycle. Let C' be a cycle in $G - \{V(F_1) \cup V(F_2)\}$. One can observe that if $G(C')$ does not contain F_2 , then there would be edge crossings among the edges from the vertices on C' to the vertices on F_1 and F_2 , and hence Γ would not be a plane embedding of G . Therefore $G(C')$ contains F_2 . Furthermore, $G - \{V(F_1) \cup V(F_2)\}$ contains exactly one cycle C' , and C' contains all vertices of $G - \{V(F_1) \cup V(F_2)\}$; otherwise, Γ would not be a planar embedding of G , a contradiction. Therefore in Γ , $G(F_1)$ contains C' and $G(C')$ contains F_2 and hence Γ is a doughnut embedding. \square

A *1-outerplanar* graph is an embedded planar graph where all vertices are on the outer face. It is also called *1-outerplane* graph. An embedded graph is a *k-outerplane* ($k > 1$) if the embedded graph obtained by removing all vertices of the outer face is a $(k - 1)$ -outerplane graph. A graph is *k-outerplanar* if it admits a *k-outerplanar* embedding. A planar graph G has *outerplanarity* k ($k > 0$) if it is *k-outerplanar* and it is not *j-outerplanar* for $0 < j < k$.

We now show that the outerplanarity of a p -doughnut graph G is 3. Since none of the faces of G contains all vertices of G , G does not admit 1-outerplanar embedding. We now need to show that G does not admit a 2-outerplanar embedding. We have the following fact.

Fact 3.2.7 *A graph G having outerplanarity 2 has a cut-set of four or less vertices.*

Proof. Deleting vertices from the outer face of a 2-outerplanar graph leaves a 1-outerplanar graph. A 1-outerplanar graph has a cut-set of at most 2. From the definition of 2-outerplanar graph and the cut-set size of 1-outerplanar graph, one can observe that a graph G having outerplanarity 2 has a cut-set of four or less vertices. \square

Since G is 5-connected graph, G has no cut-set of four or less vertices. Hence by Fact 3.2.7 the graph G has outerplanarity greater than 2. Thus the following lemma holds.

Lemma 3.2.8 *Let G be a p -doughnut graph for $p \geq 4$. Then G is neither an 1-outerplanar graph nor a 2-outerplanar graph.*

We now prove the following theorem.

Theorem 3.2.9 *The outerplanarity of a p -doughnut graph G is 3.*

Proof. A doughnut embedding of G immediately implies that G has a 3-outerplanar embedding. By Lemma 3.2.8 G is neither an 1-outerplanar graph nor a 2-outerplanar graph. Therefore the outerplanarity of a p -doughnut graph is 3. \square

Based on the properties of doughnut graphs shown in this section, one can recognize a doughnut graph as follows. A planar graph G is a doughnut graph if (i) G has $4p$ -vertices where p (≥ 4) is an integer, (ii) it is 5-regular, and (iii) for any planar embedding Γ of G , Γ has two vertex disjoint faces each of which has exactly p -vertices and all the other faces has exactly three vertices. The verification of Properties (i) and (ii) are trivial. Using an efficient algorithm for face traversing of G , one can easily verify the Property (iii). Thus one can recognize a doughnut graph in linear time using a linear-time face traversing algorithm.

3.3 Straight-Line Grid Drawings of Doughnut Graphs

In this section we give a linear-time algorithm for finding a straight-line grid drawing of a doughnut graph on a grid of linear area.

Let G be a p -doughnut graph. By Theorem 3.2.6 G has a doughnut embedding. Let Γ be a doughnut embedding of G as illustrated in Fig. 7.5(a). Let C_1 , C_2 , and C_3 be

the outer cycle, the middle cycle and the inner cycle of Γ , respectively. We now have the following facts.

Lemma 3.3.1 *Let G be a p -doughnut graph and let Γ be a doughnut embedding of G . Let C_1 , C_2 , and C_3 be the outer cycle, the middle cycle and the inner cycle of Γ , respectively. For any two consecutive vertices z_i, z_{i+1} on C_2 , one of z_i, z_{i+1} has exactly one neighbor on C_1 and the other has exactly two neighbors on C_1 .*

Proof. In Γ $G(C_1)$ contains C_2 and $G(C_2)$ contains C_3 . By Theorem 3.2.4 G is 5-regular, and hence each vertex on C_1 has exactly three neighbors on C_2 . Since each of the faces which contains vertices on both C_1 and C_2 is triangulated, one can observe that one of z_i, z_{i+1} has exactly one neighbor on C_1 and the other has exactly two neighbors on C_1 . \square

Lemma 3.3.2 *Let G be a p -doughnut graph and let Γ be a doughnut embedding of G . Let C_1, C_2 and C_3 be the outer cycle, the middle cycle and the inner cycle of Γ , respectively. Let z_i be a vertex on C_2 , then either the following (a) or (b) holds.*

(a) z_i has exactly one neighbor on C_1 and exactly two neighbors on C_3 .

(b) z_i has exactly one neighbor on C_3 and exactly two neighbors on C_1 .

Proof. In Γ $G(C_1)$ contains C_2 and $G(C_2)$ contains C_3 . By Theorem 3.2.4 G is 5-regular, and hence each vertex on C_2 has exactly three neighbors on C_1 and C_2 . These neighbors are on C_1 or on C_3 or on both. Lemma 3.3.1 yields z_i has either one or two neighbors on C_1 . If z_i has one neighbor on C_1 then (a) holds. Otherwise, (b) holds. \square

Before describing our algorithm we need some definitions. Let z_i be a vertex on C_2 such that z_i has two neighbors on C_1 . Let x and x' be the two neighbors of z_i on C_1 such that x' is the counter clockwise next vertex to x on C_1 . We call x the *left neighbor* of z_i on C_1 and x' the *right neighbor* of z_i on C_1 . Similarly we define the left neighbor and the

right neighbor of z_i on C_3 if a vertex z_i on C_2 has two neighbors on C_3 . We are now ready to give our algorithm.

We now embed C_1 , C_2 and C_3 on three nested rectangles R_1 , R_2 and R_3 , respectively on a grid as illustrated in Fig. 7.5(b). We draw rectangle R_1 on grid with four corners on grid point $(0,0)$, $(p+1, 0)$, $(p+1, 5)$ and $(0, 5)$. Similarly the four corners of R_2 are $(1, 1)$, $(p, 1)$, $(p, 4)$, $(1, 4)$ and the four corners of R_3 are $(2,2)$, $(p-1, 2)$, $(p-1, 3)$, $(2,3)$.

We first embed C_2 on R_2 as follows. Let z_1, z_2, \dots, z_{2p} be the vertices on C_2 in counter clockwise order such that z_1 has exactly one neighbor in C_1 . We put z_1 on $(1, 1)$, z_p on $(p, 1)$, z_{p+1} on $(p, 4)$ and z_{2p} on $(1, 4)$. We put the other vertices of C_2 on grid points of R_2 preserving the relative positions of vertices of C_2 .

We now put vertices of C_1 on R_1 as follows. Let x_1 be the neighbor of z_1 on C_1 and let x_1, x_2, \dots, x_p be the vertices of C_1 in counter clockwise order. We put x_1 on $(0, 0)$ and x_p on $(0, 5)$. Since z_1 has exactly one neighbor on C_1 , by Lemma 3.3.1, z_{2p} has exactly two neighbors on C_1 since z_{2p} and z_1 are consecutive vertices on C_2 . Since z_1 and z_{2p} is on a triangulated face of G having vertices on both C_1 and C_2 , x_1 is a neighbor of z_{2p} . One can easily observe that x_p is the other neighbor of z_{2p} on C_1 . Clearly the edges (x_1, z_1) , (x_1, z_{2p}) , (x_p, z_{2p}) can be drawn as straight-line segment without edge crossing as illustrated in Fig. 7.5(b). We next put neighbors of z_p and z_{p+1} . Let x_i be the neighbor of z_p on C_1 if z_p has exactly one neighbor on C_1 , otherwise let x_i be the left neighbor of z_p on C_1 . We put x_i on $(p+1, 0)$ and x_{i+1} on $(p+1, 5)$. We now show that the edges from z_p and z_{p+1} to x_i and x_{i+1} can be drawn as straight-line segments without edge crossings. We first consider the case where z_p has exactly one neighbor on C_1 . In this case x_i is the only one neighbor of z_p on C_1 . Then, by Lemma 3.3.1, z_{p+1} has two neighbors on C_1 . Since each face having vertices on both C_1 and C_2 is triangulated, x_i and x_{i+1} are the two neighbors of z_{p+1} on C_1 . Clearly the edges (z_p, x_i) , (z_{p+1}, x_i) and (z_{p+1}, x_{i+1}) can be drawn as straight-line segments without edge crossings, as illustrated in Fig. 7.5(b). We next consider the case where z_p has two neighbors on C_1 . In this case let x_i and x_{i+1} be

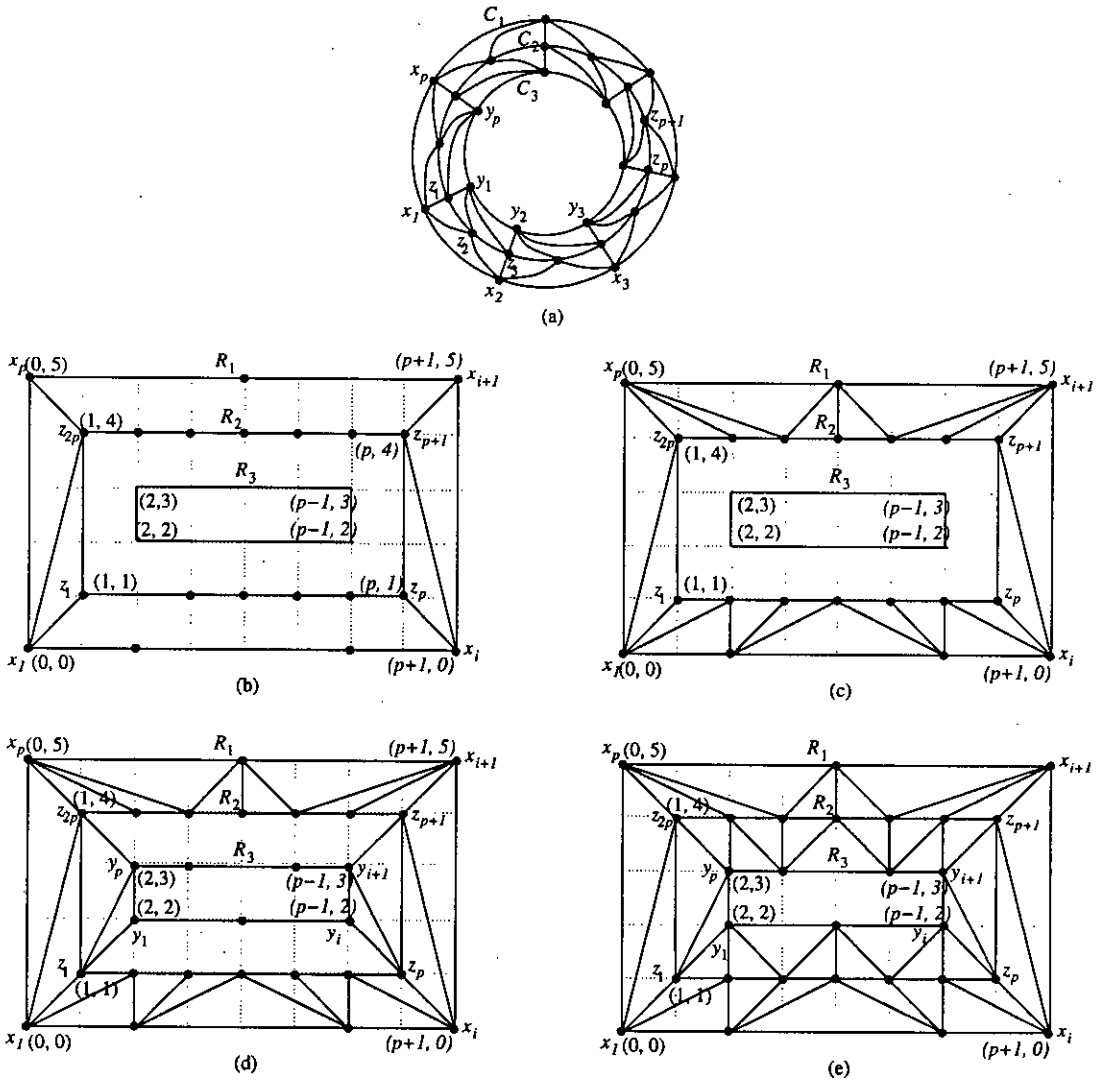


Figure 3.4: (a) A doughnut embedding of a p -doughnut graph of G , (b) edges between four corner vertices of R_1 and R_2 are drawn as straight-line segments, (c) edges between vertices on R_1 and R_2 are drawn, (d) edges between four corner vertices of R_2 and R_3 are drawn as straight-line segments and (e) a straight-line grid drawing of G .

the two neighbors of z_p on C_1 . By Lemma 3.3.1, z_{p+1} has exactly one neighbor on C_1 . Since the face of G containing z_p, z_{p+1} and the neighbor of z_{p+1} on C_1 is triangulated, x_{i+1} is the neighbor of z_{p+1} . Clearly the edges $(z_p, x_i), (z_p, x_{i+1})$ and (z_{p+1}, x_{i+1}) can be drawn as straight-line segments without edge crossings as illustrated in Fig. 3.5. We put the other vertices of C_1 on grid points of R_1 arbitrarily preserving their relative positions on C_1 .

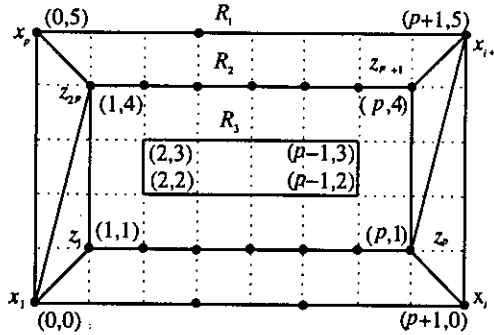


Figure 3.5: Illustration of case where z_p has two neighbors on C_1 .

Since vertices z_2, z_3, \dots, z_{p-1} are put in counter clockwise order on the horizontal segment between points $(1, 1)$ and $(p, 1)$ on R_2 preserving the relative positions of vertices on C_2 and x_2, x_3, \dots, x_{i-1} are put on the horizontal segment between points $(0, 0)$ and $(p + 1, 0)$ of R_1 preserving the relative positions of the vertices on C_1 , all edges of G connecting vertices in $\{z_2, z_3, \dots, z_{p-1}\}$ to vertices in $\{x_2, x_3, \dots, x_{i-1}\}$ can be drawn as straight-line segments without edge crossings. Similarly, the edges of G connecting vertices in $\{z_{p+1}, z_{p+2}, \dots, z_{2p-1}\}$ to vertices in $\{x_{i+2}, x_{i+3}, \dots, x_{p-1}\}$ can be drawn as straight-line segments without edge crossings. See Fig. 7.5(c).

We now put the vertices of C_3 on R_3 as follows. Since z_1 has exactly one neighbor on C_1 , by Lemma 3.3.2(a), z_1 has exactly two neighbors on C_3 . Then, by Lemma 3.3.2(b), z_{2p} has exactly one neighbor on C_3 since z_{2p} has exactly two neighbors on C_1 . Let y_1, y_2, \dots, y_p be the vertices on C_3 in counter clockwise order such that y_1 is the right neighbor of z_1 . Then y_p is the left neighbor of z_1 . We put y_1 on $(2, 2)$ and y_p on $(2, 3)$. Clearly

the edges (y_1, z_1) , (y_1, z_{2p}) , (y_p, z_1) can be drawn as straight-line segments without edge crossings, as illustrated in Fig. 7.5(d). We next put neighbors of z_p and z_{p+1} on C_3 . Let y_i be the neighbor of z_p on C_2 if z_p has exactly one neighbor on C_3 , otherwise y_i be the left neighbor and y_{i+1} be the right neighbor of z_p on C_3 . We put y_i on $(p-1, 2)$ and y_{i+1} on $(p-1, 3)$, and hence the edges of G from z_p and z_{p+1} to y_i and y_{i+1} can be drawn as straight-line segments without edge crossing as illustrated in Fig. 7.5(d). We put the other vertices of C_3 on grid points of R_3 arbitrarily preserving their relative positions on C_3 .

Since vertices z_2, z_3, \dots, z_{p-1} are put in counter clockwise order on a horizontal segment between points $(1, 1)$ and $(p, 1)$ on R_2 preserving the relative positions of vertices and y_2, y_3, \dots, y_{i-1} are put on a horizontal segment between points $(2, 2)$ and $(p-1, 2)$ of R_3 , all edges of G connecting vertices in $\{z_2, z_3, \dots, z_{p-1}\}$ to vertices in $\{y_2, y_3, \dots, y_{i-1}\}$ can be drawn as straight-line segments without edge crossings. Similarly, the edges of G connecting vertices in $\{z_{p+1}, z_{p+2}, \dots, z_{2p-1}\}$ to vertices in $\{y_{i+2}, y_{i+3}, \dots, y_{p-1}\}$ can be drawn as straight-line segments without edge crossings. Fig. 7.5(e) illustrates the complete straight-line grid drawing of a p -doughnut graph.

The area requirement of the straight-line grid drawing of a p -doughnut graph G is equal to the area of rectangle R_1 and the area of R_1 is $= (p+1) \times 5 = (n/4 + 1) \times 5 = O(n)$, where n is the number of vertices in G . Thus we have a straight-line grid drawing of a p -doughnut graph on a grid of linear area. Clearly the algorithm takes linear time. Thus the following theorem holds.

Theorem 3.3.3 *A doughnut graph G of n vertices has a straight-line grid drawing on a grid of area $O(n)$. Furthermore, the drawing of G can be found in linear time.*

3.4 Conclusion

In this chapter we introduced a new class of planar graphs, called doughnut graphs, which is a subclass of 5-connected planar graphs. A graph in this class has a straight-line grid drawing on a grid of linear area, and the drawing can be found in linear time. We also showed that a doughnut graph is a 5-regular graph and the outerplanarity of a doughnut graph is 3. Thus we identified a subclass of 3-outerplanar graphs that admits straight-line grid drawing with linear area. Although the class doughnut graphs contains exactly one graph for each integer $p \geq 4$, the class contains an infinite number of graphs. Using a linear-time face traversing algorithm, one can recognize a doughnut graph in linear time.

Chapter 4

Spanning Subgraphs of Doughnut Graphs

In this chapter, we deal with the straight-line drawing of the spanning subgraphs of doughnut graphs. In Chapter 3, we have seen that a doughnut graph admits a straight-line grid drawing with linear area. One can easily observe that a spanning subgraph of a doughnut graph also admits a straight-line grid drawing with linear area. Fig. 4.1(f) illustrates a straight-line grid drawing with linear area of a graph G' in Fig. 4.1(d) where G' is a spanning subgraph of a doughnut graph G in Fig. 4.1(a). Using a transformation from the “subgraph isomorphism” problem [GJ79] one can easily prove that recognition of a spanning subgraph of a given graph is an NP-complete problem in general. Hence recognition of a spanning subgraph of a doughnut graph is not a trivial problem. We thus restrict ourselves only in 4-connected planar graphs. In this chapter, we give a necessary and sufficient condition for a 4-connected planar graph to be a spanning subgraph of a doughnut graph which immediately gives a sufficient condition for a 4-connected planar graph that admit straight-line grid drawing with linear area.

This chapter is organized as follows. In Section 4.1, we give some definitions. Section 4.2 provides a necessary and sufficient condition for a 4-connected spanning subgraph

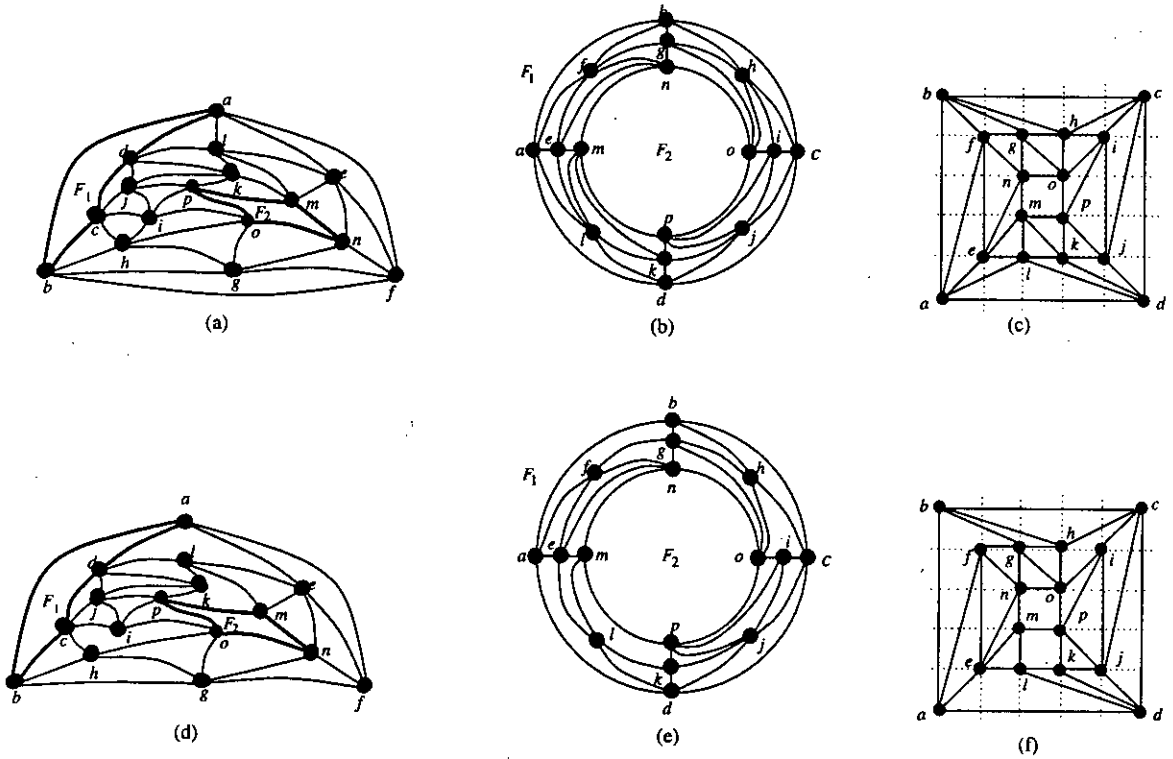


Figure 4.1: (a) A doughnut graph G , (b) a doughnut embedding of G , (c) a straight-line grid drawing of G with linear area, (d) a spanning subgraph G' of G , (e) an embedding of G' where face F_1 is embedded as the outerface and (f) a straight-line grid drawing of G' with area $O(n)$.

of a doughnut graph. In Section 4.3, we give a sufficient condition for a 4-connected planar graph that admits straight-line drawing with linear area. Finally Section 4.4 concludes the chapter. Our results presented in this chapter are published in [KR08].

