BAR-REPRESENTABLE VISIBILITY GRAPHS AND A RELATED NETWORK FLOW PROBLEM

By

Stephen Kenneth Wismath

August 1989

Technical Report 89-24

Department of Computer Science
University of British Columbia
Vancouver, B.C. V6T 1W5 Canada

Abstract

A bar layout is a set of vertically oriented non-intersecting line segments in the plane called bars. The visibility graph associated with a layout is defined as the graph whose vertices correspond to the bars and whose edges represent the horizontal visibilities between pairs of bars.

This dissertation is concerned with the characterization of bar-representable graphs: those graphs which are the visibility graphs of some bar layout. A polynomial time algorithm for determining if a given graph is bar-representable, and the subsequent construction of an associated layout are provided. Weighted and directed versions of the problem are also formulated and solved; in particular, polynomial time algorithms for the layout of such graphs are developed.

The Planar Full Flow problem is to determine a plane embedding and an (acyclic) orientation of an undirected planar network that admits a feasible flow, that uses all arcs (except those incident upon the source or sink) to full capacity and maintains planarity. The connection of this flow problem to bar-representable graphs is exploited to solve the weighted case of the latter. As evidence that both the acyclicity and planarity constraints are necessary to obtain a polynomial algorithm for this problem, two natural variants of the Full Flow problem are shown to be strongly NP-Complete.

Contents

Abstract ii

List of Figures vi

Acknowledgments ix

1 Introduction 1

1.1 The Flow Existence Problems 5

1.1.1 The Planar Flow Existence Problem 6

1.1.2 The Undirected Planar Flow Problems 7

1.2 Thesis Outline 7

1.3 Previous Work and Motivation 9

1.4 Algorithmic Issues 13

1.5 Notation 14

2 Polarized, and Measured Polarized Bar-Representable Graphs 16

2.1 Introduction 16

2.2 Measured Polarized Graphs 18

2.2.1 Embedded Measured Polarized Graphs 18

2.2.2 Measured Polarized Graphs (Unordered) 21

2.3 Polarized Graphs 23

2.3.1 Embedded Polarized Graphs 24

2.3.2 Polarized Graphs (Unordered) 29

2.4 Reformulation as a Flow Existence Problem 30

2.4.1 Ordered Directed Weighted Graphs 30

2.4.2 Directed Weighted Graphs 31

2.4.3 Ordered Directed Graphs 32

2.4.4 Directed Graphs 33

3 Bar-Representable Graphs **35**

3.1 Introduction 35

3.2 Embedded Bar-Representable Graphs 37

3.3 Bar-Representable Graphs (Unordered) 41

3.4 The Undirected Planar Positive Flow Reformulation 46

3.4.1 Ordered Graphs 46

3.4.2 Unordered Graphs 47

4 Measured Graphs **48**

4.1 Introduction 48

4.2 Flow Existence Formulation 50

4.3 Ordered Graphs 51

4.3.1 Extreme Edges and Their Weights Specified 52

4.3.2 Algorithm to Orient the Edges of a Network With Extreme Edges and Weights Specified 54

4.3.3 Extreme Edges of $N(G)$ With Positive Demands Specified 55

4.3.4 Algorithm to Orient the Edges of a Network with Positively-Weighted Extreme Edges Specified 74

4.3.5 Extreme Edges of $N(G)$ Specified 75

4.4 The Unordered Case 80

4.4.1 Algorithm to Determine a Full Flow for an Unordered Graph 102

5 NP-Complete Full Flow Versions **104**

5.1 Introduction 104

5.2 The Non-Planar Full Flow Problem 105

5.3 The Undirected Planar Cyclic Full Flow Problem 112

6 New Directions and Related Results **116**

6.1 Multiple Lines-of-Sight 119

6.2 Optimization Problems 122

7 Conclusions and Contributions **123**

List of Figures

1.1 Bar Layout and Associated Graph 3

1.2 Non Bar-Representable Graph 4

2.1 A Graph H and its Directed Weighted Bar Layout 17

2.2 $N_w(H)$ 23

2.3 Blockable Visibility 26

2.4 Strong vs. ϵ-Visibility 27

2.5 Violation of The In-Out Property 28

3.1 a) An s-t Numbered Graph, b) Embedded Into Sections, and c) a realization. 38

3.2 G^+ of a Non Bar-Representable Graph 43

4.1 A Non-Measured Graph 50

4.2 Form of $\mathcal{N}(\mathcal{G})$ and $\mathcal{N}(\mathcal{G})^{-v}$ 53

4.3 Corresponding Edges 59

4.4 Corresponding Bundle Weights are Equal 60

4.5 Bundle Order 61

4.6 Partitioning Non-Extreme Vertices 62

4.7 Forced Bundles on Extreme Vertices 63

4.8 Examples of the Four Cases 65

4.9 Nets - Forced and Induced 67

4.10 Proper Intersection of M and N 68

4.11 Case 1 69

4.12 Case 2 69

4.13 Case 3 70

4.14 Case 4 70

4.15 Preferred Cycles 72

4.16 No Forced Bundles 76

4.17 Forced Balanced Vertex 77

4.18 Pseudo-arc 77

4.19 Case 5: Form of I 78

4.20 Case 6: Form of I 79

4.21 Polynomial Number of Exterior Faces 81

4.22 Exponential Number of Exterior Faces 82

4.23 τ and $G - \tau$ 83

4.24 The Two Degenerate Cases 86

4.25 Case 2.i 86

4.26 Case 2.ii 87

4.27 Case 2.iii 88

4.28 Case 3 88

4.29 Case 4 89

4.30 Five Chunks 90

4.31 Redirection of Chord $< v, u >$ 91

4.32 The Three Forms of the Inside Path of τ 93

4.33 Invalid Split on p 94

4.34 G'' 98

4.35 *Out* Flow Type 99

4.36 G''' 101

5.1 $G(B)$ 107

5.2 Component 108

5.3 *N(B)* 110

5.4 Variable Pair 111

5.5 Edge Crossing: Same Weights 113

5.6 Substitute Component 114

6.1 K_8: 2-Line-of-Sight Representation 121

6.2 Layout of a Graph of Thickness k 121

Acknowledgments

There are many people to thank.

At the University of Lethbridge, my colleagues in the Department of Mathematical Sciences and three Deans of Arts and Science have shown great patience and much encouragement over the past 6 years. Their support is gratefully acknowledged.

At U.B.C., the faculty, staff and graduate students helped to create an exciting and enjoyable working environment. Special thanks to: Barry Brachman for his technical help, wit and friendship; Lisa, Norm and Theresa for their collegiality; Carol for her tennis; and Marc, Rick, Jim, *et al* for their commiseration and craziness.

My supervisory and examining committee members had the disadvantage of working with an off-campus graduate student. Thank you for your patience and efforts.

David Kirkpatrick was my supervisor - perhaps mentor better describes the level of his involvement. Thank you for sharing your time, energy, experience and also for many pleasant runs.

Finally, this thesis is dedicated to my family: my wife Shelly (who started after and finished before), and my daughter Alice (who waited until after).

Chapter 1

Introduction

Many problems in computer science may be formulated in terms of the *visibility* among objects in a given domain. Frequently, the existence of these visibilities has been taken for granted, and only recently have the computational issues and the underlying structure been investigated. One of the earliest results in the area is due to Chvátal [4] and is known as the Art Gallery Problem [1]. Visibility problems typically have arisen in VLSI wire routing, printed circuit board layout, graphics, motion planning, and also in less applied areas such as combinatorial mathematics and computational geometry.

O'Rourke [17] suggests that

> ... some of the fundamental unsolved problems involving visibility in computational geometry will not be solved until the combinatorial structure of visibility is more fully understood. Perhaps the purest condensation of this structure is a *visibility graph*.

In general, the nodes of a **visibility graph** correspond to objects and the edges represent a visibility relation among the objects. Two objects are defined as visible if a line

[1] Determine the minimum number of guards sufficient to view the interior of an n-wall art gallery.

segment could be drawn between them intersecting no other object. Attempts at characterizing visibility graphs have for the most part been unsuccessful even for problems of a fairly restricted nature. Several types of objects have been investigated, including "boxes", non-intersecting curves [10],[30], line segments [23],[5],[16],[20],[21], and the sides of a polygon [11],[17]. Various notions of visibility have also been considered. The direction or directions of visibility (known as the **line-of-sight**) may be specified and the *amount* of visibility between objects may be of concern. Perhaps the most constrained version (and the one investigated here) involves parallel line segments as the objects which are visible along a single fixed line-of-sight. This appears to be a natural starting point for the understanding of the more general versions.

A **layout** is defined as a set of vertically oriented non-intersecting line segments (called **bars**) in the plane. A visibility relation exists between pairs of bars: two bars $\overline{v}$ and $\overline{w}$ are **visible** if a non-degenerate rectangle R can be drawn with two opposite sides consisting entirely of portions of $\overline{v}$ and $\overline{w}$ and such that the interior of R contains no portion of any bar. For a pair of visible bars, the vertical width of the visibility rectangle of maximum area is used as a measure of the amount of visibility between the bars along the horizontal line-of-sight [2].

Such a layout has an associated visibility graph whose vertices correspond to the bars and whose edges represent the visibility relation. Figure 1.1 shows a bar layout and its associated graph.

Remark: The restriction to vertical line segments also covers the case when the objects are horizontally convex (no horizontal line intersects the boundary of the object more than

[2]To avoid the introduction of multiple edges, the bars are assumed to be displaced vertically so that there is at most one visibility rectangle between any pair of bars. See section 1.4 for a discussion of this issue.

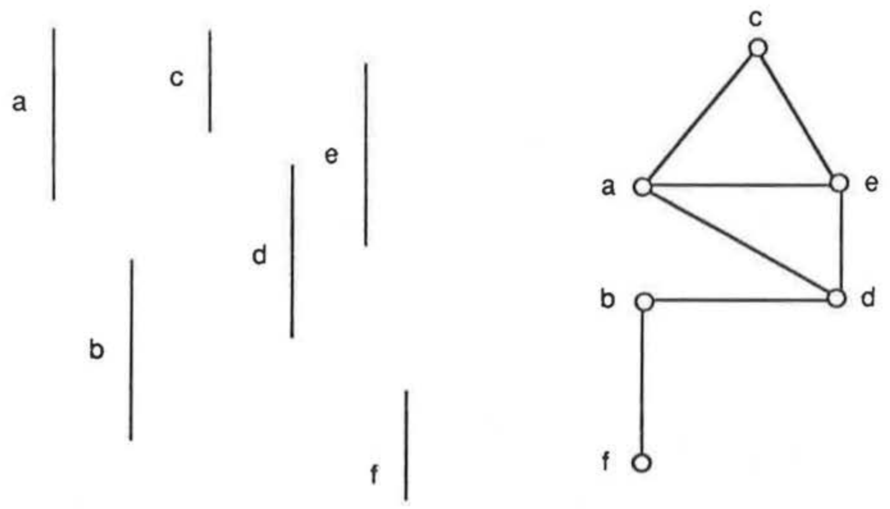


Figure 1.1: Bar Layout and Associated Graph

twice).

This thesis succeeds in characterizing those graphs that correspond to some bar layout - the **bar-representable** graphs - and in providing efficient algorithms for recognizing such graphs and constructing an associated layout.

Bar-representable graphs are certainly planar; since the rectangles representing the visibilities of a layout do not intersect, the edges of the corresponding graph need not intersect, as pointed out by Johnson [30]. (In fact, this planarity argument holds for curves other than bars too). Not all planar graphs, however, are bar-representable. Figure 1.2 shows a planar non bar-representable graph - the smallest such graph in terms of the number of edges. Certain classes of planar graphs, for example, trees, outerplanar graphs, Hamiltonian, and biconnected graphs, can be shown to be bar-representable [29].

As defined, the visibility relation is symmetric, but the bars of a layout exhibit a natural partial ordering, namely a left-to-right order. If bars $\overline{u}$ and $\overline{v}$ are visible, then one is further

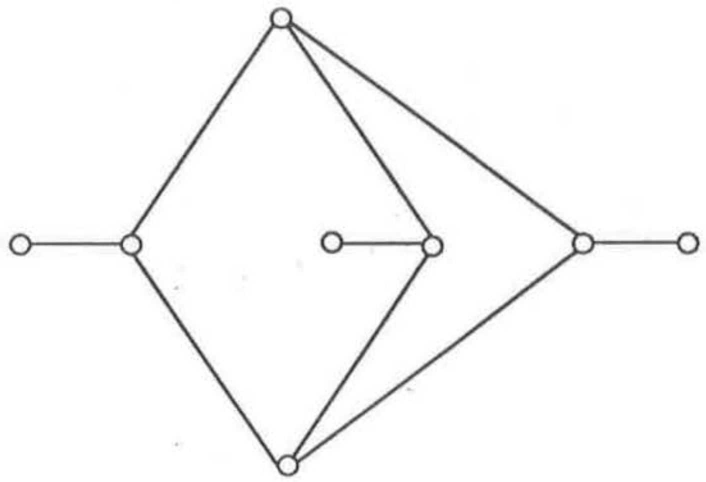


Figure 1.2: Non Bar-Representable Graph

to the left than the other. *Arcs* (directed edges) in the graph will be used to capture this notion of ordering. A directed graph G is a **polarized bar-representable** graph, if there exists an associated layout $L(G)$ consistent with the orientations of the arcs - namely that bar $\overline{u}$ is left of and visible to $\overline{v}$ in $L(G)$ iff $< u, v >$ is an arc of G (directed from u to v) [3].

The amount of visibility between bars (i.e. the vertical width of the maximum visibility rectangle) may also be represented in the associated graph, by the addition of **weights** to the edges. (Hereafter, "weighted" will mean "edge-weighted"). A weighted graph G is a **measured bar-representable** graph, if there exists an associated layout whose visibilities between bars are commensurate with the weights of the corresponding edges of G. If, in addition, G is a directed graph and a polarized bar-representable graph, then G is called a polarized measured bar-representable graph.

Note however, that a particular layout corresponds more precisely to a graph in which the cyclic ordering about each vertex is fixed, corresponding to the ordering of visible

[3]Section 1.5 outlines the notation used in the thesis.

neighbours about each bar. In addition, in any layout, there is a set of bars visible to the "exterior" - bars with some unblocked visibility to the left or right. A graph will be called **ordered** if in addition to the vertex adjacency information, both the (counter-clockwise) cyclic ordering about each vertex and a set of vertices to appear on the exterior face are specified. If an ordered graph G can be embedded in the plane with no edge crossings, preserving the cyclic ordering of the vertices and with the set of specified vertices on the exterior face, then G will be called an **embeddable** graph. Thus, any particular plane embedding is a concrete realization of an ordering and this ordering may be specified in linear time. Conversely, given an ordering G of a graph, the problem of determining whether G has a plane embedding is solvable in linear time, as is the actual construction of the embedding as shown by Kirkpatrick [14].

Bar-representability questions can be extended to include weighted, directed and ordered graphs; the eight cases that arise as a result, form the framework of the investigation in the thesis. The general form for each case is: given a graph G, either directed or undirected, either weighted or unweighted, and either ordered or unordered, determine if G is a (polarized), (measured), (embeddable) bar-representable graph.

1.1 The Flow Existence Problems

All of the bar-representable cases may be reformulated as versions of the following two flow problems. Informally, the first problem is, given a directed acyclic capacitated network, find a feasible flow such that the edges with non-zero flow form a planar graph. The second problem is, given an *undirected* graph, determine an orientation of the arcs that creates a directed acyclic network for which the first problem is solvable.

1.1.1 The Planar Flow Existence Problem

A directed graph is called **capacitated**, if associated with each arc a, there is a weight $c(a)$, representing the maximum capacity of the arc a.

A **source** (respectively **sink**) of a directed graph is defined as a vertex with no incoming (respectively outgoing) arcs incident upon it.

A **network** N is defined as a (possibly non-planar) directed capacitated acyclic graph with unique source s and unique sink t joined by the arc $< s, t >$ with associated capacity of 1.

A **feasible flow** f of a network N, is a function mapping $A(N) \rightarrow \Re$, such that:

- for all arcs a, $0 \leq f(a) \leq c(a)$.
- for all vertices $v \neq s, t$, $\sum_w f(< v, w >) = \sum_u f(< u, v >)$.

A feasible flow in which $f(a)$ is strictly greater than 0, for all arcs a, will be called a **positive flow**.

Define N^f as the directed graph with vertex set $V(N)$ and arc set $A(N^f)$ given by $< u, v >$ is a member of $A(N^f)$ iff $f(< u, v >) > 0$. The arcs $< s, u >$ and $< v, t >$ of N^f for which $f > 0$ will be referred to as the **extreme** arcs. Arcs not incident upon s or t will be referred to as **internal** arcs.

If a flow is feasible and also uses all internal arcs to full capacity (i.e. $f(a) = c(a)$), then the flow will be called **full**.

The **Planar Positive Flow (PPF)** problem can now be stated as, given a network N, determine if there exists a positive flow f such that N^f is planar. (The flow f will then also be called planar).

The **Planar Full Flow (PFF)** problem is to determine if there exists a *full* flow f on network N, such that N^f is planar.

1.1.2 The Undirected Planar Flow Problems

Define an **orientation** of an undirected capacitated graph M with specified vertices s and t as an imposed direction on each edge of M, thus creating a directed graph. This function mapping edges to arcs (directed edges) will be denoted by g, and the resulting directed capacitated graph will be denoted by $g(M)$.

The function g will be called **network creating** if in the resulting graph, $g(M)$:

- there is no directed cycle, and
- s and t appear as the unique source and sink respectively.

The **Undirected Planar Positive Flow (UPPF)** problem is, given an undirected graph M with specified vertices s and t, to determine if there exists a network creating orientation g, such that there is a positive flow f and $g(M)^f$ is planar.

The **Undirected Planar Full Flow (UPFF)** problem is to determine such an orientation which also admits a planar *full* flow.

1.2 Thesis Outline

In general, the solutions to the unordered cases rely heavily on the solutions to the ordered cases and hence these two are paired together in each chapter. In chapter 2 the directed case is solved, namely under what conditions is a directed graph a *polarized* bar-representable graph? Characterizations for both the directed weighted and the directed unweighted cases are presented. These graph-theoretic characterizations for polarized and measured-polarized bar-representable graphs lead to linear time algorithms for their detection and for the construction of an associated layout.

Determining if an unweighted undirected graph is bar-representable was the original motivation of this thesis. Its solution (a linear time characterization and layout algorithm) is described in chapter 3. The construction of the layout is based upon a modification of an s-t-numbering of a graph.

The problem of determining whether a weighted, undirected graph is a measured bar-representable graph is solved in chapter 4 by reformulating it in terms of the full flow existence problem. A sequence of successively less constrained subproblems leads to the final algorithm. The algorithms in this chapter for both the ordered and unordered cases are fairly involved but their time complexities are shown to be polynomially bounded.

As circumstantial evidence that the problems in chapter 4 are inherently difficult, chapter 5 contains proofs that two variants of the undirected flow existence problem are NP-Complete. The problem of finding an acyclic orientation of an undirected capacitated network for which a (possibly) non-planar full flow exists, is shown to be NP-Complete. The second NP-Complete variant is to determine an orientation of a capacitated undirected graph for which a planar full flow exists *but with directed cycles permitted* in the network.

Chapter 6 outlines some related visibility results and open problems; in particular, multiple lines-of-sight and curves other than bars are discussed.

In conclusion, in chapter 7, the main contributions of the thesis are summarized.

1.3 Previous Work and Motivation

A few authors have considered visibility problems involving line segments, but the notion of *visibility* is defined in various ways throughout the literature. Perhaps the most comprehensive treatment of visibility is by Tamassia and Tollis [23], who distinguish between:

- **strong-visibility,** in which a pair of bars is visible if a horizontal *line segment* can be drawn between them that intersects no other bar and
- **ϵ-visibility**, in which a horizontal *rectangle* of non-zero width not properly intersecting other bars is required between bars to be visible.

Remark: Note that ϵ-visibility is the model of visibility assumed in this thesis and as a consequence, in the weighted cases, edge weights are strictly greater than 0. One justification for this choice of model is that it is not clear how edges of 0 weight would be interpretted. It also provides a solution to the "unit-visibility" case, in which a minimum amount of visibility between bars is specified *a priori*, again since 0-weighted edges are not created. Finally, the ϵ-visibility model produces a larger class of graphs than the strong-visibility model, in fact, a superset. However, note that the two models are equivalent if the bars are redefined to be interval line segments with a closed top and an open bottom.

The characterization of bar-representable graphs under the ϵ-visibility model as described in chapter 3, and first presented in [28], was independently discovered by Tamassia and Tollis [23] who also conjectured a characterization for strong-visibility bar-representable graphs. A similar result relating to blocking relations was independently presented by Rival and Urrutia [19].

Schlag *et al* [21] present linear time algorithms for computing all visibility pairs (and hence the visibility graph) of bars in a given layout, employing a strong visibility model, if the bars are presorted by X-coordinate, top Y-coordinate, or bottom Y-coordinate. The application to VLSI layout compaction is also discussed.

Luccio *et al* [16], use the strong-visibility model, however the layout of bars is severely constrained in their work and under their given restriction, the two models of visibility are equivalent. The Y-coordinates of all endpoints of bars are assumed to be different. As a consequence, all the interior faces of the associated planar ordered visibility *multi-graph* are triangular (bounded by 3-cycles). The authors characterize the visibility graphs that arise under this model as **ipo-triangular** (transformable into a triangular multi-graph by successive duplications of existing edges). Their constructive proof provides a layout scheme for a given visibility multi-graph M. This algorithm appears to be linear in the number of edges of M. (For *triangular* multi-graphs, the number of edges is also linear in the number of vertices). However, the complexity of determining whether a graph is ipo-triangular, is not addressed.

Several authors have investigated graphs and their layouts under a *weak-visibility* [4] model. A graph G is a **weak visibility** graph if there exists a layout of bars corresponding to the vertices of G such that, *if* (u, v) is an edge of G *then* the corresponding bars $\overline{u}$ and $\overline{v}$, are visible (rather than *iff*). Thus every weak-visibility graph is a subgraph of some strong-visibility graph. This notion of weak-visibility graphs is motivated by practical VLSI routing concerns in which every pair of *adjacent* vertices in a specified graph must have a "channel" or route available in the layout and the associated bars are physically joined along these visibility channels. Extra visibilities between bars are ignored by not physically connecting them.

[4] using the notation of Tamassia and Tollis

Duchet *et al* [5], showed that every planar graph is a weak-visibility graph (or equivalently that every planar graph is a subgraph of some strong-visibility graph). Tamassia and Tollis [23], presented an $O(|V|)$ algorithm to create a weak-visibility layout.

Rosenstiehl and Tarjan [20], also employ the weak-visibility model and assume the given graph is biconnected. (If it is not, edges may be added and ignored in the final layout). A notion equivalent to an s-t numbering (see chapter 3) is that of a **bipolar orientation**, in which the edges are directed to form an acyclic directed graph with unique source (no incoming arcs) and unique sink (no outgoing arcs). The construction of an associated layout of a biconnected graph in chapter 3 is similar but not identical to theirs. They note that the dual graph may be laid out perpendicularly and "interlocked". The area of the layout is of particular concern and they conjecture that finding a bipolar orientation that minimizes the layout area is NP-Hard. A technique for generating the bipolar orientations (there may be an exponential number of them) is provided.

Garey, Johnson and So [10], proposed an elegant method of testing printed circuit boards for the existence of short circuits based on colouring the associated line-of-sight graph. A **net** is defined as a tree of vertical and horizontal line segments (representing a conductor path on a printed circuit board). The problem is, given a set of n non-intersecting nets, to test if some pair of nets contains a short circuit. The brute force method tests all pairs of nets by applying an electrical signal to one net and checking if it propagates to the other net. Under various assumptions about the form of the short circuits, they note that not all $\binom{n}{2}$ nets need be tested. They show that if all short circuits were known to occur along a vertical or horizontal line-of-sight for example, then the resulting line-of-sight graph could be coloured with at most 12 colours. The nets of each colour are then connected to form 12 supernets and the $\binom{12}{2}$ pairwise tests of the supernets reveal the existence of a short circuit. In addition to this 2-line-of-sight version, they investigate a 1-line-of-sight case and the case when the lines-of-sight are of bounded length. In [30], Johnson notes that a 2-line-of-sight graph must have thickness [5] at most two and poses the problem of determining if all graphs of thickness two have a 2-line-of-sight layout.

In an *orthogonal polygon*, all sides are vertical or horizontal. A natural visibility problem is to consider the two graphs that arise from using a vertical line-of-sight with the horizontal sides and a horizontal line-of-sight with the vertical sides. Booth and O'Rourke [17] noted that these *edge visibility graphs* must be trees and investigated which pairs of trees "mesh" together to represent polygons. In particular, they show that a subclass of trees, called caterpillars, mesh with any tree to form an orthogonal polygon. The complete characterization however, remains an open problem.

El Gindy [7] considered the visibilities among the *corners* of polygons with the lines-of-sight not fixed and discovered some classes of these *vertex visibility graphs.* For example, every maximal outerplanar graph is a vertex visibility graph. Again, the complete characterization is still an open question. (A good exposition is contained in [17].) Ghosh [11], has proven three necessary conditions for a graph to be a vertex visibility graph and conjectured that the conditions are sufficient for simple polygons.

A significantly different definition of a visibility graph G_s for a set S of n line segments, requires that the vertices of G_s correspond to the $2n$ *endpoints* of the line segments. Vertices are adjacent if the line segment joining the associated endpoints intersects no other line segment of S. Such a graph need not be planar. The lower and upper bounds on the number of edges are $5n - 4$ and $2n^2 - n$. (See Shen and Edelsbrunner [22]). Welzl [26] provides an $O(n^2)$ time and space algorithm for constructing the visibility graph. These

[5]Graph G has thickness k if there exists a decomposition of G into k planar subgraphs and k is minimum.

results are optimal in the worst case, however determining an output-sensitive algorithm (i.e. one whose complexity is a function of the size of the output as well as the input), remains an open problem. These endpoint visibility graphs have been used extensively in some shortest path obstacle avoidance problems - see for example [15].

Rival and Urrutia [19] consider a separation problem on convex figures in the plane. The problem is to remove one at a time the figures from the collection by translation along an assigned direction without collision. In particular, they solve a one-directional version in which all figures are assigned the identical direction of translation. They define an **obstructing relation** between pairs of objects: "For figures A and B we say that B *obstructs* A if there is a line joining a point of A to B which follows the direction assigned to B." A binary "blocking" relation is defined between a pair of objects if there exists a sequence of obstructing objects between them. Their characterization of this blocking relation involves *truncated planar lattices.* A **lattice** is an ordered set in which every pair of elements has a least upper bound and a greatest lower bound; it is **planar** if its "diagram" (graph representation in the plane) can be drawn without crossings. A **truncated planar lattice** is obtained by deleting the (unique) maximum and minimum elements. They prove that there is a one-to-one correspondence between the class of all (one-directional) blocking relations and the class of truncated planar lattices. This characterization is equivalent to the characterization of embedded bar-representable graphs in chapter 3 and indicates a useful relationship between visibility problems and ordered sets.

1.4 Algorithmic Issues

The underlying model of computation assumed for the formulation and analysis of algorithms is a random access machine (RAM). Only the comparison, addition and multiplication [6] of input values are allowed as operations on real numbers, with a uniform cost measure; see [18], for example, for a discussion of these issues. A *worst case* asymptotic complexity analysis will be used as the measure of the running time of the algorithms.

Strictly speaking, a visibility graph corresponding to a layout could include multiple edges. There could exist more than one maximal width visibility rectangle between a pair of bars; however, these rectangles will be assumed to have been merged by vertically displacing intervening bars as far as possible without disturbing the visibilities. The resulting layout then corresponds to a *simple* planar graph. Recall, that for simple, planar graphs, the number of vertices, $|V|$, the number of edges, $|E|$, and the number of faces, $|F|$, are all linearly related (a consequence of Euler's Theorem) and hence the time complexity of a graph algorithm may be presented as a function of the number of vertices.

Graphs are assumed to be represented in doubly-connected-edge-list form (DCEL), (see [18]). As a consequence, all of the algorithms have a space bound that is a linear function in the size of the input.

1.5 Notation

Standard graph theoretic terminology will be used throughout; see, for example, Harary [12]. Graphs are assumed to be connected, simple, and finite. The set of vertices of graph G will be denoted by *V(G)* and the edge set as *E(G)*. If G is directed, the arc set is $A(G)$. Edges will be denoted in parentheses (ex. (u,v)), and arcs (directed edges) in angle brackets (for example, the arc from u to v as: $< u, v >$). A plane embedding of graph G will be written as: $\mathcal{G}$. The existence of weights or directions on edges or topological information describing a specific ordering must be determined from context. The weight

[6] In fact, a slightly weaker model of computation not requiring multiplication could be used.

of an edge or arc e, will be denoted by weight(e). For a vertex in a directed graph, **inweight(v)** is $\sum_{u \in V}$ weight($< u, v >$) and **outweight(v)** is $\sum_{w \in V}$ weight($< v, w >$). If the inweight(v) is equal to the outweight(v), then v is **balanced**.

The bar corresponding to vertex v will be denoted by $\overline{v}$.

Since it is necessary to specify the layout of bars explicitly, the layout is assumed to be constructed in a 2-dimensional coordinate system. Each bar $\overline{v}$ will be defined by specifying $X(\overline{v})$, $TOP(\overline{v})$, and $|\overline{v}|$. (The x and y coordinates of the top of $\overline{v}$ and the length of $\overline{v}$.) Let $BOTTOM(\overline{v})$ be $TOP(\overline{v}) - |\overline{v}|$.

Chapter 2

Polarized, and Measured Polarized Bar-Representable Graphs

2.1 Introduction

Four of the eight bar-representable cases are solved in this chapter, namely all variations of the directed case: when edges are weighted or unweighted, and the graph is ordered or unordered. The most constrained bar-representable problem is, given an ordered weighted directed graph, determine if it is an embeddable measured polarized bar-representable graph. Theorem 1 characterizes such graphs and its constructive proof provides an associated layout in linear time. The layout procedures for all of the subsequent cases rely on this layout scheme. Each of these less constrained cases, in effect reduce to this case by special techniques.

Figure 2.1 shows an example of a directed weighted graph H, and an associated layout. Therefore, H is a measured polarized graph.

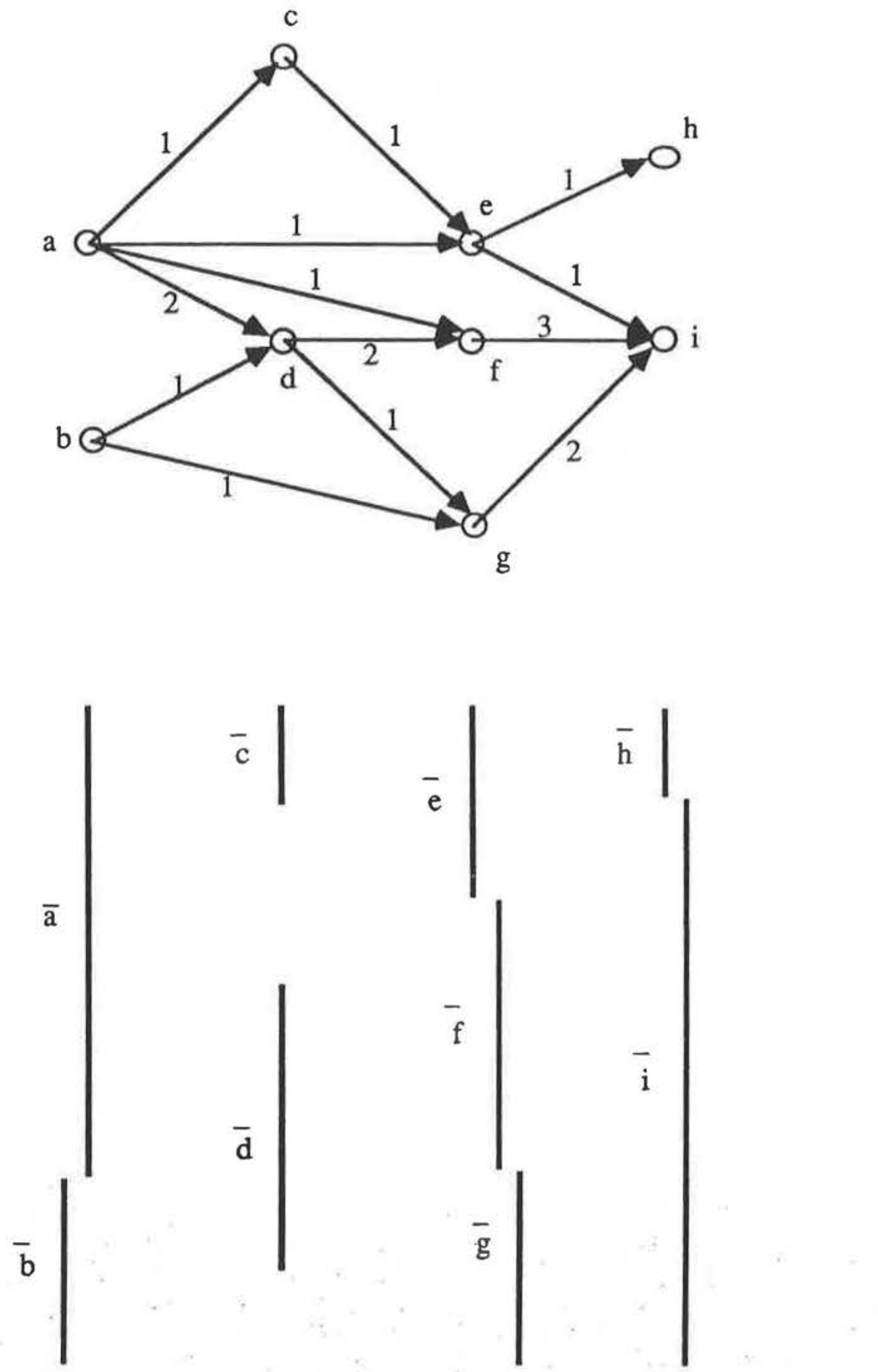


Figure 2.1: A Graph H and its Directed Weighted Bar Layout

2.2 Measured Polarized Graphs

2.2.1 Embedded Measured Polarized Graphs

Before characterizing embedded measured polarized graphs, a few properties of *bar layouts* should be noted.

Firstly, bars with equal amounts of left and right visibility correspond to balanced vertices - those with equal inweight and outweight.

Unbalanced vertices come in two types: those with insufficient inweight and those with insufficient outweight. Such vertices will now be shown to correspond to bars on the "outside" of the layout. Define a bar $\overline{u}$ as **exterior** in layout L if it is possible to draw an unbounded horizontal half line intersecting $\overline{u}$ and no other bar of L. (For example, $\overline{a}$, $\overline{b}$ and $\overline{h}$, $\overline{i}$ in figure 2.1). Exterior bars are further classified as having unblocked visibility to the right or to the left depending on the direction of the unbounded horizontal half line. The vertices corresponding to bars not totally blocked to the left (respectively right), have inweight $<$ outweight (respectively inweight $>$ outweight). Such unbalanced vertices certainly must appear on the exterior face of the ordering. Furthermore, their cyclic ordering about the exterior face must not be mixed - all vertices with insufficient inweight appear cyclicly together and all vertices with insufficient outweight appear cyclicly together. A particularly effective way to express this is to imagine two "super" bars bounding the layout - a leftmost bar $\overline{S}$ and a rightmost bar $\overline{T}$. The bottoms of each of these two bars will be the bottom of the lowest bar of the layout and the tops will be one unit higher than the top of the highest bar in the layout. In the resulting associated supergraph, there are exactly two unbalanced vertices: S and T. The two sets of previously unbalanced vertices are attached to S and T.

Define two sets of vertices V_1, V_2 on the exterior face of an ordering as **separable** if it is possible to introduce two new vertices v_1', v_2' into the exterior face and join each member of V_1 to v_1' and each member of V_2 to v_2' preserving planarity.

Theorem 1 *Ordered weighted digraph G is an embedded measured polarized bar-representable graph iff G corresponds to a planar embedding $\mathcal{G}$ with the following properties:*

- *$\mathcal{G}$ contains no directed cycle, and*
- *the vertices with insufficient inweight and the vertices with insufficient outweight of $\mathcal{G}$ are on the exterior face and separable.*

Proof:

The necessity of the above conditions is evident from the properties of layouts; their sufficiency can be demonstrated by constructing a bar layout from such a graph. The directions impose a partial ordering on the vertices which may be exploited to construct the associated layout. Embed $\mathcal{G}$ crossing free in the plane with the specified exterior face. Introduce two extra vertices in the exterior face: S and T. Vertices with insufficient inweight (respectively outweight) are joined to S (respectively T), thus balancing the in- and out- weights of all vertices (except S and T). The embedding is then modified; the vertices are arranged in discrete vertical "columns" with all arcs oriented left-to-right. The layout is formed by deforming the vertices carefully into bars as follows.

Deform the embedding of $\mathcal{G}$ with S and T attached so that the vertices lie in k vertical sections:

- S is in section 1

- $\forall v \neq S$, v is in section i iff all left neighbours of v are in sections $\leq i - 1$ and at least one left neighbour is in section $i - 1$.

Note that no two vertices of the same section are adjacent.

On each arc a passing through a section, subdivide the arc by introducing a "pseudo-vertex", thus creating a new plane embedding $\mathcal{G}'$. The two arcs created by the split each inherit the weight of arc a.

The vertices in a section will be indexed from top to bottom. Let $v_{i\,j}$ denote the jth vertex (real or pseudo) in section i. Construct an associated layout as follows.
Let $|\overline{S}| = \sum \text{weight}(< S, v >)$, $X(\overline{S}) = 1$, and $TOP(\overline{S})$ be arbitrarily chosen. Then the length and the top of each bar $\overline{v} \neq \overline{S}$ will be a function of the length and top of its neighbours in the previous section.

$|\overline{v}_{i\,j}| = \sum_k \text{weight}(v_{i-1,k}, v_{i\,j})$ (i.e. the sum of the weights of all back edges).

$TOP(\overline{v}_{i\,1}) = TOP(\overline{S})$

$TOP(\overline{v}_{i\,j}) = BOTTOM(\overline{v}_{i,\,j-1})$,$\forall j > 1$

$X(\overline{v}_{i\,j}) = i$

The construction assures that

$(v_{i\,j}\,, v_{i\,+\,1,\,k}) \in E(\mathcal{G}')$ iff $\overline{v}_{i\,j}$ *sees* $\overline{v}_{i\,+\,1,\,k}$ for the specified width.

Note that all vertices have at least one neighbour in the previous section and therefore the length of all bars is greater than 0.

Removal of the bars corresponding to pseudo-vertices and $\overline{S}$ and $\overline{T}$ yields a layout corresponding to G.

□

Remark: Note that the construction is linear in the number of vertices and edges of the embedding $\mathcal{G}'$. The number of pseudo-vertices introduced in the algorithm may be $O(n^2)$, however, the construction need not actually create (and later delete) them, they are used here merely to simplify the description of the algorithm. The length and top of bars corresponding to real vertices may be computed by considering the only the original arcs of G and thus the algorithm is linear in the size of G.

As an aside, note that $\mathcal{G}'$ can be viewed as a network in which the edge weights describe a flow from S to T. Specifically, a "preservation of flow" property holds, namely that in any vertical cut across the arcs, between two sections, the sum of the weights of the edges is constant. At each vertex, the sum of the weights of the back edges is equal to the sum of the weights of the forward edges.

Finally, note that the structure of an embedded measured polarized graph is so highly constrained that its layout as constructed above, is in fact, unique.

2.2.2 Measured Polarized Graphs (Unordered)

The unordered version hinges on the ordered case in the sense that G is a measured polarized bar-representable graph *if there exists* an ordering of G with the properties described in the previous subsection. Although there are potentially an exponential number of orderings, the existence of an appropriate ordering may be determined in linear time. The characterization is based on the following supergraph of G which includes two new vertices, S and T, connected to the vertices with excess outweight and inweight respectively.

2 Define $N_w(G)$ as:

$$\begin{aligned}
V(N_w(G)) &= V(G) \cup \{S, T\} \\
A(N_w(G)) &= A(G) \cup \\
&\quad \{< S, v > \mid v \in V(G) \text{ and inweight}(v) < \text{outweight}(v)\} \cup \\
&\quad \{< v, T > \mid v \in V(G) \text{ and inweight}(v) > \text{outweight(v)}\} \cup \\
&\quad \{< S, T >\}. \\
\text{Weight}(< S, T >) &= 1 \\
\text{Weight}(< u, v >) &= \text{weight}(< u, v >), \quad \text{for all arcs } < u, v > \in A(G) \\
\text{Weight}(< S, v >) &= \text{outweight}(v) - \text{inweight}(v) \\
\text{Weight}(< v, T >) &= \text{inweight}(v) - \text{outweight}(v)
\end{aligned}$$

Figure 2.2 shows $N_w(H)$ for the graph H of figure 2.1.

Theorem 2 *A weighted digraph G is a measured polarized bar-representable graph iff $N_w(G)$ is planar and contains no directed cycle.*

Proof:

$(\Leftarrow)$

The construction of $N_w(G)$ ensures that S and T are the only two vertices with insufficient in- and out-weights. Thus, a plane embedding of $N_w(G)$ with S and T on the exterior face represents an ordering for which $N_w(G)$ is an embedded measured polarized graph and the layout scheme of theorem 1 may be applied. Since $\overline{S}$ and $\overline{T}$ are extreme bars, their removal creates no new visibilities among bars and the resulting layout corresponds to G.

$(\Rightarrow)$

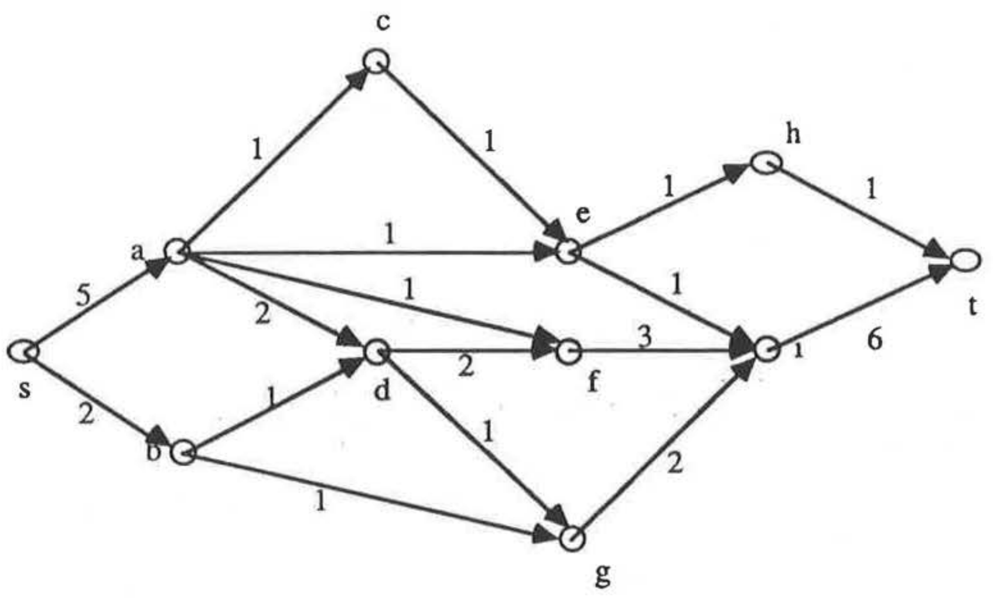


Figure 2.2: $N_w(H)$

Given a layout for G, adding extreme bars $\overline{S}$ and $\overline{T}$ yields an embedding of $N_w(G)$.

□

Remark: Construction of $N_w(G)$ can be clearly performed in linear time. There are well known linear time algorithms for both detecting cycles in directed graphs and for planarity testing - see [1], for example.

The graph $N_w(G)$ is unique and highly structured. The number of layouts of G is exactly the number of planar orderings of $N_w(G)$ with S and T on the exterior face.

2.3 Polarized Graphs

If the amount of visibility is not specified, the characterization is similar to the measured polarized case. Vertices with 0 indegree or 0 outdegree effect the role of the unbalanced vertices in the previous case and appropriate weights may be assigned to the arcs by the algorithm as the layout is constructed. The length of a bar is known when the amount of visibility provided by its left neighbours is established. This length may be shared arbitrarily among its right neighbours. Thus, a single left-to-right sweep suffices.

2.3.1 Embedded Polarized Graphs

Theorem 3 *Ordered digraph G is an embedded polarized bar-representable graph iff G corresponds to a planar embedding $\mathcal{G}$ with the following properties:*

- *G contains no directed cycle and*
- *all vertices with 0 indegree or 0 outdegree appear on the exterior face and are separable.*

Proof:

($\Rightarrow$)

In a layout of G, the bars with no left (respectively right) neighbours correspond to vertices with 0 indegree (respectively 0 outdegree). Such bars are clearly extreme and cyclically orderable and hence their associated vertices are on the exterior face and separable.

($\Leftarrow$)

Given a plane embedding $\mathcal{G}$, of G, add vertices S and T in the exterior face, and attach S to the vertices with 0 indegree and T to the vertices of 0 outdegree. Since $\mathcal{G}$ is embeddable and these sets of vertices are separable, the resulting graph is planar. Appropriate weights for the arcs must be determined.

For each arc $< v_{i-1,k},\ v_{i\ j} >$, choose the weight $\frac{|\overline{v}_{i-1,\ k}|}{d^f(v_{i-1,\ k})}$, where $d^f(v)$ is the forward degree of vertex v. Then $|\overline{v}_{i\ j}| = \sum \text{weight}(v_{i-1,k},\ v_{i\ j})$ (i.e. the sum of the weights of all back edges).

Let $|\overline{S}| = 1$, $X(\overline{S}) = 1$ and $TOP(\overline{S})$ be arbitrarily chosen.

Since all vertices other than S and T now have balanced in- and out-weights, the layout scheme described in the proof of theorem 1 provides an associated layout for $\mathcal{G}$.

□

The sufficiency of these conditions is perhaps somewhat surprising, as in a layout, there may exist a bar $\overline{b}$ adjacent to $\overline{S}$ that does *not* have indegree(b) $= 0$, for example. Such a bar could be blocked from $\overline{S}$ by increasing the weight of one of its left neighbours appropriately. The above construction leads to the following corollary which shows the existence of a *canonical* layout of ordered digraph G.

Corollary 1 *Every embedded polarized graph has an associated layout in which all and* only *the bars corresponding to vertices of 0 indegree and 0 outdegree are on the exterior.*

Proof:

Let $\overline{u}$ be a bar in a layout with $\overline{S}$ and $\overline{T}$ added and visible to $\overline{S}$ (without loss of generality) with indegree(u) > 0. Consider a maximal visibility rectangle involving $\overline{S}$ and $\overline{u}$. In the associated graph, the cyclic ordering about u must be $S, v, \ldots, w, S$ and one of v or w (or both) must be a left neighbour of u and the corresponding bar, say $\overline{v}$, may be lengthened for the width of the visibility between $\overline{S}$ and $\overline{v}$. See figure 2.3.

Note that $\overline{v}$ may also have indegree(v) strictly greater than 0, however, the above process may be applied to $\overline{v}$ and repeated application of the process will ultimately result in the visibility rectangle being blocked by a bar with no left neighbours.

□

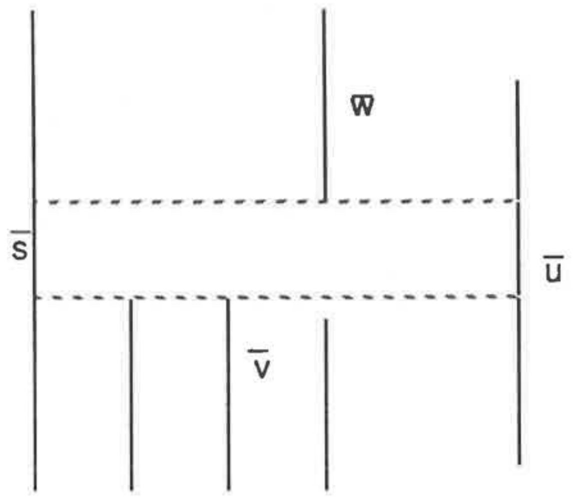


Figure 2.3: Blockable Visibility

Remark: This ability to block a visibility between a pair of bars without introducing a new visibility nicely demonstrates the difference between strong- and ϵ-visibility. In figure 2.4, under an ϵ-visibility model, the visibility between $\overline{u}$ and $\overline{x}$ may be blocked by a pair of bars $\overline{v}$ and $\overline{w}$ that abut without seeing each other. However, with a strong-visibility model, either $\overline{u}$ and $\overline{x}$ or $\overline{v}$ and $\overline{w}$ are visible under any minor perturbation of the ends of $\overline{v}$ and $\overline{w}$.

Tamassia and Tollis [23] strengthen the undirected version of this informal argument to show that the class of strong visibility graphs is a proper subset of the class of ϵ-visibility graphs.

The following crucial lemma will be exploited in later sections and will be referred to as the "consecutive-in-out-flow property" [1].

[1] Both Rosenstiehl and Tarjan [20] and Tamassia and Tollis [23] also note and prove this property.

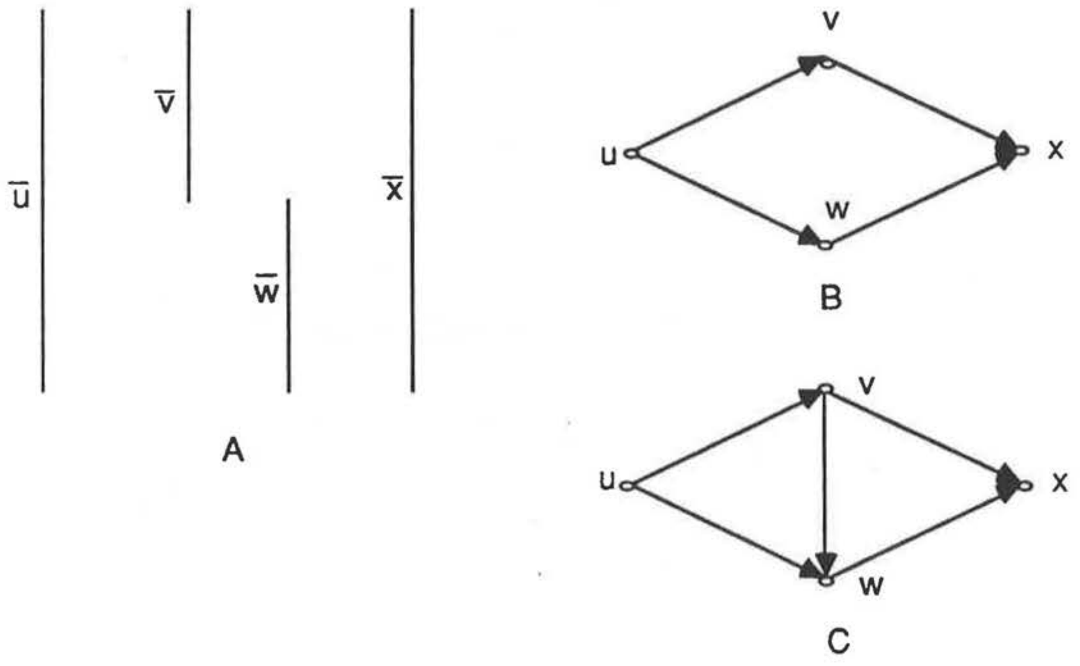


Figure 2.4: Strong vs. ϵ-Visibility

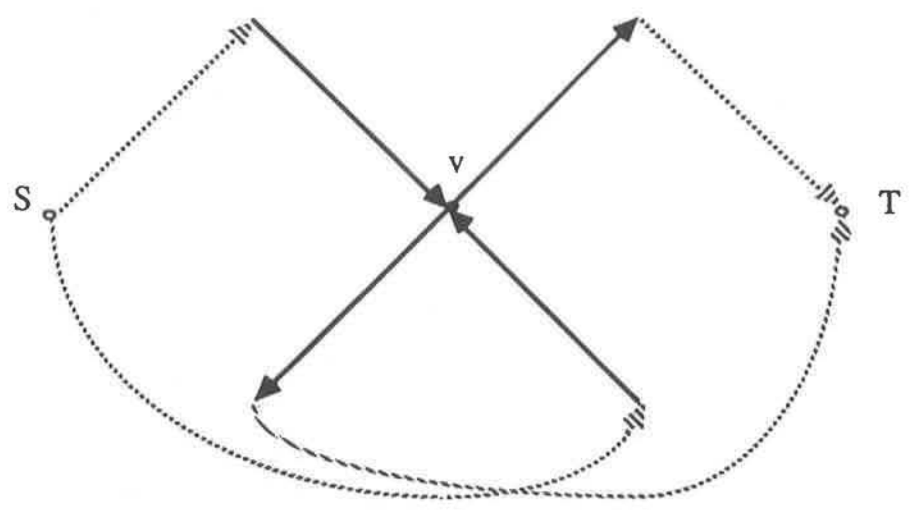


Figure 2.5: Violation of The In-Out Property

The bars of a layout are partially ordered by their left-to-right visibility relation. Thus, in the associated directed graph, there must be no directed cycle. An important consequence of this acyclic requirement is that the in and out arcs about each vertex in any embedding are consecutively ordered.

Lemma 1 *For each vertex v of ordered directed bar-representable graph G, the in-arcs (respectively out-arcs) are arranged consecutively about v.*

Proof:

Consider an embedding of G with S and T included as created in the previous proof, and in which some vertex v has at least two in and two out arcs arranged as in figure 2.5. The two in-arcs must be a part of a path from S, creating a Jordan curve, with one of the two out-arcs at v on the inside and the other on the outside. Since these two arcs must each form a path to T, one of these two paths must intersect the Jordan curve, creating a directed cycle.

□

Note that the converse of Lemma 1 is not true. If all vertices satisfy this property, acyclicity is *not* guaranteed. (As a simple counter-example consider a unit-weighted directed ring of vertices.)

2.3.2 Polarized Graphs (Unordered)

As in the measured polarized case, the unordered version of the polarized case hinges on the existence of some ordering with the required properties. As before, an appropriate ordering can be found without resorting to enumeration of all orderings. In fact, a layout is constructible in linear-time by considering a supergraph of G, $M(G)$, defined as follows:

$$\begin{aligned} V(M(G)) &= V(G) \cup \{S, T\} \\ A(M(G)) &= A(G) \cup \{< S, v > \mid v \in V(G) \text{ and indegree}(v) = 0\} \cup \\ &\quad \{< v, T > \mid v \in V(G) \text{ and outdegree}(v) = 0\} \cup \{< S, T >\}. \end{aligned}$$

Theorem 4 *Digraph G is a polarized bar-representable graph iff $M(G)$ is planar and contains no directed cycle.*

Proof:

($\Rightarrow$)

By Corollary 1, there exists a canonical layout for G in which the only bars on the exterior have indegree of 0 or outdegree of 0. Insertion of a leftmost bar $\overline{S}$ and a rightmost bar $\overline{T}$, each with bottom defined by the lowest bar and top defined as 1 + top of the highest bar, yields a layout corresponding to $M(G)$.

($\Leftarrow$)

If $M(G)$ is planar, then there exists an embedding in the plane with S and T on the exterior face. Since S and T are the only two vertices with 0 indegree and 0 outdegree respectively, they are separable. Finally, $M(G)$ is acyclic, and therefore all of the conditions of theorem 3 are satisfied and the construction described in the proof may be applied.

□

2.4 Reformulation as a Flow Existence Problem

Construction of a layout for all four of the directed cases examined in this chapter may be formulated in terms of finding a planar feasible flow in a directed acyclic network. The network is defined slightly differently for each case.

2.4.1 Ordered Directed Weighted Graphs

Given an ordered directed weighted acyclic graph G, define $N(G)$ as the network obtained by:

- introducing two new vertices s and t,
- associating the capacity weight($< u, v >$) with each arc $< u, v >$ of G,
- joining s and t to *all vertices on the exterior face* of G, with associated capacity ∞,
- joining s to t with capacity 1.

Recall that the Planar Full Flow problem requests a feasible flow function f' such that $N^{f'}(G)$ is planar and full. (There may of course, exist no such flow).

Corollary 2 *$N(G)$ has a planar full flow iff G is an embedded measured polarized graph.*

Proof:

($\Leftarrow$) This follows directly from Theorem 4.

($\Rightarrow$)

Since the capacities of all arcs of G are specified, determining a full flow function f', amounts to selecting which arcs $< s, u >$ and $< v, t >$ are non-zero and selecting an appropriate flow through each such arc. If some vertex not joined to s or t has inweight not equal to outweight, then clearly there exists no feasible flow function. Vertices with insufficient inweight (respectively outweight) must receive from s (respectively t) an amount of flow sufficient to satisfy the balancing condition. Satisfying the planarity requirement for $N^{f'}$ implies that the two groups of unbalanced vertices are separable.

□

Thus, the results of section 2.2.1 solve both the problem of characterizing the ordered directed weighted graphs for which the planar flow existence question is answered in the affirmative, and the problem of providing the feasible flow.

2.4.2 Directed Weighted Graphs

If the ordering of a directed weighted acyclic graph G is not specified, $N(G)$ is defined as above except s and t are joined to *all vertices* of G with associated capacity ∞.

Corollary 3 *$N(G)$ has a planar full flow iff G is a measured polarized graph.*

Proof:

Recall the graph $N_w(G)$ defined in section 2.2.2. Graph G was shown to be a measured polarized bar-representable graph iff $N_w(G)$ is planar and acyclic (Theorem 2). Network $N_w(G)$ is identical to the graph $N^{f'}(G)$ for feasible flow f' since the same attachments and weights to s and t are required and both graphs are required to be planar and acyclic. Thus, determining these attachments and weights is equivalent to construction of $N_w(G)$.

□

As a consequence of the above two equivalences, for both the ordered and unordered versions of the measured polarized case, the planar full flow through G is unique if it exists.

2.4.3 Ordered Directed Graphs

If the weights of ordered directed acyclic graph G are not specified, then it is the Planar Positive Flow problem that remains to be solved. For this case, the corresponding Planar Positive Flow formulation defines $N(G)$ as the network with:

- capacity ∞ associated with each arc of G,
- s and t joined to all vertices on the exterior face of $\mathcal{G}$ with associated capacity ∞,
- arc $< s, t >$ with associated capacity 1.