4.1 Preliminaries

In this section we give some definitions.

Let $G = (V, E)$ be a connected simple plane graph with vertex set V and edge set E . A *path* in G is an ordered list of distinct vertices $v_1, v_2, \dots, v_q \in V$ such that $(v_{i-1}, v_i) \in E$ for all $2 \leq i \leq q$. Vertices v_1 and v_q are end-vertices of the path v_1, v_2, \dots, v_q . Two paths are *vertex-disjoint* if they do not share any common vertex except their end vertices. The *length* of a path is the number of edges on the path. We call a path P an *even path* if the number of edges on P is even. We call a path P an *odd path* if the number of edges on P is odd. For a face F in G we denote by $V(F)$ the set of vertices of G on the boundary of face F . We call two faces F_1 and F_2 are *vertex-disjoint* if $V(F_1) \cap V(F_2) = \emptyset$. We call a face a *quadrangle face* if the face has exactly four vertices.

An *isomorphism* from a simple graph G to a simple graph H is a bijection $f : V(G) \rightarrow V(H)$ such that $(u, v) \in E(G)$ if and only if $(f(u), f(v)) \in E(H)$. Let G_1 and G_2 are two graphs. The *subgraph isomorphism problem* asks to determine whether G_2 contains a subgraph isomorphic to G_1 . Subgraph isomorphism is known to be an NP-complete problem [GJ79]. It is not difficult to prove that recognition of a spanning subgraph of a graph is an NP-complete problem using a transformation from subgraph isomorphism problem.

4.2 A Characterization for Spanning Subgraphs

In this section, we give a necessary and sufficient condition for a 4-connected planar graph to be a spanning subgraph of a doughnut graph as in the following theorem.

Theorem 4.2.1 *Let G be a 4-connected planar graph with $4p$ vertices where $p > 4$ and let $\Delta(G) \leq 5$. Let Γ be a planar embedding of G . Assume that Γ has exactly two vertex disjoint faces F_1 and F_2 each of which has exactly p vertices. Then G is a spanning subgraph of a p -doughnut graph if and only if Γ holds the following conditions.*

- (a) G has no edge (x, y) such that $x \in V(F_1)$ and $y \in V(F_2)$.
- (b) Every face f of Γ has at least one vertex $v \in \{V(F_1) \cup V(F_2)\}$.
- (c) For any vertex $x \notin \{V(F_1) \cup V(F_2)\}$, total number of neighbors of x on faces F_1 and F_2 are at most three.
- (d) Every face f of Γ except the faces F_1 and F_2 has either three or four vertices.
- (e) For any x - y path P such that $V(P) \cap \{V(F_1) \cup V(F_2)\} = \emptyset$ and x has exactly two neighbors on face $F_1(F_2)$. Then the following conditions hold.
 - (i) If P is even, then the vertex y has at most two neighbors on face $F_1(F_2)$ and at most one neighbor on face $F_2(F_1)$.
 - (ii) If P is odd, then the vertex y has at most one neighbor on face $F_1(F_2)$ and at most two neighbors on face $F_2(F_1)$.

Fact 3.1.1 implies that decomposition of a 4-connected planar graph G into its facial cycles is unique. Throughout the section we thus often mention faces of G without mentioning its planar embedding where description of the faces are valid for any planar embedding of G , since $\kappa(G) \geq 4$ for every graph G considered in this section.

Before proving the necessity of Theorem 4.2.1, we have the following fact.

Fact 4.2.2 *Let G be a 4-connected planar graph with $4p$ vertices where $p > 4$ and let $\Delta(G) \leq 5$. Assume that G has exactly two vertex disjoint faces F_1 and F_2 each of which has exactly p vertices. If G is a spanning subgraph of a doughnut graph then G can be augmented to a 5-connected 5-regular graph G' through triangulation of all the non-triangulated faces of G except the faces F_1 and F_2 .*

One can easily observe that the following fact holds from the construction **Construct-Doughnut** given in Section 3.2 of Chapter 3.

Fact 4.2.3 *Let G be a doughnut graph, and let P be any x - y path such that $V(P) \cap \{V(F_1) \cup V(F_2)\} = \emptyset$ and x has exactly two neighbors on face $F_1(F_2)$. Then the following conditions hold.*

(i) *If P is even, then the vertex y has exactly two neighbors on face $F_1(F_2)$ and exactly one neighbor on face $F_2(F_1)$.*

(ii) *If P is odd, then the vertex y has exactly one neighbor on face $F_1(F_2)$ and exactly two neighbors on face $F_2(F_1)$.*

We are now ready to prove the necessity of Theorem 4.2.1.

Proof for Necessity of Theorem 4.2.1

Assume that G is a spanning subgraph of a p -doughnut graph. Then by Theorem 3.2.4 G has $4p$ vertices. Clearly $\Delta(G) \leq 5$ and G satisfies the Conditions (a), (b), and (c), otherwise G would not be a spanning subgraph of a doughnut graph. The necessity of Condition (e) is obvious by Fact 4.2.3. Hence it is sufficient to prove the necessity of Condition (d) only.

(d) G does not have any face of two or less vertices since G is a 4-connected planar graphs. Then every face of G has three or more vertices. We now show that G has no face of more than four vertices. Assume for a contradiction that G has a face f of q vertices such that $q > 4$. Then f can be triangulated by adding $q - 3$ extra edges. These extra edges increase the degrees of $q - 2$ vertices, and the sum of the degrees will be increased

by $2(q - 3)$. Using pigeon hole principle one can easily observe that there is a vertex among the $q(> 4)$ vertices whose degree will be raised by at least 2 after triangulation of f . Then G' would have a vertex of degree six or more where G' is a graph obtained after triangulation of f . Hence we cannot augment G to a 5-regular graph through triangulation of all the non-triangulated faces of G other than the faces F_1 and F_2 . Therefore G cannot be a spanning subgraph of a doughnut graph by Fact 4.2.2, a contradiction. Hence each face f of G except F_1 and F_2 has either three or four vertices. \square

In the remaining of this section we give a constructive proof for the sufficiency of Theorem 4.2.1. Assume that G satisfies the conditions of Theorem 4.2.1. We now have the following lemma.

Lemma 4.2.4 *Let G be a 4-connected planar graph satisfying the conditions in Theorem 4.2.1. Assume that all the faces of G except F_1 and F_2 are triangulated. Then G is a doughnut graph.*

Proof. To prove the claim, we have to prove that (i) G is 5-connected, (ii) G has two vertex disjoint faces each of which has exactly p , $p > 4$ vertices, and all the other faces of G has exactly three vertices, and (iii) G has the minimum number of vertices satisfying the properties (i) and (ii).

(i) We first prove that G is a 5-regular graph. Every face of G is a triangle except F_1 and F_2 . Furthermore each of F_1 and F_2 has exactly p , $p > 4$ vertices. Then G has $3(4p) - 6 - 2(p - 3) = 10p$ edges. Since none of the vertices of G has degree more than five and G has exactly $10p$ edges, each vertex of G has degree exactly five. We now prove that vertices of G lie on three vertex-disjoint cycles C_1 , C_2 and C_3 such that cycles C_1 , C_2 , C_3 contain exactly p , $2p$ and p vertices, respectively. We take an embedding Γ of G such that F_1 is embedded as the outer face and F_2 is embedded as an inner face. We take the contour of face F_1 as cycle C_1 and contour of face F_2 as cycle C_3 . Then each of C_1 and C_2 contains exactly p , $p > 4$ vertices. Since G satisfies Conditions (a), (b) and

(c) of Theorem 4.2.1 and all the faces of G except F_1 and F_2 are triangulated, the rest $2p$ vertices of G form a cycle in Γ . We take this cycle as C_2 . $G(C_2)$ contains C_3 since G satisfies Condition (b) in Theorem 4.2.1. Clearly C_1 , C_2 and C_3 are vertex-disjoint and cycles C_1 , C_2 , C_3 contain exactly p , $2p$ and p vertices, respectively. We now prove that G is 5-connected. Assume for a contradiction that G has a cut-set of less than five vertices. In such a case G would have a vertex of degree less than five, a contradiction. Hence G is 5-connected.

(ii) The proof of this part is obvious since G has two vertex disjoint faces each of which has exactly p vertices and all the other faces of G has exactly three vertices.

(iii) The number of vertices of G is $4p$. Using Lemma 3.2.1, we can easily proof that it is the minimum number of vertices required to construct a graph G that satisfies the properties (i) and (ii).

Therefore G is a doughnut graph. □

We thus assume that G has a non-triangulated face f except faces F_1 and F_2 . By Condition (d) of Theorem 4.2.1, f is a quadrangle face. It is now sufficient to show that we can augment the graph G to a doughnut graph by triangulating each of the quadrangle faces of G . However, we can not augment G to a doughnut graph by triangulating each quadrangle face arbitrarily. For example, the graph G in Fig. 4.2(a) satisfies all the conditions of Theorem 4.2.1 and it has exactly one quadrangle face $f_1(a, b, c, d)$. If we triangulate f_1 by adding an edge (a, c) as illustrated in Fig. 4.2(b) the resulting graph G' would not be a doughnut graph since a doughnut graph does not have an edge (a, c) such that $a \in V(F_1)$ and $c \in V(F_2)$. But if we triangulate f_1 by adding an edge (b, d) as illustrated in Fig. 4.2(c) the resulting graph G' is a doughnut graph. Hence every triangulation of a quadrangle face is not always valid to augment G to a doughnut graph. We call a triangulation of a quadrangle face f of G is a *valid triangulation* if the resulting graph G' obtained after the triangulation of f does not contradict any condition of Theorem 4.2.1. We call a vertex v on the contour of a quadrangle face f a *good vertex* if v is

one of the end vertex of an edge which is added for a valid triangulation of f .

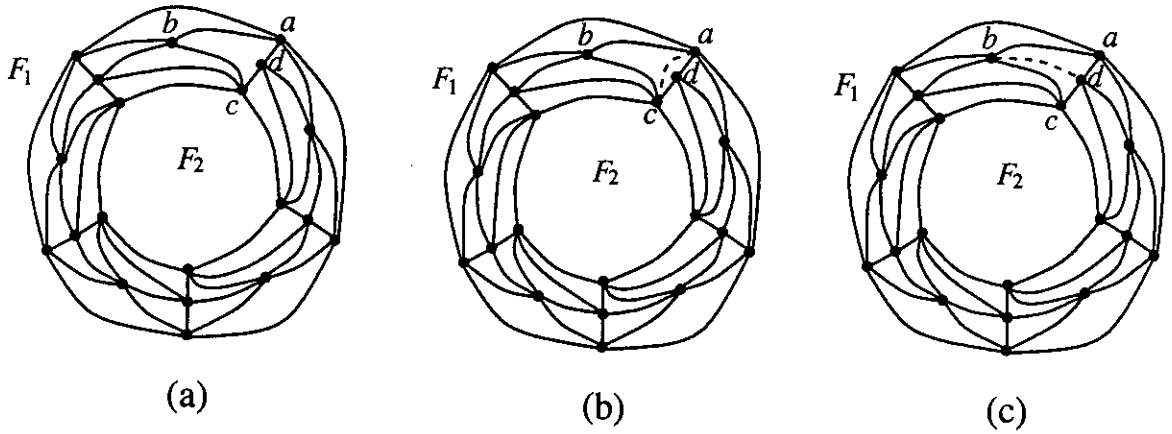


Figure 4.2: (a) $f_1(a, b, c, d)$ is a quadrangle face, and (b) and (c) are two different triangulations of f_1 .

We call a quadrangle face f of G an α -face if f contains at least one vertex from each of the faces F_1 and F_2 . Otherwise we call a quadrangle face f of G a β -face. In Fig. 4.3, $f_1(a, b, c, d)$ is an α -face whereas $f_2(p, q, r, s)$ is a β -face.

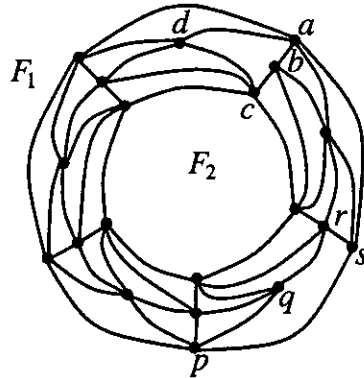


Figure 4.3: $f_1(a, b, c, d)$ is an α -face and $f_2(p, q, r, s)$ is a β -face.

In a valid triangulation of an α -face f of G no edge is added between any two vertices $x, y \in V(f)$ such that $x \in V(F_1)$ and $y \in V(F_2)$. Hence the following fact holds on an α -face f .

Fact 4.2.5 *Let G be a 4-connected planar graph satisfying the conditions in Theorem 4.2.1. Let f be an α -face in G . Then f admits a unique valid triangulation and the triangulation is obtained by adding an edge between two vertices those are not on F_1 and F_2 .*

Faces $f_1(a, b, c, d)$ and $f_2(p, q, r, s)$ in Fig. 4.4(a) are two α -faces and Fig. 4.4(b) illustrates the valid triangulations of f_1 and f_2 . Vertices b and d of f_1 , and vertices q and s of f_2 , are good vertices.

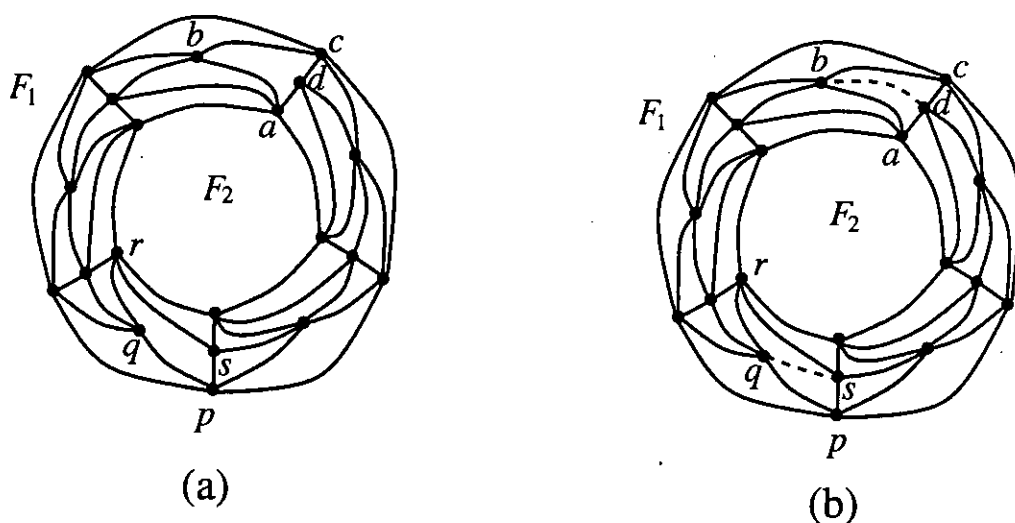


Figure 4.4: (a) $f_1(a, b, c, d)$ and $f_2(p, q, r, s)$ are two α -faces, and (b) valid triangulations of f_1 and f_2 .

We call a β -face a β_1 -face if the face contains exactly one vertex either from F_1 or from F_2 . Otherwise we call a β -face a β_2 -face. In Fig. 4.5, $f_1(a, b, c, d)$ is a β_1 -face whereas $f_2(p, q, r, s)$ is a β_2 -face. We call a vertex v on the contour of a β_1 -face f a *middle* vertex of f if the vertex is in the middle position among the three consecutive vertices other than the vertex on F_1 or F_2 . In Fig. 4.5, vertex c of f_1 and vertex r of f_2 are the middle vertices of f_1 and f_2 respectively.

In a valid triangulation of a β_1 -face f of G no edge is added between any two vertices $x, y \in V(f)$ such that $x, y \notin V(F_1) \cup V(F_2)$. Hence the following fact holds on a β_1 -face

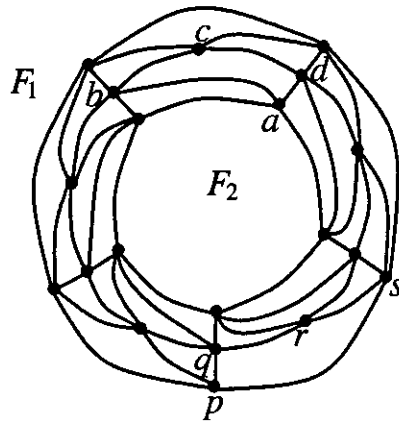


Figure 4.5: $f_1(a, b, c, d)$ is a β_1 -face and $f_2(p, q, r, s)$ is a β_2 -face.

f.

Fact 4.2.6 *Let G be a 4-connected planar graph satisfying the conditions in Theorem 4.2.1. Let f be a β_1 -face of G . Then f admits an unique valid triangulation and the triangulation is obtained by adding an edge between the vertex on F_1 or F_2 and the middle vertex.*

Faces $f_1(a, b, c, d)$ and $f_2(p, q, r, s)$ in Fig. 4.6(a) are two β_1 -faces and Fig. 4.6(b) illustrates the valid triangulations of f_1 and f_2 . Vertices a and c of f_1 , and vertices p and r of f_2 , are good vertices.

In a valid triangulation of a β_2 -face f of G no edge is added between any two vertices $x, y \in V(f)$ where $x \in V(F_1)(V(F_2))$, $y \notin \{V(F_1) \cup V(F_2)\}$ and G has either (i) an even q - y path P such that q has exactly two neighbors on $F_2(F_1)$ and $V(P) \cap \{V(F_1) \cup V(F_2)\} = \emptyset$, or (ii) an odd q - y path P such that q has exactly two neighbors on $F_1(F_2)$ and $V(P) \cap \{V(F_1) \cup V(F_2)\} = \emptyset$. Hence the following fact holds on a β_2 -face f .

Fact 4.2.7 *Let G be a 4-connected planar graph satisfying the conditions in Theorem 4.2.1. Let f be a β_2 -face of G . Then f admits an unique valid triangulation and the triangulation is obtained by adding an edge between a vertex on face F_1 or F_2 and a vertex $z \notin V(F_1) \cup V(F_2)$.*

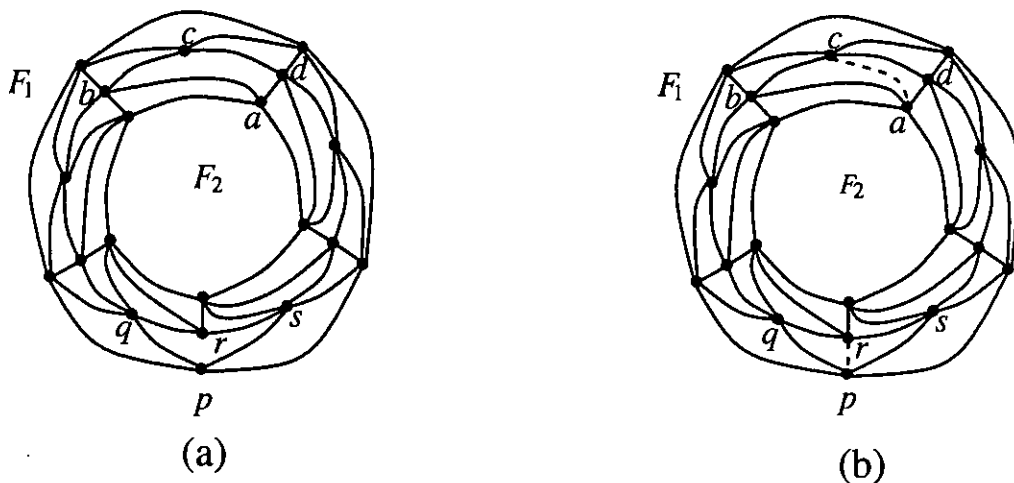


Figure 4.6: (a) $f_1(a, b, c, d)$ and $f_2(p, q, r, s)$ are two β_1 -faces and (b) valid triangulations of f_1 and f_2 .

Face $f_1(a, b, c, d)$ in Fig. 4.7(a) is a β_2 -face and G has an even u - d path P such that u has exactly two neighbors g and h on F_2 , and $V(P) \cap \{V(F_1) \cup V(F_2)\} = \emptyset$. Fig. 4.7(c) illustrates the valid triangulation of f_1 . Vertices a and c are the good vertices of f_1 . Face $f_2(l, m, n, o)$ in Fig. 4.7(b) is a β_2 -face and G has an odd v - o path P such that v has exactly two neighbors s and t on F_1 , and $V(P) \cap \{V(F_1) \cup V(F_2)\} = \emptyset$. Fig. 4.7(d) illustrates the valid triangulation of f_2 . Vertices l and n are the good vertices of f_2 .

We now have the following Lemmas 4.2.8 and 4.2.9.

Lemma 4.2.8 *Let G be a 4-connected planar graph satisfying the conditions in Theorem 4.2.1. Then any quadrangle face f of G admits an unique valid triangulation such that after triangulation $d(v) \leq 5$ holds for any vertex v in the resulting graph.*

Proof. By Facts 4.2.5, 4.2.6 and 4.2.7, f admits an unique valid triangulation. Since a valid triangulation increases the degree of a good vertex by one, it is sufficient to show that each good vertex of f has degree less than five in G . Assume for a contradiction that a good vertex v has degree more than four in G . Then one can observe that G would violate a condition in Theorem 4.2.1.