Corollary 4 *Network $N(G)$ has a positive planar flow iff G is an embedded polarized bar-representable graph.*

Proof:

The outweight (respectively inweight) of vertices with 0 indegree (respectively 0 outdegree) is unknown but strictly greater than 0 in any positive flow. Therefore, all such vertices must be attached to s (respectively t) and hence appear on the exterior face of G separably. Furthermore, a positive flow is acyclic since $N(G)$ is an acyclic network. Theorem 3 requires exactly the same conditions for G to be an embedded polarized bar-representable graph.

The weighted graph corresponding to a layout of G with super bars $\overline{S}$ and $\overline{T}$ corresponds to a planar acyclic weighted graph which is equivalent to $N^{f'}(G)$ for some positive flow f'.

□

Recall that Corollary 1 ensured the existence of a canonical layout of G for embedded polarized graphs. In flow terms, the interpretation of this corollary is the existence of a feasible flow in which only the vertices of 0 indegree and 0 outdegree receive non-zero flow from s and t respectively.

2.4.4 Directed Graphs

The situation for *unordered* unweighted directed graph G is similar to that for the ordered case. Define $N(G)$ as in the previous case except s and t are joined to all vertices of G with associated capacity ∞.

Corollary 5 *Network $N(G)$ has a planar positive flow iff G is a polarized bar-representable graph.*

Proof:

Theorem 4 characterized the polarized bar-representable graphs in terms of the unweighted supergraph $M(G)$. A layout for $M(G)$ was implicitly constructed via the technique described in the proof of Theorem 3. The resulting weighted digraph exhibits the same properties as $N^{f'}(G)$, for a particular positive planar flow f', since in any positive flow, the vertices with 0 indegree require non-zero flow assistance from s and those with 0 outdegree must issue non-zero flow to t.

Let f' be a planar positive flow. The weighted graph $N^{f'}(G)$ is planar and acyclic. Removal of the s and t connections yields a weighted version of digraph G. Theorem 2 may be used to show that G is a measured polarized bar-representable graph since the supergraph $N_w(G)$ is identical to $N^{f'}(G)$.

□

Chapter 3

Bar-Representable Graphs

3.1 Introduction

If neither the weights nor directions for G are specified, the problem of determining a layout of bars may be reformulated as making suitable selections for each edge. However, it is the *structure* of the graph that is the critical factor in bar-representability. This structural requirement has been inherent but somewhat hidden in the results of the previous chapter. The undirected unweighted characterization more explicitly displays the fundamental nature of bar-representability.

When constructing a layout for graph G, the difficulty is not so much in adjusting bars to *achieve* visibility, but rather to *deny* visibility between non-consenting bars. (In weak-visibility models, this denial is not an issue. Recall for example, the results of Duchet *et al* [5] who showed that *all* planar graphs have a layout under the weak-visibility model.)

Biconnected components are the "building blocks" of bar layouts and it is the manner of their connection that determines bar-representability. G is **biconnected** if it contains no articulation vertex [1]. Note that it is neither the existence nor the number of articulation

[1] A vertex is an **articulation vertex** if its removal would disconnect the graph.

vertices that is significant. For example, all trees are bar-representable, yet all non-leaf vertices of trees are articulation vertices. Nonetheless, the first crucial step in characterizing the class of bar-representable graphs is to prove that all biconnected graphs are bar-representable. Any bar-representable graph will then be shown to have a biconnected planar "completion".

The intuition for the characterization may be informally described as follows. Consider an embedding of a bar-representable graph consisting of several biconnected blocks. Each block must have a leftmost and rightmost bar in any layout. If there exists an articulation vertex not on the exterior face of the embedding, then some biconnected block is contained in an interior face and can be shown to have an extreme bar that must have access to the outside (i.e. be unblocked in one direction), which leads to a contradiction. It is not sufficient to have only all the articulation vertices on the exterior face in the embedded case; each biconnected block must be embedded in the exterior face also. For the unembedded case, it does suffice to check that all articulation vertices are embeddable on the exterior face, as the biconnected blocks may then also be safely embedded. The characterization for the unembedded version is expressed in a slightly more elegant (but perhaps less intuitive) form.

The layout for biconnected graphs is the first important step.

An alternate definition of biconnectedness is that G must be s-t-numberable. An **s-t-numbering** of graph G with distinguished edge (s,t), is a one-to-one function $\lambda : V \rightarrow \{1, 2, ..., |V|\}$ such that for all vertices $v \neq s, t$, there exist vertices u, w adjacent to v with $\lambda(u) < \lambda(v) < \lambda(w)$. $\lambda(s) = 1$, $\lambda(t) = |V|$.

Even and Tarjan showed [8] that an s-t-numbering of a biconnected graph may be computed in linear time. An s-t-numbering of G has a natural and useful relationship to a bar layout of G. In bar terms, the s-t-numbering asserts that every bar must have a left neighbour and a right neighbour - except for a leftmost and rightmost bar. As shall be demonstrated, with a little care a layout for a biconnected graph can be generated from its s-t-numbering.

Lemma 2 *All planar biconnected graphs are bar-representable.*

Proof:

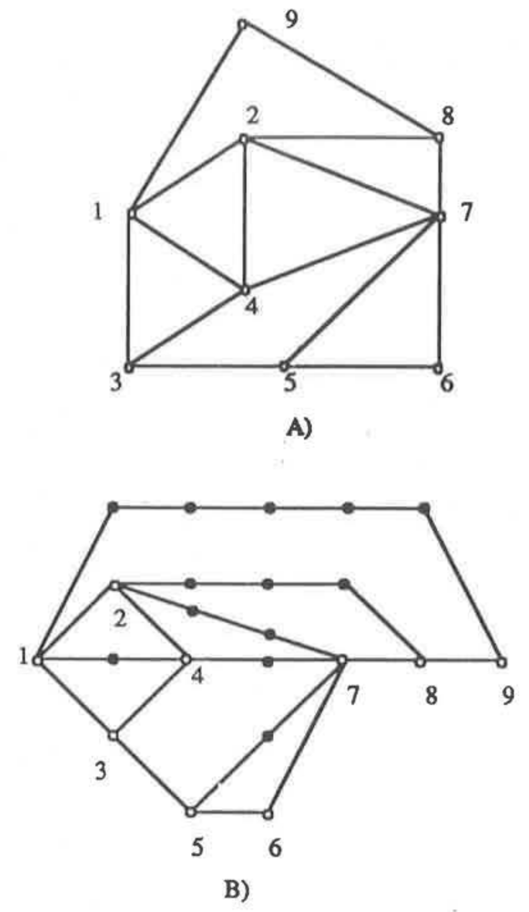
Choose any edge as the distinguished edge (s, t) and compute an s-t numbering for biconnected planar graph G. Create a directed acyclic graph G' by orienting each edge (u, v) from u to v, if $\lambda(u) < \lambda(v)$ in the s-t numbering of G. If G' can be shown to be a polarized graph, then G is bar-representable. Consider $N(G')$ as defined in section 2.3.2. Since s and t are the only vertices with respectively 0 indegree and 0 outdegree, the only additional arcs in $N(G')$ are $< S, s >$, $< t, T >$ and $< S, T >$ and these arcs create no directed cycle in $N(G')$. Since there is a plane embedding of G' with s and t on the exterior face, therefore $N(G')$ is planar. These are the conditions required by Corollary 4 for G' to be a polarized bar-representable digraph and the layout scheme of the previous section provides a layout for G' and therefore also for G, showing that G is bar-representable.

□

Figure 3.1 exhibits the three stages of the algorithm for this case.

3.2 Embedded Bar-Representable Graphs

If an ordering of G is specified, then the associated embedding must be "potentially biconnectable", in the sense that the edge connections to super vertices S and T must create a planar biconnected graph. The actual vertices chosen to be joined to S and T are not



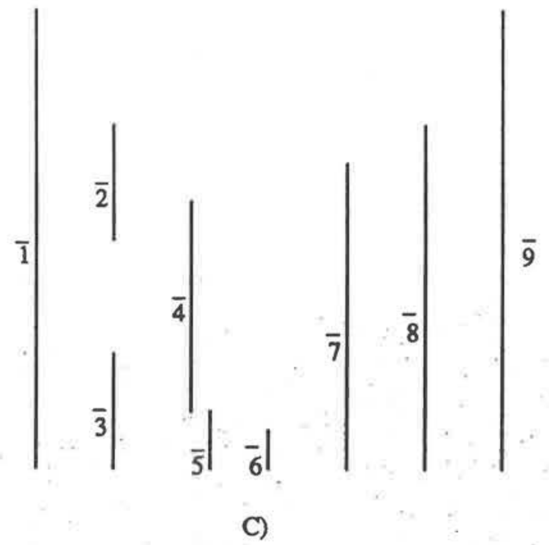


Figure 3.1: a) An s-t Numbered Graph, b) Embedded Into Sections, and c) a realization.

as clearly imposed as for the polarized cases. This flexibility reflects the fact that there are many more possible layouts when the edges are undirected. However, note that not any planar biconnected supergraph of G is sufficient; the bars corresponding to any newly added vertices must be removable without creating visibilities that are not edges in G. Since $\overline{S}$ and $\overline{T}$ will appear in the super layout as extreme bars, their removal creates no new visibilities.

An **end-block** of a graph G is defined as a biconnected block containing exactly one articulation vertex of G. It has been shown, [2], that if a graph G has at least one articulation vertex, then G has at least two end-blocks. These end-blocks play a crucial role in the layout of bar-representable graphs.

Theorem 5 *Ordered graph G is an embedded bar-representable graph iff G has a plane embedding $\mathcal{G}$ in which all biconnected components appear in the specified exterior face.*

Proof:

($\Rightarrow$)

(by contradiction) Suppose there exists a layout corresponding to an ordered graph G in which not all biconnected blocks appear on the exterior face. Let B' be a maximal connected collection of biconnected components of G in an interior face of a plane embedding $\mathcal{G}$. Let a be the unique articulation vertex that connects B' to $\mathcal{G}$. Any layout for B' must contain at least one leftmost and one rightmost bar, at most one of which may correspond to $\overline{a}$. Suppose, without loss of generality, that $\overline{c}$ is a rightmost bar of B' and $\overline{c} \neq \overline{a}$. Since B' is interior, so is $\overline{c}$ and hence, $\overline{c}$ has no unblocked visibility to the right. But if $\overline{c}$ is a rightmost bar of B', then $\overline{c}$ must be visible to some bar of $G - B'$, which contradicts the assumption that a was an articulation vertex. Hence, no such layout for G exists.

($\Leftarrow$)

If *all* biconnected blocks are on the exterior face of some embedding of $\mathcal{G}$, then the end-blocks of $\mathcal{G}$ certainly are. Let $E_1, E_2, \ldots, E_k$ be the endblocks of $\mathcal{G}$ with associated articulation vertices $a_1, a_2, \ldots, a_k$. Define G' as the ordered graph obtained by:

- inserting vertices S and T in the exterior face of G,
- joining S by an edge to a vertex $v \in E_1$ on the exterior face of $\mathcal{G}$ other than a_1,
- joining T to each of the end-blocks $E_2, E_3, \ldots, E_k$ with an edge (v_i, T), $v_i \in E_i$, $v_i \neq a_i$, for $2 \leq i \leq k$ and v_i is on the exterior face of G,
- adding the edge (S, T).

G' is planar since $\{v_1\}$ and $\{v_2, \ldots, v_k\}$ are separable.

Furthermore, G' is biconnected as shall now be shown. Suppose G' is not biconnected. Then there exists an articulation vertex v whose removal disconnects G'. Since G' is a supergraph of $\mathcal{G}$, the only candidates for v are: S, T, and the articulation vertices of $\mathcal{G}$.

- $v \neq S$. By construction, $G' - S$ is the graph $\mathcal{G}$ with T attached to all but one of the endblocks of $\mathcal{G}$, and hence $G' - S$ is not disconnected.
- $v \neq T$. $G' - T$ consists of the graph $\mathcal{G}$ with S attached and hence $G' - T$ is not disconnected.

Therefore, v must be an articulation vertex of $\mathcal{G}$ whose removal disconnects $\mathcal{G}$ into connected components $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_m$. Each $\mathcal{G}_i$ either:

- is biconnected, in which case it was an end-block of $\mathcal{G}$ and hence connected to S or T in G', or

- contains an articulation vertex and hence at least one of the endblocks of $\mathcal{G}_i$ was an end-block of $\mathcal{G}$ and hence connected to S or T in G'.

Since S and T are connected, all the $\mathcal{G}_i$ are, in fact, connected in G'. Hence G' contains no articulation vertex.

Since G' is planar and biconnected, by Lemma 2 there is a layout for G' with (S,T) as the distinguished edge used for the $s-t$ numbering in the construction in the proof. The bars $\overline{S}$ and $\overline{T}$ can be removed without introducing new visibilities since they are on the exterior, to obtain a layout for $\mathcal{G}$.

□

3.3 Bar-Representable Graphs (Unordered)

Given an unordered graph, checking for bar-representability is equivalent to determining if there exists an ordering with all biconnected blocks on the exterior face,. Again, enumeration of all orderings is computationally intractable. Two characterizations of bar-representability will be presented. The first is primarily constructive and the second is more graph-theoretic in nature; each leads to linear-time algorithms for determining the bar-representability of a graph.

The following modification of s-t numberings allows unordered bar-representable graphs to be characterized exactly.

Firstly, for a numbering scheme, λ, of the vertices of a graph, a **local λ-max** (respectively **λ-min**) is defined as a vertex which has no higher (respectively lower) λ-numbered neighbour.

A planar graph G has an s-t* **numbering** if there exists a plane embedding of G and a one-to-one function $\lambda : V \rightarrow \{1,2,...,|V|\}$ such that all local λ-max and local λ-min vertices are on the same face and separable.

Lemma 3 *G is bar-representable iff G is (planar and) has an s-t* numbering.*

Proof:

($\Leftarrow$) Introduce two new vertices S and T into the exterior face of the plane embedding of G, and join all local λ-max (respectively λ-min) vertices crossing free to S (respectively T), and join S to T, to form a graph which is clearly S-T numberable and therefore by Lemma 2 has a bar layout L in which the $\overline{S}$ and $\overline{T}$ bars are the left-most and right-most bars. Removal of the S and T bars, yields a layout representing G.

($\Rightarrow$) Define a **standard form** layout as one in which all bars are embedded (without overlapping) on the lines $X(\overline{v}) = 1, 2, \ldots k$ and all bars are as far to the left as possible.

A layout can be converted to standard form by a sequence of horizontal translations without affecting the visibility relations.

If the bars of a layout in standard form are numbered from left to right and top to bottom, then all bars with no left neighbour appear on the line $X = 1$. In the associated ordered graph G, the vertices corresponding to these bars are local λ-minima. Since these λ-min vertices appear together on the exterior face, the two groups of local λ- min and λ-max vertices are separable and hence the numbering is, in fact, an s-t^* numbering of the associated ordered graph G.

□

The following lemma shows an equivalent characterization of the bar-representable graphs.

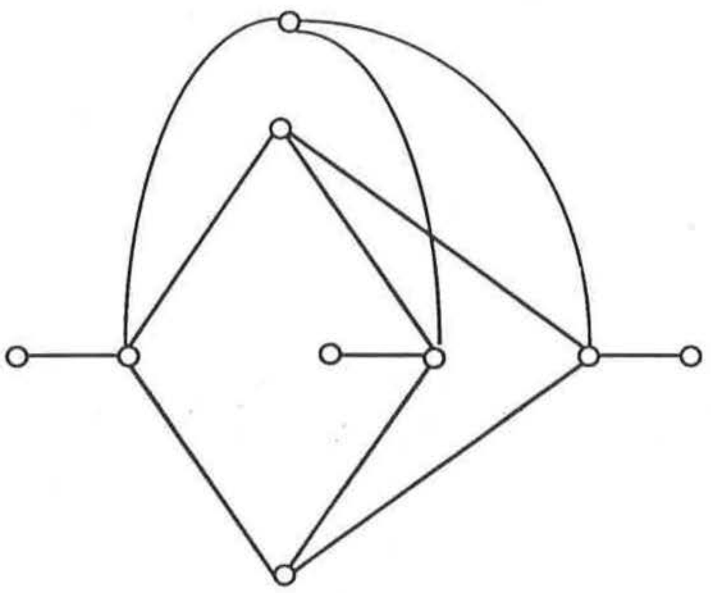


Figure 3.2: G^+ of a Non Bar-Representable Graph

Define G^+ as the one vertex extension of G:

$V(G^+) = V(G) \cup \{x\}$

$E(G^+) = E(G) \cup \{(x,v) \mid v \text{ is an articulation vertex of } G\}$.

Figure 3.2 shows the graph H^+ corresponding to the non bar-representable graph H of figure 1.2.

Lemma 4 *Graph G^+ is planar iff G is planar and has an s-t* numbering.*

Proof:

Note that G^+ is planar iff G can be embedded in such a way that all articulation vertices appear on the exterior face.

($\Leftarrow$) (By contradiction).

Let G be s-t^* numberable with $\mathcal{G}$ as the plane embedding realizing the numbering. Suppose that $\mathcal{G}$ contains an articulation vertex c not on the exterior face. Then c is contained within some circuit of $\mathcal{G}$. Let F be the innermost circuit with c on the inside. Since c is an articulation vertex, there is a component entirely inside F which must contain a local λ-max or λ-min not on the exterior face and hence $\mathcal{G}$ does not realize an s-t^* numbering.

($\Rightarrow$) Let $\mathcal{G}$ be an embedding of G with all articulation vertices s_j and biconnected components B_i on the exterior face. The blocks of $\mathcal{G}$, will be s-t* numbered in a depth-first manner. Modify the s-t numbering algorithm of Even and Tarjan, [8]:
STNUMBER(s, t, B, beg) will s-t number biconnected block B with distinguished edge (s, t) without labelling s, and numbering other vertices starting with beg. Note that $\lambda(t) = beg + |B| - 1$.

```
Procedure STSTAR($s, start$)
for all unnumbered blocks $B_i$ containing articulation vertex $s$ do:
    • choose an unmarked vertex $t$ adjacent to $s$
      on the exterior face in $B_i$
    • STNUMBER($s, t, B_i, start$)
    • $start := start + |B_i| - 1$
    • for all articulation vertices $s_j$ in $B_i$ do
      STSTAR($s_j, start$)
return.
```

Then the following call s-t* numbers G.

$\lambda(s_1) := 1$

STSTAR(s_1,2)

Note that the numbering assures that articulation vertex s_1 is the only local λ-min value, and all λ-max values (t) are on the exterior face (and therefore the two sets are separable) and hence G is s-t* numbered.

□

Determining the articulation vertices, biconnected blocks and s-t numberings are linear operations ([8]), and hence the STSTAR algorithm also requires only linear time.

Finally,

Theorem 6 *Graph G is bar-representable iff G^+ is planar.*

Proof:

Directly from the previous lemma and Lemma 3.

□

Corollary 6 *Both determining if a graph G is bar-representable and providing a bar layout of G can be performed in linear time.*

Proof:

Although Theorem 6 does not directly indicate how a layout for G may be obtained, it does provide a means for checking for bar-representability in linear time via planarity testing.

Note that the proof of Lemma 3 does provide the means for orienting the edges of G and as a consequence, for obtaining a layout for G in linear time.

□

3.4 The Undirected Planar Positive Flow Reformulation

3.4.1 Ordered Graphs

For an ordered undirected unweighted graph G, $M(G)$ is defined as the capacitated graph obtained by:

- assigning capacity ∞ to each edge,
- adding vertices s and t in the exterior face of G, joined to all vertices on the exterior face of G with associated capacity ∞.

Corollary 7 *Ordered graph G is an embedded bar-representable graph iff the Undirected Planar Positive Flow problem is solvable for $M(G)$.*

Proof:

An s-t*-numbering of G induces a network-creating orientation on $M(G)$, and the proof of Lemma 3 provides a layout for G. The set of visibility widths between bars in this layout corresponds to the range of a positive flow function for $M(G)$.

Given a positive flow f' for an orientation of $M(G)$, weight and direct G by the flow values of f' under the given orientation. The resulting graph is an embedded measured polarized bar-representable graph as shown in Corollary 2, and thus G is an embedded bar-representable graph.

□

3.4.2 Unordered Graphs

In the least constrained version of the Flow reformulation, G is an unordered, unweighted, undirected graph. The network $M(G)$ is defined as above with the exception that s and t are each joined to all vertices with associated capacity of ∞.

Corollary 8 *Graph G is a bar-representable graph iff the Undirected Planar Positive Flow problem is solvable for $M(G)$.*

Proof:

The proof of the previous corollary may be used together with the result of corollary 3.

□

Chapter 4

Measured Graphs

4.1 Introduction

When the given graph is weighted but not directed, the problem of determining whether a given graph is a measured bar-representable graph becomes significantly more difficult, for both the ordered and unordered versions. The previous two chapters have solved both the more constrained (directions imposed), and the less constrained (weights unspecified) cases.

In the previous two unweighted cases, the balancing conditions at interior vertices were of little real concern - suitable weights could always be chosen. In the two directed cases, the consecutive-in-out property severely constrained the topology of the layout.

When the weights are fixed and the directions are unspecified, the problems reduce to a partitioning-type problem about each interior vertex since the inweight and outweight of such vertices must be balanced. The problem of partitioning a set of weights into two subsets of equal weight is known to be NP-Complete (in the weak sense [9]). However, the partitions about the interior vertices are not independent and furthermore, vertices which have no exactly balanced partition must appear on the exterior face. As shall be shown, the structure of the graph induces sufficient dependency among the partitionings for the entire problem to be solvable in polynomial time.

Assuring that each interior vertex is balanced, requires that the edges incident upon each such vertex be partitioned into two groups of equal weight. While there may exist many valid partitions of the edges about each vertex, the decisions on how to balance the vertex depend not only on the local possibilities, but also on the global structure. In the ordered case, the consecutive-in-out-flow property helps to limit the choices. The necessary absence of directed cycles further reduces the number of possible partitions. It is the exploitation of these two properties that in effect drives the algorithms for this weighted undirected case. The existence of an algorithm with a polynomially bounded time complexity for this problem is somewhat surprising, in light of the NP-Completeness of the partitioning problem and the NP-Hardness results in the following chapter. The degrees of the polynomials in the analyses of the algorithms for the two cases are not small. However, the efficiency of the algorithms can almost certainly be improved at the expense of clarity.

Figure 4.1 shows a weighted graph H that is not a measured bar-representable graph but that is bar-representable in its unweighted form. That H is not a measured bar-representable graph can be proven by checking all possible orientations of the edges and then applying the results obtained on measured polarized graphs. However, this procedure is clearly not computationally viable in general, as it involves an overhead factor of $2^{|E|}$.

For both the ordered and unordered versions of the measured bar-representable case, it is the undirected planar full flow formulation of the problem that will be solved.

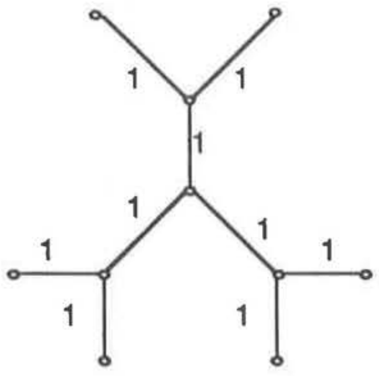


Figure 4.1: A Non-Measured Graph

4.2 Flow Existence Formulation

The embedded measured bar-representable problem may be formulated as a flow problem as follows. Given an ordered weighted graph G, define $N(G)$ as the capacitated graph obtained by:

- associating the capacity w with each edge of G of weight w,
- adding vertices s and t into the exterior face and joining s and t to all vertices on the exterior face of G with associated capacity ∞,
- joining s to t with associated capacity of 1.

The problem then is to determine a network-creating orientation g, of the edges such that $g(N(G))$ has a planar full flow.

The unordered formulation defines $N(G)$ as above with the exception that s and t are joined to all vertices of G, with associated capacity of ∞.

In the previous chapters, the attachments of the s and t vertices did not really play a major role. There was either little flexibility (the directed case) or almost all attachments successfully led to a layout (the undirected unweighted case). The current case however is solved by considering a sequence of subcases, successively relaxing the specifications of the s and t attachments. There are two parameters:

- the specification of the vertices to which s and t are to be adjacent in the final flow. That is, the edges carrying non-zero flow in the full flow (called the **extreme edges**) are specified.
- whether the weights on these extreme arcs are also specified.

A useful notation in this regard is the introduction of **0-weight** extreme edges. The interpretation of weight(s, v) being 0 under some planar feasible flow f, is that the addition of edge (s, v) into N^f would not introduce a non-planarity.

4.3 Ordered Graphs

Given an ordered weighted undirected graph G, determining if G is an embedded measured bar-representable graph, is equivalent to solving the undirected planar full flow problem on $N(G)$, where $N(G)$ is created by joining s and t to all vertices of G on the exterior face with associated capacity of ∞, as defined in the previous section. The solution to this problem is based upon the case when the extreme edges of $N(G)$ are specified (i.e. s and t are adjacent only to two separable subsets of the exterior vertices of G). This problem is in turn based upon the case when the extreme edges of $N(G)$ with non-zero flow are specified. Finally, the solution to this case depends upon the case in which the extreme edges with non-zero flow together with their flow values are specified.

4.3.1 Extreme Edges and Their Weights Specified

If the non-zero s and t attachments and weights are specified, then a fairly straightforward greedy algorithm solves the undirected planar full flow problem. The technique is similar to topological sorting ([1]), as it relies on the property that all acyclic digraphs contain a vertex with no incoming arcs.

Lemma 5 *If $N(G)$ has a planar full flow, then there exists a vertex v such that weight(s,v) $= \sum_{w \neq s}$ weight(v,w).*

Proof:

Let g' be an acyclic orientation of $N(G)$ and let f' be a planar full flow of $g'(N(G))$. Then $g'(N(G))$ is an acyclic digraph with a single source s. Removal of the vertex s creates an acyclic digraph which must also contain at least one vertex v with no incoming arcs. Vertex v has the required property under the flow f' since it must have balanced in- and out-weights.

□

The algorithm will sequentially remove the vertices provided by lemma 5. The order of removal of the vertices determines an orientation for the edges of G and a flow for the resulting ordered digraph may be found with the results of chapter 2.

Define $\mathcal{N}(\mathcal{G})^{-v}$ as the embedded graph obtained by deleting s-adjacent vertex v from $\mathcal{N}(\mathcal{G})$ and replacing each edge incident upon v by one from s, with the same weight. See figure 4.2 for an illustration.

Lemma 6 *If there exists a vertex v adjacent to s such that weight$(s,v) = \sum_{w \neq s}$ weight(v,w) and $\mathcal{N}(\mathcal{G})^{-v}$ has a planar full flow, then $\mathcal{N}(\mathcal{G})$ has a planar full flow.*

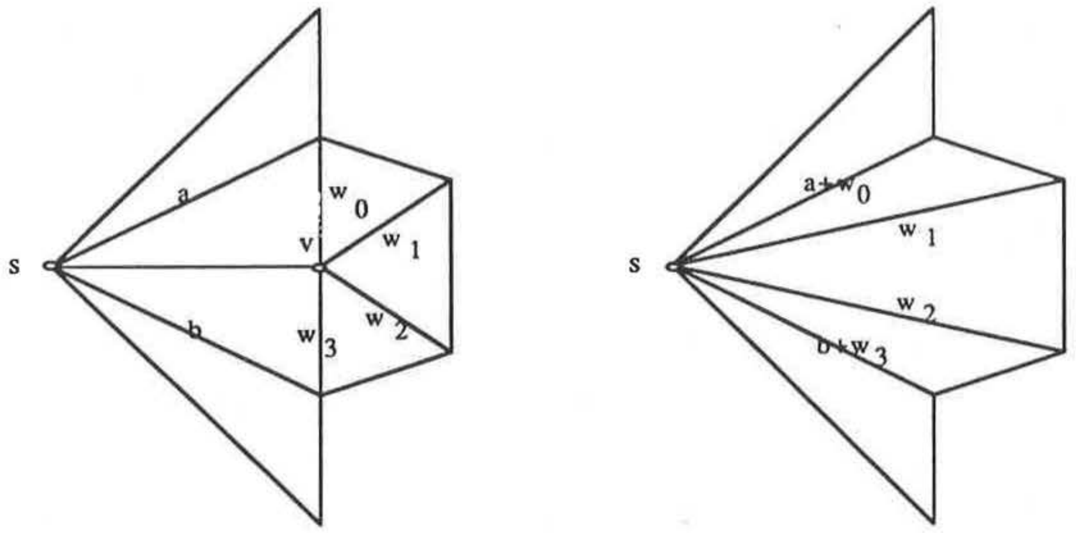


Figure 4.2: Form of $\mathcal{N}(\mathcal{G})$ and $\mathcal{N}(\mathcal{G})^{-v}$

Proof:

Let v be a vertex with the required property adjacent to $s, u_1, u_2, \ldots, u_n$ in clockwise order and with associated weights $x, w_1, w_2, \ldots, w_n$. If u_1 and u_n are visible to s, then let their weights be w_0 and w_{n+1}. Then $\mathcal{N}(\mathcal{G})$ and $\mathcal{N}(\mathcal{G})^{-v}$ have the form as in figure 4.2.