□

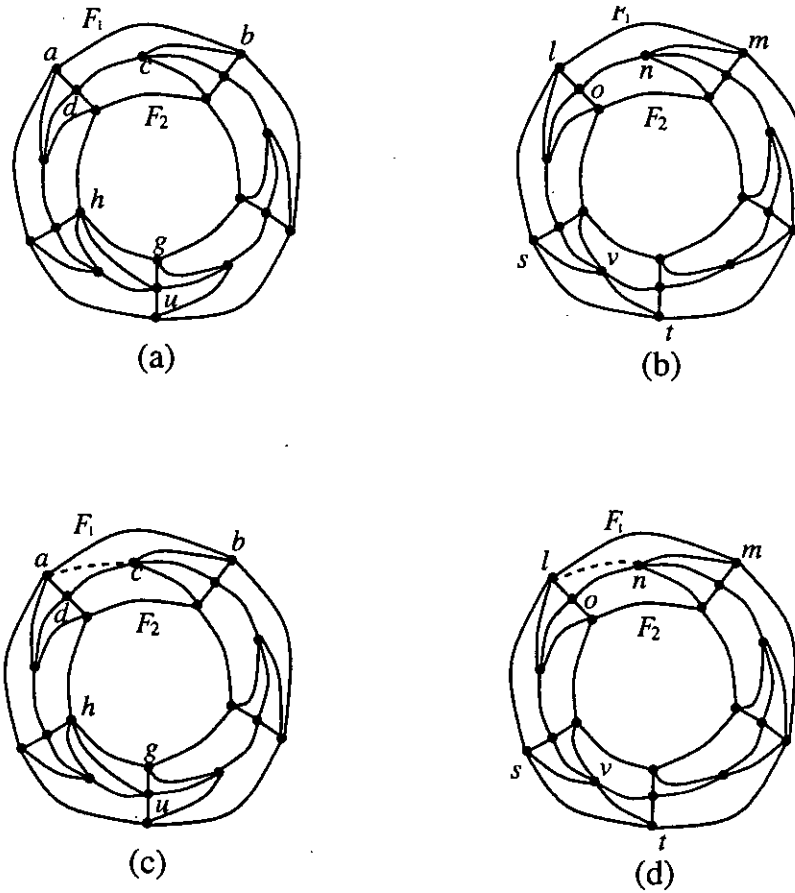


Figure 4.7: Illustration for valid triangulation of β_2 -face; (a) $f_1(a, b, c, d)$ is a β_2 -face and G satisfies condition (i), (b) $f_2(l, m, n, o)$ is a β_2 -face and G satisfies condition(ii), and (c) and (d) illustrate the valid triangulations of f_1 and f_2 respectively.

Lemma 4.2.9 *Let G be a 4-connected planar graph satisfying the conditions in Theorem 4.2.1. Also assume that G has quadrangle faces. Then no two quadrangle faces f_1 and f_2 have a common vertex which is a good vertex for both the faces f_1 and f_2 .*

Proof. Assume that faces f_1 and f_2 have a common vertex u . If u is neither a good vertex of f_1 nor a good vertex of f_2 , then we have done. We thus assume that u is a good vertex of f_1 or f_2 . Without loss of generality we assume that u is a good vertex of f_1 . Then u is not a good vertex of f_2 , otherwise u would not be a common vertex of f_1 and f_2 , a contradiction. Thus faces f_1 and f_2 can not have a common vertex which is a good vertex for both the faces. \square

We are now ready to give a proof for the sufficiency of the Theorem 4.2.1.

Proof for Sufficiency of Theorem 4.2.1

Assume that the graph G satisfies all the conditions of Theorem 4.2.1. If all the faces of G except F_1 and F_2 are triangulated, then by Lemma 4.2.4 G is a doughnut graph. Otherwise, we triangulate each quadrangle face of G , using its valid triangulation. Let G' be the resulting graph. For each vertex v in G' $d(v) \leq 5$, since according to Lemma 4.2.8 degree of each vertex of the graph remains five or less after valid triangulation of each quadrangle face and by Lemma 4.2.9 no two quadrangle faces of G have a common vertex which is a good vertex for both the faces. Since G satisfies the conditions in Theorem 4.2.1, G' is obtained from G using valid triangulations of quadrangle faces and $d(v) \leq 5$ for each vertex v in G' , G' satisfies the conditions in Theorem 4.2.1. Hence G' is a doughnut graph by Lemma 4.2.4. Therefore G is a spanning subgraph of a doughnut graph. \square

4.3 A Sufficient Condition for Linear Area Drawings

In this section we give a sufficient condition for a 4-connected planar graph that admits straight-line grid drawing with linear area.

In Theorem 4.2.1 we have given a necessary and sufficient condition for a 4-connected planar graph to be a spanning subgraph of a doughnut graph. We now have the following lemma.

Lemma 4.3.1 *Let G be a 4-connected planar graph satisfying the conditions in Theorem 4.2.1. Then G can be augmented to a doughnut graph in linear time.*

Proof. We first embed G such that F_1 is embedded as the outer face and F_2 is embedded as an inner face. We then triangulate each of the quadrangle faces of G using its valid triangulation if G has quadrangle faces. Let G' be the resulting graph. As shown in the sufficiency proof of Theorem 4.2.1, G' is a doughnut graph. One can easily find all quadrangle faces of G and perform their valid triangulations in linear time, hence G' can be obtained in linear time. \square

The proof of Lemma 4.3.1 immediately gives us a linear-time algorithm to augment a 4-connected planar graph G to a doughnut graph if G satisfies the conditions in Theorem 4.2.1. We have thus provided a sufficient condition for a 4-connected planar graph that admits straight-line grid drawings with linear area as stated in the following theorem.

Theorem 4.3.2 *Let G be a 4-connected planar graph satisfying the conditions in Theorem 4.2.1. Then G admits a straight-line grid drawing on a grid of area $O(n)$. Furthermore, the drawing of G can be found in linear time.*

Proof. Using the method described in the proof of Lemma 4.3.1, we augment G to a doughnut graph G' by adding dummy edges (if required) in linear time. By Theorem 3.3.3, G' admits a straight-line grid drawing on a grid of area $O(n)$. We finally obtain a drawing of G from the drawing of G' by deleting the dummy edges (if any) from the drawing of G' . By Lemma 4.3.1, G can be augmented to a doughnut graph in linear time and by Theorem 3.3.3, straight-line grid drawing of a doughnut graph can be found in linear time. Moreover, the dummy edges can also be deleted from the drawing of a doughnut graph in linear time. Hence the drawing of G can be found in linear time. \square

105995

4.4 Conclusion

In this chapter, we established a necessary and sufficient condition for a 4-connected planar graph G to be a spanning subgraph of a doughnut graph. We also gave a linear-time algorithm to augment a 4-connected planar graph G to a doughnut graph if G satisfies the necessary and sufficient condition. By introducing the necessary and sufficient condition, in fact, we have provided a sufficient condition for a 4-connected planar graph that admits straight-line grid drawings with linear area.

Chapter 5

Label-Constrained Outerplanar Graphs

In this chapter, we introduce a subclass of outerplanar graphs which has a straight-line grid drawing on a grid of area $O(n \log n)$. We give a linear-time algorithm to find such a drawing. We call this class *label-constrained outerplanar graphs* since a “vertex labeling” of the dual tree of this graph satisfies certain constraints. Fig. 3.1(a) illustrates a label-constrained outerplanar graph G , and a straight-line grid drawing of G with $O(n \log n)$ area is illustrated in Fig. 3.1(b). It is well known that a planar graph of n vertices admits a straight-line grid drawing on a grid of area $O(n^2)$ [Sch90, FPP90]. A lower bound of $\Omega(n^2)$ on the area-requirement for straight-line grid drawings of certain planar graphs are also known [CN98, FPP90]. Garg and Rusu showed that an n -node binary tree has a planar straight-line grid drawing with area $O(n)$ [GR02, GR04b]. Although trees admit straight-line grid drawings with linear area, it is generally thought that triangulations may require a grid of quadratic size. Hence finding nontrivial classes of planar graphs of n vertices richer than trees that admit straight-line grid drawings with area $o(n^2)$ is posted as an open problem in [BEGKLM04]. The problem of finding straight-line grid drawings of outerplanar graphs with $o(n^2)$ area was first posed by Biedl in [Bie02], and Garg and

Rusu showed that an outerplanar graph with n vertices and the maximum degree d has a planar straight-line drawing with area $O(dn^{1.48})$ [GR04a]. Di Battista and Frati showed that a “balanced” outerplanar graph of n vertices has a straight-line grid drawing with area $O(n)$ and a general outerplanar graph of n vertices has a straight-line grid drawing with area $O(n^{1.48})$ [DF06]. Recently Frati showed that a general outerplanar graphs with n vertices admits a straight-line grid drawing with area $O(dn \log n)$, where d is the maximum degree of the graph [Fra07].

This chapter is organized as follows. In Section 5.1, we give some definitions. Section 5.2 provides the drawing algorithm. Section 5.3 presents a linear-time algorithm for recognition of a label-constrained outer planar graph, and Section 5.4 concludes the chapter. Our results presented in this chapter are going to appear in [KAR09].

5.1 Preliminaries

In this section we introduce a labeling of a tree, give some definitions and define a class of outerplanar graphs which we call “label-constrained outerplanar graphs.”

We now define a vertex labeling of a rooted binary tree. Let T be a binary tree and let r be the root of T . Then the *labeling of a vertex u of T with respect to r* , which we denote by $L_r(u)$, is defined as follows:

- (i) if u is a leaf node then $L_r(u) = 1$;
- (ii) if u has only one child q and $L_r(q) = k$ then $L_r(u) = k$;
- (iii) if u has two children s and t , and $L_r(s) = k$ and $L_r(t) = k'$ where $k > k'$, then $L_r(u) = k$; and
- (iv) if u has two children s and t , and $L_r(s) = k$ and $L_r(t) = k$ then $L_r(u) = k + 1$.

This vertex labeling is similar to Horton-Strahler number [Hor45, Str52] which was originally introduced to classify the river systems.

We denote by $L_r(T)$ the labeling of all the vertices of T with respect to r . Fig. 5.1 illustrates the vertex labeling of a binary tree T rooted at r where an integer value represents the label of the associated vertex. The following lemma is immediate from the

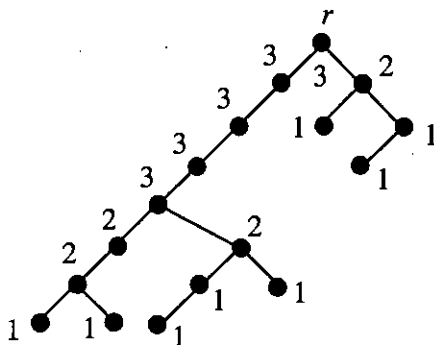


Figure 5.1: Vertex labeling of a binary tree T .

labeling defined above.

Lemma 5.1.1 *Let T be a binary tree and let r be the root of T . Let u and v be two vertices of T such that u is an ancestor of v . If $L_r(u) = k$ and $L_r(v) = k'$, then $k \geq k'$.*

The following lemma gives an upper bound on the value of the vertex labeling.

Lemma 5.1.2 *Let T be a binary tree with n vertices and let r be the root of T . Assume that $L_r(r) = k$. Then $k = O(\log n)$.*

Proof. We first show that T has at least $2^k - 1$ vertices using an induction on k . The claim is obvious for $k = 1$. Assume that $k \geq 2$ and the result is true for all trees T' with root of label k' such that $k' < k$. Let T be a tree with the root of label k . We first consider the case where r has a single child. In this case r has a successor q such that each of the left and the right children of q is labeled with $k - 1$, and hence by induction hypothesis each of the subtrees of q has at least $2^{k-1} - 1$ vertices. Then T has at least 2^k vertices. We now consider the case where r has two children. In this case, either (i) each of the children of r has label $k - 1$ or (ii) one of the children of r has label k and the other one has label k' where $k' < k$. In case of (i), by induction hypothesis, each of the

subtrees of r of T has at least $2^{k-1} - 1$ vertices, and hence T has at least $2^k - 1$. In case of (ii), we assume that the subtree of r of T rooted at a vertex p is labeled with k . Then p has a successor s such that each of the left and the right children of s is labeled with $k - 1$, and hence by induction hypothesis each of the subtrees of s has at least $2^{k-1} - 1$ vertices. Then T has at least 2^k vertices. Therefore T has at least $2^k - 1$ vertices. Hence $n \geq 2^k - 1$, i.e., $k = O(\log n)$. \square

We have the following lemma also.

Lemma 5.1.3 *Let T be a binary tree and let r be the root of T . Assume that all the vertices of T are labeled with respect to r using vertex labeling. Let $V(k)$ be a set of vertices such that for all $v \in V(k)$, $L_r(v) = k$. Then any connected component of the subgraph of T induced by $V(k)$ is a path.*

Proof. Let a connected component of the subgraph induced by $V(k)$ in T be $T'(k)$. Assume for a contradiction that $T'(k)$ is not a path. Then a vertex $v \in T'(k)$ has degree three. In such a case, v and the two children of v have the same label in T which is a contradiction to the definition of vertex labeling of T . Hence any connected component of the subgraph of T induced by $V(k)$ is a path. \square

A binary tree T is *ordered* if one child of each vertex v of T is designated as the *left child* and the other is designated as the *right child*. (Note that the left child or the right child of a vertex may be empty.) Let T be a rooted ordered binary tree. For any vertex $v \in T$, we call the subtree of T rooted at the left child (if any) of v the *left subtree* of v . Similarly we define the *right subtree* of v . We call an u - v path of T a *left-left path* if u is an ancestor of all the vertices of the path and each vertex except u of this path is the left child of its parent. Similarly we define a *right-right path* of T . We call a path of T a *cross path* if the path is neither a left-left path nor a right-right path. In Fig. 5.2(a) the path $P_0 = v_1, v_2, v_3, v_4$ is a left-left path; $P_1 = v_0, v_8, v_9$ is a right-right path; and $P_2 = v_5, v_6, v_{16}$ is a cross path. We call a maximal left-left path of T the *leftmost path* of T if

one of the end vertex of the path is the root of T . Similarly we define the *rightmost* path of T . In Fig. 5.2(b) the path v_0, v_1, \dots, v_7 is the leftmost path; and the path v_0, v_8, v_9 is the rightmost path of T , where v_0 is the root of T . For any vertex $x \in T$, we call a

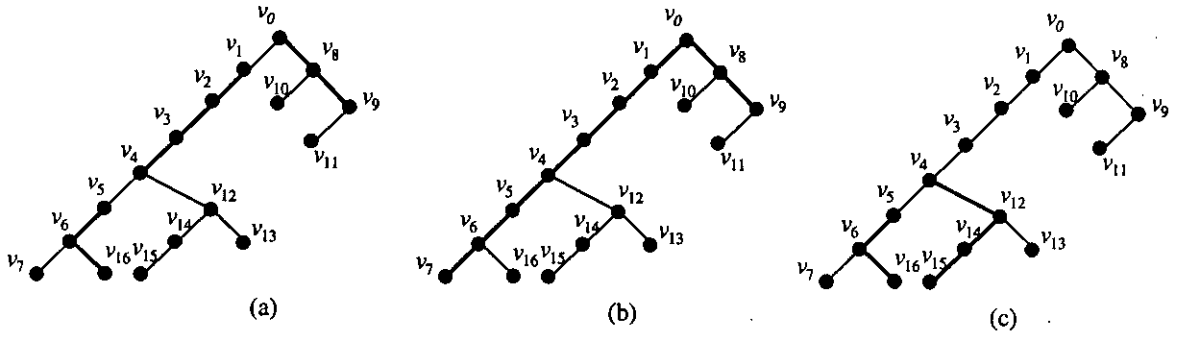


Figure 5.2: Illustration for different types of paths in an ordered binary tree T rooted at v_0 .

path x, v_1, \dots, v_m is the *left-right* path of x such that v_1 is the left child of x , and v_{i+1} is the right child of v_i where $1 \leq i \leq m - 1$. Similarly we define the *right-left* path of x . The path v_5, v_6, v_{16} is the left-right path of vertex v_5 , and the path $v_4, v_{12}, v_{14}, v_{16}$ is the right-left path of v_4 in Fig. 5.2(c).

We call $L_r(T)$ a *flat labeling* if any path induced by the vertices of T of the same label is either a left-left or a right-right path. We now have the following fact.

Fact 5.1.4 *Let T be an ordered binary tree and let r be the root of T . Let $L_r(T)$ be a flat labeling. Then for any vertex $x \in T$, each of the vertices of the left-right (right-left) path of x except the left (right) child of x has a smaller label than the label of x .*

Let G be a maximal outerplanar graph and let T be the dual tree of G . We convert T as a rooted ordered binary tree T_r by assigning its root r and ordering of the children of each vertex of T , as follows. Let r be a vertex of T such that r corresponds to an inner face f_r of G containing an edge (u, v) on the outerface. (Note that the degree of r is either one or two.) We regard r as the root of T . Let w be the vertex of f_r other than

u, v such that u, v and w appear in the clockwise order on f_r . We call u and v the *poles* of f_r and w the *central vertex* of f_r . We also call u the *left vertex* of f_r and v the *right vertex* of f_r . The vertex of T corresponding to the face sharing the vertices v and w (if any) of f_r is the *right child* of r and the vertex of T corresponding to the face sharing the vertices u and w (if any) of f_r is the *left child* of r . Let p and q be two vertices of T such that p is the parent of q , and let f_p and f_q be the two faces of G corresponding to p and q in T . Let v_1, v_2 and v_3 be the vertices of f_q in the clockwise order such that v_1 and v_2 are also on f_p . Then v_1 and v_2 are poles of f_q , and v_3 is the central vertex of f_q . The vertex v_1 is the left vertex of f_q and the vertex v_2 is the right vertex of f_q . The vertex of T corresponding to the face sharing the vertices v_2 and v_3 (if any) of f_q is the right child of q and the vertex of T corresponding to the face sharing the vertices v_1 and v_3 (if any) of f_q is the left child of q . Thus we have converted the dual tree T of a maximal outerplanar graph G to a rooted ordered dual tree T_r .

We are now ready to give the definition of “label-constrained outerplanar graphs.” Let G be a maximal outerplanar graph and let T be the dual tree of G . We call G a *label-constrained outerplanar graph* if T can be converted to a rooted ordered binary dual tree T_r such that $L_r(T_r)$ is a flat labeling. Fig. 5.3(a) illustrates a label-constrained outerplanar graph G since G is a maximal outerplanar graph and $L_r(T_r)$ a flat labeling as illustrated in Fig. 5.3(b). The $L_p(T_p)$ is not a flat labeling since T_p has a cross path induced by the label 2 vertices as illustrated in Fig. 5.3(c).

5.2 Drawings of Label-Constrained Outerplanar Graphs

In this section we give a linear-time algorithm for finding a straight-line grid drawing of a label-constrained outerplanar graph with $O(n \log n)$ area.

Let G be a maximal outerplanar graph and let T_r be the rooted ordered binary dual tree of G taking a vertex r of T_r as the root. In [DF06], Di Battista and Frati defined

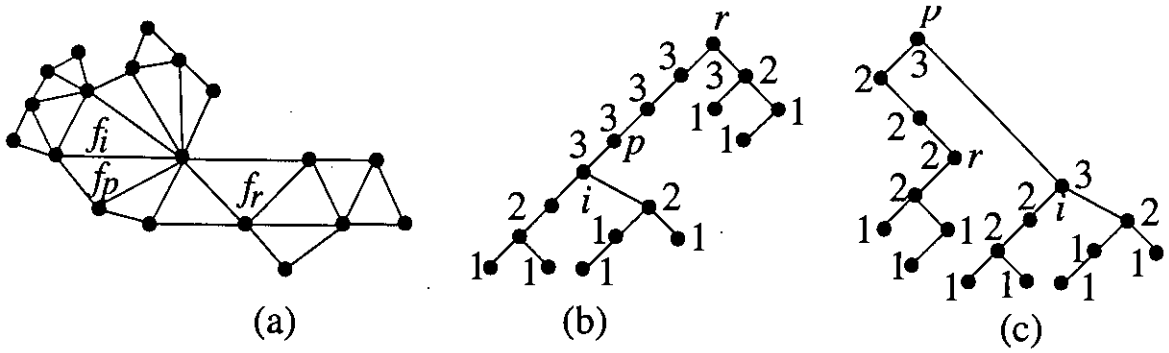


Figure 5.3: (a) A maximal outerplanar graph G , (b) a rooted ordered binary dual tree T_r of G where the vertex r of T_r corresponds to the face f_r of G , and (c) another rooted ordered binary dual tree T_p of G where the vertex p of T_p corresponds to the face f_p of G .

a bijection function γ between the vertices of T_r and vertices of G except for the poles of the face f_r corresponding to the root of T_r where each of the vertex of T_r is mapped to the central vertex of the corresponding face of G . We immediately have the following lemma from [DF06].

Lemma 5.2.1 *Let G be a maximal outerplanar graph and let T_r be the rooted ordered binary dual tree of G . Then G contains a copy of T_r which is a spanning tree T' of $G - \{u, v\}$, where u and v are the poles of the face f_r corresponding to the root of T_r .*

Fig. 5.4(b) illustrates the dual tree of G in Fig. 5.4(a) where the root r of T_r corresponds to the face f_r of G containing vertices u , v and w in the clockwise order. G contains a copy of T_r , which is a spanning tree T' of $G - \{u, v\}$, such that each of the vertices of T_r is mapped to the central vertex of the corresponding face of G as illustrated in Fig. 5.4(c) where the edges of T' are drawn by solid lines.

Our idea is as follows. We first draw the rooted ordered binary dual tree T_r of a label-constrained outerplanar graph G , and then we put the poles of the face f_r corresponding to the root r of T_r , and add each of the edges of G which are not in the drawing of T_r .

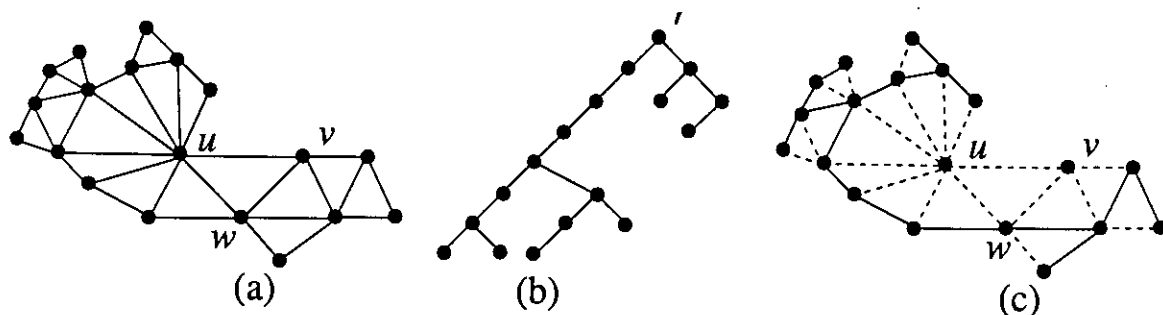


Figure 5.4: (a) A maximal outerplanar graph G , (b) the dual rooted ordered tree T_r of G and (c) a spanning-tree T' of $G - \{u, v\}$ is drawn by the solid lines.

The x -coordinates of the vertices of T_r are assigned in the order of the inorder traversal of T_r in the increasing order starting from 1. The y -coordinate of a vertex of T_r is the label of the vertex minus one. We now put the vertices of T_r at the calculated coordinates and add the required edges to complete the drawing of T_r . Fig. 5.5(c) illustrates a straight-line grid drawing of T_r in Fig. 5.5(b).

We now put the pole vertices of the face of G corresponding to the root of T_r . The left vertex and the right vertex of the face corresponding to the root of T_r are put at $(0, k)$ and at $(n - 1, k)$, respectively where k is the label of the root of T_r . We now add each of the edges of G which are not in the drawing of T_r using straight-line segments, and thus we complete the drawing of G . We call the algorithm described above for drawing an outerplanar graph **Algorithm Draw-Graph**. We now have the following theorem.

Theorem 5.2.2 *Let G be a label-constrained outerplanar graph. Then Algorithm Draw-Graph finds a straight-line grid drawing of G with $O(n \log n)$ area in linear time.*

In the rest of the section, we give a proof of Theorem 5.2.2. We first show that Algorithm **Draw-Graph** produces a straight-line drawing of G . The following lemmas are immediate from the assignment of x -coordinates and y -coordinates of the vertices of T_r .

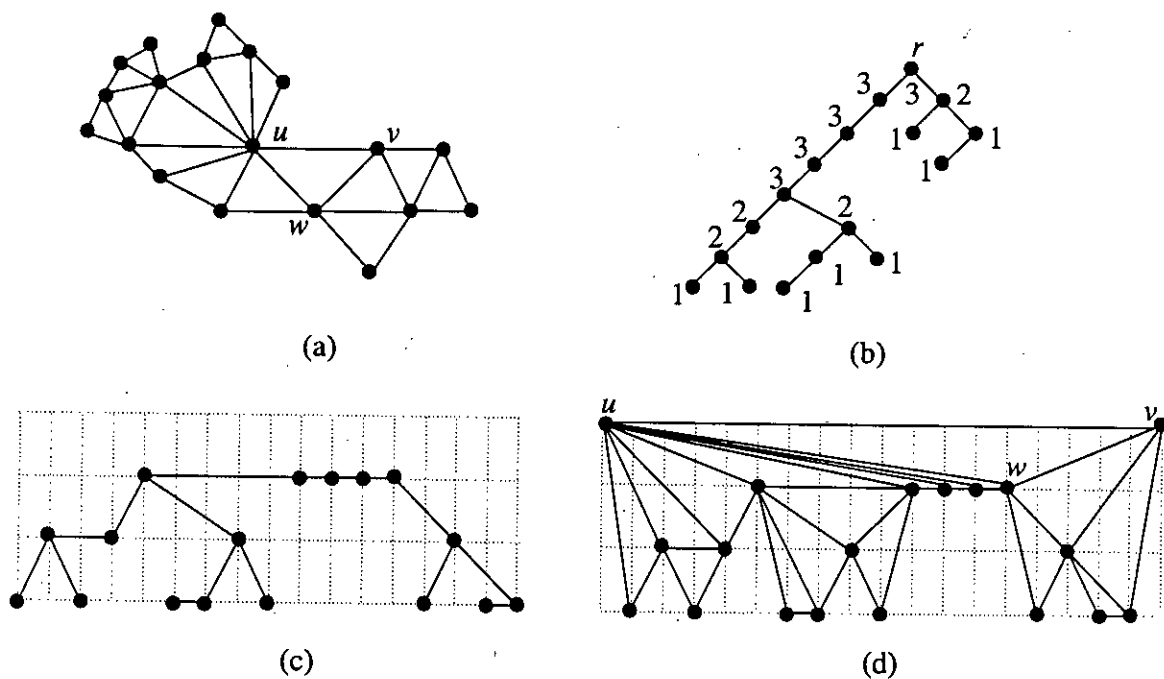


Figure 5.5: (a) A label-constrained outerplanar graph G , (b) the dual rooted ordered tree T_r of G , (c) a straight-line grid drawing of T_r , and (d) a straight-line grid drawing of G .

Lemma 5.2.3 *Let G be a label-constrained outerplanar graph and let T_r be a rooted ordered binary dual tree of G . Let u be a vertex of T_r , and let s and t be the left and the right child of u , respectively. Then the x -coordinate of any vertex of the subtree rooted at s is less than the x -coordinate of any vertex of the subtree rooted at t .*

Lemma 5.2.4 *Let G be a label-constrained outerplanar graph and let T_r be a rooted ordered binary dual tree of G . Let u and v be vertices of T_r where u is an ancestor of v . Then the y -coordinate of u is greater than or equal to the y -coordinate of v .*

We also have the following lemma.

Lemma 5.2.5 *Let G be a maximal outerplanar graph and let T_r be a rooted ordered binary dual tree of G . Let q be any vertex of T_r , and let x and y be the left and the right child of q , respectively. Let f_q , f_x and f_y be the faces of G corresponds to the vertices q , x and y in T_r . Then the left vertex of f_q is the left vertex of f_x and the right vertex of f_q is the right vertex of f_y .*

Proof. Immediate from the definition of the left and the right vertex of a face in G . □

The drawing of T_r is a straight-line grid drawing immediate from the Lemmas 5.2.3 and 5.2.4. We now show that each of the edges of G those are not in T_r can be drawn using straight-line segments without any edge crossings. An edge between the two pole vertices of the face corresponding to the root of T_r can be drawn using a straight-line segment without any crossings with the existing drawing of T_r since each of the pole vertices is placed above all the vertices of T_r . By Lemma 5.2.5, the left vertex of the face corresponding to the root of T_r is the left vertex of the faces corresponding to the vertices of the leftmost path of T_r . Therefore the left vertex of the face corresponding to the root of T_r is adjacent to all the vertices of the leftmost path of T_r in G . These edges can be drawn using straight-line segments without any edge crossings since the left vertex

of the face corresponding to the root of T_r is placed strictly to the left and above of all the vertices on the leftmost path of T_r . Similarly, we can draw the edges between the right vertex of the face corresponding to the root of T_r and the vertices on the rightmost path of T_r using straight-line segments without any edge crossings. In a maximal outerplanar graph, the rest of the edges are between any vertex $v \in T_r$ and a vertex on the left-right or the right-left path of v . From the x -coordinate of any vertex $v \in T_r$ and by Fact 5.1.4, one can see that a vertex $v \in T_r$ is placed strictly to the left (right) and above of all the vertices of the right-left (left-right) path of v except the right (left) child of v . Thus all such edges can be drawn using straight-line segments without any edge crossings and hence the following lemma holds.

Lemma 5.2.6 *Let G be a label-constrained outerplanar graph. Then Algorithm Draw-Graph finds a straight-line grid drawing of G .*

Fig. 5.5(d) illustrates the straight-line grid drawing of G in Fig. 5.5(a).

Proof of Theorem 5.2.2: By Lemma 5.2.6, the drawing of G is a straight-line grid drawing. The height of the drawing of G is the label of the root of the dual tree of G . By Lemma 5.1.2, the height of the drawing of G is $O(\log n)$. The width of the drawing is $O(n)$. Therefore the area of the drawing is $O(n \log n)$. One can easily see that the drawing of G can be found in linear time.

5.3 Recognition of Label-Constrained Outerplanar Graphs

In this section we give a linear-time algorithm for recognition of a label constrained outerplanar graph.

Let G be a maximal outerplanar graph and let T_r be a rooted ordered binary dual tree of G taking a vertex r as the root. (Note that r corresponds to an inner face f_r of G having an edge on the outer face.) From the definition of the vertex labeling of a binary tree in Section 5.1, one can easily see that $L_r(T_r)$ can be done by a bottom-up

computation in linear time. The verification whether $L_r(T_r)$ is a flat labeling can also be done in linear time. In case $L_r(T_r)$ is not a flat labeling, $L_p(T_p)$ might be a flat labeling where T_p is a rooted ordered binary dual tree of G rooted at a vertex p of degree one or two other than r . Therefore to examine whether G is a label-constrained outerplanar graph or not, we have to compute the vertex labeling of the rooted ordered binary dual tree of G rooted at each vertices of degree one or degree two of the dual tree T of G and verify whether each such a labeling of vertices is a flat labeling or not. Hence the recognition of a label-constrained outerplanar graph requires $O(n^2)$ time by a naive approach.

Before presenting our linear-time recognition algorithm, we present the following observation. Fig. 5.6(b) illustrates T_r of a maximal outerplanar graph G in Fig. 5.6(a) where r corresponds to the face f_r of G , and Fig. 5.6(c) illustrates T_p of G in Fig. 5.6(a) where p corresponds to the face f_p of G . Note that the vertex q is the right child of p in T_r in Fig 5.6(b); but it is the left child of p in T_p as illustrated in Fig 5.6(c). Thus we can not get the rooted ordered binary dual tree T_p of G immediately from T_r by simply choosing the vertex p of T_r as the new root without taking care about the ordering of the children of each vertices. However from a close observation of the ordering of the children of a vertex in T_r and T_p , one can see that the ordering of the children is changed only for the vertices on the r - p path of T_r . For the rest of the vertices of T_r , the ordering of the children is unchanged in T_p . This change in the ordering of the children is presented in the following three lemmas which are immediate from the definition of the ordering of the children of a vertex in the rooted ordered binary dual tree.

Lemma 5.3.1 *Let G be a maximal outerplanar graph and let T_r be a rooted ordered binary dual tree of G . Let s be the right (left) child of r in T_r . Let p be a vertex of T other than r such that the degree of p is one or two, and the vertex s is remained as a child of r in T_p . Then s is the left (right) child of r in T_p .*

Lemma 5.3.2 *Let G be a maximal outerplanar graph and let T_r be a rooted ordered binary dual tree of G . Let s be a vertex of T_r other than r such that the degree of vertex s is one*

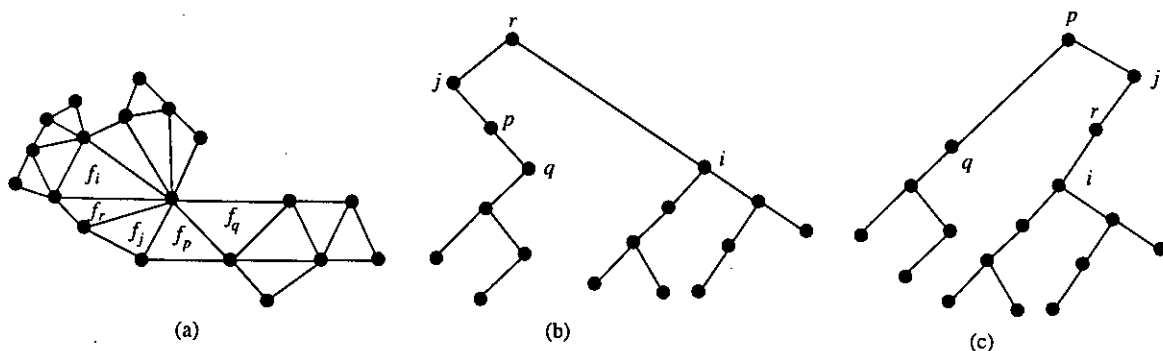


Figure 5.6: (a) A maximal outerplanar graph G , (b) the rooted ordered binary dual tree T_r of G and (c) the rooted ordered binary dual tree T_p of G .

or two, and let t be the right (left) child of s in T_r . Then t is the left (right) child of s in T_s .

Lemma 5.3.3 *Let G be a maximal outerplanar graph and let T_r be a rooted ordered binary dual tree of G . Let x be a degree three vertex such that s is the parent of x , and p and q are the left and the right children of x in T_r . Let y be a descendant of x in the left (right) subtree of x in T_r such that the degree of y is one or two. Then s (p) is the right child of x and q (s) is the left child of x in T_y .*

We now can compute the labeling of a rooted ordered dual tree T_x of a maximal outerplanar graph G from a given labeling of a rooted ordered dual tree T_y of G using the Lemmas 5.3.1, 5.3.2, 5.3.3, where vertex x and y corresponds to the faces f_x and f_y of G , as in the following lemmas.

Lemma 5.3.4 *Let G be a maximal outerplanar graph and let T_r be a rooted ordered binary dual tree of G . Then $L_p(T_p)$ can be computed in constant time if $L_r(T_r)$ is given where p is a child of r in T_r and the degree of p is either one or two.*

Proof. Without loss of generality, we may assume that p is the left child of r in T_r . We also assume that q is the right child of r (if any) in T_r , and p has the right child s .

(if any) in T_r . The labeling of the vertices of the subtrees rooted at q and s is the same in both of the $L_r(T_r)$ and the $L_p(T_p)$ since the labeling is done by bottom-up approach. Therefore we need to compute the label of r and p with respect to p to compute $L_p(T_p)$. By Lemma 5.3.1, q becomes the left child of r in T_p , and the label of r is same as the label of q . A cross path is detected at r in T_p if q has the same label as its right child. By Lemma 5.3.2, s becomes the left child of p in T_p . Then r becomes the right child of p in T_p . The label of p is computed from the label of r and s . A cross path is detected at p in T_p if the label of p is same as the label of r (s) and the label of r (s) is the same as the label of its right (left) child. Thus $L_p(T_p)$ can be computed in constant time. Fig. 5.7 illustrates the computation of $L_p(T_p)$ from $L_r(T_r)$. □

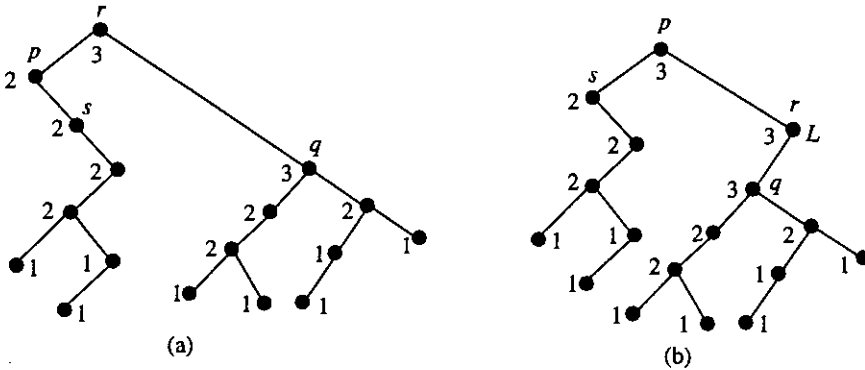


Figure 5.7: (a) $L_r(T_r)$ and (b) $L_p(T_p)$.

Lemma 5.3.5 *Let G be a maximal outer planar graph and let T_r be a rooted ordered binary dual tree of G . Then $L_x(p)$ can be computed in constant time if $L_r(T_r)$ is given where p is a child of r , and x is a descendant of p and the degree of x is either one or two.*

Proof. Without loss of generality, we assume that p is the left child of r in T_r . We also assume that q is the right child of r (if any) in T_r . We need to compute the label of r and p with respect to x to compute $L_x(p)$. By Lemma 5.3.1, q becomes the left child of

r in T_x , and the label of r is same as the label of q . A cross path is detected at r if q has the same label as its right child. We now have two cases to consider.

Case 1: The degree of p is two.

In this case, we assume that s is the child of p in T_r . The label of p would be same as r since r is the only child of p . However, if s is the left child of p in T_r , then r is the right child of p in T_x and there is a cross path at p as illustrated by thick line in Fig. 5.8(i)(b). Otherwise, r is the left child of p in T_x and there is no cross path at p as illustrated in Fig. 5.8(i)(d). Then $L_x(p)$ can be computed from the labeling of r .

Case 2: The degree of p is three.

In this case, we assume that s and t are the left and the right child of p in T_r . If x is in the left subtree of p in T_r , then by Lemma 5.3.3, r is the right child and t is the left child of p in T_x as illustrated in Fig. 5.8(ii)(b). Fig. 5.8(ii)(d) illustrates the case where x is in the right subtree of p in T_r . Then $L_x(p)$ can be computed from the labeling of r and t (s).

Thus we can compute $L_x(p)$ in constant time. Furthermore, one can determine in constant time whether there is a cross path at p or not. □

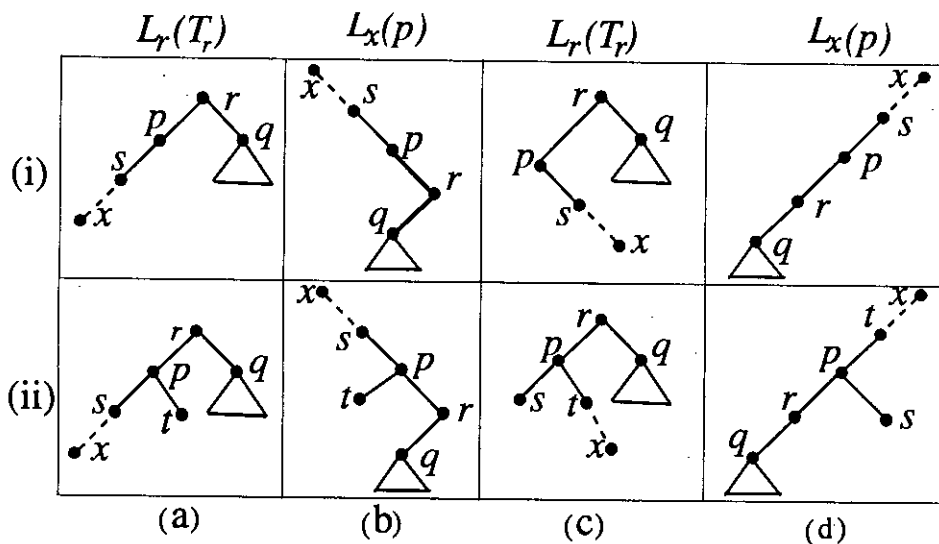


Figure 5.8: Illustration of different cases of computing $L_x(p)$ from given $L_r(T_r)$.

We are now ready to present our algorithm. Let G be a maximal outerplanar graph and let T_r be a rooted ordered binary dual tree of G . We first compute $L_r(T_r)$. We have done if $L_r(T_r)$ is a flat labeling. Then G is a label-constrained outerplanar graph. We thus assume that $L_r(T_r)$ is not a flat labeling. We now have the following lemma.

Lemma 5.3.6 *Let G be a maximal outerplanar graph and let T_r be a rooted ordered binary dual tree of G . If $L_r(T_r)$ is not a flat labeling and there exists a vertex $x \in T_r$ such that both of the subtrees of x contains a cross path, then G is not a label-constrained outerplanar graph.*

Proof. Consider first the case where x is the root of T_r . Then we can have another rooted ordered dual tree T_q of G such that q corresponds to an inner face f_q of G . (Note that the degree of q is either one or two.) The vertex q is either from the left or from the right subtree of r in T_r . Without loss of generality, we may assume that q is in the left subtree of r . Then $L_q(T_q)$ is not a flat labeling since the labeling of vertices as well as the cross path of the right subtree of r in T_r remain unchanged in T_q . Consider next the case x is an internal vertex in T_r and the degree of x is three. Then for any vertex v of T_r which is not a descendant of x in T_r , $L_v(T_v)$ can not be a flat labeling since labeling is done by a bottom-up computation. We can prove that there exists no descendant s of x in T_r such that $L_s(T_s)$ is a flat labeling using the same reasoning as we have used to prove that $L_q(T_q)$ is not a flat labeling. Therefore G is not a label-constrained outerplanar graph. \square

We thus assume that $L_r(T_r)$ is not a flat labeling and there exists no vertex $x \in T_r$ such that both the subtrees of x contain a cross path. In this case, each of the vertices of T_r at which a cross path is detected lies on a single path and r is an end vertex of this path. Let us assume that u is the other end vertex of this path, i.e., u is the farthest vertex from r at which a cross path is detected. Then for any vertex v of T_r which is neither u nor a descendant of u in T_r , $L_v(T_v)$ can not be a flat labeling since labeling is

done in a bottom-up approach. We thus concentrate our attention only on the vertices of the subtree rooted at u in T_r as a probable new root for computing labeling. We now have two cases to consider.

Case 1: u is not the root in T_r .

In this case, we have two subcases to consider.

Subcase 1(a): The degree of u is two.

By Lemmas 5.3.4 and 5.3.5, we can compute $L_u(T_u)$ in the $O(l)$ time where l is the length of the r - u path. If a cross path is detected at any ancestor x of u in T_r during this step then for any descendant y of x in T_r , $L_y(T_y)$ can not be a flat labeling. Since u and all the descendants of u are also descendants of x in T_r , for any vertex v in T_r , $L_v(T_v)$ can not be a flat labeling. Hence G is not label-constrained. Thus we assume that no cross path is detected at any ancestor x of u . We have done if $L_u(T_u)$ is a flat labeling. Otherwise, a cross path is detected at u . In this case, we have to compute $L_x(T_x)$ for any descendant x of u in T_r with degree one or two and verify whether $L_x(T_x)$ is a flat labeling for recognizing G a label-constrained outerplanar graph, and this can be done in linear time by Lemma 5.3.4 and Lemma 5.3.5.

Subcase 1(b): The degree of u is three.

Let v be any vertex on the r - u path other than u then for all the descendants x of u in T_r the value of $L_x(v)$ is the same. We can compute $L_x(v)$ for all such vertices v on the r - u path other than u in $O(l)$ time where l is the length of the r - u path and x is any degree one or two descendant of u . Again, for any ancestor y of u , if a cross path is detected at y , then G can not be a label-constrained outerplanar graph. Otherwise, we have to compute $L_x(T_x)$ for any descendant x of u in T_r with degree one or two and verify whether $L_x(T_x)$ is a flat labeling for recognizing G a label-constrained outerplanar graph, and this can be done in linear time by Lemma 5.3.4 and Lemma 5.3.5.

Case 2: u is the root in T_r .

In this case, we have to compute $L_x(T_x)$ for any descendant x of r in T_r with de-

gree one or two and verify whether $L_x(T_x)$ is a flat labeling for recognizing G a label-constrained outerplanar graph, and this can also be done in linear time by Lemma 5.3.4 and Lemma 5.3.5.

Thus one can recognize a label-constrained outerplanar graph in linear time. We have the following theorem.

Theorem 5.3.7 *Let G be a maximal outerplanar graph and let T be the dual tree of G . Then one can decide in linear time whether there exists a vertex r of T such that $L_r(T_r)$ is a flat labeling where T_r is the rooted ordered binary dual tree of G . Furthermore, one can find such a vertex r of T in linear time if such a vertex exists.*

5.4 Conclusion

In this chapter we introduced a subclass of outerplanar graphs, which we call label-constrained outerplanar graphs. A graph in this class has a straight-line grid drawing on a grid of $O(n \log n)$ area, and the drawing can be found in linear time. We gave an algorithm to recognize a label-constrained outerplanar graph in linear time. Our drawing algorithm is based on a very simple and natural labeling of a tree. The labeling is bounded by $O(\log n)$, and the labeling might be adopted for solving some other tree-related problems.