Suppose there exists a planar full flow for $\mathcal{N}(\mathcal{G})^{-v}$. Then s supplies weight w_i to each vertex u_i. A corresponding flow for $\mathcal{N}(\mathcal{G})$ may be obtained by supplying the weight w_i to u_i via v. (i.e. all edges (v, u_i) are directed from v to u_i.) Thus, the outflow(v) is $\sum_{u_i}$ weight(v, u_i). By assumption, weight(s, v) is equal to this amount and hence v is balanced in such a flow. No directed cycle is introduced in this flow for $\mathcal{N}(\mathcal{G})$, since a directed cycle would necessarily involve s or v and s is maintained as source and v has only a single incoming arc, from s. □

4.3.2 Algorithm to Orient the Edges of a Network With Extreme Edges and Weights Specified

Given a planar ordering $N(G)$ with s and t arcs and demands specified, the following algorithm orients the edges to create an acyclic directed network. Determining a planar full flow for the resulting network may then be achieved by appealing to the result in Chapter 2 on directed weighted networks.

```
For i := 1 to n do
  • Choose a vertex $v \neq s,t$ such that $\sum_{w \neq s}$weight$(w,v)$ = weight$(s,v)$. (If no such vertex
    exists, HALT there is no layout)
  • label$(v)$ := i
  • For all vertices $w$ adjacent to $v$ do
    {Collapse $\mathcal{N}(\mathcal{G})$ to $\mathcal{N}(\mathcal{G})^{-v}$:}
      – delete edge $(v,w)$ and add arc $< s,w >$ (with initial weight of 0), if not present.
      – weight$(s,w)$ := weight$(s,w)$ + weight$(v,w)$.
  • Delete vertex $v$.
```

Orient the edges (u,v) of $N(G)$ so that label(u) < label(v) iff arc $< u,v > \in N(G)$.

Determine a planar full flow for $N(G)$ with the algorithm for directed weighted graphs (chapter 2). 1

4.3.3 Extreme Edges of $N(G)$ With Positive Demands Specified

Assumptions: Two simplifying assumptions will be made for this section.

- Assume that no articulation vertex of G is adjacent to both s and t in $N(G)$. (Such a graph may be split into its biconnected components. The flow for the components may be computed independently and then merged upon completion of the algorithm).
- Assume that there does exist an orientation of $N(G)$ for which there is a planar full flow. (The algorithm will not make this assumption, it is made simply to avoid awkwardness in expressing the following lemmas).

A somewhat involved algorithm with time complexity of $O(n^4)$ is developed for the conditions of this section. One critical observation is that knowledge of the direction of any edge in the graph has consequences that propagate throughout the graph. Edges are divided into equivalence classes called *nets* so that knowing the direction of any edge in the net forces a direction on all others in that net. Consider, for example, figure 4.9 b. If edge (v, w) were known to be directed from v to w then edge (w, x) must be directed out to balance the flow at vertex v. In figure 4.9a, vertex v is adjacent to s and since some non-zero flow is accepted from s, (v, w) must be directed from v to w. (Otherwise, either a directed cycle is produced or v is not balanced). Suppose (v, w) is oriented from v to w, then the direction of some "corresponding" edges is also known (see 4.9b). The flow in a net propagates its way through the network in this manner.

All the edges of the network are initially partitioned into nets. It will be shown that there is a single net whose direction is *fixed* and $O(n)$ mirror pairs of unfixed nets. The algorithm then consists of a four stage process:

- Unfixed nets may be forced by the manner of their intersection with the fixed net.

- Pairs of unfixed nets may intersect in such a way that forces their direction.
- Unfixed nets may intersect the new fixed net in a manner that forces them.
- The remaining unfixed nets are merged into the fixed net.

The resulting weighted directing of the graph may then be laid out with the techniques of chapter 2.

Each edge (u, v) has potentially two possible directions, namely, the arcs $< u, v >$ and $< v, u >$.

Define a relation **forces** between pairs of directions of edges:

$$< u, v > \textbf{ forces } < w, x > \text{ if}$$

for all acyclic orientations g for which there exists a planar full flow

$$g((u, v)) \ = \ < u, v > \text{ iff } g((w, x)) \ = \ < w, x >.$$

As a consequence, $< u, v >$ forces $< w, x >$ iff $< v, u >$ forces $< x, w >$.

The relation is reflexive, symmetric and transitive and therefore creates a set of equivalence classes of directions of edges called **nets.** Each edge appears in exactly two nets which necessarily form a pair of mirror images of sets of directions of edges.

The problem initially could be viewed as one of choosing one of the two possible directions for each edge of $N(G)$, but once these nets have been constructed, the problem reduces to choosing one net from each pair of mirror nets. Once a direction for an edge has been established, an entire net of arcs is determined. Consider, for example, an edge of the form (s, v); there is only one valid direction for such an edge, namely $< s, v >$, since

s must be the source in any flow. Thus, the entire net containing $< s, v >$ must be chosen in any flow.

It is the construction of the nets that is computationally involved. The algorithm determines a single net whose orientation is **fixed** by the topology of the network, and many pairs of mirror nets that are "unfixed". Nets are built from dependent subnets which are initially constructed by considering local information at each vertex, and then subsequently merged based upon global relationships.

Two nets will be called **disjoint** if they have no edge (in either direction) in common. Define an **extreme** vertex as one adjacent to s or t in $N(G)$.

The local properties of subnets (from the fixed net or unfixed nets) about a single vertex will first be investigated. Informally, the following lemmas show that for non-extreme vertex v:

- A net N incident upon v "passes through" v. (And as a consequence, nets terminate only at extreme vertices.)
- The two sets of edges of net N, the in-directed edges and the out-directed edges, incident on v are each consecutively ordered.
- The inweight of net N at v is equal to the outweight of N at v.
- The cyclic ordering of any pair of disjoint nets on v has a fixed form.

Next, properties of the single fixed net F will be considered. It will be shown that:

- F passes through every extreme vertex.

Lemma 7 *For all non-extreme vertices v, if $< u, v >$ is a direction of an edge of net N, then there exists some corresponding directed edge $< v, w >$ of N, forced by $< u, v >$.*

Proof:

Let the incident weights about v be in clockwise order, $w_0,\ w_1, \ldots, w_n,\ (n \geq 1)$ on edges $e_0,\ e_1, \ldots, e_n$ with e_0 representing arc $< u, v >$. Let $T = \frac{1}{2}\sum_{i=0}^{n} w_i$ and let $w_{n+1} = w_0$, for notational convenience.

Determine two groups of consecutively ordered edges from e_0 clockwise and counter-clockwise whose weights sum to no more than T. Let k be the subscript of the edge which is $max_k \sum_{i=0}^{k} w_i \leq T$,

and let l be the subscript of the edge which is $min_l \sum_{i=l}^{n+1} w_i \leq T$.

Then, $\sum_{i=0}^{k} w_i + \sum_{i=l}^{n+1} w_i \leq 2T$, (by the definition of T) and therefore,

$\sum_{i=0}^{n} w_i + w_0 - \sum_{i=k+1}^{l-1} w_i \leq 2T$.

But $\sum_{i=0}^{n} w_i = 2T$ and hence,

$\sum_{i=k+1}^{l-1} w_i \geq w_0$.

Finally, the weights w_i, are all assumed to be greater than 0 and therefore $k+1$ is less than or equal to $l-1$.

Thus, for vertex v to be balanced in some flow, there must be at least one edge between e_k and e_l which is (locally) forced by e_0.

□

For constructing the nets, it is necessary to note that if w_0 is strictly less than $\sum_{i=k+1}^{l-1} w_i$ in the above proof, then the edges $e_{k+1}, \ldots, e_{l-1}$ may be conceptually collapsed into a single edge and the lemma applied again, relabelling the new edge as the known edge e_0, but directed out of v. The set of edges forced by the new (collapsed) edge will include the previous e_0 and at least one other edge.

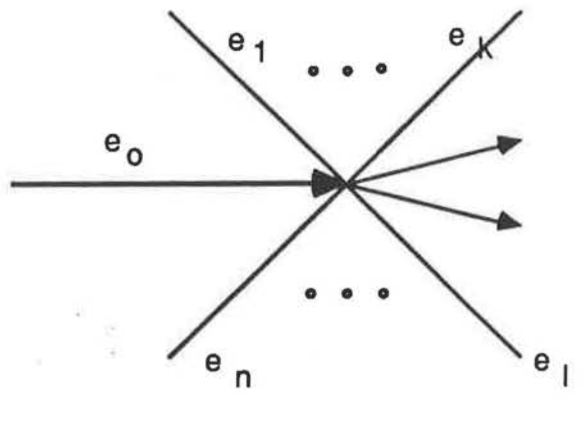


Figure 4.3: Corresponding Edges

This lemma may be interpreted as asserting that a net "passes through" vertex v. A natural question then is to ask at which vertices do nets *terminate* (i.e. not pass through). Nets that are not part of the fixed net, have their endpoints at the extreme vertices. The fixed net shall be shown to terminate at s and t and, in fact, that is why its direction is determined, and also why there is only a single fixed net.

Another consequence of the previous lemma is that at any non-extreme vertex v, the relation "forces", locally partitions the directed edges of net N at v into two corresponding subsets, called **bundles:** those forced into v and those forced out of v by any planar acyclic orientation g'. A different planar acyclic orientation may force the mirror net, however, the mirror net must contain the same bundles, only in the opposite direction. The two bundles associated with net N, at non-extreme vertex v will be denoted as $A_{N,v}$ and $A'_{N,v}$, respectively, the in-directed edges and the out-directed edges of N at v. The subscripts will be dropped if the specification of the particular net and vertex is clear or not relevant.

Define for a set X of edges, weight(X) as $\sum_{e\in X}$weight(e).

Lemma 8 *Weight($A_{N,v}$) = weight($A'_{N,v}$), for all nets N and all non-extreme vertices v.*

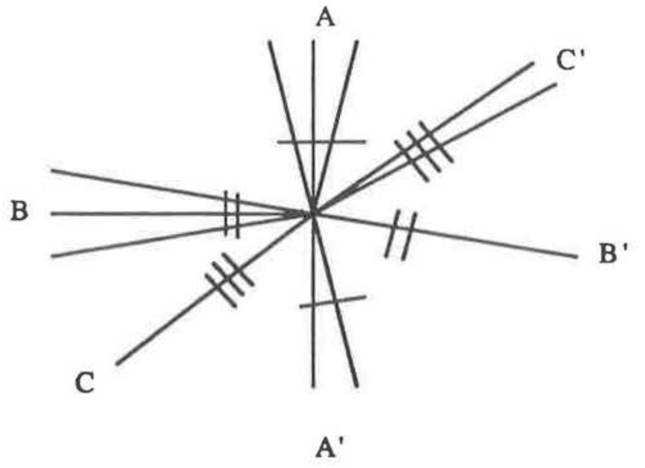


Figure 4.4: Corresponding Bundle Weights are Equal

Proof:

If bundle A "weighs more" than A' then collapse the edges of A into a single edge e_0 with weight equal to the sum of the weights of bundle A. By Lemma 7, there exists some sequence of edges $e_{k+1}, \ldots, e_{l-1}$ in the corresponding bundle (and thus in A') whose weights sum to at least w_0, contradicting the initial assumption.

□

Lemma 9 *Let M and N be two disjoint nets that intersect at non-extreme vertex v, with bundles $A_{M,v}$, $A'_{M,v}$ and $B_{N,v}$, $B'_{N,v}$. Then, either the clockwise or the counter-clockwise order about v is $ABA'B'$.*

Proof:

Assume the clockwise order is $AA'B'B$. Let g be any acyclic orientation that directs A into v and A' out of v. Then g must not direct B out of v and B' into v, since then a directed cycle would be created. (Recall the "consecutive-in-out" lemma, 1). Thus, A forces B, contradicting the assumption that A and B were bundles from different nets.

□

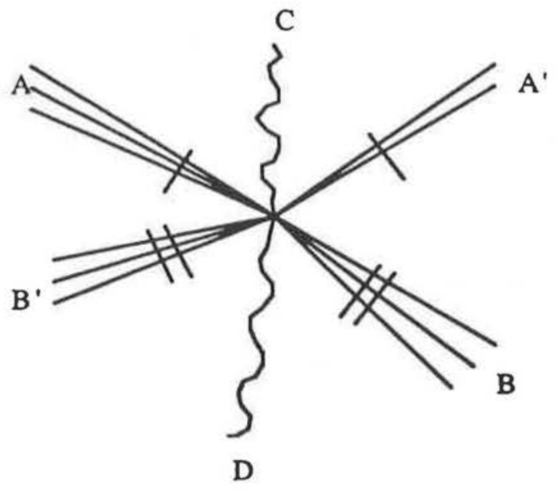


Figure 4.5: Bundle Order

Lemma 10 *For any non-extreme vertex v, when any two consecutive oppositely directed arcs incident on v are known, the entire partition about v can be determined.*

Proof:

Let a and b' be two such arcs with corresponding bundles A' and B. By the previous lemma, the clockwise order is $a, b', A'B$. Since there are no arcs between a and b', there are none between A' and B. Any edges between b' and A' must be oriented in the same direction as b', to preserve the "consecutive in-out property". Similarly, for edges between B and a. Again, the previous lemma assures that such edges form corresponding bundles.

□

The following lemma establishes the existence of an initial fixed subnet.

Lemma 11 *For each extreme vertex v, there exists a directed bundle (forced out of v, if v is s-extreme and forced into v if v is t-extreme), under any acyclic orientation with a planar full flow.*

Proof:

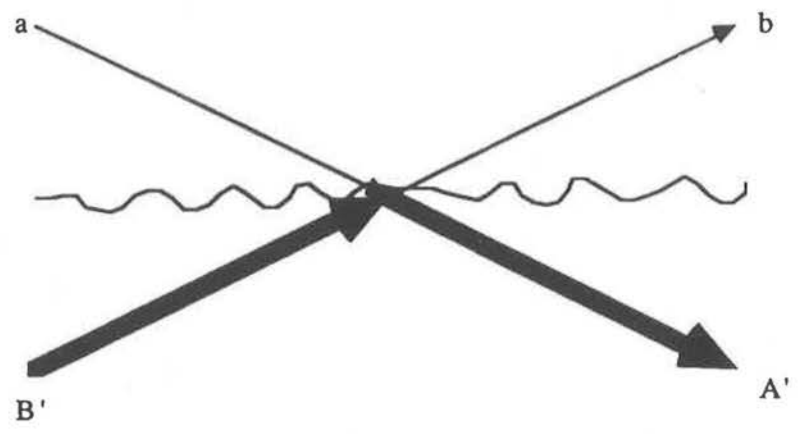


Figure 4.6: Partitioning Non-Extreme Vertices

Assume v is adjacent to s and not to t with weight$(s,v) = \epsilon$, $\epsilon > 0$). If there is only a single edge other than (s,v), then the lemma is trivial, therefore assume there are at least two such edges. Label the edges clockwise from (s,v) as $e_1, e_2, \ldots, e_n$ with weights $w_1, w_2, \ldots, w_n$. For notational convenience, introduce two fictitious edges e_0 and e_{n+1} with weights of 0.

Define

$f_i = \sum_{j=1}^{i} w_j$, (the "forward sum" to i)

$b_i = \sum_{j=i}^{n} w_j$, (the "backward sum" from n)

$d_i = | f_i - b_i |$, (the absolute difference)

(See figure 4.7).

Consider any edge e_i with the property that

1) $f_i \geq b_{i+1}$ and

2) $b_i \geq f_{i-1}$.

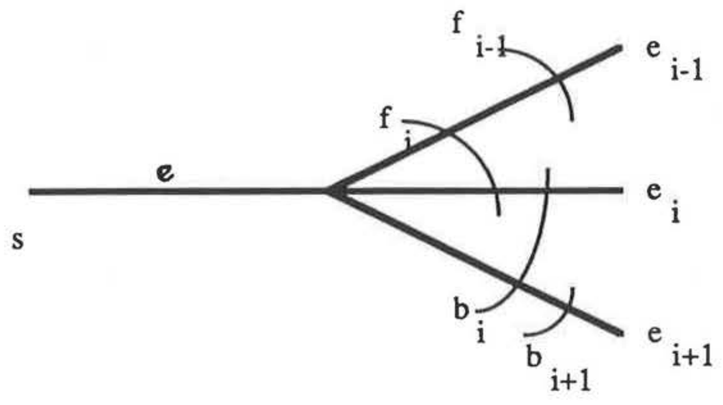


Figure 4.7: Forced Bundles on Extreme Vertices

Such an edge must go out since if it came in, either $e_1, \ldots, e_{i-1}$ come in (but then $f_i + \epsilon > b_{i+1}$ and the flow-in flow-out property fails) or $e_{i+1}, \ldots, e_n$ come in (and again this property fails).

It remains to be shown that such an edge must exist.

Note that the f_i are monotonely increasing and the b_i are monotonely decreasing. There are two cases to consider based on whether

Case 1: There exists an edge e_i such that $f_i = b_i$.

Then, $f_i > b_{i+1}$ (since $b_{i+1} = b_i - w_i$).

Hence, $b_i = f_i > f_{i-1}$.

Thus e_i satisfies the two required properties and therefore must be directed out.

Case 2: There does not exist an edge e_i with $f_i = b_i$. Consider the edge e_i with the largest subscript such that $f_i < b_i$. Then either e_i or e_{i+1} (or both) must be directed out, as shall now be shown. This case is further subdivided based upon the relationship between f_i and b_{i+1}, namely whether $f_i > b_{i+1}$, or $f_i < b_{i+1}$, or $f_i = b_{i+1}$.

Case 2a: $f_i < b_{i+1}$.

Then, $f_{i+1} > b_{i+1} > b_{i+2}$.

But these are exactly the conditions 1 and 2 for edge e_{i+1}. Therefore e_{i+1} is directed out.

Case 2b: $f_i > b_{i+1}$.

Then, $b_i > f_i > f_{i-1}$

Hence by conditions 1 and 2, e_i is directed out.

Case 2c: $f_i = b_{i+1}$.

Substituting equality for inequality in the previous two cases demonstrates that both e_i and e_{i+1} are directed out.

Figure 4.8 shows examples of these four cases.

□

For edge (u,v), define bundle(u,v) as:

- the set of edges computed by Lemma 7, if v is not extreme.
- the set of edges computed by Lemma 11, if $u = s$ or t.
- $\{(s,v)\}$ or $\{(v,t)\}$, if $(u,v) \in$ bundle(s,v) or bundle(t,v) respectively.
- ϕ, otherwise

The previous lemmas indicate a strategy for the preliminary computation of the fixed and unfixed pairs of subnets:

Compute the bundling information for each edge and merge, via set union, the subnets induced by the lemmas, yielding a single fixed net and a set of mirror pairs of induced

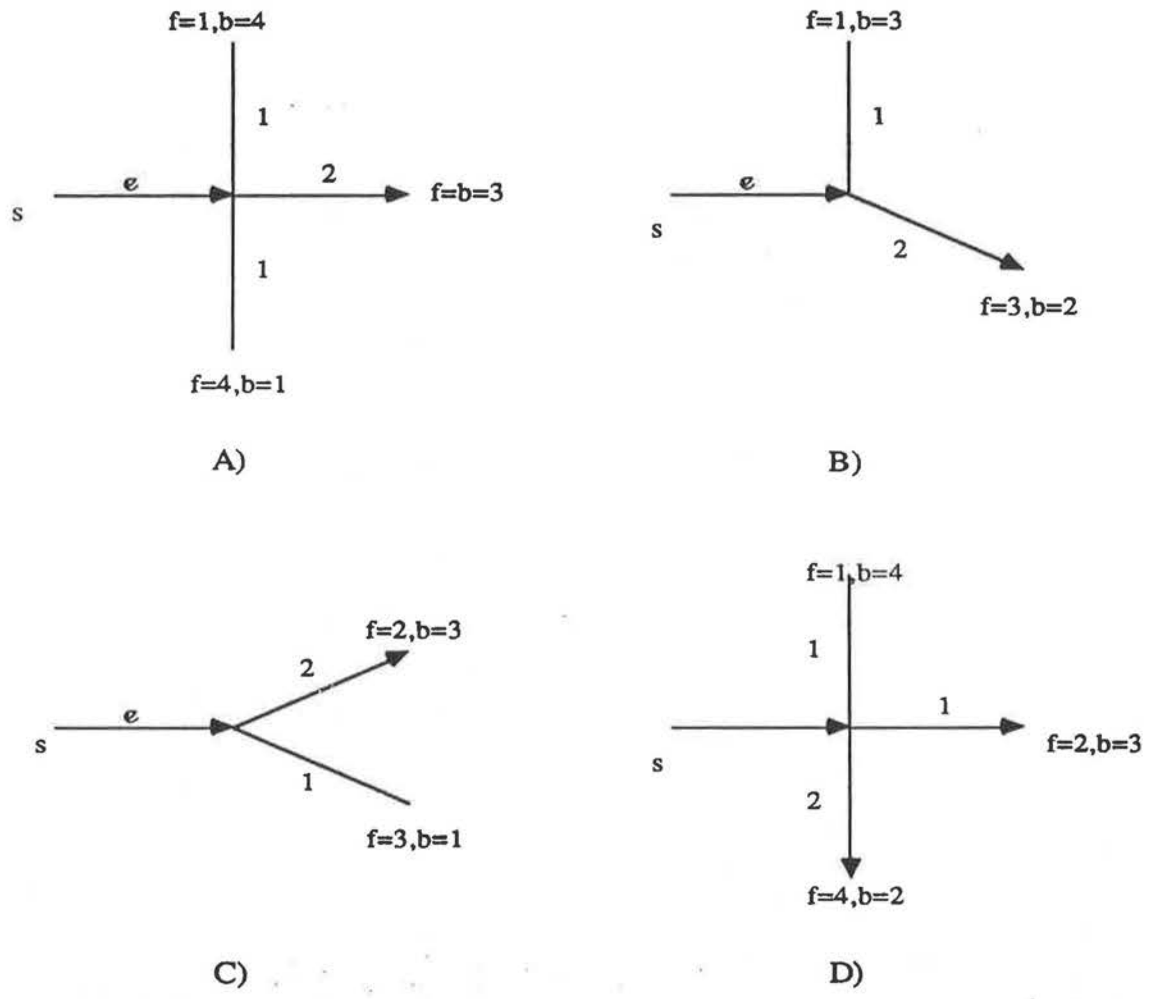


Figure 4.8: Examples of the Four Cases

subnets.

Note that these induced subnets may not be independent; they may intersect in a manner which implies they must be merged either to each other or to the fixed net. Fortunately, these intersections may be detected efficiently. The algorithm treats two cases typified by the following examples. See figure 4.9.

- Case 1: In 4.9c induced subnet I intersects the fixed subnet F twice, in effect forcing I to flow in the direction from u to v. (Otherwise a directed cycle is created.)
- Case 2: Induced subnets may also be forced by the nature of their intertwining. In 4.9d, neither I nor J is forced when considered individually, however their intersection forces them. Again only one pair of orientations from the mirror pairs, (I "down" and J "up") does not introduce a cycle.

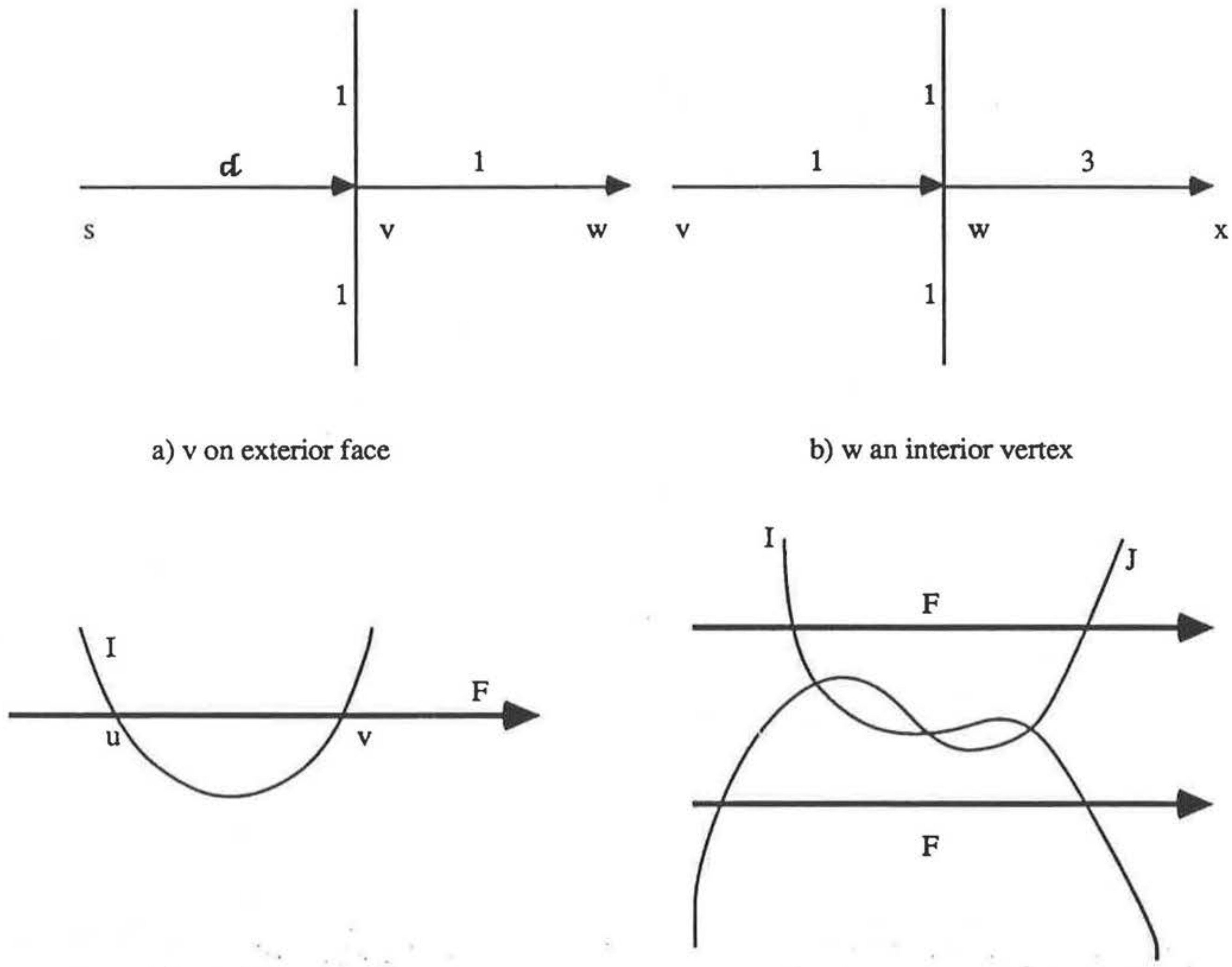
The testing for forcedness for a given pair of subnets may be achieved by determining if a directed graph contains a directed cycle - a linear time operation (via topological sorting) [1]. Still, there may exist $\Theta(n)$ induced pairs of subnets. To check all subsets of subnets for interdependent forcedness would be prohibitively expensive. Fortunately, not all subsets of subnets need be checked, a crucial lemma shows that checking disjoint *pairs* suffices.

The induced subnets remaining at this stage reflect the fact that there may exist many layouts for $N(G)$.

Once a direction for each edge has been established (i.e. an acyclic orientation), the algorithm for laying out weighted, directed, ordered graphs will complete the procedure.

The details of implementing the above overview will now be supplied.

Let N_0 be the initial fixed subnet of maximal size obtainable by lemmas 7, and 11.



a) v on exterior face

b) w an interior vertex

c) I flows u to v

d) I and J are forced

Figure 4.9: Nets - Forced and Induced

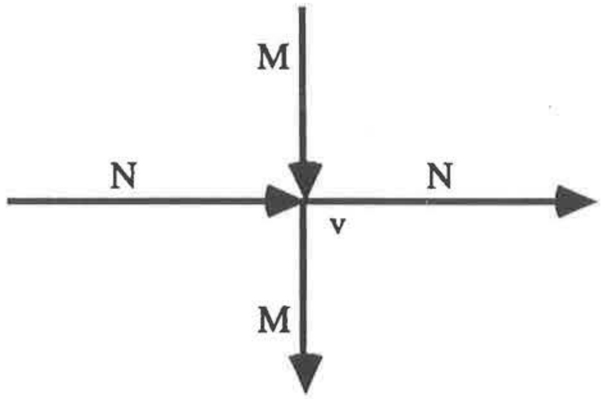


Figure 4.10: Proper Intersection of M and N

The direction of the fixed net is forced under all planar acyclic orientations, however, either subnet from each mirror pair may appear in different orientations. The next step is to consider the effect of choosing one of the induced subnets from a mirror pair of subnets - a process called **forcing.**

If directed net M intersects directed net N at vertex v, then M is said to intersect N **properly**, if the inarcs (respectively outarcs) of M appear clockwise of the inarcs (respectively outarcs) of N at v; and M intersects N **counter-properly** if N intersects M properly.