The previously best known area bound for an outerplanar graph is $O(dn \log n)$ due to [Fra07], where d is the maximum degree of the outerplanar graph G . This immediately gives an $O(n \log n)$ area bound if the maximum degree of G is bounded by a constant. But the maximum degree of an outerplanar graph is not always bounded by a constant. A trivial outerplanar graph may have the maximum degree $n - 1$ although it requires $O(n)$ area for straight-line grid drawing as illustrated in Fig. 5.9. However, there are an infinite number of outerplanar graphs, as one illustrated in Fig. 5.10 which have the maximum degree $O(n^{0.5})$. A straight-line grid drawing of such a graph in this class obtained by the algorithm of Frati [Fra07] requires $O(n^{1.5} \log n)$ area whereas our algorithm produces the

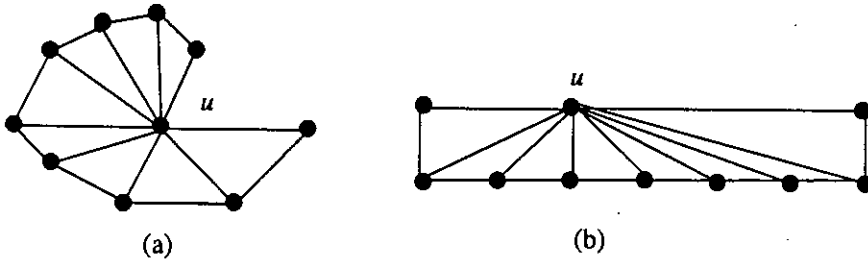


Figure 5.9: (a) A trivial outerplanar graph G with maximum degree $n - 1$ and (b) a straight-line grid drawing of G with $O(n)$ area.

drawing with $O(n \log n)$ area.

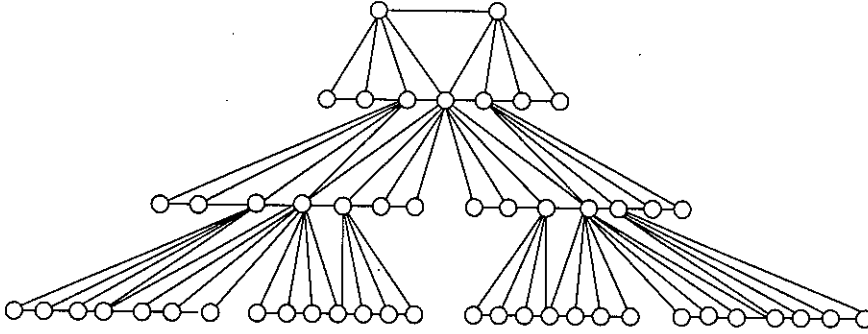


Figure 5.10: An example of an outerplanar graph G which has maximum degree $O(n^{0.5})$.

Chapter 6

Partitioning of Doughnut Graphs

In this chapter we give a linear-time algorithm for finding a k -partition of a doughnut graph G . Our algorithm is based on finding the Hamiltonian path between any pair of vertices of a doughnut graph. We exploit the simple structure of a doughnut graph for finding Hamiltonian path between any pair of vertices of a doughnut graph. Our algorithm is also applicable for finding k -partition of a doughnut graph with basis two.

This chapter is organized as follows. Section 6.1 presents some definitions and preliminary results. In Section 6.2, we give an algorithm for finding a Hamiltonian path between any pair of vertices of a doughnut graph. Section 6.3 provides a linear-time algorithm for finding k -partition of a doughnut graph. Finally Section 6.4 concludes the chapter.

6.1 Preliminaries

In this section we give some definitions and preliminary results. Let $G = (V, E)$ be a connected simple planar graph with vertex set V and edge set E . A *path* in G is an ordered list of distinct vertices $v_1, v_2, \dots, v_q \in V$ such that $(v_{i-1}, v_i) \in E$ for all $2 \leq i \leq q$. Let $P_1 = x_1, \dots, x_k$ and $P_2 = x_m, \dots, x_o$ be two paths. We denote by P_1P_2 the concatenation of two paths P_1 and P_2 where the last vertex of P_1 and the first vertex of P_2 are adjacent,

i.e, $P_1P_2 = x_i, \dots, x_k, x_m, \dots, x_o$ where x_m is a neighbor of x_k .

Let Γ be a doughnut embedding of a doughnut graph G . Let z_i be a vertex on C_2 such that z_i has two neighbors on C_1 . Let x and x' be the two neighbors of z_i on C_1 such that x' is the counter clockwise next vertex to x on C_1 . We call x the *left neighbor* of z_i on C_1 and x' the *right neighbor* of z_i on C_1 . Similarly we define the left neighbor and the right neighbor of z_i on C_3 if a vertex z_i on C_2 has two neighbors on C_3 . Then the cycle C_1 contains p vertices, cycle C_2 contains $2p$ vertices and cycle C_3 contains p vertices. Let z_1, z_2, \dots, z_{2p} be the vertices of C_2 in counter clockwise order such that z_1 has exactly one neighbor on C_1 . Let x_1 be the neighbor of z_1 on C_1 , and let x_1, x_2, \dots, x_p be the vertices of C_1 in the counter clockwise order. Let y_1, y_2, \dots, y_p be the vertices on C_3 in counter clockwise order such that y_1 and y_p are the right neighbor and the left neighbor of z_1 , respectively. Fig. 6.1(b) illustrates the labeling as mentioned above. In the rest of the thesis, we consider a doughnut embedding Γ such that the vertices of cycles C_1, C_2 and C_3 are labeled as mentioned above where x_1, z_1 and y_1 are the leftmost vertex on C_1, C_2 and C_3 , respectively.

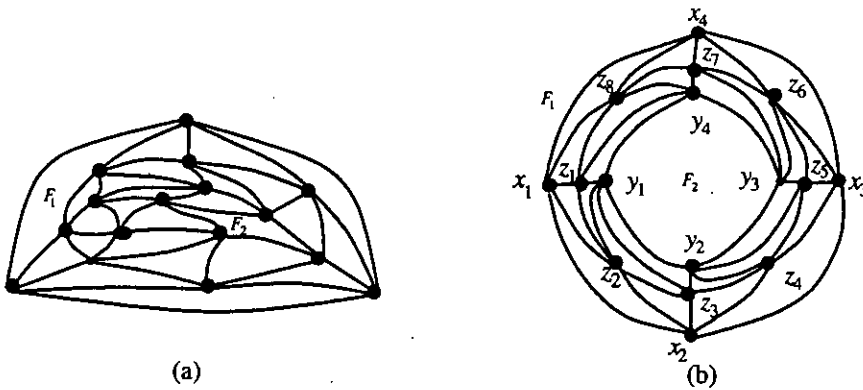


Figure 6.1: (a) G is a p -doughnut graph where $p = 4$ and (b) doughnut embedding of G .

We now have the following lemmas.

Lemma 6.1.1 *Let G be a p -doughnut graph and let Γ be a doughnut embedding of G . Let z_i be a vertex of C_2 . Then the following conditions hold.*

- (a) z_i has exactly two neighbors on C_1 and exactly one neighbor on C_3 if i is even. The neighbors of z_i on C_1 are x_p and x_1 if $i = 2p$ otherwise $x_{i/2}$ and $x_{i/2+1}$ in a counter clockwise order, and the neighbor of z_i on C_3 is y_p if $i = 2p$ otherwise $y_{i/2}$.
- (b) z_i has exactly two neighbors on C_3 and exactly one neighbor on C_1 if i is odd. The neighbors of z_i on C_3 are y_1 and y_p if $i = 1$ otherwise $y_{\lfloor i/2 \rfloor - 1}$ and $y_{\lfloor i/2 \rfloor}$ in a counter clockwise order, and the neighbor of z_i on C_1 is $x_{\lfloor i/2 \rfloor}$.

Lemma 6.1.2 *Let G be a p -doughnut graph and let Γ be a doughnut embedding of G . Let x_i be a vertex of C_1 . Then x_i has exactly three neighbors z_{2p}, z_1, z_2 if $i = 1$ otherwise $z_{2i-2}, z_{2i-1}, z_{2i}$ on C_2 in a counter clockwise order.*

Lemma 6.1.3 *Let G be a p -doughnut graph and let Γ be a doughnut embedding of G . Let y_i be a vertex of C_3 . Then y_i has exactly three neighbors z_{2p-1}, z_{2p}, z_1 if $i = p$ otherwise $z_{2i-1}, z_{2i}, z_{2i+1}$ on C_2 in a counter clockwise order.*

A *Hamiltonian cycle (path)* of a graph G is a cycle (path) which contains all the vertices of G . We call a graph G *Hamiltonian* if G contains a Hamiltonian cycle. The Hamiltonian cycle problem asks whether a given graph contains a Hamiltonian cycle, and the problem is NP-complete even for 3-connected planar graphs [GJT76]. However the problem becomes polynomial-time solvable for 4-connected planar graphs: Tutte proved that a 4-connected planar graph necessarily contains a Hamiltonian cycle [Tut56]. We call a graph G is *Hamiltonian-connected* if G has a Hamiltonian path between any pair of vertices of G . Thomassen proved that 4-connected planar graphs are Hamiltonian-connected [Tho83].

6.2 Finding Hamiltonian Path in Doughnut Graphs

A doughnut graph G is Hamiltonian-connected since G is 5-connected. One can find a Hamiltonian path in a doughnut graph using algorithm proposed by Chiba and Nishizeki

[CN89]. In their paper, they gave a proof of Tutte's theorem based on Thomassen's short proof avoiding decomposition of a 4-connected planar graph into nondisjoint subgraphs. Their proof is constructive and yields an algorithm for finding Hamiltonian path. Their algorithm clearly runs in $O(n^2)$ time, since one step of divide-and-conquer can be done in $O(n)$. The key idea for linear implementation of this algorithm is to use, in place of the Hopcroft and Tarjan's algorithm [HT73], a new algorithm to decompose a plane graph into small subgraphs by traversing some facial cycles. Although a sophisticated analysis shows that each of the edge is traversed at most constant time during one execution of Hamiltonian path finding algorithm and hence the algorithm runs in linear time, the linear-time implementation of the algorithm looks non-trivial. In this section we present a very simple linear-time algorithm for finding Hamiltonian path between any pair of vertices of a doughnut graph. In our algorithm we exploit the simple structure of a doughnut graph.

We have the following theorem on a doughnut graph.

Theorem 6.2.1 *Let G be a doughnut graph. Then a Hamiltonian path between any pair of vertices of G can be found in linear time.*

Proof. We first show a Hamiltonian path between any pair of vertices of a doughnut graph. Let Γ be a doughnut embedding of G and let C_1, C_2 and C_3 be the outer cycle, the middle cycle and the inner cycle of Γ . We have the following four cases to consider.

Case 1: Both the vertices u, v are either on C_1 or on C_3 .

We assume that both of u and v are on C_1 , since the case where both of u and v are on C_3 is similar. Let $u = x_i$ and $v = x_j$. Without loss of generality, we assume that $i < j$. We take the following paths. (i) $P_1 = x_i, z_{2i-1}, z_{2i}, x_{i+1}, z_{2i+1}, z_{2i+2}, \dots, x_{j-1}, z_{2j-3}, z_{2j-2}$; (ii) $P_2 = y_{j-1}, y_{j-2}, \dots, y_j$; (iii) $P_3 = z_{2j-1}, z_{2j}, \dots, z_{2i-2}$; and (iv) $P_4 = x_{i-1}, x_{i-2}, \dots, x_j$. The path P_1 contains vertices of C_1 and C_2 . By Lemma 6.1.2, z_{2i-1} is a neighbor of x_i . By Lemma 6.1.1, x_{i+1} is a neighbor of z_{2i} since $2i$ is even. The path P_2 contains

all the vertices of C_3 . The path P_3 contains all the vertices of C_2 those are not appear in P_1 and the Path P_4 contains vertices of C_1 those are not appear in the path P_1 . We can concatenate the paths P_1 and P_2 since y_{j-1} is a neighbor of z_{2j-2} by Lemma 6.1.1. The paths P_2 and P_3 can be concatenated since z_{2j-1} is a neighbor of y_j by Lemma 6.1.3. The paths P_3 and P_4 can also be concatenated since x_{i-1} is a neighbor of z_{2i-2} by Lemma 6.1.1. Thus we can concatenate the four paths and the resulting path is HP_{x_i, x_j} where $HP_{x_i, x_j} = P_1 P_2 P_3 P_4$. The path HP_{x_i, x_j} is a Hamiltonian path since P_1, P_2, P_3 and P_4 contain all the vertices of G . Fig. 6.2 illustrates the case where $u = x_2$ and $v = x_5$. In this example (i) $P_1 = x_2, z_3, z_4, x_3, z_5, z_6, x_4, z_7, z_8$; (ii) $P_2 = y_4, y_3, y_2, y_1, y_5$; (iii) $P_3 = z_9, z_{10}, z_1, z_2$; and (iv) $P_4 = x_1, x_5$. The Hamiltonian path is $HP_{x_2, x_5} = P_1 P_2 P_3 P_4$.

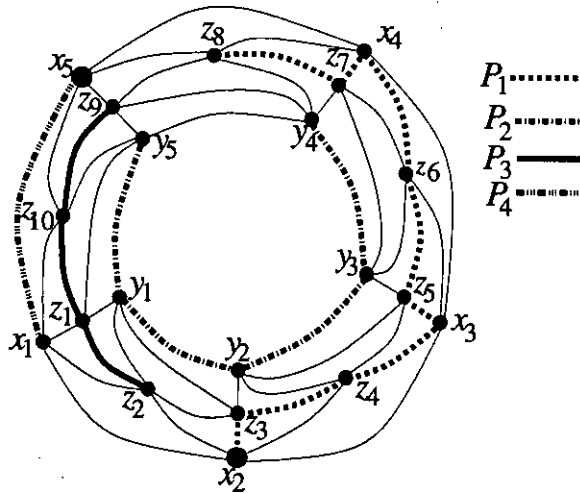


Figure 6.2: Illustration for Hamiltonian path between two vertices on cycle C_1 .

Case 2: Both vertices u, v are on C_2 .

Let $u = z_i$ and $v = z_j$. Without loss of generality we may assume that $i < j$. We have two subcases to consider.

Subcase 2(a): i is odd.

We take the following paths. (i) $P_1 = z_i, x_{[i/2]}, x_{[i/2]+1}, \dots, x_{[i/2]-1}$; (ii) $P_2 = z_{i-1}, y_{(i-1)/2}, z_{i-2}, z_{i-3}, y_{(i-3)/2}, \dots, y_{(j+1)/2}$ if j is odd, otherwise $P_2 = z_{i-1}, y_{(i-1)/2}, z_{i-2}, z_{i-3}, y_{(i-3)/2}, \dots, z_{j+1}$; (iii) $P_3 = y_{(j+1)/2} - 1, y_{(j+1)/2-2}, \dots, y_{[i/2]}$ if j is odd, otherwise $P_3 =$

$y_{[(j+1)/2]-1}, y_{[(j+1)/2]-2}, \dots, y_{[i/2]}$; and (iv) $P_4 = z_{i+1}, z_{i+2}, \dots, z_j$. By Lemmas 6.1.1, 6.1.2, 6.1.3, we can prove the adjacency between two consecutive vertices of each path. We can also prove the adjacency between the end vertex and the starting vertex of paths P_1 and P_2 , P_2 and P_3 , P_3 and P_4 using the Lemmas 6.1.1, 6.1.2, 6.1.3. Therefore we can concatenate the paths P_1, P_2, P_3, P_4 ; and the resulting path $HP_{x_i, x_j} = P_1 P_2 P_3 P_4$ is a Hamiltonian path since the paths P_1, P_2, P_3, P_4 contain all the vertices of the graph. Fig. 6.3 illustrates the case where $u = z_3$ and $v = z_8$. In this example (i) $P_1 = z_3, x_2, x_3, x_4, x_5, x_1$; (ii) $P_2 = z_2, y_1, z_1, z_{10}, y_5, z_9$; (iii) $P_3 = y_4, y_3, y_2$; and (iv) $P_4 = z_4, z_5, z_6, z_8$. The Hamiltonian path is $HP_{z_3, z_8} = P_1 P_2 P_3 P_4$.

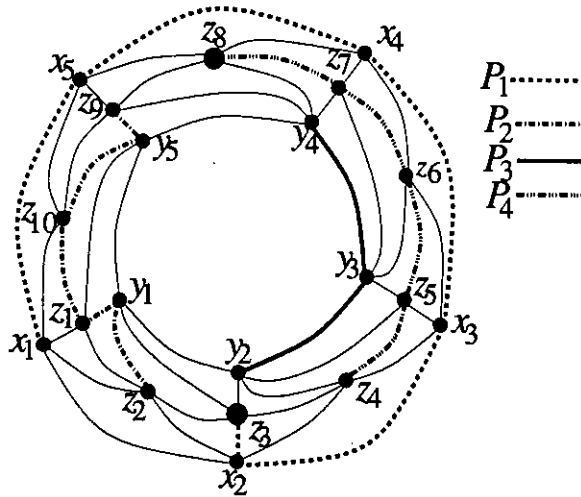


Figure 6.3: Illustration for Hamiltonian path between two vertices on C_2 where one of the vertex index is odd.

Subcase 2(b): i is even.

We take the following paths. (i) $P_1 = z_i, x_{i/2+1}, x_{i/2+2}, \dots, x_{i/2}$; (ii) $P_2 = z_{i-1}, z_{i-2}, y_{(i-2)/2}, z_{i-3}, \dots, z_{j+1}$; (iii) $P_3 = y_{(j+1)/2}, y_{(j+1)/2-1}, \dots, y_{[i/2]}$ and (iv) $P_4 = z_{i+1}, z_{i+2}, \dots, z_j$. Using the same arguments as in Subcase 2(a), we can prove that $HP_{x_i, x_j} = P_1 P_2 P_3 P_4$ is a Hamiltonian path. Fig. 6.4 illustrates the case where $u = z_2$ and $v = z_8$. In this example (i) $P_1 = z_2, x_2, x_3, x_4, x_5, x_1$; (ii) $P_2 = z_1, z_{10}, y_5, z_9$; (iii) $P_3 = y_4, y_3, y_2, y_1$; and (iv) $P_4 = z_3, z_4,$

z_5, z_6, z_7, z_8 . The Hamiltonian path is $HP_{z_2, z_8} = P_1 P_2 P_3 P_4$.

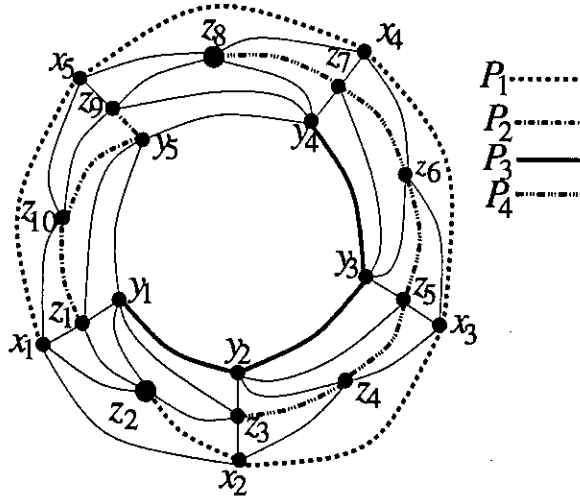


Figure 6.4: Illustration for Hamiltonian path between two vertices on C_2 where one of the vertex index is even.

Case 3: The vertex u either on C_1 or on C_3 and v on C_2 .

We assume that u is on C_1 and v is on C_2 , since the case where u is on C_3 and v is on C_2 is similar. Let $u = x_i$ and $v = z_j$.

We take the following paths. (i) $P_1 = x_i, x_{i+1}, \dots, x_{i-1}$; (ii) $P_2 = z_{2i-3}, z_{2i-2}, y_{i-1}, z_{2i-1}, \dots, z_{j-1}$ if j is even, otherwise $P_2 = z_{2i-3}, z_{2i-2}, y_{i-1}, z_{2i-1}, \dots, z_{(j-1)/2}$; (iii) $P_3 = y_{j/2}, y_{j/2+1}, \dots, y_{i-2}$ if j is even, otherwise $P_3 = y_{(j-1)/2+1}, y_{(j-1)/2+2}, \dots, y_{i-2}$; and (iv) $P_4 = z_{2i-4}, z_{2i-5}, \dots, z_j$. Using the same arguments as in Subcase 2(a), we can prove that $HP_{x_i, z_j} = P_1 P_2 P_3 P_4$ is a Hamiltonian path. Fig. 6.5 illustrates the case where $u = x_5$ and $v = z_2$. In this example (i) $P_1 = x_5, x_1, x_2, x_3, x_4$; (ii) $P_2 = z_7, z_8, y_4, z_9, z_{10}, y_5, z_1$; (iii) $P_3 = y_1, y_2, y_3$; and (iv) $P_4 = z_6, z_5, z_4, z_3, z_2$. The Hamiltonian path is $HP_{x_5, z_2} = P_1 P_2 P_3 P_4$.

Case 4: The vertex u on C_1 and v on C_3 .

We assume that u is on C_1 and v is on C_3 since the case where u is on C_3 and v is on C_1 is similar. Let us assume that $u = x_i$ and $v = y_j$. We take the following paths. (i) $P_1 = x_i, x_{i+1}, \dots, x_{i-1}$; (ii) $P_2 = z_{2i-3}, y_{\lfloor (2i-3)/2 \rfloor}, z_{2i-4}, z_{2i-5}, y_{\lfloor (2i-5)/2 \rfloor}, \dots, z_{2j-1}$; (iii) P_3

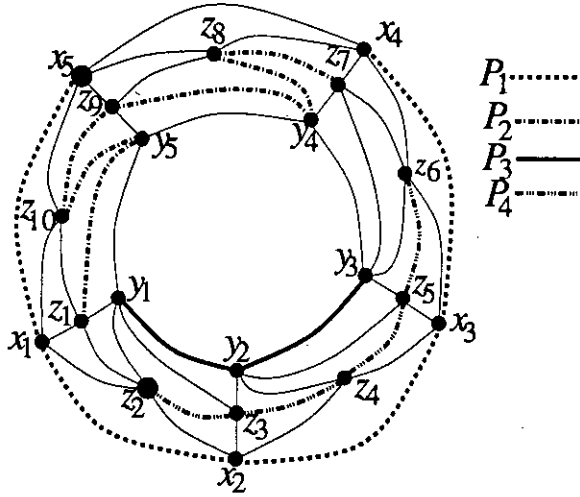


Figure 6.5: Illustration for Hamiltonian path between a vertex on C_1 and a vertex on C_2 .