(Informally, M intersects N properly if N flows left to right iff M flows top to bottom.)

Lemma 12 *If induced subnet I intersects the fixed net F at two vertices v_1, v_2, once properly and once counter-properly, then I is forced.*

Proof:

There is one case to consider and three further cases which will be shown to reduce to the first.

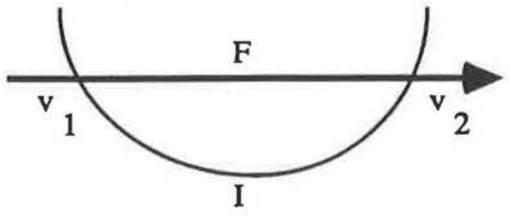


Figure 4.11: Case 1

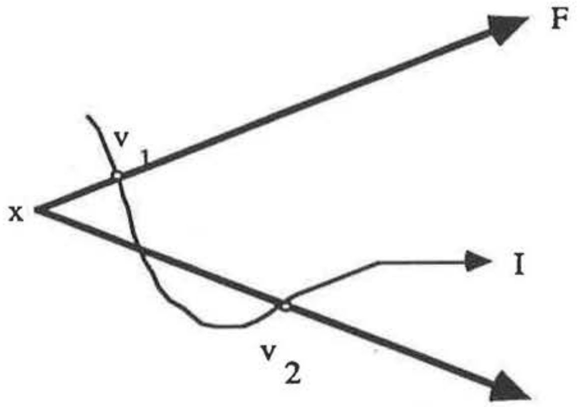


Figure 4.12: Case 2

Case 1: There exists a directed path in I and one in F between v_1 and v_2. Then I is forced to flow from v_1 to v_2, otherwise a directed cycle would be introduced.

Case 2: There exists a directed path in I from v_1 to v_2 and not in F. Vertices v_1 and v_2 must have a common ancestor or descendent in F. Suppose v_1 and v_2 have a common ancestor x in F. Net I must cross the (x,v_2) path properly or the (x,v_1) path counter-properly. Hence case 1 applies for some pair of vertices on I.

Case 3: There exists a directed path in F from v_1 to v_2 and not in I. Then vertices v_1 and v_2 must have a common ancestor or descendent in I. Suppose v_1 and v_2 have a common ancestor x in I. Again, case 1 applies as either path (x,v_1) or (x,v_2) crosses F a

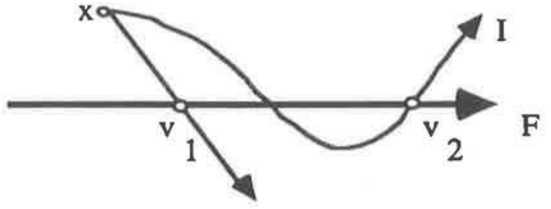


Figure 4.13: Case 3

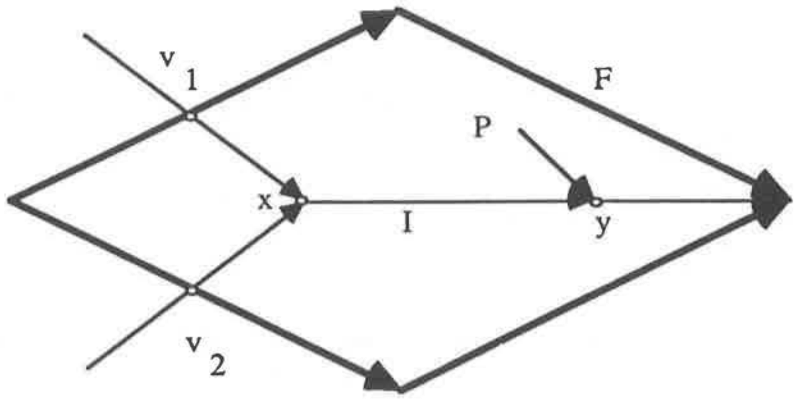


Figure 4.14: Case 4

second time.

Case 4: There is no directed path in F and no directed path in I from v_1 to v_2. Vertices v_1 and v_2 must have a common ancestor or descendent in I. Suppose v_1 and v_2 have a common descendent x in I. Then they have some descendent y that is adjacent to t. If F intersects path (x, y), then case 1 applies. Otherwise, y has a forced path P from s by Lemma 11. Path P must cross F or I and hence case 1 applies for some pair of vertices on I.

□

Induced subnets of the form in the above lemma can be merged into the fixed network N_0 yielding a fixed net N_1, and the mirror subnet discarded. For each remaining unforced mirror pair of subnets, label as I the subnet forming a **preferred orientation**, namely the one in which the net intersects N_1 properly at all vertices. Since each net contains at least two extreme vertices each of which is also in N_1, there are at least two intersections. The mirror subnet of I (which flows in counter-preferred orientation) will be denoted by $\overline{I}$.

The dependencies among the induced subnets themselves can now be determined by checking the pairwise union of disjoint subnets (in preferred orientation) for cycles. In the algorithm, if a pair of induced disjoint subnets A and B in preferred direction contains a cycle, then the directions for A and B will be determined by checking for cycles in $\overline{A} \cup B \cup N_1$ and $\overline{B} \cup A \cup N_1$, at most one of which will be cycle free. (Note that $\overline{A} \cup \overline{B} \cup N_1$ is not a candidate since it must also contain a cycle if $A \cup B$ does.)

It is necessary to check all pairs of induced intersecting disjoint subnets and to merge them appropriately with the fixed net N_1 if they would create a cycle in preferred orientation.

The following lemma proves that checking *pairs* of subnets is sufficient; not all subsets need be considered.

Lemma 13 *If the union of k ($\geq$ 3) induced subnets with preferred orientation contains a directed cycle, then some pair of them also creates a cycle.*

Proof:

It suffices to prove that if the hypothesis is true, then some l ($< k$) of the nets also create a directed cycle.

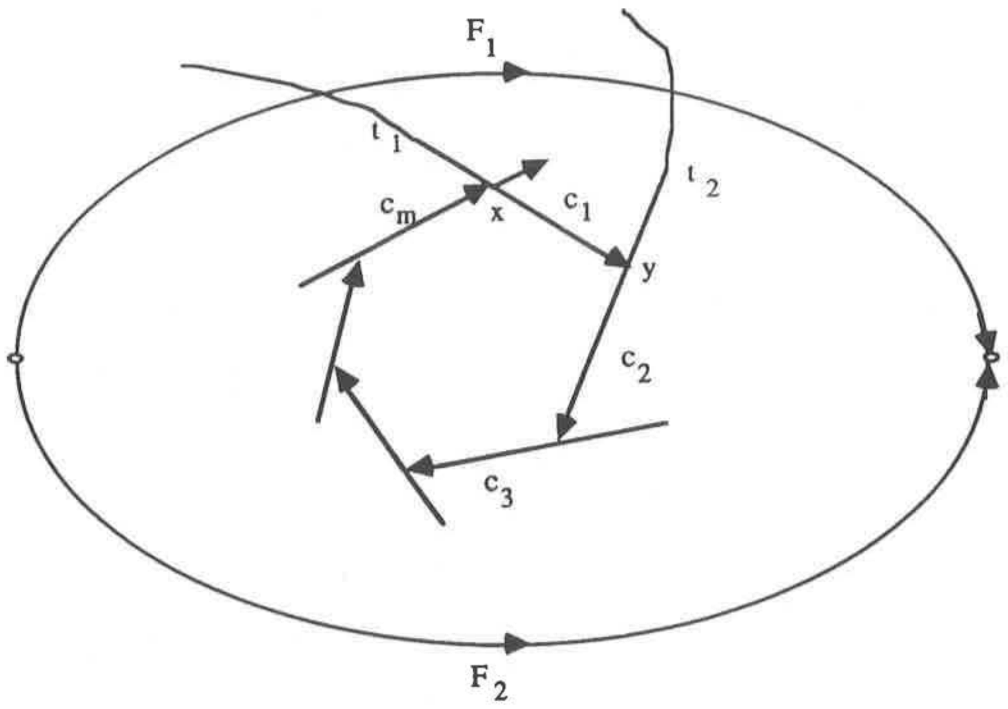


Figure 4.15: Preferred Cycles

Let C be a directed cycle involving k nets in preferred orientation. Assume C is directed clockwise and consists of m ($\geq k$) maximal net paths $c_1, c_2, \ldots, c_m$. Note that none of the c_i may be part of the fixed net, since then one of the (adjacent) induced nets would cross it counter-properly; such counter-proper intersections have been removed in the previous stage. Furthermore, F does not pass through C since again one of the induced nets would then cross it improperly. See figure 4.15.

Define Tail(c_i) as the subset of the net containing c_i that precedes c_i (in preferred direction), and Head(c_i) as the subset of the net containing c_i that follows c_i (in preferred direction).

Let x be the vertex of intersection of c_m and c_1. Let y be the vertex of intersection of c_1 and c_2. Let v_1 be an arbitrary extreme vertex on Tail(c_1) and t_1 ($\in$ Tail(c_1)) be a path from v_1 to x. Let v_2 be an arbitrary extreme vertex on Tail(c_2) and t_2 ($\in$ Tail(c_2)) be a path from v_2 to y. Since v_1 and v_2 are extreme vertices, they are also on the forced net.

Let h be an arbitrary path in Head(c_m) from x to an extreme vertex. Then h enters a region bounded by t_1,c_1, t_2 and F. Since h must not cross F improperly, it must first cross one of:

- t_1 : Then $t_1 \cup h$ creates a directed cycle and the *two* associated nets create a cycle.
- t_2: Then there exists a cycle with *m-1* paths from nets (namely without c_1)
- c_1: If h enters the cycle, it must cross one of the c_i ($i > 1$) creating a cycle with fewer than m paths (at least c_1 may be omitted)

In the two later cases, the cycle discovered may still involve k nets, however repeated applications of the argument must produce a directed cycle with fewer nets as m is strictly decreasing.

□

There is one final stage required. Let N_2 be the fixed net resulting from merging these forced pairs of induced nets into N_1. An induced (unforced) subnet I may now intersect N_2 both properly and improperly since N_2 is a superset of N_1. Such subnets are forced and it is therefore necessary to check each of the remaining induced subnets once again for forcedness with N_2, and choose either subnet I or $\overline{I}$ accordingly. 0

4.3.4 Algorithm to Orient the Edges of a Network with Positively-Weighted Extreme Edges Specified

Compute an initial approximation F of the fixed net and a set of induced subnets $I_1, I_2, \ldots, I_k$ with mirror subnets $\overline{I}_1, \overline{I}_2, \ldots, \overline{I}_k$ via the bundling information.

In the following, whenever a subnet is chosen, its mirror subnet is discarded.

```
**Repeat until** no change to $F$:

    • for all unforced subnets $I$ **do**

        – **if** $I \cup F$ contains a cycle **then** $F := F \cup \overline{I}$

        – **if** $\overline{I} \cup F$ contains a cycle **then** $F := F \cup I$

{all remaining subnets intersect $F$ consistently}

Relabel all remaining pairs of induced subnets so that $I$ flows in the preferred direction.

**For** all induced disjoint pairs of subnets $A$, $B$ **do**

    • **If** $A \cup B \cup F$ contains a cycle **then**

        – **if** $\overline{A} \cup B \cup F$ contains a cycle then $F := F \cup A \cup \overline{B}$
          else $F := F \cup \overline{A} \cup B$

    {Induced nets must be checked once again individually for forcedness}

**Repeat until** no change to $F$:

    • for all unforced subnets $I$ **do**

        – **if** $I \cup F$ contains a cycle **then** $F := F \cup \overline{I}$

        – **if** $\overline{I} \cup F$ contains a cycle **then** $F := F \cup I$
```

```
{all remaining nets intersect $F$ consistently}
Merge the remaining induced nets into $F$ (in preferred direction).
If $F$ contains a cycle, then $N(G)$ has no planar feasible flow.
```

Analysis: The FOR loop involving pairs of subnets determines the complexity of the algorithm. There may be $\theta(n)$ subnets and hence $O(n^2)$ calls to the first cycle detection procedure, which requires a second call to the cycle detector. Since cycles in directed graphs may be detected in linear time, the overall complexity of the algorithm is $O(n^4)$.

4.3.5 Extreme Edges of $N(G)$ Specified

Note that the previous algorithm depended rather heavily on the knowledge of the *non-zero* extreme edges of $N(G)$. Given the ordering G without the s and t adjacencies specified, there are $O(\binom{n}{2})$ ways to separate the vertices on the exterior face into two groups: "potentially s-adjacent" and "potentially t-adjacent". However, each vertex on the exterior face may be attached (weight > 0) or not attached (weight $= 0$). This would incur an $O(2^n)$ overhead if the previous algorithm were used in a straight-forward manner. The overhead may be contained to $O(n^2)$ if the previous algorithm can be modified to handle the case where the s and t attachments are specified *but may equal 0.*

Lemma 11 is the first lemma that appears to fail to hold for the current case; s may not force a directed bundle on an extreme vertex v. See figure 4.16 for example.

Using the notation of lemma 11, if there exists a pair of consecutive edges e_i and e_{i+1} such that $f_i = b_{i+1}$, then no bundle may be forced. This corresponds to case 2c in the proof. An examination of the other 3 cases shows that a forced bundle can be determined if no such pair of edges exists. Call an extreme vertex v **balanced** if such a pair of edges can be determined. This pair of edges will be denoted by e_i^v and e_{i+1}^v, although the superscript

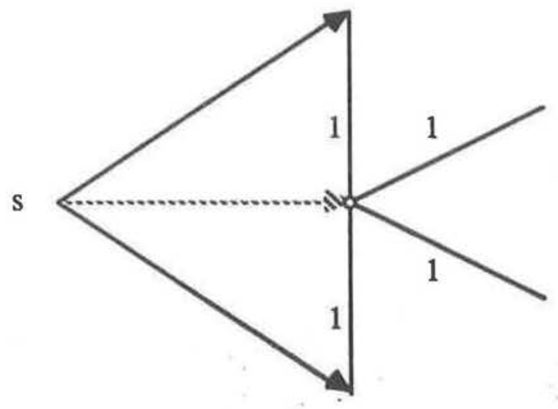


Figure 4.16: No Forced Bundles

will be omitted if the vertex v is unambiguous.

The following lemma shows that even for balanced vertices, one of the edges e_i or e_{i+1} is part of the fixed net; however, it is not possible to compute which one at this stage.

Lemma 14 *If v is balanced and s-adjacent (respectively t-adjacent) then one of e_i or e_{i+1} must be directed out (respectively in).*

Proof:

Suppose that v is s-adjacent.

Suppose e_i and e_{i+1} are both forced in to v. If $e_i = e_1$ or $e_{i+1} = e_n$, then the inflow(v) > outflow(v), so assume $i > 1$ and $n > i + 1$. There exists some path p from s to v involving e_i. The cycle $p,(v,s)$ must contain in its interior either $e_1, e_2, ..., e_{i-1}$ or $e_{i+1}, ..., e_n$. See figure 4.17.

Case 1: Suppose the former. Then $e_1, e_2, ..., e_{i-1}$ must be directed *in* to v else a directed cycle would occur as the path to t would cross p. But then the inflow(v) > outflow(v), yielding a contradiction.

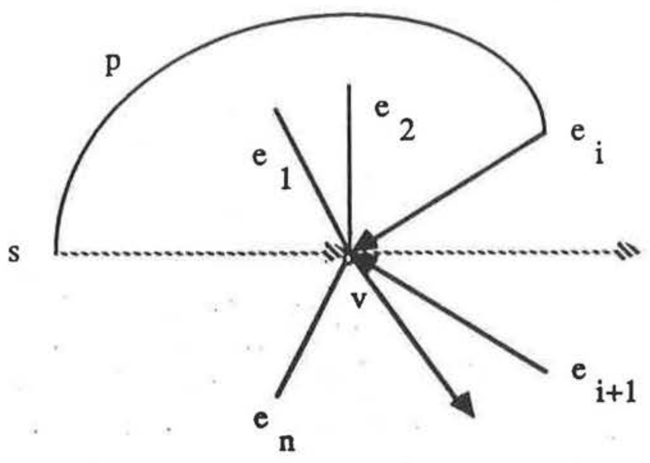


Figure 4.17: Forced Balanced Vertex

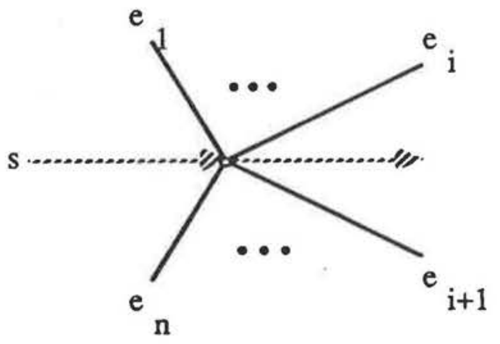


Figure 4.18: Pseudo-arc

Case 2: Suppose the latter. Then at least one edge from $e_{i+2}, ..., e_n$ must be directed out to balance v. But this edge must intersect p, creating a cycle and the required contradiction.

□

Since it is not possible to determine *a priori* which of the two edges of a balanced vertex v is forced, insert an unweighted **pseudo-arc** p between e_i and e_{i+1} - out if v is s-adjacent, in if v is t-adjacent. See figure 4.18

Although $< s, v >, p$ is not itself a true forced net, it may be treated as such since one

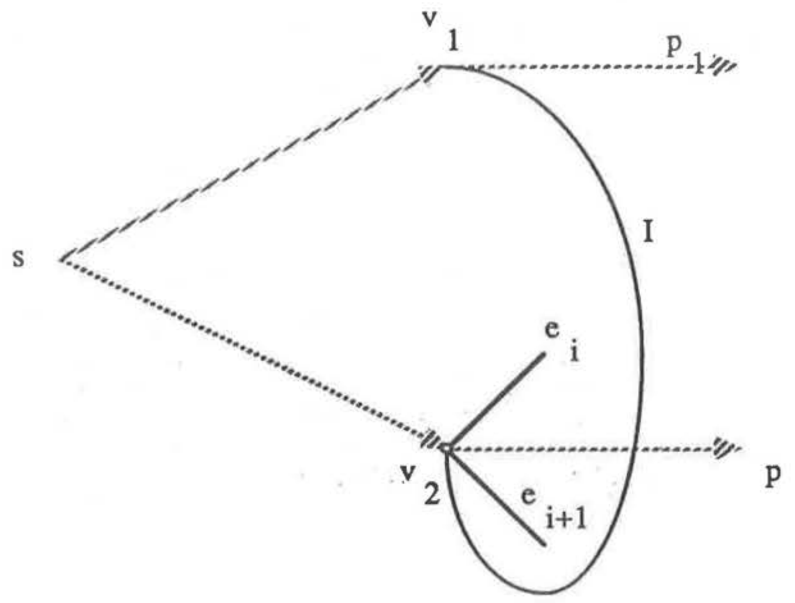


Figure 4.19: Case 5: Form of I

of e_i or e_{i+1} must be part of a directed net.

Lemma 11 now holds with p as the forced bundle. To show that lemma 12 holds requires the following extra cases involving the intersection of an induced net with a forced pseudo-net. Note that the notions of proper and counter-proper intersections still hold for pseudo-nets.

Proof: (of lemma 12 continued)

Induced net I intersects two pseudo-nets at v_1 and v (once properly and once counter-properly). Note that v_1 and v must be extreme vertices. Hence there are two cases:

Case 5: v_1 and v are both s-adjacent. Then I is of the form (or symmetric to) figure 4.19.

Suppose I is directed from v_1 to v. Consider the edges e_i^v and e_{i+1}^v that define p (e_{i+1}^v could be an edge on I). Lemma 14 shows e_i^v or e_{i+1}^v is directed out. If e_{i+1}^v is an edge of I then e_i^v is forced out. In any case, there exists an edge contained entirely within the closed

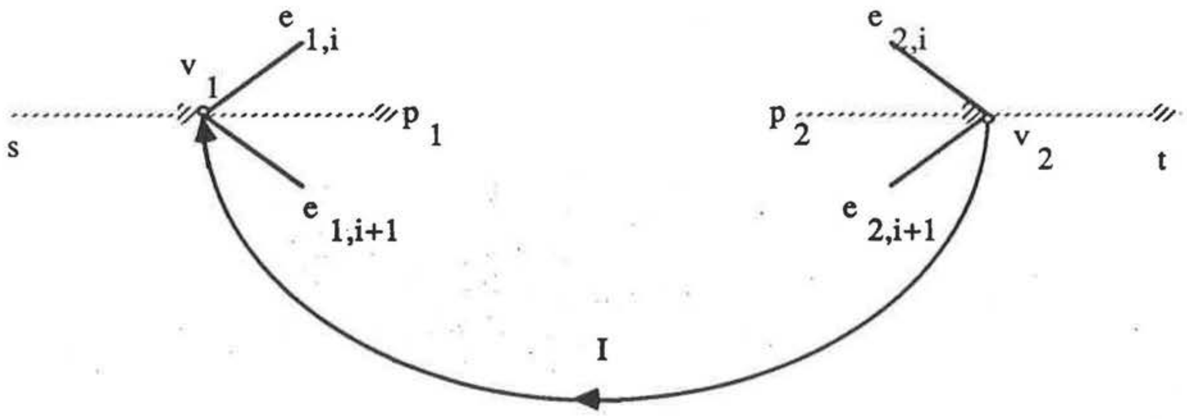


Figure 4.20: Case 6: Form of I

simple curve defined by $(s, v_1), I, (v, s)$ that is out-forced. The induced net associated with this edge must cross I forming a directed cycle. Therefore I is forced to flow from v to v_1 (and no such directed cycle occurs). The case when v_1 and v are t-adjacent is similar.

Case 6: v_1 is s-adjacent and v_2 is t-adjacent. Then I is of the form (or symmetric to) figure 4.20.

The pseudo-nets p_1 and p_2 have defining edges e_i^v, e_{i+1}^v and $e_i^{v_2}$, $e_{i+1}^{v_2}$ respectively.

Suppose I is directed from v_2 to v_1. Then there exists at least one edge e_1' out of v_1 and one edge e_2' into v_2 by lemma 14. The nets induced by them must then lead to t and from s respectively. Furthermore, they must either intersect each other or I and form a directed cycle. Thus I must be forced from v_1 to v_2.

The above arguments may also be applied if I intersects a forced net once and a pseudo-net once.

Thus lemma 12 holds when pseudo-nets are included as part of the fixed net.

□

The remaining lemmas are unaffected by the pseudo-nets.

The following modification to the algorithm is required:

Step 1: When computing the bundling information, create a pseudo-edge p_i for each balanced extreme vertex v_i.

i.e. bundle$(s, v_i) = \{p_i\}$, bundle$(v_i, p_i) = \phi$, if v_i is adjacent to s, or

bundle$(v_i, t) = \{p_i\}$, bundle$(p_i, v_i) = \phi$, if v_i is adjacent to t.

The complexity analysis of the algorithm is unchanged.

This 0-weighted version is the final link required to show that the general ordered case has a polynomial time solution. Given an ordering G, running the previous algorithm for each of the $O(n^2)$ s- and t- adjacent choices for creation of $N(G)$, determines in polynomial time, an orientation which has a planar full flow, if one exists.

4.4 The Unordered Case

Let G be a weighted (undirected) graph, and let $N(G)$ be the graph with s and t attached to all vertices, with associated capacity ∞, as specified in the undirected full flow formulation.

The planar full flow problem requests an orientation of the edges of $N(G)$ such that the resulting network is acyclic and contains a planar full flow. In any such successful planar flow, s and t will be attached only to vertices that form an exterior face of G in some ordering. Considering all orderings of G is in general not computationally feasible, however, it may yet be possible to check if G is a measured bar-representable graph from the results of the ordered cases. If G comes from a class of graphs which has only a polynomial number of distinct orderings, then each ordering may be checked with the

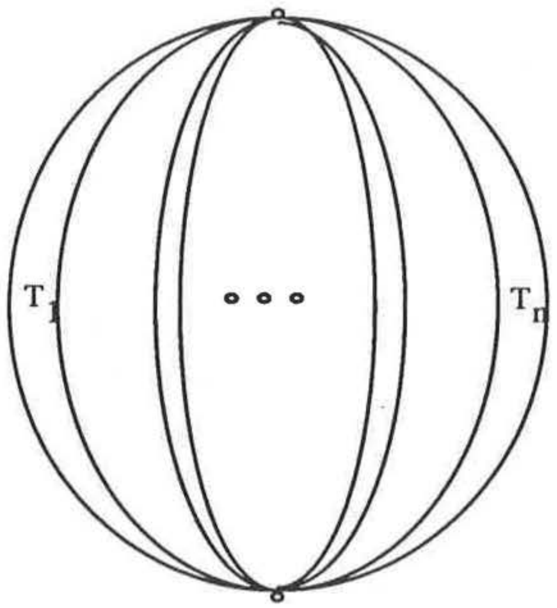


Figure 4.21: Polynomial Number of Exterior Faces

previous algorithm. Triconnected graphs, [1] for example, have only a linear number of planar orderings, as each of the $O(|V|)$ faces may be chosen as the exterior face. (This is a consequence of a result due to Whitney [27].)

The number of orderings alone however is not the crucial factor. The outline of the graph in figure 4.21 has n triconnected components and is itself biconnected but not triconnected. It is a representative of a class of graphs which has $O(n!)$ orderings but only $O(\binom{n}{2})$ possible exterior faces and hence the algorithm of subsection 4.1.3 may be applied.

Each biconnected component of G must be embedded in the exterior face by Theorem 5, and hence as argued in the previous section, G may be decomposed into its biconnected blocks. These blocks may be merged upon completion of the algorithm.

Assume G is biconnected and not triconnected. Then G has at least one **cutpair** of vertices whose removal would disconnect G. A naive algorithm would check each possible

[1] G is triconnected if there is no pair of vertices whose removal would disconnect G.

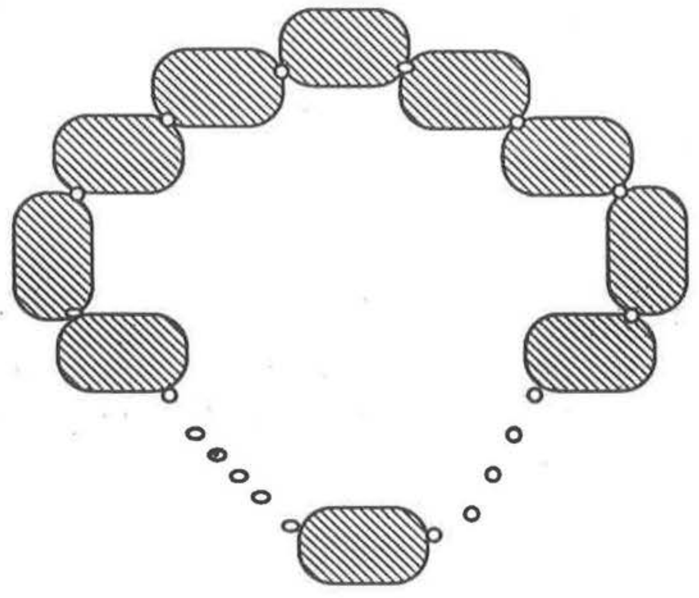


Figure 4.22: Exponential Number of Exterior Faces

exterior face, but again, this may be prohibitively expensive. Consider for example, a class of graphs in which there are $O(n)$ triconnected components arranged as in figure 4.22. Then there are $O(2^n)$ possible exterior faces, as each triconnected component may be flipped about its cutpair.

Triconnected subgraphs are the key to solving this undirected unordered weighted case. Their structure is sufficiently fixed that the number of acyclic orientations is limited, and since they are attached to the rest of the graph on *exactly* two vertices, their interaction with the remainder is somewhat predictable. Note that G may be decomposed into its triconnected components in linear time and furthermore, these components are unique (see Hopcroft and Tarjan [13]).

The algorithm consists of several stages in which the triconnected components of G are successively collapsed until a sufficiently small graph is obtained. Initially, the triconnected components of G are identified and classified by type. Triconnected components of the

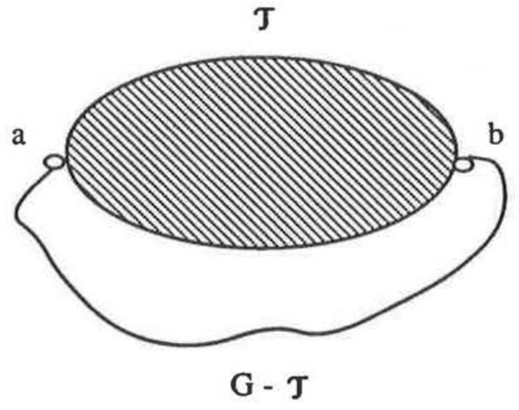


Figure 4.23: τ and $G - \tau$

simplest type are collapsed (recursively) into weighted edges. In the resulting graph, the remaining uncollapsible triconnected components must all appear on the exterior face, but, as in figure 4.22, both sides of each component apparently must be checked. To avoid this exponential overhead, a canonical form (called a *chunk*) is chosen for each triconnected component and these chunks are carefully merged until a graph of constant size is created. A layout for this final graph can be converted into a layout for the original graph by substituting the layouts for the collapsed triconnected components, in reverse order.

Let T be a triconnected component of G with cutpair a, b. Since T is triconnected, there exists a unique face F, containing both a and b. Let τ represent the (unique) ordering of T with a and b on the exterior face. Note that a and b divide the exterior face of τ into two chains. The remainder of G will be denoted by $G - \tau$ (see figure 4.23 for a sketch). Let G be an ordering which admits a bar layout and let $\mathcal{N}(\mathcal{G})$ be the embedding of the associated network with s and t attached. Then either s and t belong to face F, or to some interior face of T.

Suppose T appears as τ in $\mathcal{N}(\mathcal{G})$. Then there are two possible forms:

1. The flow through τ is entirely via a and b and no other vertex of τ is adjacent to

s or t. (In bar terms, all bars of τ lie entirely between $\overline{a}$ and $\overline{b}$). τ is said to be **collapsible.**

2. Some of the vertices on one of the two chains of τ are adjacent to s or t.

The "third" form in which vertices on *both* chains of τ are visible to s or t may occur only in the trivial case when G is in fact triconnected. Note that triconnected components of the first type act effectively like "super edges", in the sense that the flow through τ is strictly from a to b or from b to a. If τ is collapsible then there exists a full flow for τ in which the vertices on *both* chains are not visible to s or t. Assume without loss of generality that the flow is into a and out of b. Then the flow through τ is $\sum_{v \epsilon \tau} weight(a, v)$ or equivalently $\sum_{u \epsilon \tau} weight(u, b)$. Call this value w_τ and denote by G/τ the graph G with τ replaced by an edge (a, b) of weight w_τ. (If (a, b) is already an edge of G then the two edges are merged with the weights summed). Clearly, if there exists a full flow for G/τ then there exists a full flow for G: the flow for τ may simply be substituted for the associated edge of G/τ. The following crucial theorem confirms the converse.

First, it is worth noting the interpretation of the following theorem in terms of the associated bar layout. Component τ, and $G - \tau$ are not visible to each other except via bars $\overline{a}$ and $\overline{b}$. Given a bar layout in which τ does not appear entirely between $\overline{a}$ and $\overline{b}$, the theorem insures that $\overline{a}$ or $\overline{b}$ can be modified to permit such a layout of τ without disturbing the layout of $G - \tau$. There are many possible arrangements of τ and $G - \tau$ about $\overline{a}$ and $\overline{b}$, but since $\overline{a}$ and $\overline{b}$ are not visible, they can be classified as one of only two basic forms:

- $\overline{a}$ and $\overline{b}$ overlap but are blocked by τ or $G - \tau$, or
- the top of one bar abuts with the bottom of the other.

Lemma 15 *If τ is collapsible and there exists a full flow for G with T embedded as τ, then there exists a full flow for G/τ.*

Proof:

[As an aid in checking that all cases are considered, the bar interpretation of each case will be indicated in square parentheses.]

Let $\mathcal{F}$ be a full flow for G in which τ receives assistance from s or t. A modified full flow, $\mathcal{F}'$, for G/τ will be exhibited that behaves on $G - \tau$ like $\mathcal{F}$.

Note that either τ consists of a single vertex v and a and b (the degenerate case) or τ receives the assistance on one path referred to as the **outside path** of τ, and the **inside path** of τ has no s or t adjacencies.

Case 1: (Degenerate Case):

Suppose τ consists solely of a,b and one vertex v. There are only two subcases as illustrated in figure 4.24. In A, the full flow for G will appear as a full flow in G/τ as in B. The full flow for C, will appear as one of the two flows in D (one may create a cycle since the flow in $G - \tau$ may contain a directed path between a and b).

Case 2: [$\overline{a}$ and $\overline{b}$ abut] Suppose there is no directed path from a to b in $\mathcal{F}$, in either τ or $G - \tau$. Then there are three subcases:

Case 2.i: [τ is entirely on one side of $\overline{a}$ and $\overline{b}$] Component τ is adjacent to s and the flow to a in τ is *in* and the flow to b in τ is *in* as in figure 4.25 A. Neither of the two flows in 4.25 B creates a cycle with $G - \tau$ and either represents a valid flow in G/τ.

Case 2.ii: [τ wraps around one bar] Component τ is adjacent to s and to t in $\mathcal{F}$ (without loss of generality).

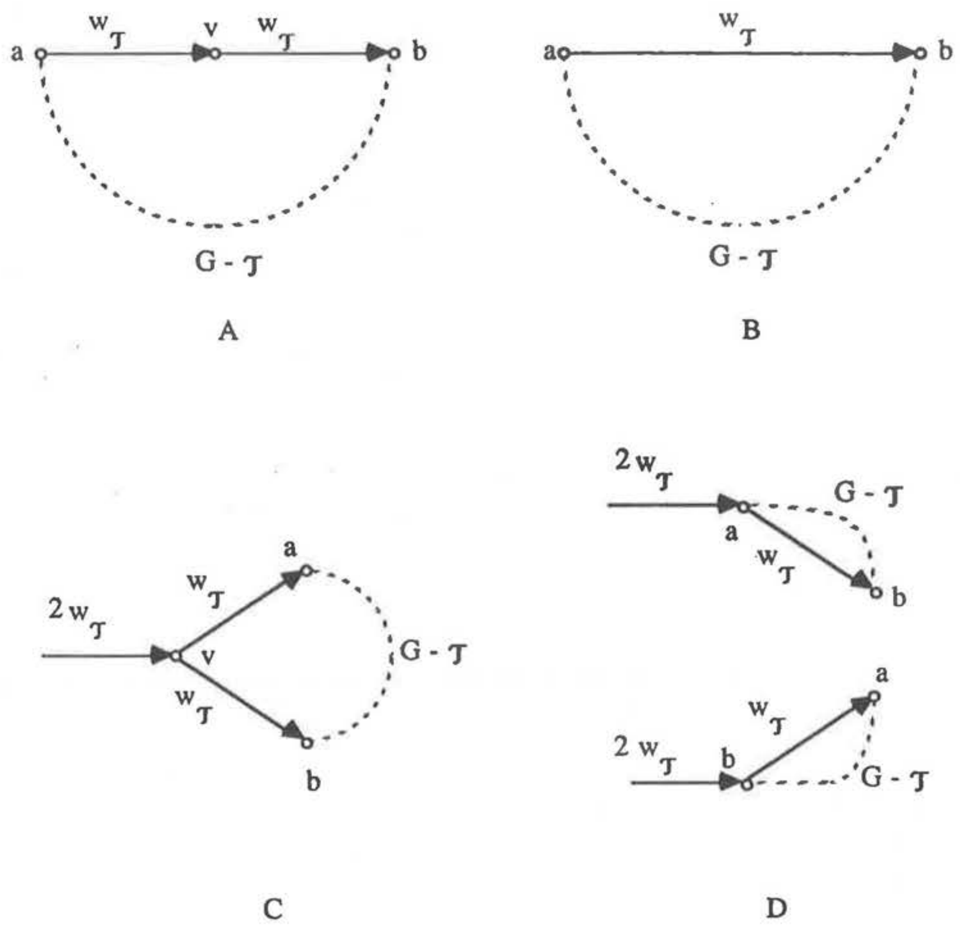


Figure 4.24: The Two Degenerate Cases

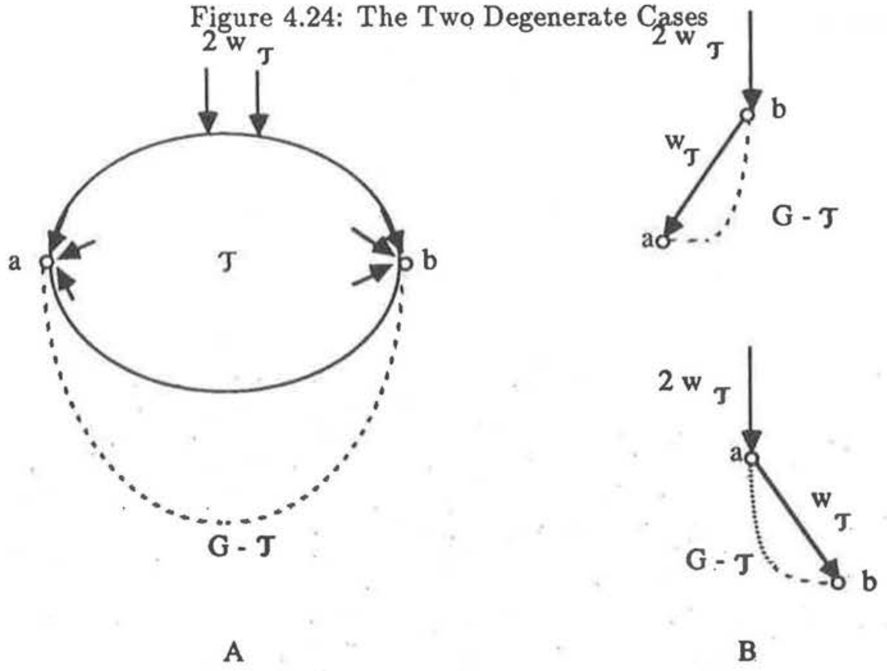


Figure 4.25: Case 2.i

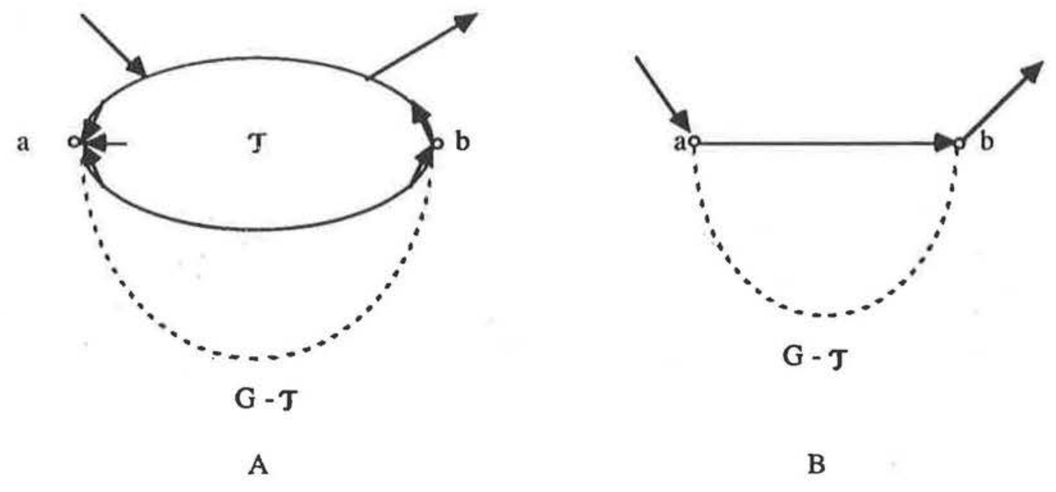


Figure 4.26: Case 2.ii

Then the full flow $\mathcal{F}$ has the general form as in 4.26 A. The full flow for G/τ as shown in 4.26 B substitutes for τ in $\mathcal{F}$ without affecting the flow in $G - \tau$.

Case 2.iii: [τ wraps around both bars (or equivalently, $G - \tau$ is entirely on one side)] τ is adjacent to t, to s and to t on the outside path as in figure 4.27 A.

As a consequence, $G - \tau$ may only be adjacent to t and thus $G - \tau$ must receive all of its flow via a and b. Either of the flows for G/τ in figure 4.27 B can supply the identical assistance to $G - \tau$, introduces no directed cycle and satisfies the s and t adjacency requirements.

Case 3: [Part of τ blocks $\overline{a}$ and $\overline{b}$]: There exists a path in τ from a to b (without loss of generality).

Substituting the flow in 4.28 B does not interfere with $G - \tau$ and thus G/τ has a full flow.

Case 4: [Part of $G - \tau$ blocks $\overline{a}$ and $\overline{b}$, and τ does not] There is a path in $G - \tau$ from a to b and there is no such path in τ.

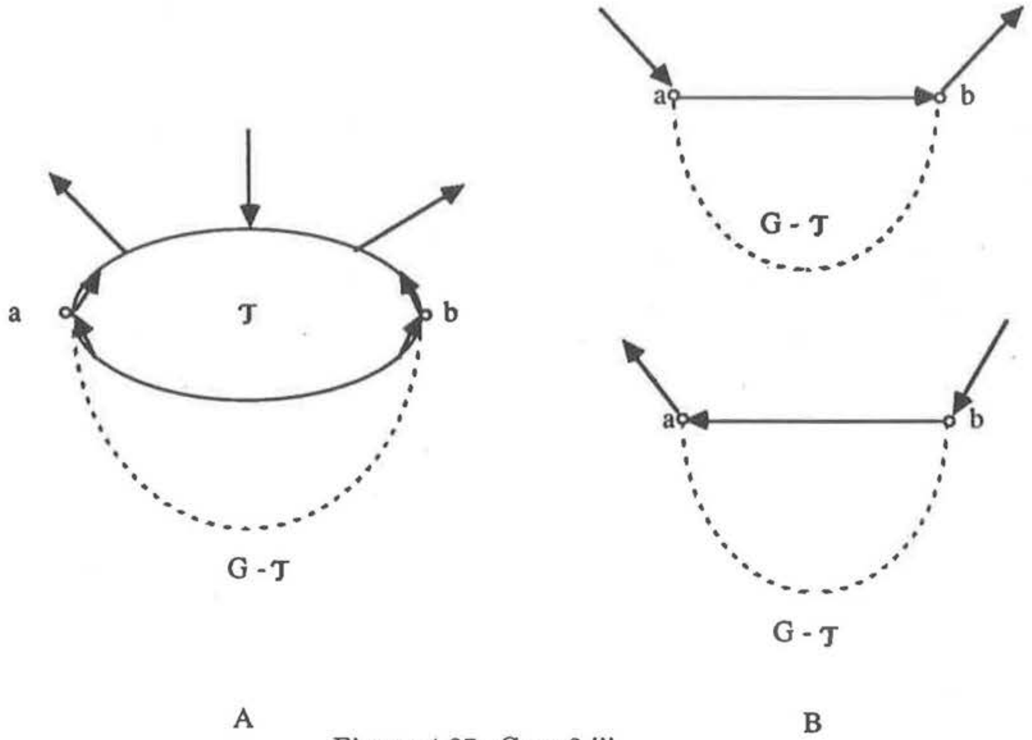


Figure 4.27: Case 2.iii

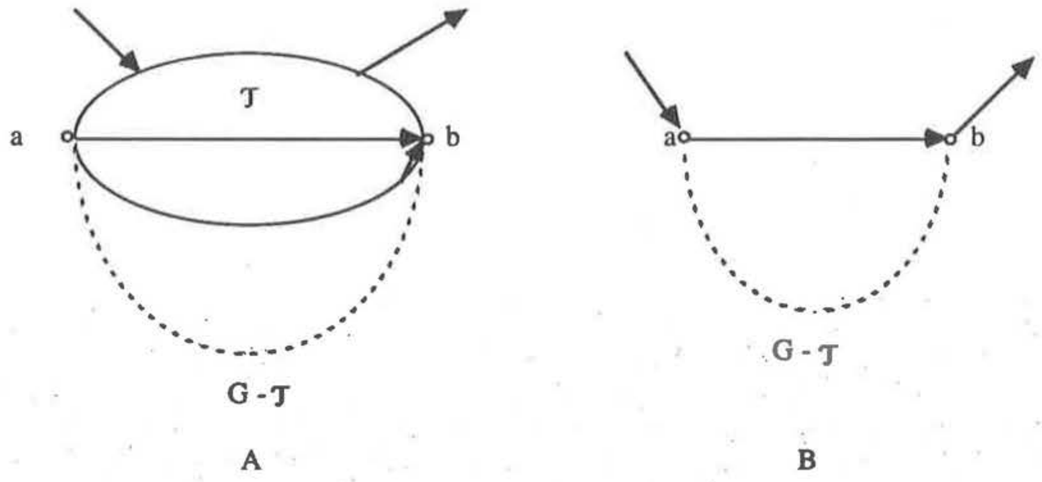


Figure 4.28: Case 3

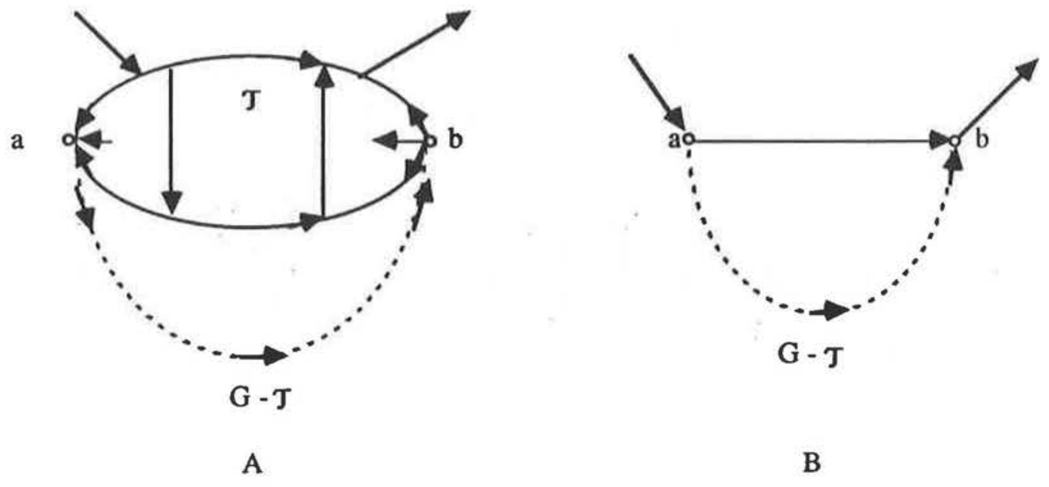


Figure 4.29: Case 4

The flow represented in figure 4.29 B for G/τ is a valid modification of $\mathcal{F}$ as shown in figure 4.29 A since one of a or b can absorb any extra flow without disturbing $G - T$.

□

If T does not appear as τ in $\mathcal{N}(\mathcal{G})$ then s and t are embedded in some interior face of T and $G - T$ is entirely contained in a (different) interior face of $\mathcal{N}(\mathcal{G})$ without direct access to s or t. Hence, all other triconnected components of G (and in fact $G - T$ itself) must have form 1 above.

The first stage of the algorithm identifies all triconnected components of G (a linear time operation [13]) and classifies them. Collapsible triconnected components are recursively replaced by suitably weighted edges.

The resulting graph, G', is either triconnected (in which case the number of embeddings is linear and the algorithm for the embedded case may be used) or consists of edges and triconnected components requiring assistance from s or t (i.e. non-collapsible components). Hence, if this graph does not have an embedding with all triconnected components on

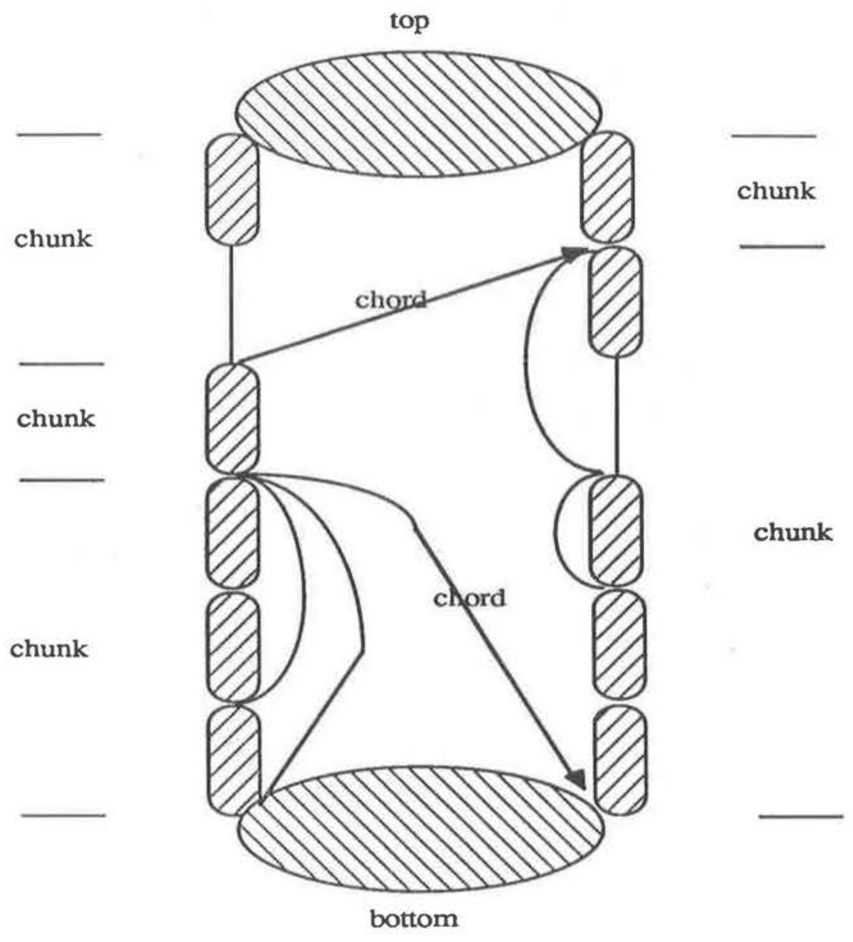


Figure 4.30: Five Chunks

the exterior face, then there does not exist a full flow. Therefore, assume G' is partially embedded with all triconnected components on the exterior face in a ring. See for example, figure 4.30.

The embedding is *not* however fixed - the appropriate "side" of each component must yet be chosen (either of the two "exterior" paths of τ may potentially be chosen as exterior for $\mathcal{G}$). There may still be $\Theta(n)$ components.

A second degree of freedom involves the "direction" for each component. There are potentially three choices: the flow is generally from a to b, from b to a, or the flow splits. Stage 2 of the algorithm avoids the seemingly exponential ambiguity of the partial "embedding", by exploiting the structure of *chains* of triconnected components.

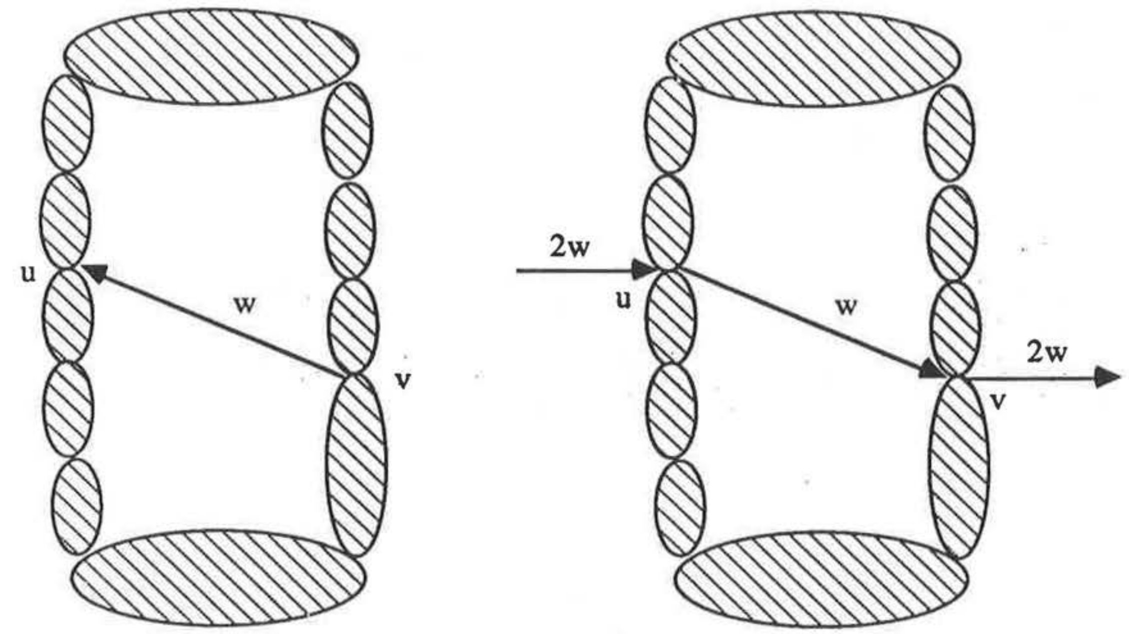


Figure 4.31: Redirection of Chord $< v, u >$

Some pair of triconnected components must appear in the flow as potentially visible to *both* s and t. Call these the **top** and **bottom** components. The algorithm will try all pairs of triconnected components for top and bottom. There are at most $\mathcal{O}(\binom{n}{2})$ pairs and two sides for each of these components. Once a pair has been chosen for top and bottom, those components visible only to s and those only to t are known; they form adjacent partial rings on either side of the top and bottom, referred to as the **s-side** and the **t-side**. The "interior face" of G' consists of edges between vertices of the set of cutpairs.

Define a **chord** as an edge with endpoints on both the s-side and the t-side. A full flow might require chords directed from the t-side to the s-side. However, many such chords may be safely redirected in a flow from the s-side to the t-side, without disturbing the rest of the flow. Figure 4.31 shows how the chord may be redirected maintaining the balance at u and v.

This redirection might create a directed cycle; the following lemma describes a sufficient condition for redirecting a batch of such chords avoiding the creation of a directed cycle.

Label the chords from top to bottom as $c_1, c_2, \ldots, c_k$ and let the interior paths of the top and bottom components be considered honourary chords c_0 and c_{k+1}.

Lemma 16 *Let $\mathcal{F}$ be a full flow in which chords c_i and c_j are directed from the s-side to the t-side and $c_{i+1}, \ldots, c_{j-1}$ are directed from the t-side to the s-side. Then there exists a full flow $\mathcal{F}'$ in which all chords between c_i and c_j are directed from the s-side to the t-side.*

Proof:

The balancing conditions may easily be conserved as in the figure. Suppose a directed cycle is introduced involving c_k $(i < k < j)$, directed from the s-side to the t-side. Since none of the chords $c_i, \ldots, c_j$ are directed from the t-side to the s-side, the cycle involves a chord c_l $(l < i$ or $j < l)$ in this direction. Assume without loss of generality that $j < l$. Then there must exist a directed cycle in flow $\mathcal{F}$ involving c_j and c_l - a contradiction since $\mathcal{F}$ is a full flow. Hence, no cycle may arise by redirecting $c_{i+1}, \ldots, c_{j-1}$.

□

A consequence of this lemma is that if there is a flow for G', then there is one in which the chords may be considered as partitioned into three sets of adjacent chords: those from the t-side to the s-side $(c_0, \ldots, c_{i-1})$, those from the s-side to the t-side $(c_i, \ldots, c_j)$, and those from the t-side to the s-side $(c_{j+1}, \ldots, c_{k+1})$.

There are $\binom{k+2}{2}$ ways to partition the chords into these three consecutive groups. Since k may be $\Theta(n)$, this contributes an $O(n^2)$ overhead to the algorithm, as each partition will be generated and examined.

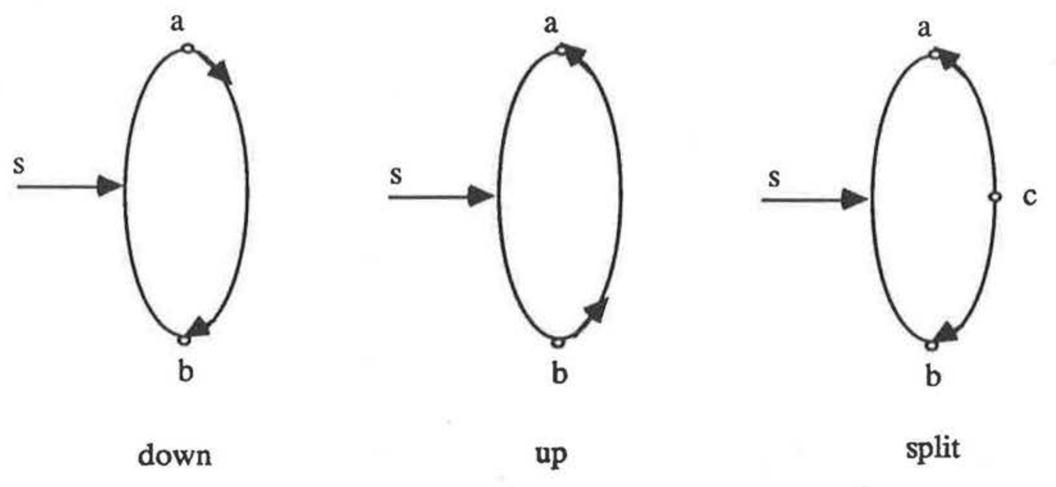


Figure 4.32: The Three Forms of the Inside Path of τ

The next several lemmas involve a (non-collapsible) triconnected component τ on the s-side with cutpair a, b. The corresponding lemmas for t-sided components are similar and are omitted.

For whichever side of τ is chosen as the inside (call it p), the flow may only take one of three possible forms (see figure 4.32):

- directed from a to b (called **down**)
- directed from b to a (called **up**)
- split at a vertex c (called **split**) with the two paths directed from c to b and c to a (oppositely for t-sided components).

These three forms are a necessary consequence of the following lemma.