$= z_{2j}, z_{2j+1}, \dots, z_{2i-2}$; and (iv) $P_4 = y_{i-1}, y_{i-2}, \dots, y_j$. Using the same arguments as in Subcase 2(a), we can prove that $HP_{x_i, x_j} = P_1 P_2 P_3 P_4$ is a Hamiltonian path. Fig. 6.6 illustrates the case where $u = x_5$ and $v = y_2$. In this example (i) $P_1 = x_5, x_1, x_2, x_3, x_4$; (ii) $P_2 = z_7, y_4, z_8, z_9, y_5, z_{10}, z_1, y_1, z_2, z_3$; (iii) $P_3 = z_4, z_5, z_6$; and (iv) $P_4 = y_3, y_2$. The Hamiltonian path is $HP_{x_5, y_2} = P_1 P_2 P_3 P_4$.

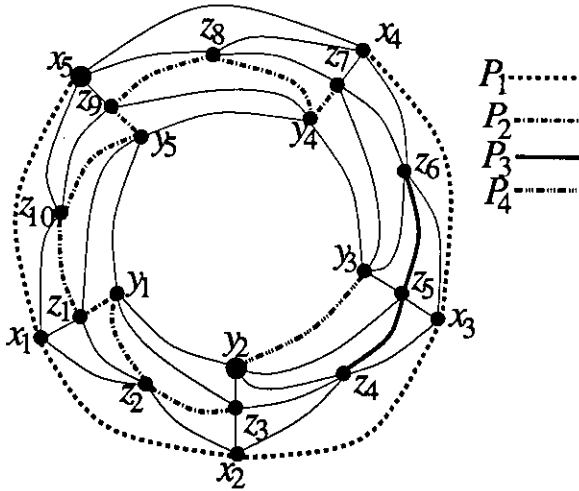


Figure 6.6: Illustration for Hamiltonian path between a vertex on C_1 and a vertex on C_3 .

Therefore G has a Hamiltonian path between any pair of vertices. One can find such

a path in linear time easily. □

6.3 k -Partition of a Doughnut Graph

Given a graph $G = (V, E)$, k natural numbers n_1, n_2, \dots, n_k such that $\sum_{i=1}^k n_i = |V|$, we wish to find a k -partition V_1, V_2, \dots, V_k of the vertex set V such that $|V_i| = n_i$ and V_i induces a connected subgraph of G for each $i, 1 \leq i \leq k$. A k -partition of a graph G is illustrated in Fig. 6.7 for $k = 5$ where the edges of five connected subgraphs are drawn by solid lines, and the remaining edges of G are drawn by dotted lines. Let $B = u_1, u_2, \dots, u_m$

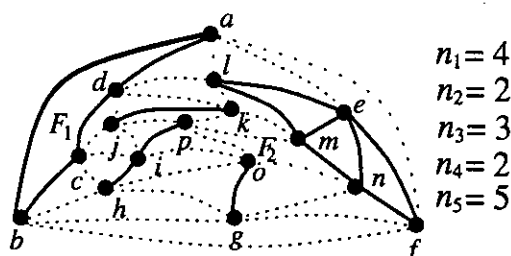


Figure 6.7: A 5-partition of a 5-connected planar graph G .

be a sequence of distinct vertices of G with $m \leq k$. A k -partition of G with basis B is a k -partition with the additional restriction that $u_i \in V_i$, for $1 \leq i \leq m$. A k -partition of a graph G with basis m is illustrated in Fig. 6.8 for $k = 5$ and $m = 5$.

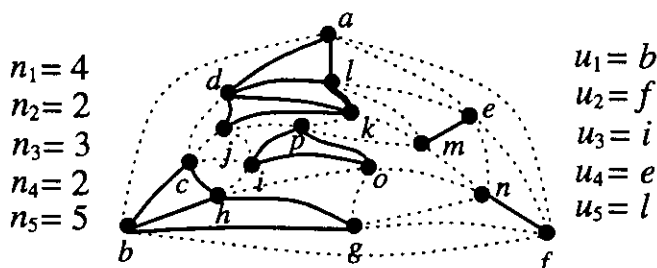


Figure 6.8: A 5-partition of a 5-connected planar graph G with basis 5.

The problem of finding a k -partition of a given graph often appears in the load distribution among different power plants and the fault-tolerant routing of communication net-

works [NRN97, NN01]. The problem is NP-hard in general even k is limited to 2 [DF85], and hence it is very unlikely that there is a polynomial-time algorithm to solve the problem. Although not every graph has a k -partition, Györi and Lovász independently proved that every k -connected graph has a k -partition [Gyr78, Lov77]. However, their proofs do not yield any polynomial-time algorithm for finding a k -partition of a k -connected graph.

For the case of $k = 2, 3, 4$ and 5 , the following algorithms have been known:

- (i) a linear-time algorithm to find a bipartition of a biconnected graph [STN90, STNMU90];
- (ii) an algorithm to find a tripartition of a triconnected graph in $O(n^2)$ time, where n is the number of vertices of a graph [STNMU90];
- (iii) a linear-time algorithm to find a tripartition of a triconnected planar graph [JSN94];
- (iv) a linear-time algorithm to find a 4-partition of a 4-connected plane graph G when four basis vertices u_1, u_2, u_3, u_4 are located on the same face of G [NRN97]; and
- (iv) a linear-time algorithm to find a 5-partition of a 5-connected internally triangulated plane graph G when five basis vertices u_1, u_2, u_3, u_4, u_5 are located on the same face of G [NN01];

A p -doughnut graph is a 5-connected planar graph. One may think that a p -doughnut graph G where $p > 4$ can be partitioned using Nagai and Nakano's [NN01] algorithm after triangulation of one of the face of G with p -vertices. But it is not possible since after removing the dummy edges used for triangulation the partition may not be connected. In this section, we give an algorithm for finding k -partition of a doughnut graph. We have the following theorem.

Theorem 6.3.1 *Let G be a doughnut graph. Then G admits k -partitioning where $k \leq n$. Furthermore, one can find such a partition in linear time.*

Proof. By Theorem 6.2.1, G has a Hamiltonian path between any pair of vertices. We first find a Hamiltonian path $HP_{u,v}$ between any pair of vertices u and v of G . Then starting from one end vertex of $HP_{u,v}$, we divide the path into k subpaths where each subpath contains the number of vertices exactly equal to the natural number associated with the corresponding partition. Each of the partition is a subgraph induced by the vertices of the corresponding subpaths. Fig. 6.9 illustrates k -partitioning of G . Fig. 6.9(a) illustrates a Hamiltonian path of G between vertices x_2 and z_6 . Fig. 6.9(b) illustrates k -partition of G for $k = 7$ where the natural numbers are 3, 2, 5, 3, 2, 4, 1, respectively. Fig. 6.9(c) illustrates k -partition of G for $k=4$ where the natural numbers are 4, 6, 3, 7, respectively. The edges of Hamiltonian path and the connected subgraphs are drawn by thick lines, and the remaining edges are drawn by thin lines. One can find a Hamiltonian path by Theorem 6.2.1 in linear time and a subgraph induced by the vertices on a subpath can also be obtained in linear time.

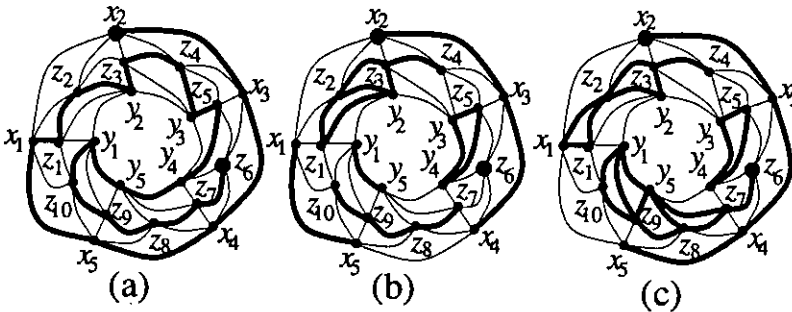


Figure 6.9: (a) Hamiltonian path HP_{x_2,z_6} of G , (b) a 7-partition of G and (c) a 4-partition of G .

□

Our k -partition algorithm is based on finding a Hamiltonian path between any pair of vertices of a doughnut graph. The two end vertices of a Hamiltonian path can be used as two basis vertices of a k -partition. So, the following theorem also holds.

Theorem 6.3.2 *Let G be a doughnut graph. Then k -partitioning of G with basis two can*

be found in linear time.

By using the Chiba and Nishizeki's [CN89] algorithm, we now have the following result for any 4-connected planar graph.

Theorem 6.3.3 *Let G be a 4-connected planar graph. Then G admits k -partitioning with the basis at most two.*

6.4 Conclusion

In this chapter, we gave a linear-time algorithm for finding a k -partition of a doughnut graph. A doughnut graph G is a fault tolerant graph since the vertices of G lies on three vertex disjoint cycles and G is 5-regular. Therefore k -partitioning of G is interesting. We can also have a k -partition for a 4-connected planar graph using the same method.

Chapter 7

Doughnut Graphs for Interconnection Networks

An interconnection network provides communication among processors in parallel processing and distributed systems. An *interconnection network* is usually modeled as an undirected graph G where each nodes of G corresponds to a processor and an edge corresponds to the communication channel between the two processors corresponding to the end vertices of the edge. Thus, the terms *graph* and *network*, *node* and *processor*, and *edge* and *channel* are used interchangeably throughout this chapter. In parallel processing or distributed systems, the design of interconnection networks is an important issue and many networks have been proposed [AHK87, AK89, CFF00]. The performance of such systems significantly depends on the choice of network topology. Several parameters like connectivity, degree, diameter, symmetry and fault tolerance for building interconnection networks are discussed in [Xu01]. A class of Cayley graphs based on permutation groups has proven to be suitable for designing interconnection networks. One of the most efficient and widely used interconnection network is hypercubes (which are also Cayley graphs) [BA84]. These graphs are regular and symmetric but the degree of vertices changes as the size of the graph is increased. Therefore using these graphs is prohibitive in

network with a large number of nodes [CAB93]. There are applications where the computing nodes in an interconnection network only have fixed number of I/O ports. Moreover, fixed degree networks are important from the viewpoint of VLSI implementation [SP89].

In this chapter, we propose a new family of graphs called *doughnut graphs* for building interconnection networks. A graph of this class is regular and has maximal fault tolerance, smaller diameter and recursive structure. Moreover, the graphs of this class admits linear area drawing. The proposed family of graphs offers a better alternative for VLSI implementation in terms of regularity, greater fault tolerance and area efficiency. We also give an efficient routing scheme in a doughnut graph.

This chapter is organized as follows. In Section 7.1, we give some parameters for performance evaluation of an interconnection networks. Section 7.2 provides a simple routing scheme. A recursive structure of a doughnut graph is presented in Section 7.3. Section 7.4 summarizes the topological properties of doughnut graphs. Finally Section 7.5 concludes the chapter.

7.1 Parameters of Interconnection Networks

The following parameters [Xu01] are used to compare different interconnection networks.

Number of vertices The number of vertices represent the number of processor.

Diameter In a network, information needs to travel from one processor to another. It will take d steps to communicate between two processors if the distance between two processor is d . Naturally we are interested in the maximum distance between any pair of processors. We call this maximum distance the *network diameter*. Network with smaller diameter is preferable.

Vertex degree The vertex degree is a very important factor. There are applications where the computing nodes in interconnection network only have a fixed number of

I/O ports. Moreover, it is important that the degree of each vertex is fixed from the view point of VLSI implementation.

Connectivity The fault tolerance of a vertex of an undirected graph G is measured by the vertex connectivity of G . A graph G is said to have a vertex connectivity k if the graph G remains connected when an arbitrary set of less than k vertices are removed.

Fault tolerance A graph G is said to have *maximal fault tolerance* if its vertex connectivity equals the minimum degree of G . Any degree regular graph is a maximal fault tolerant graph.

Embedding It is another important factor. We can apply the algorithm of simple network to a richer one if the richer one contain simple one as a subgraph. For example, array of any size and dimension can be embedded in hypercubes. Therefore any algorithm design for simple array is also applicable for hypercube.

Recursive Structure A network has *recursive structure* if every instance of it can be created by connecting smaller instances of the same network. For example, a network with n vertices is created by connecting in some fixed way four networks of $n/4$ vertices such that each of the network has a recursive structure. This property often makes scalable computers, an important feature of successful machines.

7.2 A Simple Routing Scheme

In this section, we present a simple routing scheme of a doughnut graph for interconnection network. We have the following lemmas.

Lemma 7.2.1 *Let G be a p -doughnut graph and let Γ be a doughnut embedding of G . Let C_1 , C_2 and C_3 be the three vertex disjoint cycles of Γ such that C_1 is the outer cycle, C_2*

is the middle cycle and C_3 is the inner cycle. Then the the shortest path between any two vertices on cycle C_1 or C_3 contains only the vertices of cycle C_1 or C_3 , respectively.

Proof. We only prove the case where both of the vertices are on C_1 since the case is similar if both of the vertices are on C_3 . Let x_i and x_j be two vertices of C_1 . For contradiction, we assume that P is a shortest path between x_i and x_j which contains vertices other than vertices of cycle C_1 . Then (i) G would have a non-triangulated face other than F_1 and F_2 or (ii) a vertex of C_2 would have degree more than five or (iii) the graph G would be non-planar, a contradiction to the properties of a doughnut graph. Therefore the shortest path between any two vertices of C_1 contains only the vertices of cycle C_1 . \square

Lemma 7.2.2 *Let G be a p -doughnut graph and let Γ be a doughnut embedding of G . Let C_1, C_2 and C_3 be the outer, the middle and the inner cycle of Γ , respectively. Let z_i and z_j be two non-adjacent vertices on the C_2 cycle of Γ and the length of the shortest (between clockwise and counter clockwise) path between them along C_2 is l . Then the length of any path between z_i and z_j is at least $\lceil l/2 \rceil + 1$.*

Proof. Without loss of generality we assume that $i < j$ and the shortest path between z_i and z_j along C_2 is in the counter clockwise direction. We prove the claim by induction on l . Since z_i and z_j are non-adjacent, then $l \geq 2$. The claim is true for $l = 2$ where $j = i + 2$ and the shortest path between these two vertices has length $\lceil 2/2 \rceil + 1 = 2$.

Assume that $l > 2$ and the claim is true for all pairs of vertices of C_2 with the shortest distance $l' < l$ between them along C_2 . In this case $j = i + l$. Let P be any path between z_i and z_j . We now show that the length of P is at least $\lceil l/2 \rceil + 1$.

We first consider the case where P contains some vertex z_k of cycle C_2 such that $i < k < j$. If z_k is adjacent to z_i , then by induction hypothesis, the length of any path between z_k and z_j has length $\lceil (l - 1)/2 \rceil + 1$ and therefore the length of P is at least $1 + \lceil (l - 1)/2 \rceil + 1 \geq \lceil l/2 \rceil + 1$. From the same line of reasoning, we can show that if

z_k is adjacent to z_j , then the length of P is at least $\lceil l/2 \rceil + 1$. Thus we assume that z_k is adjacent to neither z_i nor z_j . Then from induction hypothesis, the length of any path between z_i and z_k is at least $\lceil (k - i)/2 \rceil + 1$ and the length of any path between z_k and z_j is at least $\lceil (j - k)/2 \rceil + 1$. Therefore the length of P is at least $\lceil l/2 \rceil + 1$. Hence, no path containing vertices of cycle C_2 other than z_i and z_j has length less than $\lceil l/2 \rceil + 1$.

Thus we assume that P does not contain any vertices of C_2 other than z_i and z_j . Therefore there are only two different paths to consider for each pair of vertices z_i and z_j , one containing only vertices of C_1 and the other containing only vertices of C_3 other than z_i and z_j . If P contains only the vertices of C_1 other than z_i and z_j , then by Lemma 6.1.1, we find that the rightmost neighbor of z_i and the leftmost neighbor of z_j on C_1 are $x_{\lfloor i/2 \rfloor + 1}$ and $x_{\lfloor j/2 \rfloor}$ respectively and therefore the length of P is at least $1 + \lfloor j/2 \rfloor - \lfloor i/2 \rfloor - 1 + 1 \geq \lceil l/2 \rceil + 1$. On the other hand, if P contains only the vertices of C_3 other than z_i and z_j , then by Lemma 6.1.1, we find that the rightmost neighbor of z_i and the leftmost neighbor of z_j on C_3 are $y_{\lfloor i/2 \rfloor}$ and $y_{\lfloor j/2 \rfloor}$ respectively and therefore the length of P is at least $1 + \lfloor j/2 \rfloor - \lfloor i/2 \rfloor + 1 \geq \lceil l/2 \rceil + 1$.

Fig. 7.1 illustrates different cases of shortest path between two vertices of C_2 where Fig. 7.1(a) illustrates a shortest path between z_3 and z_7 , Fig. 7.1(b) illustrates a shortest path between z_4 and z_8 , Fig. 7.1(c) illustrates a shortest path between z_z and z_7 , and Fig. 7.1(d) illustrates a shortest path between z_3 and z_8 .

□

We now have the following Theorem.

Theorem 7.2.3 *Let G be a p -doughnut graph and let Γ be a doughnut embedding of G . Let C_1 , C_2 and C_3 be the three vertex disjoint cycles of Γ such that C_1 is the outer cycle, C_2 is the middle cycle and C_3 is the inner cycle. Then the shortest path between any pair of vertices u and v of G can be found in linear time.*

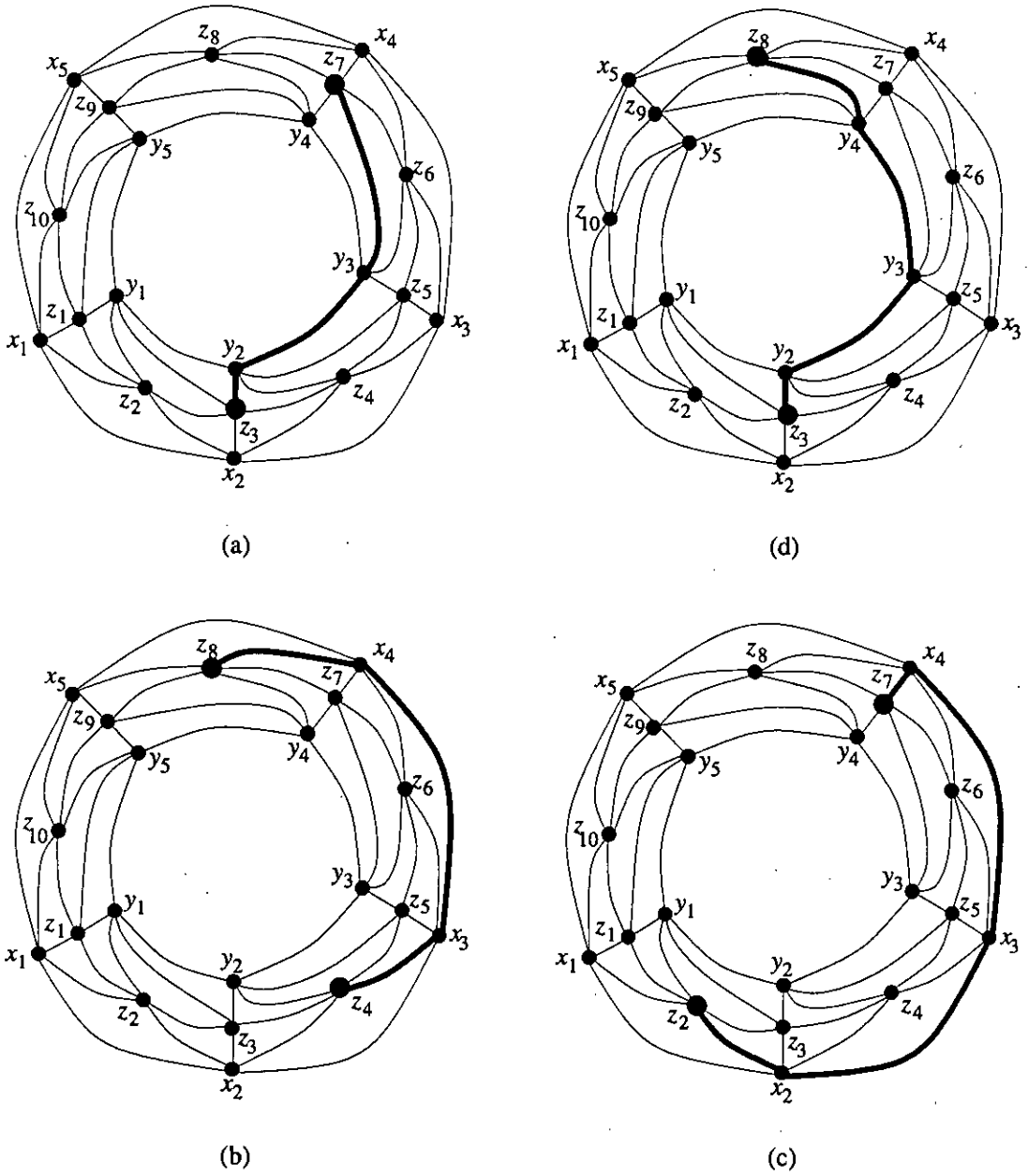


Figure 7.1: Illustration for shortest path between two vertices on C_2 of a doughnut graph.

Proof. The vertices of G lie on three vertex disjoint cycles C_1 , C_2 and C_3 where C_1 is the outer cycle, C_2 is the middle cycle and C_3 is the inner cycle. we have four cases to consider.

Case 1: Both the u and v are either on C_1 or on C_3 .