Lemma 17 *In any flow $\mathcal{F}$, no pair of arcs of p meet head to head.*

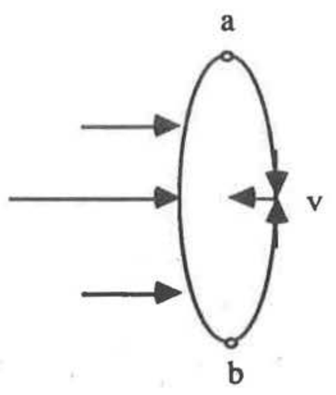


Figure 4.33: Invalid Split on p

Proof:

Suppose such a pair of arcs meets at a vertex v on p as in figure 4.33. Since τ is s-adjacent, there must exist a directed path from v to a or from v to b to balance v. This path must intersect one of the two paths from s to v creating a directed cycle.

□

For collapsible triconnected components, the sums of the incident weights in τ at a and at b were equal. For non-collapsible components, these sums may not be the same. Define $\sum_a^\tau$ as $\sum_{u \in \tau}$weight(u,a), and $\sum_b^\tau$ as $\sum_{u \in \tau}$weight(u,b). (The τ superscript will be dropped if the triconnected component is clear).

In a particular flow $\mathcal{F}$, the amounts of inflow at a and b in τ may not be respectively $\sum_a^\tau$ and $\sum_b^\tau$. Define for cutpair a, b of triconnected component τ, **inflow**$(a,\tau,\mathcal{F})$ as $\sum_{u \in \tau, <u,a> \in \mathcal{F}}$ weight$(<u,a>)$ and **inflow**$(b,\tau,\mathcal{F})$ as $\sum_{u \in \tau, <u,b> \in \mathcal{F}}$ weight$(<u,b>)$. (Again, the τ and $\mathcal{F}$ will be omitted if the context is clear).

Define **outflow** as $\sum$ − inflow.

First, it is important to investigate the relation between the inflows at a and b and the possible forms of τ in $\mathcal{F}$. These relations will allow the creation of canonical forms for the non-collapsible triconnected components, which may then be collapsed.

Lemma 18 *If the flow in τ is up (respectively down) then inflow$(a,\tau) = \sum_a^\tau$ (respectively inflow$(b,\tau) = \sum_b^\tau$). If the flow in τ splits, then both conditions hold.*

Proof:

i) (up) Suppose inflow$(a,\tau) \neq \sum_a^\tau$. Then there is an arc out of a in τ, whose flow must escape either through a or b since τ is s-sided, creating a directed cycle.

ii) (down) Similarly.

iii) (split) Suppose inflow$(b,\tau) \neq \sum_b^\tau$. Then there is a directed path q from b that must flow to t via a. Let $< c, b >$ be the arc on p incident to b. There exists a flow from s to c which necessarily intersects q creating a directed cycle.

□

If the flow in τ is up (respectively down), then the inflow(b) (respectively inflow(a)) is still unknown. In fact, these values will generally differ from flow to flow. Fortunately, it is possible to choose one value for each form that is sufficiently flexible to encode all valid flows of that form.

Let $\mathcal{F}'$ be the flow that maximizes outflow$(a,\tau,\mathcal{F})$ over all flows $\mathcal{F}$ that have the property inflow$(b,\tau,\mathcal{F}) = \sum_b^\tau$.

Define Δ_a^τ as outflow$(a,\tau,\mathcal{F}')$ − inflow$(a,\tau,\mathcal{F}')$, and define Δ_b^τ similarly. Then Δ_a^τ represents the maximum amount of flow absorbable by τ from the component above τ if

τ's flow is down.

It is not immediately clear how the Δ^τ can be computed efficiently since this appears to involve a flow optimization. Note that the possible inflows to a are discrete and the number of inflows is linear in the degree of a in τ. The embedded flow algorithm will be called for each valid partition of the edges adjacent to a in τ. Denote the two sides of τ by p_1 and p_2 and label the edges adjacent to a in τ in order as $e_1, e_2, \ldots, e_k$ with e_1 on p_1 and e_k on p_2. Let $N_j^1(a,\tau)$ be the network: τ with s joined to a with weight of $\sum_{i=j+1}^{k}$ weight(e_i) $-$ $\sum_{i=1}^{j}$ weight(e_i), t joined to b with weight of $\sum_b^\tau$, and s joined to all vertices on p_1 (except b) with weight unknown (≥ 0).

Let $N_j^2(a,\tau)$ be the network: τ with s joined to a with weight of $\sum_{i=1}^{j-1}$ weight(e_i) $-$ $\sum_{i=1}^{j}$ weight(e_{k-i}), t joined to b with weight of $\sum_b^\tau$, and s joined to all vertices on p_2 (except b) with weight unknown (≥ 0).

Now, Δ_a^τ can be computed in polynomial time, by considering the embedded flows of the $N_j^i(a,\tau)$. It is the maximum over all j of the two differences above. Note that only j values that result in non-negative values of these differences need be considered.

In any flow in which τ is directed down, for example, the amount of flow absorbed by τ from above is maximized by the flow in τ achieving Δ_a^τ without altering the amount of flow passed on by τ to the component below. Thus, if the flow type were known *a priori*, τ could be "collapsed" in a manner similar to the previous stage. The algorithm postpones the committment to a particular flow type by storing all three flow types and the $\sum$ and Δ values as an abstract triconnected component called a *chunk*. A pair of adjacent chunks will be selectively and recursively merged to form a higher order chunk that interacts with G' like a triconnected component in the sense that exactly two of its vertices communicate with G' and furthermore its flow is either up, down, or splits. Define a **chunk** inductively as either:

- a triconnected component with cutpair a, b together with the four-tuple $<\sum_a^\tau, \sum_b^\tau, \Delta_a^\tau, \Delta_b^\tau>$ and the triple of boolean values $< Up, Down, Split >$ indicating the valid flow types for τ OR
- the *merge* of two chunks OR
- the *absorption* of a chunk and an s-chord.

Let τ_1 and τ_2 be a pair of adjacent chunks with cutpairs a_1, b_1 and a_2, b_2 respectively, (b_1 is equivalent to a_2) such that b_1 has no chord incident upon it. The **merge** of τ_1 and τ_2 has cutpair a_1, b_2 and the four tuple: $< \sum_{a_1}^{\tau_1}, \sum_{b_2}^{\tau_2}, \Delta_{a_1}^{\tau_1}, \Delta_{b_2}^{\tau_2}>$. The flow type booleans are set according to the following rules:

$Down := Down_1 \wedge Down_2 \wedge (\Delta_{a_2}^{\tau_2} \geq \sum_{b_1}^{\tau_1})$

$Up := Up_1 \wedge Up_2 \wedge (\Delta_{b_1}^{\tau_1} \geq \sum_{a_2}^{\tau_2})$

$Split := Up_1 \wedge Down_2$.

(These conditions simply require that the τ_i have the appropriate flow type and that the "receiving" chunk is capable of accepting the flow from its neighbour).

If there is an s-chord with weight w incident upon the cutpair of a chunk τ, the flow in τ has only two possible forms, since it can not be split along the s-chord. The s-chord will be **absorbed** into the chunk as follows. The *split* value must be set to *false* and the flow values of τ incremented by w to reflect the extra capacity of the chunk.

If chunks are merged and s-chords absorbed until no further reductions are possible, the resulting graph, G'' consists entirely of chunks separated by chords as in figure 4.34, for example.

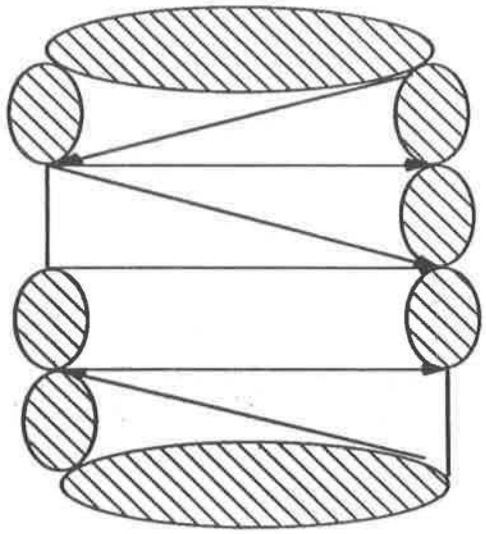


Figure 4.34: G''

Note that the *order* of merging is of no consequence. Since the merge of two chunks retains the flow values of the extreme cutpair, and since the flow type validity checks involve only comparisons between flow values of two chunks at their shared vertex, exactly the same tests are performed in any ordering of merges.

The final stage of the algorithm merges adjacent pairs of chunks sharing a chord into *c-chunks.* These c-chunks are merged until no further collapsing is possible, yielding a graph G''' consisting of a top and bottom component, at most four c-chunks on the s side, and at most four c-chunks on the t side. This final graph has a constant number of embeddings and may thus be laid out by checking all embeddings.

A c-chunk has intercourse with its two adjacent neighbours (on the same side) and possibly via a chord with a c-chunk on the other side. There are *four* flow types for a c-chunk. By choosing a canonical form for each of these types, sufficient information can be exported to the merged c-chunks that a full flow for the final graph implies a full flow for

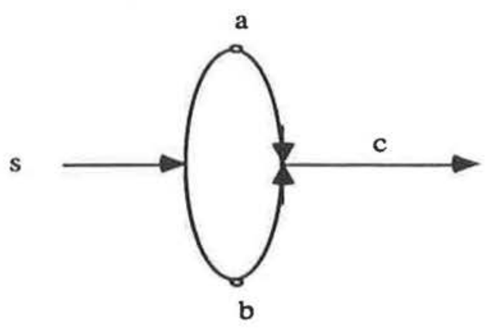


Figure 4.35: *Out* Flow Type

G'' which can be efficiently computed by unwinding the recursion. The critical observation is that a merged c-chunk also has only one of four possible flow types (not sixteen) thus avoiding a potentially exponential blowup in the merging step.

Define a **c-chunk** τ as a generalized chunk with:

- cutpair a, b
- flow value four-tuple $<\sum_{a_1}^{\tau_1}, \sum_{b_2}^{\tau_2}, \Delta_{a_1}^{\tau_1}, \Delta_{b_2}^{\tau_2}, >$.
- boolean four-tuple $< Up, Down, Split, Out >$ indicating the valid flow types of τ. (The new flow type **Out** has the form in figure 4.35).
- interior chord value c^{τ}. (This value will be positive for chords directed from the s-side to the t-side, and negative otherwise).

These c-chunks are created by merging either two chunks or two c-chunks. The merge of a c-chunk and a chunk may also be required but the conditions are sufficiently similar to be omitted. Not all adjacent chunks may be combined. Define a chunk as **mergeable** if all the chords incident upon it are consistently directed. Note that there are at most four non-mergeable chunks as a consequence of lemma 16.

Let τ_1 and τ_2 be two adjacent mergeable chunks whose shared vertex b_1 $(=a_2)$ has incident chords whose weights sum to c. Then the **merge** of τ_1 and τ_2 has cutpair a_1, b_2, flow values $<\sum_{a_1}^{\tau_1},\sum_{b_2}^{\tau_2},\Delta_{a_1}^{\tau_1},\Delta_{b_2}^{\tau_2},>$, and chord value c. The valid flow types are determined by the following checks:

$Down := (Down_1 \wedge Down_2 \wedge \sum_{b_1}^{\tau_1} \leq c + \Delta_{a_2}^{\tau_2}) \vee$
$(Down_1 \wedge Split_2 \wedge c \geq \sum_{b_1}^{\tau_1} + \sum_{a_2}^{\tau_2})$.

$Up := (Up_1 \wedge Up_2 \wedge \sum_{a_2}^{\tau_2} \leq \Delta_{b_1}^{\tau_1} + c) \vee$
$(Split_1 \wedge Up_2 \wedge c \geq \sum_{b_1}^{\tau_1} + \sum_{a_2}^{\tau_2})$.

$Split := (Split_1 \wedge Split_2 \wedge c \geq \sum_{b_1}^{\tau_1} + \sum_{a_2}^{\tau_2}) \vee$
$(Up_1 \wedge Down_2 \wedge c > 0) \vee$
$(Up_1 \wedge Split_2 \wedge \sum_{a_2}^{\tau_2} \leq \Delta_{b_1}^{\tau_1} + c) \vee$
$(Split_1 \wedge Down_2 \wedge \sum_{b_1}^{\tau_1} \leq \Delta_{a_2}^{\tau_2} + c)$.

$Out := (Down_1 \wedge Up_2 \wedge c \geq \sum_{b_1}^{\tau_1} + \sum_{a_2}^{\tau_2})$.

As in the previous merging, these conditions require that the flow be absorbable for each type and that the τ_i be of the appropriate type.

A pair of adjacent mergeable ***c-chunks*** τ_1 and τ_2 is merged in a similar fashion. The resulting c-chunk has cutpair a_1, b_2, chord value $c + c_1 + c_2$, and flow values $<\sum_{a_1}^{\tau_1},\sum_{b_2}^{\tau_2},\Delta_{a_1}^{\tau_1},\Delta_{b_2}^{\tau_2}>$. The flow type checks listed for the previous merging are augmented as follows:

$Down := \ldots \vee (Out_1 \wedge Down_2) \vee$
$(Out_1 \wedge Split_2 \wedge \Delta_{b_1}^{\tau_1} + c \geq \sum_{a_2}^{\tau_2})$

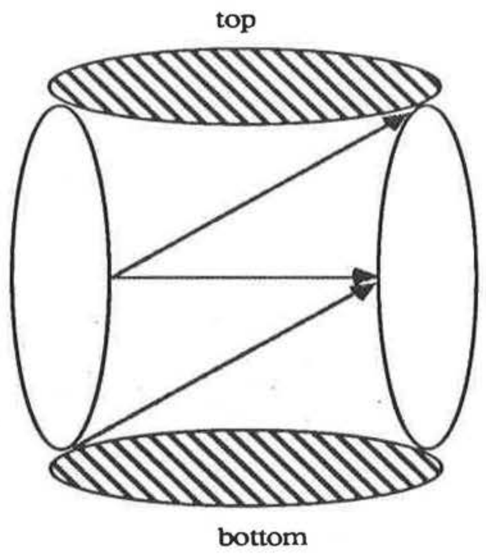


Figure 4.36: G'''

$Up := \ldots \vee (Split_1 \wedge Out_2 \wedge \Delta_{a_2}^{r_2} + c \geq \sum_{b_1}^{r_1}) \vee$
$(Up_1 \wedge Out_2)$.

$Out := \ldots \vee (Out_1 \wedge Up_2 \wedge \Delta_{b_1}^{r_1} + c \geq \sum_{a_2}^{r_2}) \vee$
$(Down_1 \wedge Out_2 \wedge \Delta_{a_2}^{r_2} + c \geq \sum_{b_1}^{r_1}) \vee$
$(Out_1 \wedge Out_2)$.

Mergeable chunks and c-chunks are collapsed as much as possible, yielding a graph G''', which has no more than four c-chunks on each side, a top and a bottom component; a sketch of such a graph is given in figure 4.36.

There are at most four flow types for each of the c-chunks and hence a constant number of calls to the embedded flow algorithm are required for a given choice of top and bottom orientations. (The inside and outside path of the top and bottom components must also be chosen).

The complete algorithm will now be presented.

4.4.1 Algorithm to Determine a Full Flow for an Unordered Graph

```
Procedure fflow($G$)
{ returns a full flow for $G$ or false.}

    If $G$ is triconnected then test each of the $\mathcal{O}(|T|)$ embeddings $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_t$ with the embedded flow algorithm and return false if all fail.
else If there exists a triconnected component $T$ of $G$ with cutpair $a, b$ such that $T$ is collapsible
then if $G - T$ is collapsible (on $a, b$) then { check with $G - t$ first }

    • Let $G_1$ be $G$ with $G - T$ collapsed.

    • For each face of $T$ as exterior face, check embeddedflow($\mathcal{G}_1$) and return the flow for $\mathcal{G}_1$
      with $G - T$ substituted, if successful.

Let $\tau$ be the unique embedding of $T$ with $a, b$ on the exterior face. Return fflow($G/\tau$) with the
collapsing of $T$ substituted

    else {there is no collapsible triconnected component}
Determine an embedding $\mathcal{G}'$ with all triconnected components on the exterior face. (If no such
embedding exists - there is no full flow for $G$ - return false.)

    For all pairs of triconnected components chosen as top and bottom do:

    For all partitions of the chords into three adjacent oppositely-directed sets do:

    • For each triconnected component, determine the valid flow types and flow values.
```

- Absorb all s-chords and merge all adjacent pairs of triconnected components and chunks that do not share a chord, yielding graph G''.

- Merge adjacent pairs of mergeable chunks and c-chunks to obtain G'''.

- Determine a full flow for G''' via the embedded flow algorithm. If there exists a flow, then

  - In reverse order of merging, backtrack, substituting the known flow.

  - Halt done.

Return *false.*

Finally, note that although the algorithm runs in polynomial time, it is certainly not a low order polynomial. The next chapter confirms that some variants of the Full Flow Existence problem are indeed NP-Complete, as may have been suspected.

Chapter 5

NP-Complete Full Flow Versions

5.1 Introduction

It is perhaps not surprising that some variants of the undirected weighted case (as expressed in the flow formulation), would be NP-Complete - there are several degrees of freedom that may be manipulated to provide a suitable environment. Three fairly natural forms of the problem can be shown to be NP-Complete. Note that two of these problems are NP-Complete *in the strong sense* - even if the values of the weights are bounded, the problems remain NP-Complete.

Define a network N as **replete** if the source s and sink t are joined to every vertex of the network with capacity ∞. Recall that the Undirected Full Flow problem requests, for a given undirected network N, an orientation g such that $g(N)$ has a planar full flow. The algorithms of chapter 4 prove the Undirected Full Flow problem has a polynomial time solution if either the network $N(G)$ is initially planar (section 4.2) or if $N(G)$ is replete (section 4.3). If however, $N(G)$ is initially non-planar but not replete, then the Undirected Full Flow problem is weakly NP-Complete [1]. (See [9] for a discussion of strong

[1] such problems may have polynomial time solutions only if the weights involved are bounded by a constant (and assuming that $P \neq NP$).

vs. weak NP-Completeness). The proof of this claim is a straightforward reduction from the Partition problem and is omitted.

A non-planar version of the problem may also be formulated and is interesting in its own right, however the connection to issues of bar-representability is tenuous. The Undirected Non-Planar Full Flow problem will request a (not necessarily planar) full flow for a given network. The problem has a trivial polynomial time solution if the network N is replete - any acyclic orientation provides a full flow since every vertex is balanceable via its connection to s or t. The problem is however strongly NP-Complete[2] if the network is not replete as shall be shown in the following section.

Finally, this result will be used to show the strong NP-Completeness of a cyclic version of the problem: Given an undirected planar network N, determine a directing of the edges which admits a full flow with directed cycles permitted. Note that this stands in bold contrast to the polynomial solution of section 4.2.

5.2 The Non-Planar Full Flow Problem

Define the **Undirected Non-Planar Full Flow** problem as follows. Given an undirected capacitated network N, with specified source s and sink t, does there exist an orientation, g, of the edges, such that $g(N)$ contains no directed cycle and there exists a full flow f, for $g(N)$? (Note that the full flow defined by f is not restricted to be planar.)

Theorem 7 *The Undirected Non-Planar Full Flow problem is NP-Complete.*

Proof:

[2]and thus NP-Complete even if the weights are bounded

a) the problem is in NP as it is possible to verify in $O(|E|)$ time if a proposed flow for a given instance is in fact a solution. (The flow on each edge must be checked to be valid and each vertex must have balanced inflow and outflow).

b) The following lemma proves a known NP-Complete problem reduces to the Undirected Non-Planar Full Flow Problem.

□

Dyer and Frieze have shown [6], that Planar 1-3 SAT is NP-Complete.

PLANAR 1-3 SAT

Instance:

Two sets of boolean variables $U = \{u_1, u_2, \ldots, u_n\}$ and $\overline{U} = \{\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_n\}$ and a collection of clauses $C = \{c_1, c_2, \ldots, c_m\}$ of clauses over U and $\overline{U}$ such that

1. $\forall\, c \in C, |c| = 3$,
2. *G(B)* is planar, where $B = c_1 \wedge c_2 \wedge \ldots \wedge c_m$ and $G(B)$ is defined as below.

Question:

Is there a truth assignment for $U \cup \overline{U}$ such that each clause in C has exactly one true literal?

Where $G(B)$ is the graph (V,E):

$V = \{c_i \mid c_i \in C\} \cup \{u_i \mid u_i \in U\} \cup \{\overline{u}_i \mid \overline{u}_i \in \overline{U}\}$

$E = \{(c_i, u_j) \mid u_j \in c_i\} \cup \{(c_i, \overline{u}_j) \mid \overline{u}_j \in c_i\} \cup \{(u_i, \overline{u}_i) \mid 1 \leq i \leq n\}$

Figure 5.1 shows $G(B)$ for $B = (a \vee \overline{b} \vee c) \wedge (b \vee \overline{b} \vee d) \wedge (\overline{a} \vee b \vee d)$.

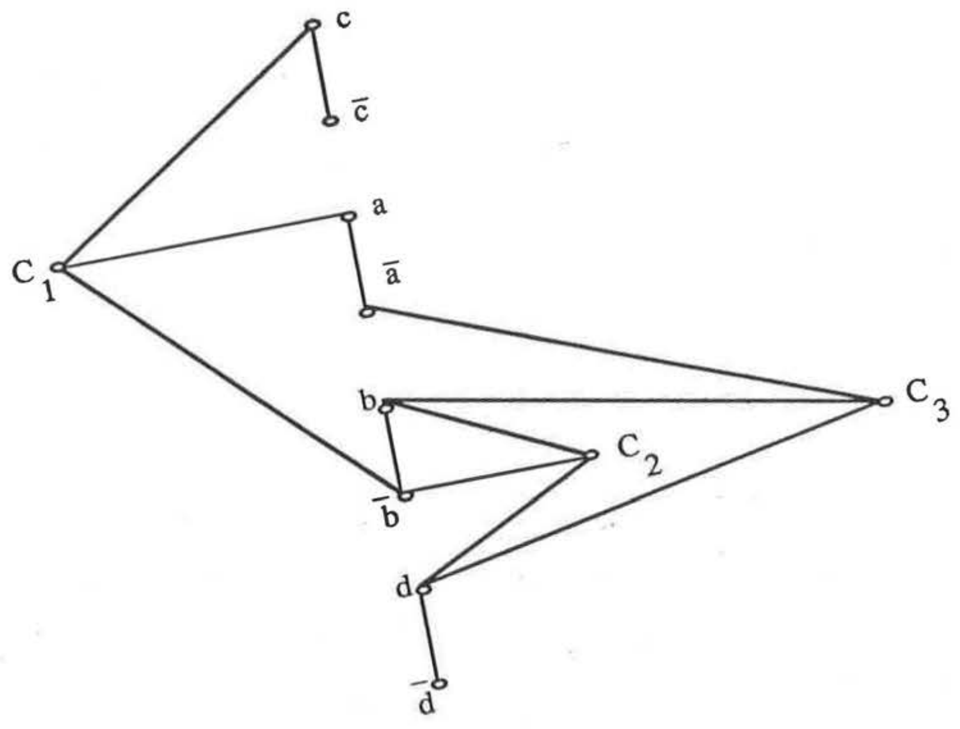


Figure 5.1: $G(B)$

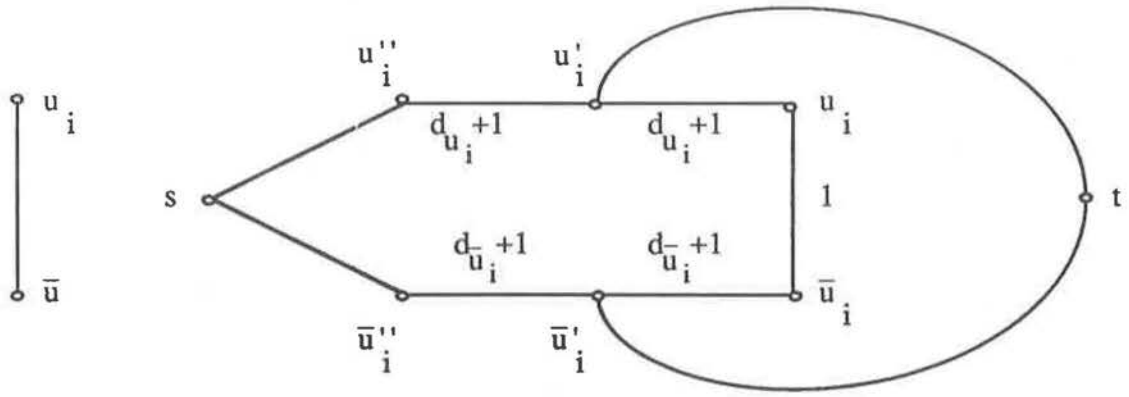


Figure 5.2: Component

Lemma 19 *PLANAR 1-3 SAT reduces to the Undirected Non-Planar Full Flow Problem.*

Proof: Given an instance of PLANAR 1-3 SAT, convert $G(B)$ to a network $N(B)$ as follows. Let d_{u_i} represent the number of times variable u_i occurs in B. Replace each occurrence of $(u_i,\overline{u_i})$ by the component in figure 5.2. Where

weight$(u_i,\overline{u_i}) := 1$

weight$(u_i',u_i) :=$ weight$(u_i'',u_i') := d_{u_i} + 1$

weight$(\overline{u_i}',\overline{u_i}) :=$ weight$(\overline{u_i}'',\overline{u_i}') := d_{\overline{u_i}} + 1$

For each edge of the form (c_i,u_j), set the weight as

weight$(c_i,u_j) :=$ weight$(c_i,\overline{u_j}) := 1$

For each clause c_i, introduce a vertex c_i' and edge (c_i,c_i') with weight 1. Introduce two new vertices s and t. Edge (s,c_i') is defined to be a member of $E(G(B))$, with an associated weight of 1, for all c_i'.

Join s to u_i'' with associated weight d_{u_i}, for all u_i''. Join s to $\overline{u_i}''$ with associated weight $d_{\overline{u_i}}$, for all $\overline{u_i}''$.

Join u_i' to t with capacity $2d_{u_i}+2$, for all u_i'. Join $\overline{u_i}'$ to t with capacity $2d_{\overline{u_i}}+2$, for all $\overline{u_i}'$.

Figure 5.3 shows *N(B)* for the previous example. Note that *N(B)* is "almost" planar in the sense that $N(B) - \{s,t\}$ is planar. It is now necessary to show that there exists a 1-3 truth assignment for B iff $N(B)$ has a full flow. First note the following facts about $N(B)$:

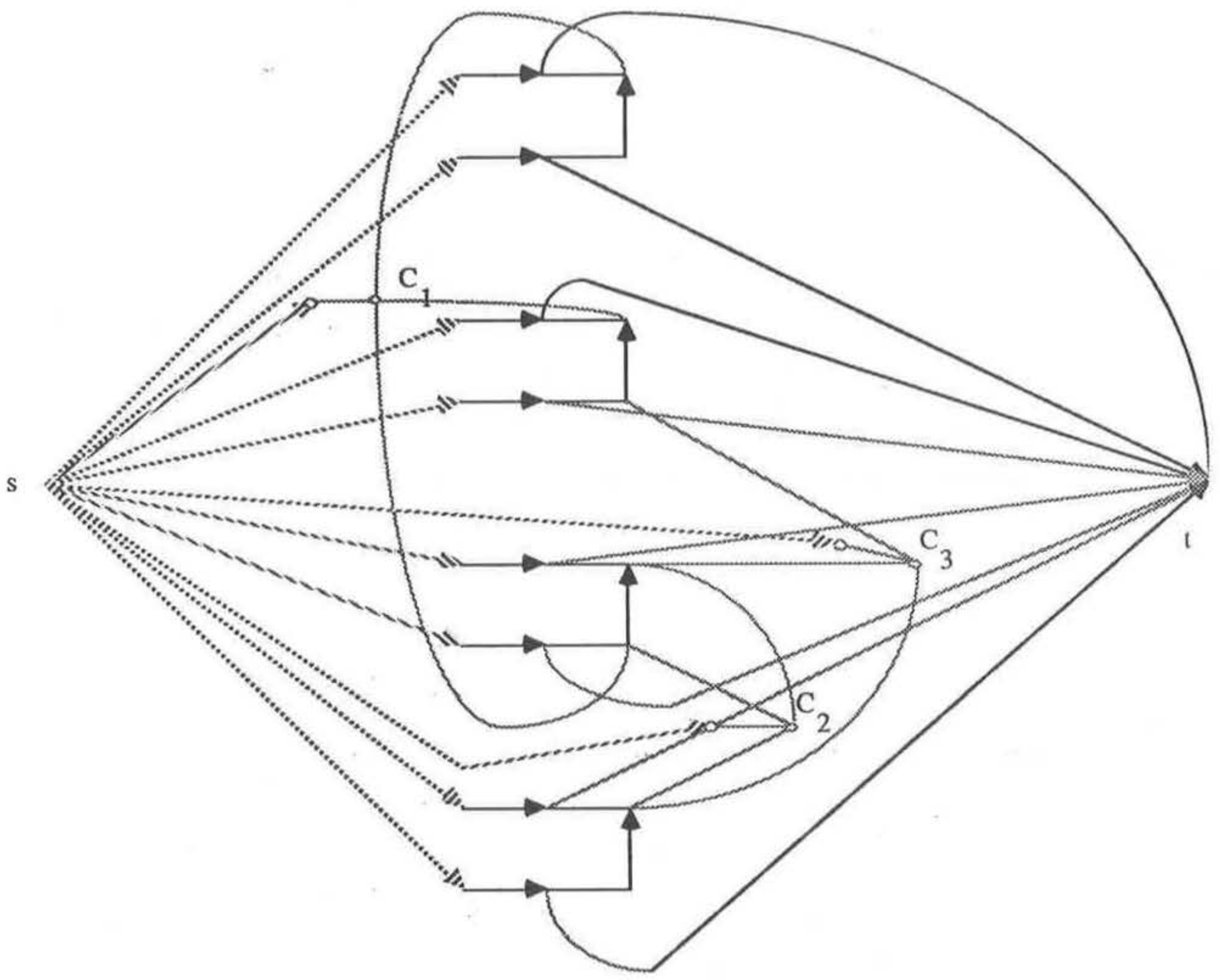
1. Since s is the source, (s,c_i') is an arc $< s,c_i'>$, and hence, so is $< c_i', c >$.