Without loss of generality, we assume that both the u and v are on C_1 , since the case where both of u and v are on C_3 is similar. Let $x_i = u$ and $x_j = v$. Without loss of generality, we may assume that $i < j$. The shortest path may be in a clockwise direction or in a counter clockwise direction. The shortest path between x_i and x_j is in a counter clockwise order of vertices of C_1 if $(j - i) < \lceil p/2 \rceil$ otherwise in an clockwise order of vertices. We take the path $P_1 = x_i, x_{i+1}, \dots, x_j$ if $(j - i) < \lceil p/2 \rceil$ otherwise $P_1 = x_i, x_{i-1}, \dots, x_j$. By Lemma 7.2.1, P_1 is the shortest path between x_i and x_j . Fig. 7.2(a) illustrates the case where $u = x_2$ and $v = x_4$, and Fig. 7.2(b) illustrates the case $u = x_2$ and $v = x_5$

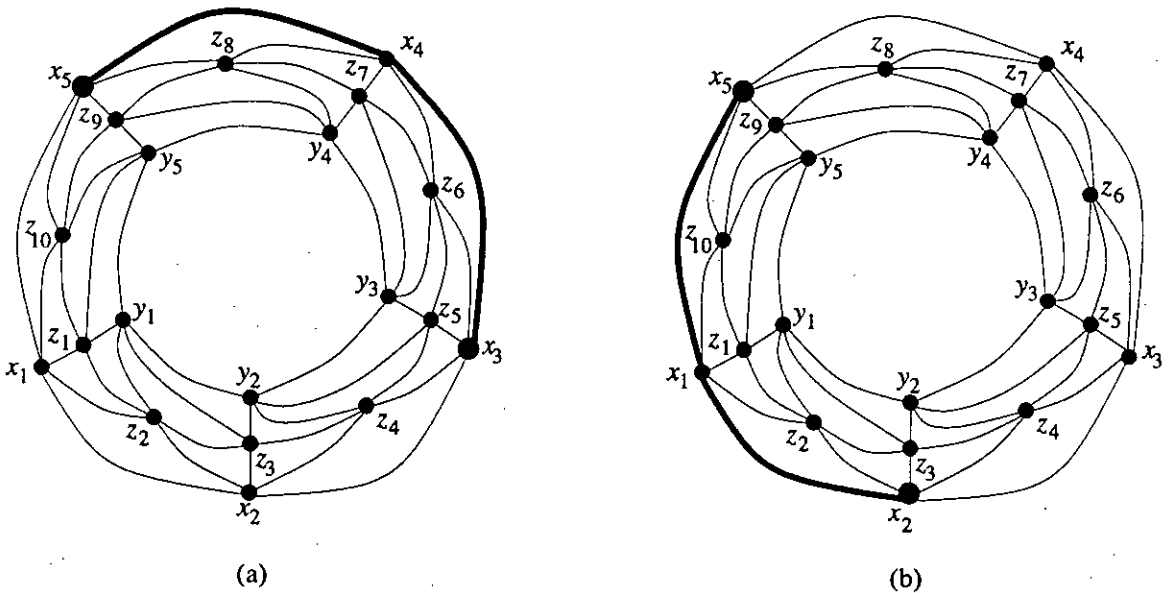


Figure 7.2: Illustration for shortest path between two vertices on Cycle C_1 .

Case 2: Both the u and v are on cycle C_2 .

We assume that $z_i = u$ and $z_j = v$, respectively. Without loss of generality, we also assume that $i < j$. In this case, the shortest path between z_i and z_j may be along the cycle C_2 only or it may contain vertices of either from cycle C_1 or from cycle C_3 along with the vertices of cycle C_2 . We have the following two subcases.

Subcase 2(a): z_i and z_j are adjacent.

In this case, the shortest path between these two vertices consists of the edge (z_i, z_j) .

Subcase 2(b): z_i and z_j are not adjacent.

If both i and j are even, take $P = (z_i, x_{i/2+1}, \dots, x_{j/2}, z_j)$. If both i and j are odd, take $P = (z_i, y_{\lceil i/2 \rceil}, \dots, y_{\lfloor j/2 \rfloor - 1}, z_j)$. If i is even and j is odd, take $P = (z_i, x_{i/2+1}, \dots, x_{\lfloor j/2 \rfloor}, z_j)$. If i is odd and j is even, take $P = (z_i, x_{\lceil i/2 \rceil}, \dots, x_{j/2}, z_j)$. It is easy to verify that all these paths have length $\lceil l/2 \rceil + 1$ and by Lemma 7.2.2, these paths are the shortest paths between z_i and z_j .

Case 3: One of the u and v is on C_2 , and the other one is on C_1 or C_3 .

We assume that u is on C_2 and the v is on C_1 . Let $z_i = u$ and $x_j = v$. We also assume that $\lceil i/2 \rceil < j$. We take path (i) $P_3 = z_i, x_{\lceil i/2 \rceil}, x_{\lceil i/2 \rceil + 1}, \dots, x_j$ if $j - \lceil i/2 \rceil < \lceil p/2 \rceil$ in case of i is odd, otherwise $P_3 = z_i, x_{\lceil i/2 \rceil}, x_{\lceil i/2 \rceil - 1}, \dots, x_j$. (ii) $P_3 = z_i, x_{i/2}, x_{i/2+1}, \dots, x_j$ if $j - \lceil i/2 \rceil < \lceil p/2 \rceil$ in case of i is even otherwise $P_3 = z_i, x_{i/2}, x_{i/2-1}, \dots, x_j$. we can prove that both of the paths are the shortest path since each of them are the subpaths of the shortest path of Subcase 2(b). Fig. 7.3(a) illustrates an example where $z_4 = u$ and $x_5 = v$. The shortest path $P_3 = z_4, x_3, x_4, x_5$. Fig. 7.3(b) illustrates an example where $z_3 = u$ and $x_4 = v$. The shortest path $P_3 = z_3, x_2, x_3, x_4$.

Case 4: One of the u and v on C_1 , and the other one is on C_3 .

We assume that the u is on C_1 and the v is on C_3 . Let $x_i = u$ and $y_j = v$. Without loss of generality, we assume that $i < j$. We take the path $P_4 = x_i, z_{2i}, y_i, y_{i+1}, \dots, y_j$ if $j - i < \lceil p/2 \rceil$ otherwise $P_4 = x_i, z_{2i-2}, y_{i-1}, y_{i-2}, \dots, y_j$. Note that in Case 3, the length of the shortest path between z_i and y_j is $j - \lceil i/2 \rceil + 1$. We now prove that P_4 is the shortest path between x_i and y_j . We prove only for the case where y_j is to the counter clockwise

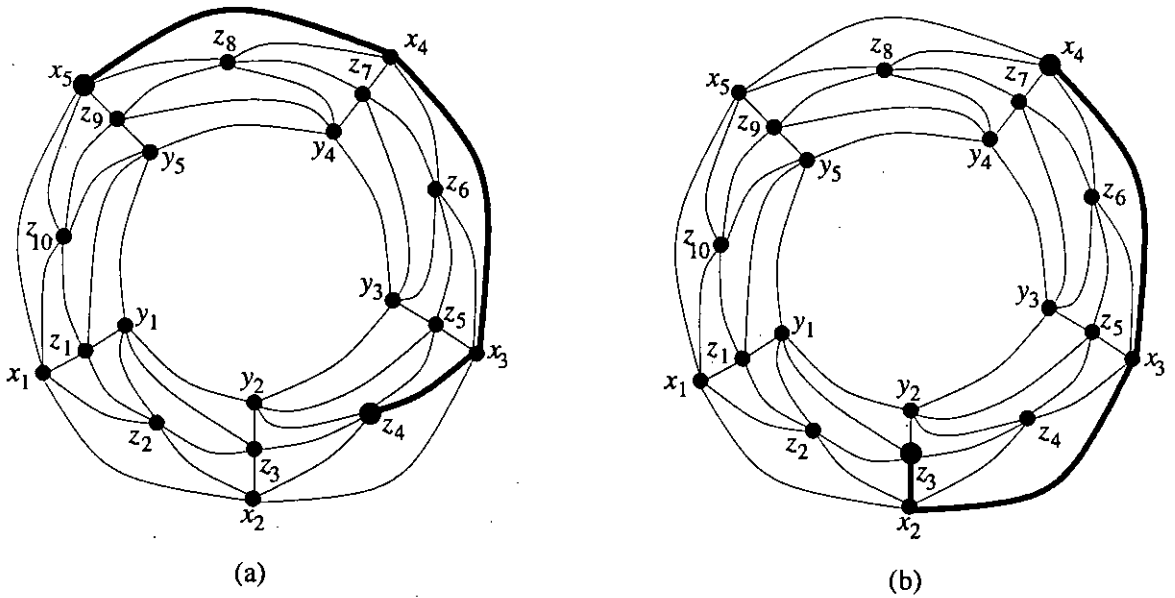


Figure 7.3: Illustration for shortest path between a vertex on cycle C_2 and a vertex on cycle C_1 .

direction of x_i . In this case, we prove that P_4 is the shortest path and the length of the shortest path is $l + 2$ using induction on length $l = j - i$. The claim is obvious for $l = 0$. Thus we assume that $l > 0$ and for any $l' < l$, the claim is true. If P_4 is not the shortest path between x_i and y_j then there is a path p' with length less than $l + 2$. Since y_j is to counter clockwise direction from x_i , the second vertex of P' is either x_{i+1} or z_{2i} . If x_{i+1} is the second vertex then by induction hypothesis, the shortest path between x_{i+1} and y_j has length $l + 1$ and the length of P' is at least $l + 2$ which contradicts our assumption. Thus we assume that the second vertex is z_{2i} . Since P_4 contains the shortest path between z_{2i} and y_j by Case 3, the length of P' can not be less than P_4 in this case also. Fig. 7.4(a) illustrates an example where $x_2 = u$ and $y_4 = v$. The shortest path $P_4 = x_2, z_4, y_2, y_3, y_4$. Fig. 7.4(b) illustrates an example where $x_2 = u$ and $y_5 = v$. The shortest path $P_4 = x_2, z_2, y_1, y_5$. Thus we can find a shortest path between any pair of vertices of a doughnut graph. One can see that the shortest path between any pair of vertices can be found in

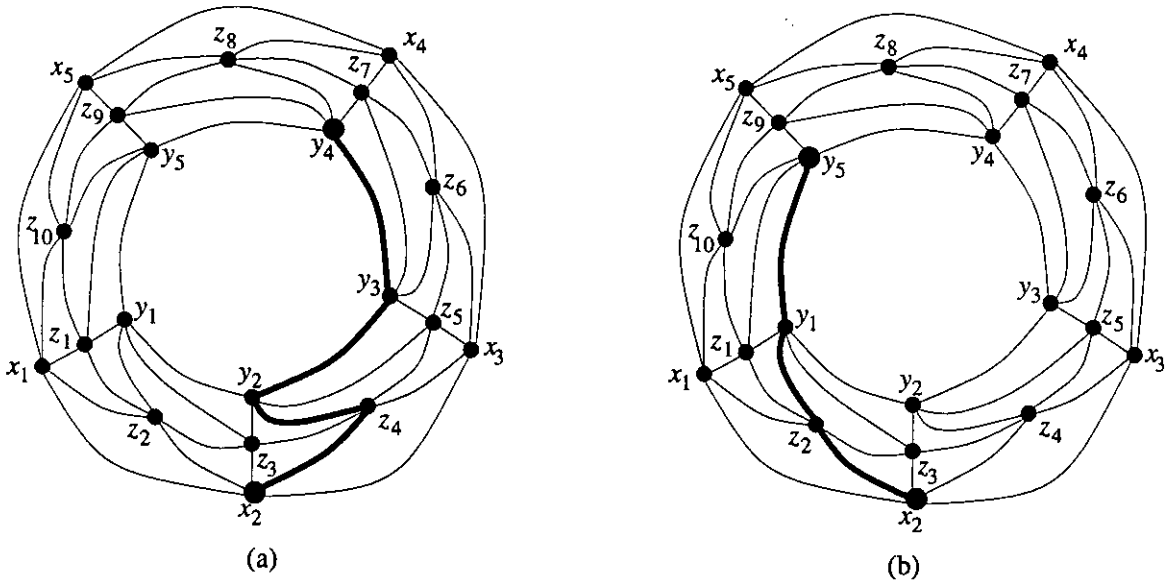


Figure 7.4: Illustration for shortest path between a vertex on C_1 and a vertex on C_3 .

linear time.

□

The above theorem gives a simple routing scheme between any pair of vertices of a doughnut graph by using the shortest path between the source and destination to route. Moreover, this path can be found in linear time.

7.3 Recursive Structure of Doughnut Graphs

A graph G has a recursive structure if every instances of it can be created by connecting the smaller instances of the same graph. We now show that the doughnut graphs have recursive structure. We use the straight-line grid drawing of a p -doughnut graph with linear area. Let Γ be a doughnut embedding of a p -doughnut graph as illustrates in Fig. 7.5(a) where vertices of cycles C_1, C_2 and C_3 are labeled using the labeling method as mentioned in Chapter 6 Section 6.1. Let z_i be a vertex on C_2 such that z_i has two neighbors on C_1 . Let x and x' be the two neighbors of z_i on C_1 such that x' is the counter

clockwise next vertex to x on C_1 . We call x the *left neighbor* of z_i on C_1 and x' the *right neighbor* of z_i on C_1 . Similarly we define the left neighbor and the right neighbor of z_i on C_3 if a vertex z_i on C_2 has two neighbors on C_3 . In the drawing algorithm, the cycles C_1 , C_2 and C_3 are embedded on the three nested rectangles R_1 , R_2 and R_3 , respectively. The four corner vertices of R_2 are the z_1, z_p, z_{p+1} and z_{2p} in counter clockwise order. The four corner vertices of rectangle R_1 are x_1, x_i, x_{i+1}, x_p in counter clockwise order such that x_i is the neighbor of z_p if z_p has exactly one neighbor on C_1 , otherwise x_i is the left neighbor of z_p and x_{i+1} be the right neighbor of z_p . The four corner vertices of R_3 are y_1, y_i, y_{i+1} and y_p in counter clockwise order such that y_i is the neighbor of z_p if z_p has exactly one neighbor on C_3 , otherwise y_i is the left neighbor and y_{i+1} be the right neighbor of z_p on C_3 . Fig. 7.5(b) illustrates the straight-line grid drawing of G in Fig. 7.5(a) where the label of four corner vertices of the rectangles R_1, R_2 and R_3 are shown. We now need some

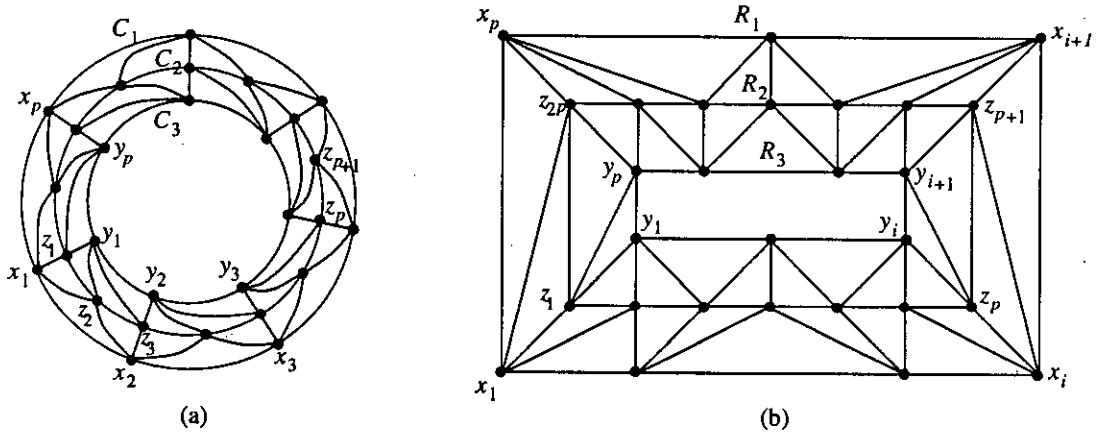


Figure 7.5: (a) A doughnut embedding of a p -doughnut graph of G and (b) straight-line grid drawing of G with linear area.

definitions. Let D be a straight-line grid drawing of a p -doughnut graph with linear area. We partition the edges of D as follows. The *left partition* consists of edges - (i) (x_1, x_p) , (ii) (z_1, z_{2p}) , (iii) (y_1, y_p) , (iv) (x_1, z_{2p}) and (v) (z_1, y_p) ; and the *right partition* consists of edges - (i) (z_p, z_{p+1}) , (ii) the edge between two neighbors of z_p on R_1 if z_p has two neighbors

on R_1 otherwise the edge between two neighbors of z_{p+1} on R_1 , (iii) the edge between two neighbors of z_p on R_3 if z_p has two neighbors on R_3 otherwise the edge between two neighbors of z_{p+1} on R_3 , (iv) the edge between z_p and its right neighbor on R_1 if z_p has two neighbors on R_1 otherwise the edge between z_{p+1} and its left neighbor on R_1 , and (v) the edge between z_p and its right neighbor on R_3 if z_p has two neighbors on R_3 otherwise the edge between z_{p+1} and its left neighbor on R_3 . The other two partitions of edges we call the *top partition* and the *bottom partition*. Fig. 7.6(b) illustrates four partitions of edges (indicated by dotted line) of a p -doughnut graph G in Fig. 7.6(a) where $p=4$.

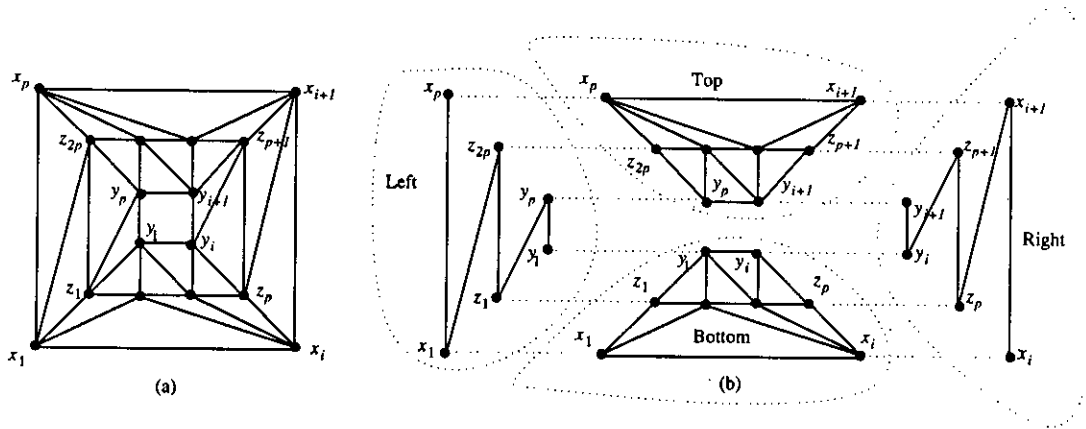


Figure 7.6: (a) A p -doughnut graph G where $p=4$ and (b) illustration for four partitions of edges of G .

We now construct a $(p_1 + p_2)$ -doughnut graph G from a p_1 -doughnut graph G_1 and a p_2 -doughnut graph G_2 as follows.

We construct the graph G'_1 from G_1 as follows. We identify the vertex x_{i+1} of the top partition to the vertex y_i of the right partition, vertex z_{p_1+1} of the top partition to the vertex z_{p_1} of the right partition, and vertex y_{i+1} of the top partition to the vertex x_i of the right partition. Fig 7.7(c) illustrates G'_1 of G_1 in Fig. 7.7(a) where $p_1 = 4$. We now construct the graph G'_2 from G_2 as follows. We identify the vertex y'_{p_2} of left partition to the vertex x'_1 of the bottom partition, vertex z'_{2p_2} of the left partition to the vertex z'_1

of the bottom partition, and the vertex x'_{p_2} of left partition to the vertex y'_1 . Fig 7.7(f) illustrates G'_2 of G_2 in Fig. 7.7(d). We put G'_1 left to the G'_2 . Finally, we identify the vertices y_{i+1} , z_{p_1+1} , x_{i+1} of G'_1 to the vertices of x'_{p_2} , z'_{2p_2} , y'_{p_2} of G'_2 , respectively; and identify the vertices of y_i , z_{p_1} , x_i of G'_1 to the vertices of x'_1 , z'_1 , y'_1 of G'_2 , respectively; and we get a $(p_1 + p_2)$ -doughnut graph G as illustrated in Fig. 7.7(h).

We now have the following theorem.

Theorem 7.3.1 *Let G_1 be a p_1 -doughnut graph and let G_2 be a p_2 -doughnut graph. Then one can construct $(p_1 + p_2)$ -doughnut graph G by combining G_1 and G_2 .*

7.4 Topological Properties of Doughnut Graphs

Let G be a p -doughnut graph. By Theorem 3.2.4, the number of vertices of G is $4p$ where $p(> 3)$ is an integer. A p -doughnut graph is maximal fault tolerant since it is 5-regular by Theorem 3.2.4. By Theorem 3.2.6, every p -doughnut graph G has a doughnut embedding Γ where vertices of G lie on three vertex disjoint cycles. In Γ , the vertices are lie on three cycles C_1 , C_2 and C_3 such that C_1 is the outer cycle containing p vertices, C_2 is the middle cycle containing $2p$ vertices and C_3 is the inner cycle containing $2p$ vertices. Then, one can easily see that the diameter of a p -doughnut graph is $\lfloor p/2 \rfloor + 2$.