2. Since each c_i has degree 4 and since $< c_i', c_i >$ is an incoming arc, c_i is balanced iff exactly one of its other three edges is incoming and the other two are outgoing.

3. Since s is the source, (s,u_i'') is an arc $<s,u_i''>$. See figure 5.4.

4. Consider a vertex u_i. If $(\overline{u_i},u_i)$ is directed from $\overline{u_i}$ to u_i then the only balancing scheme is: $<c_k,u_i>$ $\forall$ c_k and $<u_i,u_i'>$. Then weight($<u_i',t>$) must be exactly $2d_{u_i}$ + 2 implying that arc $<\overline{u_i},c_j >$ $\forall$ c_j is directed from $\overline{u_i}$ to c_j and arc $<\overline{u_i}',\overline{u_i}>$ is directed from $\overline{u_i}'$ to $\overline{u_i}$. This implies that the weight($<\overline{u_i},t>$) is 0.

($\Rightarrow$) Now given a 1-3 SAT truth assignment for B, the construction of the corresponding full flow in $N(B)$ is clear. For each variable pair $(u_i,\overline{u_i})$, if u_i is true then direct the edge from u_i to $\overline{u_i}$: $<u_i,\overline{u_i}>$, forcing the balancing scheme of 4. Consider a vertex c_i. Each true literal in its corresponding clause creates a net influx of 1 and each false literal, a net drain of 1. If clause c_i contains exactly one true literal, then vertex c_i is balanced. Thus all vertices of $N(B)$ are balanced. It remains to show that no directed cycle occurs in the directed version of $N(B)$.



a=b=c=d=false

Figure 5.3: *N(B)*

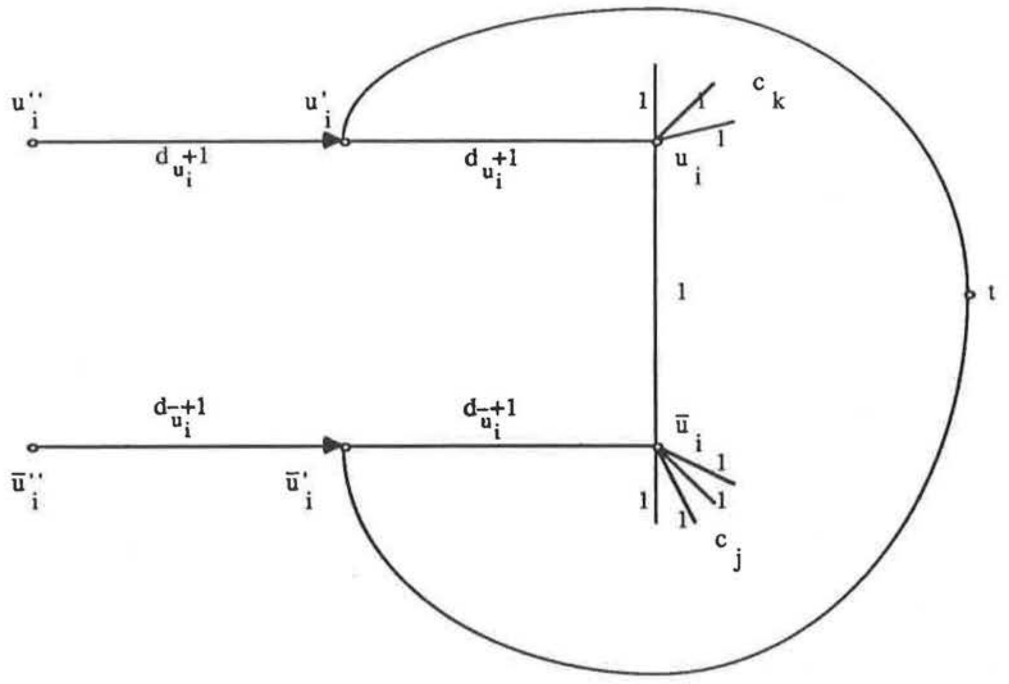


Figure 5.4: Variable Pair

By construction, a directed cycle $\overline{C}$ must involve some pair of variables, say $<\overline{u_i},u_i>$, which by fact 4 implies there does not exist an arc from u_i to any clause c_j and hence $<u_i,u_i'>$ and $<u_i',t>$ are part of the cycle, which is a contradiction since t has no outgoing arcs.

($\Leftarrow$) Given a full flow for $N(B)$, fact 4 may be used to show that exactly one of u_i and $\overline{u_i}$ is true in the associated truth assignment. Fact 2 shows each clause is satisfiable by exactly one literal. Hence the associated assignment of the full flow is a valid truth assignment for B.

□

5.3 The Undirected Planar Cyclic Full Flow Problem

It is also possible to investigate the role of the acyclicity constraint in the flow existence problem. Define the **Undirected Planar Cyclic Full Flow** problem as, given an undirected capacitated (planar) network N with specified source s and t, determine an orientation, g, of the edges such that the resulting graph $g(N)$ is a directed capacitated graph for which there exists a planar full flow. (Note that $g(N)$ is not restricted to be acyclic).

At first it may seem surprising that this seemingly minor change to the definition of the Planar Full Flow problem, results in an NP-Complete problem; consider the polynomial algorithm for the planar full flow problem of the previous chapter, for example. But in fact, some other problems involving directed graphs exhibit a similar behaviour. The Hamilton Path Problem [3] is NP-Complete for directed planar graphs but may be solved in polynomial time if the graph is also restricted to be acyclic ([9]).

The proof that the Undirected Planar Cyclic Flow problem is NP-Complete is obtained by making the graph $N(B)$ in the previous reduction, planar by substituting crossover components. For each edge crossing in $N(B)$, it is possible to substitute a planar component that enforces the same flow. Call the resulting graph $N'(B)$. Note however, that the "full flow" in $N'(B)$ in general contains directed cycles.

Theorem 8 *The Undirected Planar Cyclic Full Flow problem is NP-Complete.*

Proof:

a) The problem is clearly in NP as a proposed solution may be checked in polynomial time. Checking the capacity and balancing conditions and planarity may be done in linear time.

[3] Find a (directed) path through each vertex exactly once.

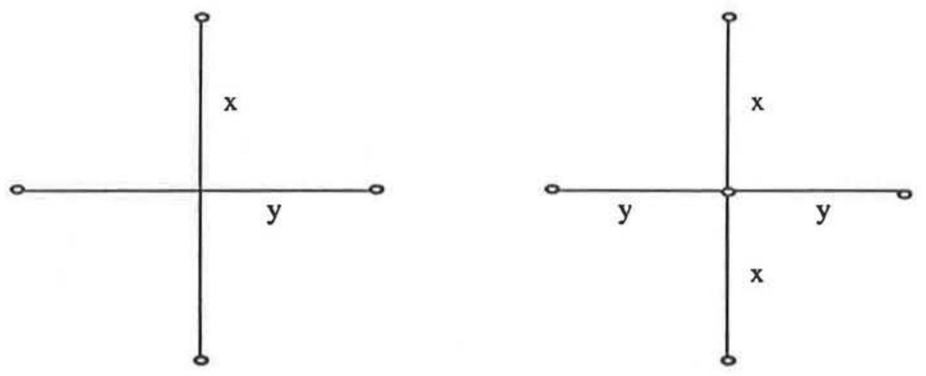


Figure 5.5: Edge Crossing: Same Weights

b) The following lemma shows PLANAR 1-3 SAT reduces to the Undirected Planar Cyclic Full Flow problem. □

Lemma 20 *PLANAR 1-3 SAT reduces to Undirected Planar Cyclic Full Flow.*

Proof:

Given an instance of PLANAR 1-3 SAT, construct $N(B)$ as described in Lemma 19. Note that in $N(B)$, the weights of all edges are specified except for those involving t. Create a new graph $N'(B)$ as follows:

For each edge crossing in $N(B)$:

1) If the two weights are known and different then introduce a new vertex v at the crossing and split the two edges into four, maintaining the weights (see figure 5.5).

2) If the edge weights are known and the same, substitute the component in figure 5.6.

3) Crossings involving edges incident upon t can be handled as above since the weights on such edges are in fact effectively known *a priori*, namely,

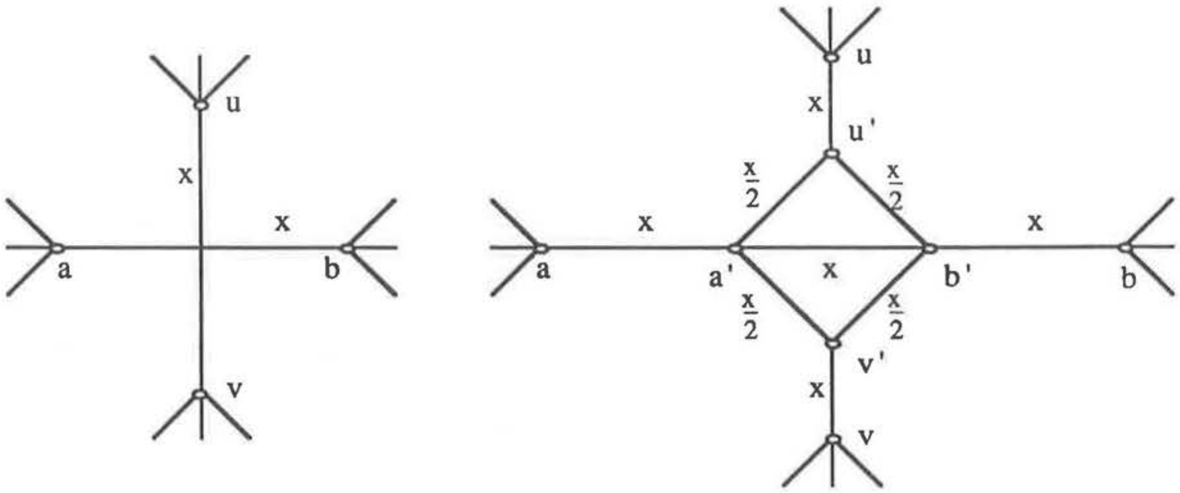


Figure 5.6: Substitute Component

weight(<$u_i{}'$,t>) = 0 or $2d_{u_i} + 2$. However, this substitution may introduce non-extreme edges of unspecified weight.

The two substitute components are designed to assure that any flow through $N(B)$ can be maintained in a flow in $N(B)'$. (Note that the weights created in the second substitute component may not be integral, but all weights in $N(B)'$ could be multiplied by 2 to preserve integrality.)

The number of vertices in $N'(B)$ is increased by at most a factor of $4d$, where d is the number of edge crossings. The crossing number of a complete graph K_n is known to be no more than $\frac{1}{64}(n)(n-1)(n-2)(n-3)$. Thus the crossing number of $N(B)$ is $O(n^4)$ and hence only a polynomial number of extra vertices need be added to obtain $N'(B)$. Thus, the reduction is polynomial.

It is clear that if $N(B)$ has a non-planar full flow then $N'(B)$ has a planar cyclic full flow. To show the converse it is sufficient to observe that in type 2 crossings, if <a,a'> (respectively <a',a>) $\in$ Flow, then <a',b'> (respectively <b',a'>) $\in$ Flow and <b',b> (respectively < b,b'>) $\in$ Flow. Hence, <a,b> (respectively <b,a>) is part of a flow in $N(B)$.

And if $<u,u'>$ (respectively $<u',u>$) $\in$ Flow, then $<u',a'>$ and $<u',b'>$ (respectively $<a',u>$ and $<b',u>$) $\in$ Flow, implying that $<a',v'>$ and $<b',v'>$ (respectively $<v',a>$ and $<v',b'>$) $\in$ Flow, in turn implying that $<v',v>$ (respectively $<v,v'>$) $\in$ Flow, and hence $<u,v>$ (respectively $<v,u>$) is part of a valid flow in $N(B)$. Thus, $N'(B)$ has a planar cyclic full flow iff there exists a truth assignment for B in which each clause contains exactly one true literal.

□

Chapter 6

New Directions and Related Results

There are two natural directions to extend the notions of bar-representability. Curves other than bars may be considered, or multiple lines-of-sight may be added.

As suggested in chapter 1, the restriction to bars for curves includes horizontally convex objects. If this convexity is relaxed even slightly, to circles or to "U"-shaped curves for example, then any planar graph may be laid out.

In fact, only the articulation vertices on an interior face of a plane embedding need to be circles: their interior components may be sheltered from the exterior world in the interior.

Permitting arbitrary non-intersecting curves does not result in a larger class of graphs being representable.

Define a graph as **O-representable** if it is the visibility graph of a layout of non-intersecting bars and circles using an ϵ-visibility model and a single horizontal line-of-sight. A circle in such a layout is said to **circumscribe** the bars and circles in its interior. A graph G will be called **circumscribable in** v if there exists a layout of G with v corresponding to the outermost circle.

Lemma 21 *All planar biconnected graphs are circumscribable with an arbitrary vertex as the outer circle.*

Proof:

In chapter 3 it was shown that any biconnected planar graph G is bar-representable. The construction produces a bar layout, based on an $s-t$ numbering of G, in which the left-most and right-most bars ($\overline{S}$ and $\overline{T}$) are of length one and all other bars lie within the rectangle defined by s and t. Since s and t are adjacent vertices of G, transforming $\overline{S}$ into a circle containing all other bars yields an O-layout of G. Since any pair of adjacent vertices may be chosen as s and t in the $s-t$-numbering, an arbitrary vertex may be chosen as the circumscribing vertex.

□

Theorem 9 *G is O-representable iff G is planar.*

Proof:

Let $\mathcal{G}$ be a plane embedding of G in which all components with articulation vertices on the exterior face are embedded in the exterior face.

All bar-representable graphs are O-representable by definition; therefore assume $\mathcal{G}$ contains articulation vertices $c_0, c_1, \ldots, c_n$ not on the exterior face. Such articulation vertices will be called **interior**.

Define the **level** of an interior articulation vertex c as the minimum number of interior articulation vertices on a path from c to a vertex on the exterior face of $\mathcal{G}$.

For an arbitrary interior articulation vertex c, $\mathcal{G}$ may be divided into subgraphs: maximal connected components in which all vertices are connected by a path not containing c. Each subgraph is assumed to also contain c. There are two types of such components: a single **exterior** component containing vertices on the exterior face and a set of **interior** components.

For all interior, level 0 articulation vertices, the exterior component $\mathcal{E}$ is a unique subgraph of $\mathcal{G}$. Since $\mathcal{E}$ contains no interior articulation vertices, it admits a bar layout. For each bar $\bar{c}$, corresponding to a level 0 articulation vertex c, substitute a circle c^o with diameter of $|\bar{c}|$. The interior components of c will be laid out inside c^o.

Let the interior components of c be $I_1, I_2, \ldots, I_k$. Layout the I_j in k disjoint horizontal bands inside c^o, as follows:

- If I is biconnected, then it may be laid out with bars circumscribed by c, by the previous lemma.
- Otherwise, I contains at least one level 1 articulation vertex. If the interior components of the level 1 articulation vertices are ignored, then I is biconnected and can be laid out with bars circumscribed by c. Substituting circles for the level 1 articulation vertices and iteratively embedding their interior components inside the circles, completes the procedure.

The number of iterations is bounded by the maximum level of the articulation vertices.

□

6.1 Multiple Lines-of-Sight

Given two lines-of-sight, horizontal and vertical, and *solid* rectangular "boxes" as objects, define a graph G as **2-box-representable** if there exists a layout of boxes whose visibility graph is G. It can be shown that all planar graphs have such a layout. Note that boxes may not be laid out in the interior of other boxes.

Theorem 10 *All planar graphs are 2-box-representable.*

Proof:

By the previous theorem, all planar graphs have a (1-line-of-sight) O-representation. The crucial step in the previous proof was to circumscribe a biconnected component by its articulation vertex. Given an O-representation, a 2-box-representation can be created by:

1. Displacing all bars and circles so that no two are in the same vertical section (with the exception of objects within their circumscribing circle).
2. Iteratively laying out components within a circle *above* their circumscribing circle, horizontally independent of other components, using the vertical line-of-sight to express the relationship.

Step 2 is well-defined since each component is either biconnected (and hence bar-representable in the vertical line-of-sight), or contains an articulation vertex circle.

Substituting solid squares for circles and rectangles for bars is easily accomplished and completes the procedure.

□

Johnson [30] has argued that given k lines-of-sight and arbitrary (non-intersecting) curves, the thickness [1] of the associated graph is $\leq k$.

Let $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ be a set of non-intersecting curves in the plane. Let $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$, $k \geq 1$ be a set of non-parallel directions: the **lines-of-sight**. Two curves C_i and C_j **see each other** if there exists a non-degenerate rectangle oriented parallel to one of the k lines-of-sight whose interior contains portions of C_i and C_j and no other curves of $\mathcal{C}$. The line-of-sight graph $G_k(\mathcal{C})$ has $\mathcal{C}$ as a vertex set and an edge between vertices if the corresponding curves can see each other.

$G_k(\mathcal{C})$ has thickness $\leq k$ since its edge set, $E_k(\mathcal{C})$, can be partitioned into k sets of "parallel" (i.e. crossing-free) layers. This prompts the following question (generalized from [30]).

For any graph G of thickness k, do there exist $\mathcal{C}$ and $\mathcal{S}$ such that $G_k(\mathcal{C}) = G$?

For example, figure 6.1 shows that K_8 has a 2-line-of-sight representation with horizontal and vertical lines-of-sight. (K_9 is the smallest graph with thickness 3).

The following theorem shows that k thickness is at least achievable with k lines-of-sight.

Theorem 11 *For all $k \geq 1$, there exists $\mathcal{C}$ and $\mathcal{S}$ such that thickness($G_k(\mathcal{C})$) $= k$.*

Proof:

Consider the line-of-sight graph G of the following set of curves. G contains the bipartite graph $K_{2k,4k^2}$ as a subgraph and hence has thickness $\geq k$. ($K_{2r+1,4r^2-2r+1}$ has thickness $r+1$ [12]).

[1] Graph G has thickness k if there exists a decomposition of G into k planar subgraphs and k is minimum.

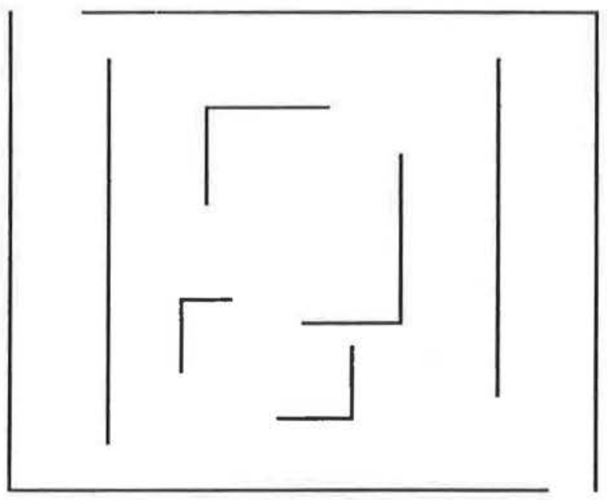


Figure 6.1: K_8: 2-Line-of-Sight Representation

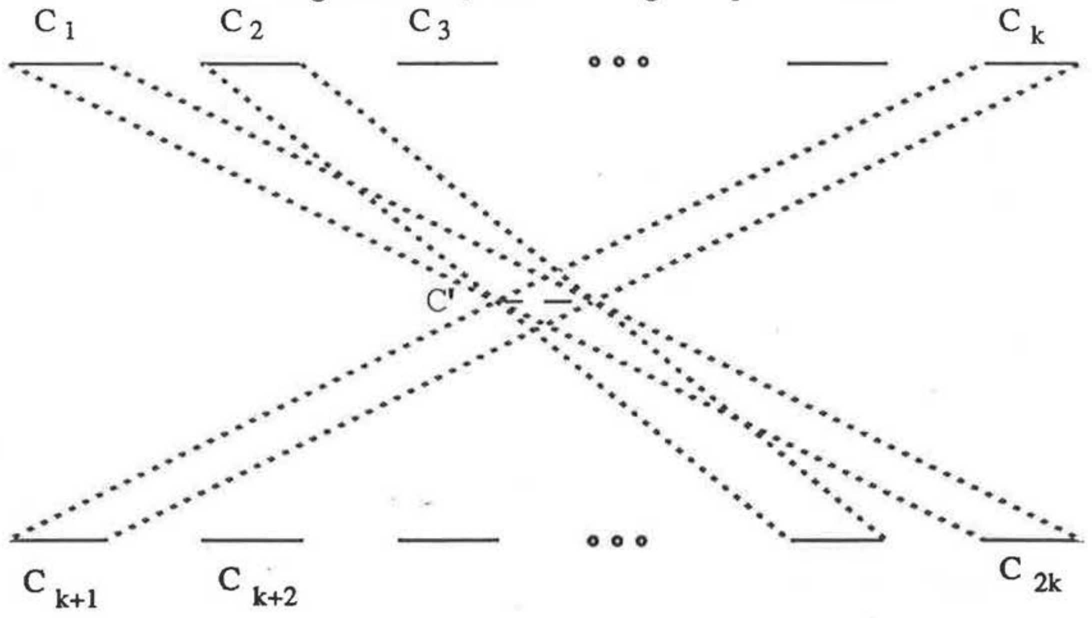


Figure 6.2: Layout of a Graph of Thickness k

In figure 6.2, C' is a collection of $(2k)^2$ segments $(C_{2k+1}, \ldots, C_{2k+(2k)^2})$ in the intersection of the k lines-of-sight, which are defined as the lines (C_i, C_{2k-i+1}), for $i = 1, 2, \ldots, k$.

□

6.2 Optimization Problems

It is not difficult to formulate optimization versions of layout problems. For weighted unordered graphs, determining the minimum width layout is at least as hard as the problem of partitioning a set of numbers into two subsets of equal weight. This Partition problem is NP-Complete ([9]) if the weights are not bounded by the size of the input.

Rosenstiehl and Tarjan conjecture [20] that minimizing the layout area of an undirected unweighted graph is NP-Hard. Note that their visibility relation is weaker however, and their layout includes the dual graph perpendicularly interlocked.

One final open problem is to interpret the weight of an edge not as the amount of visibility between a pair of bars, but as their (horizontal) distance apart.

Chapter 7

Conclusions and Contributions

The focus of this dissertation has been to characterize various types of bar-representable graphs and to determine the computational complexity of their recognition and layout. The main contributions have been the following:

- linear time algorithms for recognizing and laying out measured polarized bar-representable, (unmeasured) polarized bar-representable, and (unmeasured unpolarized) bar-representable graphs.
- a polynomial time layout algorithm for measured bar-representable graphs.
- to establish connections between these problems and natural constrained network flow problems.
- NP-Hardness results for the Undirected Non-Planar Full Flow problem, and the Undirected Cyclic Planar Full Flow problems.

For the directed cases, both weighted and unweighted, of the bar-representable problem, the characterization amounts to checking whether the vertices of insufficient indegree (or inweight) and insufficient outdegree (or outweight) are (planarly) separable into two groups on the exterior face of some embedding. This test can be performed in linear time by testing the planarity of a supergraph of the given graph.

For an undirected, unweighted graph to be bar-representable, it has been shown that there must be a planar embedding of the graph in which all the articulation vertices appear on the exterior face. This condition may also be verified in linear time by examining a supergraph of the given graph.

The weighted undirected case appears to be computationally more involved. This case was reformulated as a network flow problem requiring an orientation of the edges satisfying certain balancing conditions. In the ordered case, the flow was determined by exploiting both local and global properties based on the topology of the ordering of the graph. The resulting full flow provided an orientation of the edges for which a bar layout could be constructed via the previous results. For the unordered case, the triconnected components of the graph were the critical building blocks and a collapsing process reduced the graph to a sufficiently small size to be solved with the aid of the ordered version.

The Undirected Non-Planar Full Flow problem and the Cyclic Planar Full Flow problems were shown to be NP-Hard by reduction from Planar 1-3 SAT. These two results indicate the important role that the planarity and acyclicity requirements play in the formulation of the Full Flow problem.

There are several areas within the field of visibility that may benefit from these results obtained in the thesis and from the techniques used to solve them. The formulation of the problem(s) in terms of network flows has been particularly fruitful and may be applicable in other visibility domains.

References

[1] A. Aho, J. Hopcroft, J. Ullman, The Design and Analysis of Algorithms, Addison-Wesley Publishing Co., 1974.

[2] M. Behzad, G. Chartrand, L. Lesniak-Foster, Graphs and Digraphs, Wadsworth Inc., 1979.

[3] C. Berge, Graphs and Hypergraphs, North Holland Pub., 1973.

[4] V. Chvátal, A combinatorial theorem in plane geometry, *J. Combin. Theory Ser. B*, **18** (1975), 39-41.

[5] P. Duchet, Y. Hamidoune, M. Las Vergnas, and H. Meyniel, Representing a planar graph by vertical lines joining different levels, *Discrete Math.* **46** (1983), 319-321.

[6] M. Dyer, A. Frieze, Planar 3DM is NP-Complete, *Journal of Algorithms*, **7** (1986), 174-184.

[7] El Gindy, Hierarchical decomposition of polygons with applications, Ph.D. thesis, McGill University (1985).

[8] S. Even, R. Tarjan, Computing an s-t Numbering, *Theoretical Computing Science.* 2 (1976) 339-344.

[9] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Co., 1979.

[10] M. Garey, D. Johnson, H. So, An Application of Graph Coloring to Printed Circuit Testing, 1975 FOCS.

[11] S. Ghosh, On Recognizing and characterizing visibility graphs of simple polygons, The Johns Hopkins University/EECS Technical Report 86/14, 1986.

[12] F. Harary, Graph Theory, Holt, Reinhart and Winston, 1967.

[13] J. Hopcroft, R. Tarjan, Dividing a Graph into Triconnected Components, *SIAM J. Computing*, Vol. 2, No. 3, (1973) 135-158.

[14] D.G. Kirkpatrick, Establishing Order in Planar Subdivisions, *J. of Discrete and Computational Geometry* 3 (1988), 267-280.

[15] T. Lozano-Perez, M. Wesley, An algorithm for planning collision-free paths among polyhedral obstacles, *Communications of the ACM* 22 (1979), 560-570.

[16] F. Luccio, S. Mazzone, C. K. Wong, A Note on Visibility Graphs, *Discrete Mathematics*, **64**, (1987), 209-219.

[17] J. O'Rourke, Art Gallery Theorems and Algorithms, Oxford University Press (1987).

[18] F. Preparata, M.I. Shamos, Computational Geometry: An Introduction, Springer-Verlag (1985).

[19] I. Rival, J. Urrutia, Representing Orders on the Plane by Translating Convex Figures, *Order* 4 (1988), 319-339.

[20] P. Rosenstiehl, R. Tarjan, Rectilinear Planar Layouts and Bipolar Orientations of Planar Graphs, *Discrete and Computational Geometry*, 1 (1986), 343-353.

[21] M. Schlag, F. Luccio, P. Maestrini, D.T. Lee, C.K. Wong, A Visibility Problem in VLSI Layout Compaction, *Advances in Computing Research*, Vol. 2, 259-282.

[22] X. Shen, H. Edelsbrunner, A Tight Lower Bound on the Size of Visibility Graphs, *Information Processing Letters* 26(1987/88) 61-64.

[23] R. Tamassia, I. Tollis, A Unified Approach to Visibility Representations of Planar Graphs, *Discrete and Computational Geometry*, 1 (1986), 321-341.

[24] C. Thomassen, Interval Representations of Planar Graphs, *Journal of Combinatorial Theory*, series B **40** (1986), 9-20.

[25] G. Toussaint, Shortest path solves edge-to-edge visibility in a polygon, *Pattern Recognition Letters*, **4** (1986), 165-170.

[26] E. Welzl, Constructing the Visibility Graph for n-Line Segments in $O(n^2)$ Time, *Information Processing Letters* 20 (1985), 167-171.

[27] H. Whitney, Congruent Graphs and the Connectivity of Graphs, *American J. of Mathematics*, LIV (1932), 150-168.

[28] S. K. Wismath, Characterizing Bar Line-of-Sight Graphs, *Proc. 1st ACM Symposium on Computational Geometry*, Baltimore (1985), 147-152.

[29] S. K. Wismath, Issues in Embeddings, unpublished Thesis Proposal, U.B.C. (1983).

[30] Algorithmic Aspects of Combinatorics, *Annals of Discrete Math.* 2 (1978), North Holland Publishing Co., 243.