Hypercube [BA84], the most efficient and widely used, is a popular interconnection network. It has logarithmic diameter. It is not a scalable network since the degree of a node of this network changes with the changes in the dimension of the network. It has maximal faultolerance, Hamiltonian embedding and recursive structure. Wrapped around butterfly [Lei92] is another popular interconnection network which is scalable. The degree of each node of this network does not change with the size of the network. It has maximal fault tolerance and Hamiltonian embedding but the diameter is higher than the hypercube for same size of network. The k -degree Cayley graph [HH06] is another family of graph for interconnection networks. It has maximal fault tolerance, logarithmic diameter and

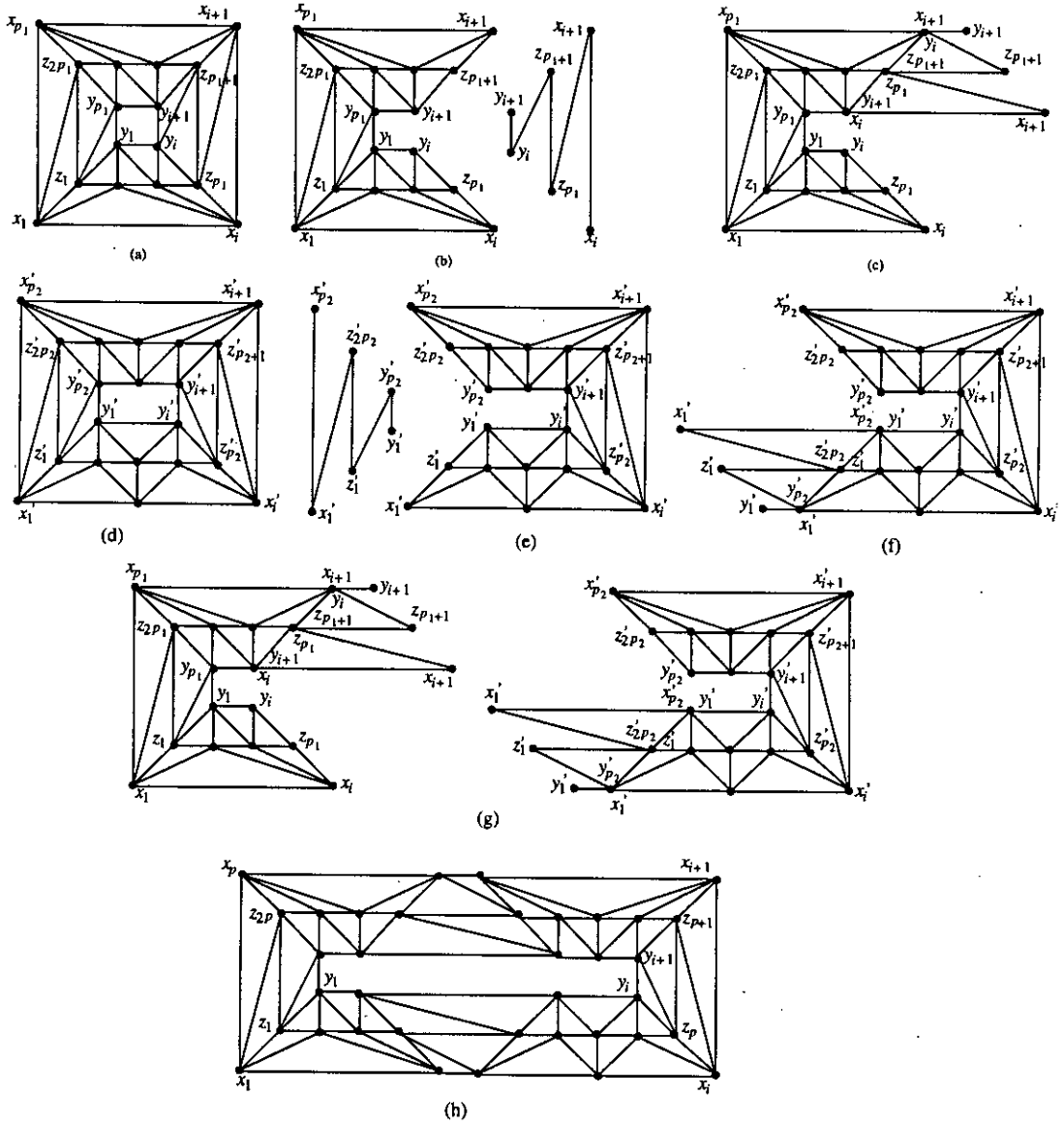


Figure 7.7: Illustration for construction of a $(p_1 + p_2)$ -doughnut graph G from a p_1 -doughnut graph G_1 and a p_2 -doughnut graph G_2 where $p_1 = 4$ and $p_2 = 5$.

the degree of each node is fixed. It has also Hamiltonian embedding.

7.5 Conclusion

In this chapter we presented the topological properties of doughnut graphs. We proposed a very simple routing scheme for doughnut graphs. Doughnut graphs have beautiful recursive structure. One of the limitation is the diameter which is linear but the coefficient is $1/8$. We may have a scalable interconnection network using doughnut graphs since the degree of a vertex of a doughnut graph does not change with the size of a doughnut graph. This is also important for VLSI implementation point of view as well as applications where the computing nodes in an interconnection networks only have fixed number of I/O ports. Thus doughnut graphs may have nice application as the interconnection networks.

Chapter 8

Conclusion

This thesis deals with classes of planar graphs that admit straight-line grid drawings with sub-quadratic area. We have introduced some classes of planar graphs that admit area-efficient drawings. We have provided linear-time algorithms for finding straight-line grid drawings of newly introduced classes of planar graphs with sub-quadratic area. We have studied k -partitioning problems for newly introduced classes of planar graphs and found some interesting results for doughnut graphs. We have also studied topological properties of doughnut graphs and identified some beautiful features so that we propose this class as a promising class of interconnection networks.

We first summarize each chapter and its contributions. In Chapter 1 we have introduced different drawing styles, different aspects of graph drawings, motivation of this study of graph drawing and summary of our results with the existing results.

In Chapter 2 we have introduced graph theoretical terminologies which have been used in this thesis.

In Chapter 3 we have presented doughnut graphs, a subclass of 5-connected as well as 3-outerplanar graphs, that admit straight-line grid drawings with linear area. We also give a linear-time algorithm for finding a straight-line grid drawing with linear-area.

In Chapter 4, we have given a necessary and sufficient condition for a 4-connected

planar graph to be a spanning subgraph of a doughnut graph. We also present a linear-time algorithm for augmenting a 4-connected spanning subgraph of a doughnut graph to a doughnut graph. Thus we have introduced a subclass of 4-connected planar graphs that admits straight-line grid drawing with linear area.

In Chapter 5, we have introduced label-constrained outerplanar graphs that admit straight-line grid drawings with $O(n \log n)$ area. We give a linear-time algorithm for finding such a drawing of a label-constrained outerplanar graph. We also give a linear-time algorithm for recognition of a label-constrained outerplanar graph.

In Chapter 6, we have given a linear-time algorithm for finding a k -partition of a doughnut graph. Our algorithm is based on finding a Hamiltonian path between any pair of vertices of a doughnut graph. We also give a linear-time algorithm for finding a Hamiltonian path between any pair of vertices of a doughnut graph.

In Chapter 7, we have proposed the class of doughnut graph as a promising class for interconnection networks. We have identified a set of topological properties of a doughnut graph like regularity, smaller diameter, maximal fault tolerance, recursive structure etc. We have also proposed a very simple and efficient routing scheme in a doughnut graph.

In the course of thesis, we have raised the following open problems.

- (a) Finding classes of planar graphs richer than binary trees, and other than the doughnut graphs and “balanced” outerplanar graphs, that admit straight-line grid drawings with linear area.
- (b) Recognition of a 4-connected spanning subgraph of a doughnut graph where two vertex disjoint faces F_1 and F_2 remain unaffected is a non-trivial problem. We have solved this problem. Recognition of a 4-connected spanning subgraph of a doughnut graph without any restriction on face even harder than the solved one. Hence recognition of a 4-connected spanning subgraph of a doughnut graph without any restriction on faces is an interesting problem. Moreover, recognition a 3-connected or

a 2-connected spanning subgraph of a doughnut graph are also interesting problems.

- (c) A doughnut graph is a 5-connected planar graph. We know that every k -connected planar graph admits a k -partition with k basis vertices. We provide the solution for k -partitioning of a doughnut graph with basis at most two. Hence 5-partitioning of a doughnut graph with 5 basis vertices is an interesting problem.
- (d) Finding a lower bound on area requirement of an outerplanar graph is another interesting problem.
- (e) Finding other classes of planar graphs that admit straight-line grid drawing with sub-quadratic area.

References

- [AHK87] S. B. Akers, D. Harel and B. Krishnamurthy, *The star graph: an attractive alternative to the n -cube*, Proc. of International Conference on Parallel Processing, pp. 555-556, 1987.
- [AK89] S. B. Akers and B. Krishnamurthy, *A group-theoretic model for symmetric interconnection networks*, IEEE Trans. Comput., 38, pp. 555-566, 1989.
- [BA84] L. Bhuyan and D. P. Agarwal, *Generalized hypercube and hyperbus structure for a computer network*, IEEE Trans. Comput. 33, pp. 323-333, 1984.
- [BEGKLM04] F. Brandenburg, D. Eppstein, M. T. Goodrich, S. Kobourov, G. Liotta and P. Mutzel, *Selected open problems in graph drawing*, Proc. of Graph Drawing 2003, Lect. Notes in Computer Science, Springer, 2912, pp. 515-539, 2004.
- [Bie02] T. C. Biedl, *Drawing outerplanar graphs in $O(n \log n)$ area*, Proc. of Graph Drawing 2002, Lect. Notes in Computer Science, Springer, 2528, pp. 54-65, 2002.
- [BL76] K. S. Booth and G. S. Lueker, *Testing the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithm*, Journal of Comput. Syst. Sci., 13, pp. 335-379, 1976.
- [CAB93] C. Chen, D. P. Agrawal and J. R. Burke, *dBcube: a new class of hierarchical multiprocessor interconnection networks with area efficient layout*, IEEE Trans. Parallel Distributed Systems, 4, pp. 1332-1344, 1993.

- [CFF00] G. H. Chen, J. S. Fu and J. F. Fang, *Hypercomplete: a pancyclic, recursive topology for large-scale distributed multicomputer systems*, Networks, 35, pp. 56-69, 2000.
- [CN89] N. Chiba and T. Nishizeki, *The Hamiltonian cycle problem is linear time solvable for 4-connected planar graphs*, Journal of Algorithms, 10, pp. 187-211, 1989.
- [CN98] M. Chrobak and S. Nakano, *Minimum-width grid drawings of a plane graphs*, Comp. Geom. Theory and Appl., 11, pp. 29-54, 1998.
- [CNAO85] N. Chiba, T. Nishizeki, S. Abe and T. Ozawa, *A linear algorithm for embedding planar graphs using PQ-trees*, Journal of Comput. Syst. Sci., 30, pp. 54-76, 1985.
- [DETT94] G. Di Battista, P. Eades, R. Tamassia and I. G. Tollis, *Algorithms for drawing graphs: an annotated bibliography*, Comp. Geom. Theory and Appl., 4, pp. 235-282, 1994.
- [DF85] M. E. Dyer and A. M. Frieze, *On the complexity of partitioning graphs into connected subgraphs*, Discrete Applied Mathematics, 10, pp. 139-153, 1985.
- [DF06] G. Di Battista and F. Frati, *Small area drawings of outerplanar graphs*, Proc. of Graph Drawing 2005, Lect. Notes in Computer Science, Springer, 3843, pp. 89-100, 2006.
- [Far48] I. Fary, *On Straight line representation of planar graphs*, Acta Sci. Math. Szeged, 11, pp. 229-233, 1948.
- [FPP90] H. de Fraysseix, J. Pach and R. Pollack, *How to draw a planar graph on a grid*, Combinatorica, 10, pp. 41-51, 1990.
- [Fra07] F. Frati, *Straight-line drawings of outerplanar graphs in $O(dn \log n)$ area*, Proc. of the 19th Canadian Conference on Computational Geometry, pp. 225-228, 2007.

- [FW91] M. Formann and F. Wagner, *The VLSI layout problem in various embedding models*, Proc. of International Workshop on Graph Theoretic Concepts in Computer Science (WG '90), Lect. Notes in Computer Science, Springer, 484, pp.130-139, 1991.
- [GJ79] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, W. H. Freeman and Company, Newyork, 1979.
- [GJ83] M. R. Garey and D. S. Johnson, *Crossing number is NP-complete*, SIAM Journal of Alg. Disc. Methods, 4(3), pp. 312-316, 1983.
- [GJT76] M. R. Garey, D. S. Johnson and R. E. Tarjan, *The planar Hamiltonian circuit problem is NP-complete*, SIAM J. Comput., 5, pp. 704-714, 1976
- [GR02] A. Garg and A. Rusu, *Straight-line drawings of binary trees with linear area and arbitrary aspect ration*, Proc. of Graph Drawing 2002, Lect. Notes in Computer Science, Springer, 2528, pp. 320-331, 2002.
- [GR04a] A. Garg and A. Rusu, *Area-efficient drawings of outerplanar graphs*, Proc. of Graph Drawing 2003, Lect. Notes in Computer Science, Springer, 2912, pp. 129-134, 2004.
- [GR04b] A. Garg and A. Rusu, *A more practical algorithm for drawing binary trees in linear area with arbitrary aspect ratio*, Proc. of Graph Drawing 2003, Lect. Notes in Computer Science, Springer, 2912, pp. 159-165, 2004.
- [GT95] A. Garg and R. Tamassia, *On the computational complexity of upward and rectilinear planarity testing*, Proc. of Graph Drawing 1994, Lect. Notes in Computer Science, 894, pp. 286-297, 1995.
- [Gyr78] E. Györi, *On division on connected subgraphs*, Proc. of 5th Hungarian Combinational Coll., pp. 485-494, 1978.

- [HH06] S. Hsieh and T. Hsiao, *The k -degree Caley graph and its topological properties*, Networks, 47(1), pp. 26-36, 2006.
- [Hor45] R. E. Horton, *Erosioned development of systems and their drainage basins, hydrophysical approach to quantitative morphology*, Bull. Geol. Soc. of America, 56, pp. 275-370, 1945.
- [HT73] J. E. Hopcroft and R. E. Tarjan, *Dividing graph into triconnected components*, SIAM J. Comput., 2(3), pp. 135-158, 1973.
- [HT74] J. E. Hopcroft and R. E. Tarjan, *Efficient planarity testing*, Journal of Assoc. Comput. Mach., 21, pp. 549-568, 1974.
- [JSN94] L. Jou, H. Suzuki and T. Nishizeki, *A linear algorithm for finding a nonseparating ear decomposition of triconnected planar graphs*, Tech. Rep. of Information Processing Society of Japan, AL40-3, 1994.
- [KAR09] M. R. Karim, M. J. Alam and M. S. Rahman, *Straight-line grid drawings of label-constrained outerplanar graphs with $O(n \log n)$ area*, Proc. of Workshop on Algorithms and Computation, 2009, Lect. Notes in Computer Science, Springer, to appear.
- [KR07] M. R. Karim and M. S. Rahman, *Straight-line grid drawings of planar graphs with linear area*, Proc. of Asia-Pacific Symposium on Visualisation, 2007, IEEE, pp. 109-112, 2007.
- [KR08] M. R. Karim and M. S. Rahman, *Four-connected spanning subgraphs of doughnut graphs*, Proc. of Workshop on Algorithms and Computation, 2008, Lect. Notes in Computer Science, Springer, 4921, pp. 132-143, 2008.
- [Kur30] C. Kuratowski, *Sur le probleme des courbes gauches en topologie*, Fundamenta Math, 15, pp. 271-283, 1930.

- [KvL84] M. R. Kramer and J. van Leeuwen, *The complexity of wire routing and finding minimum area layouts for arbitrary VLSI circuits*, (Ed.) F. P. Preparata, *Advances in Computing Research 2: VLSI Theory*, JAI Press, pp. 129-146, 1984.
- [Lei92] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays-Trees-Hypercubes*, Morgan Kaufmann Publishers, Inc., San Mateo, 1992.
- [Lov77] L. Lovász, *A homology theory for spanning trees of a graph*, *Acta Math, Acad. Sci. Hungar.* 30, pp. 241-251, 1977.
- [MNN01] K. Miura, S. Nakano and T. Nishizeki, *Grid drawings of four-connected planar graphs*, *Discrete and Computational Geometry*, 26(1), pp. 73-78, 2001.
- [Mut92] P. Mutzel, *A fast linear time embedding algorithm based on the Hopcroft-Tarjan planarity test*, Tech. Report, 92.107, University of Cologne, 1992.
- [NC88] T. Nishizeki and N. Chiba, *Planar Graphs: Theory and Algorithms*, North-Holland, Amsterdam, 1988.
- [NN01] S. Nagai and S. Nakano, *A Linear-time algorithm for five-partitioning five-connected internally triangulated plane graphs*, *IEICE Trans. Fundamentals*, E84-A(9), pp. 2330-2337, 2001.
- [NRN97] S. Nakano, M. S. Rahman and T. Nishizeki, *A linear-time algorithm for four-partitioning four-connected planar graphs*, *Information Processing Letters*, 62, pp. 315-322, 1997.
- [NR04] T. Nishizeki and M. S. Rahman, *Planar Graph Drawing*, World Scientific, Singapore, 2004.
- [PV81] F. P. Preparata and J. Vuillemin, *The cube-connected-cycles: a versatile network for parallel computation*, *Commun ACM*, 24, pp. 300-309, 1981.

- [Sch90] W. Schnyder, *Embedding planar graphs on the grid*, Proc. of First ACM-SIAM Symp. on Discrete Algorithms, pp. 138-148, 1990.
- [SH99] W. K. Shin and W. L. Hsu, *A new planarity test*, Theoretical Computer Science, 223, pp. 179-191, 1999.
- [SP89] M. R. Samatham and D. K. Pradhan, *The De Bruijn multiprocessor network: a versatile parallel processing and sorting network for VLSI*, IEEE Trans. Comput., 38, pp. 567-581, 1989.
- [Ste51] K. S. Stein, *Convex maps*, Proc. of Amer Math. Soc., 2, pp. 464-466, 1951.
- [STN90] H. Suzuki, N. Takahashi and T. Nishizeki, *A linear algorithm for bipartition of biconnected graphs*, Information Processing Letter, 33(5), pp. 227-232, 1990.
- [STNMU90] H. Suzuki, N. Takahashi, T. Nishizeki, H. Miyano and S. Ueno, *An algorithm for tripartitioning 3-connected graphs*, Journal of Inform. Process. Soc. of Japan, 31(5), pp. 584-592, 1990.
- [Str52] A. N. Strahler, *Hypsometric (area-altitude) analysis of erosional topology*, Bull. Geol. Soc. of America, 63, pp. 1117-1142, 1952.
- [Tho83] C. Thomassen, *A theorem on paths in planar graphs*, Journal of Graph Theory, 7, pp. 129-141, 1983.
- [Tut56] W. T. Tutte, *A theorem on planar graphs*, Trans. Amer. Math. Soc. 82, pp. 99-116, 1956.
- [Wag36] K. Wagner, *Bemerkungen zum vierfarbenproblem*, Jahresber. Deutsch. Math-Verien., 46, pp. 26-32, 1936.
- [Wes01] D. B. West, *Introduction to Graph Theory*, Pearson Education Inc., Singapore, 2001.

- [Whi33] H. Whitney, *2-isomorphic graphs*, Amer. Journal Math., 55, pp. 245-254, 1933.
- [Xu01] J. Xu, *Topological Structure and Analysis of Interconnection Networks*, Kluwer Academic Publishers, Dordrecht, 2001.

List of Publications

1. M. R. Karim and M. S. Rahman, *Straight-line grid drawings of planar graphs with linear area*, Proc. of Asia-Pacific Symposium on Visualisation, 2007, IEEE, pp. 109-112, 2007.
2. M. R. Karim and M. S. Rahman, *Four-connected spanning subgraphs of doughnut graphs*, Proc. of Workshop on Algorithms and Computation, 2008, Lect. Notes in Computer Science, Springer, 4921, pp. 132-143, 2008.
3. M. R. Karim, M. J. Alam and M. S. Rahman, *Straight-line grid drawings of label-constrained outerplanar graphs with $O(n \log n)$ area*, Proc. of Workshop on Algorithms and Computation, 2009, Lect. Notes in Computer Science, Springer, to appear.
4. M. R. Karim and M. S. Rahman, *Straight-line grid drawings of planar graphs with linear area*, submitted to a journal.
5. M. R. Karim, K. M. Nahiduzzaman and M. S. Rahman, *A linear-time algorithm for k -partitioning of doughnut graphs*, submitted to a journal.

Index

- $G(C)$, 41
- HP , 88
- $L_r(T)$, 67
- $L_r(u)$, 66
- NP -complete, 29
- O , 29
- Δ , 20
- Γ , 35
- Ω , 29
- δ , 20
- f_r , 69
- k -connected, 22
- o , 29
- algorithm, 28
 - deterministic, 30
 - exponential, 29
 - linear-time, 29
 - nondeterministic, 30
 - polynomial, 29
 - running time, 28
- connectivity, 22
- cut-vertex, 22
- cycle, 22
- doughnut graph, 11
 - construction, 37
 - doughnut embedding, 41
 - p-doughnut graph, 11
- drawing
 - angular resolution, 7
 - area, 7
 - aspect ratio, 7
 - grid drawing, 5
 - shape of faces, 8
 - straight line drawing, 5
 - straight-line, 6
 - symmetry, 8
- Euler's formula, 26
- face, 25
 - α -face, 57
 - β -face, 57
 - β_1 -face, 58
 - β_2 -face, 58
 - central vertex, 70

- inner face, 25
- left vertex, 70
- outer face, 25
- pole vertex, 70
- quadrangle face, 52
- right vertex, 70
- triangulating a face, 33
- valid triangulation, 56
- vertex-disjoint, 33, 52
- flat labeling, 69
- good vertex, 56
- graph, 19
 - component, 21
 - connected, 21
 - degree, 20
 - disconnected, 21
 - dual, 27
 - Hamiltonian graph, 86
 - isomorphism, 52
 - weak dual, 28
- graph traversal
 - DFS, 31
- Hamiltonian connected, 86
- inner edge, 26
- inner vertex, 25
- interconnection network, 96
- k-partition, 14
- loop, 20, 23
- maximal fault tolerance, 98
- middle vertex, 58
- multiple edges, 20
- network diameter, 97
- node, 23
 - depth, 24
 - height, 24
 - level, 24
- open problems, 112
- outer boundary, 25
- outer cycle, 25
- outer edge, 26
- outer face, 25
- outer vertex, 25
- outerplanar graph, 26
 - k*-outerplanar, 26
 - dual tree, 28
 - label-constrained outerplanar graph,
13
 - maximal outerplanar graph, 27
 - outerplanarity, 42
- partition of edges
 - bottom, 107

- left, 106
- right, 106
- top, 107
- path, 22
 - cross path, 68
 - even path, 52
 - Hamiltonian path, 86
 - left-left path, 68
 - left-right path, 69
 - leftmost path, 68
 - length, 52
 - odd path, 52
 - right-left path, 69
 - right-right path, 68
 - rightmost path, 69
 - vertex-disjoint, 52
- planar drawing, 3
- planar graph, 4, 25
- plane graph, 25
 - triangulated plane graph, 35
- recursive structure, 98
- simple graph, 20
- subgraph, 20
 - induced subgraph, 20
 - spanning subgraph, 21
 - subgraph isomorphism, 52
- tree, 23
 - ancestor, 23
 - child, 23
 - descendant, 23
 - height, 24
 - ordered rooted, 24
 - parent, 23
 - root, 23
 - rooted tree, 23
 - spanning tree, 21
 - subtree, 24
 - vertex labeling, 66
- tree traversal, 24
 - inorder, 24
 - postorder, 24
 - preorder, 24
- walk, 22

