Via Minimization for IC and PCB Layouts

by N.J. Naclerio
Advisor: K. Nakajima

Ph.D. 87-2
Formerly TR 87-191

Via Minimization for IC and PCB Layouts

by

Nicholas Joseph Naclerio

Dissertation submitted to the Faculty of the Graduate School
of The University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1987

Acknowledgement

I would like to express my sincerest gratitude to the members of my committee, especially my advisor Professor Kazuo Nakajima for his encouragement and dedication; Professor Joseph JáJá who served as my original research advisor; and Professor H.C. Lin who arranged most of my financial support as well as several valuable meetings with researchers at Westinghouse. I am also very grateful for the help I received from Dr. Sumio Masuda without which I could never have accomplished so much in the last year. He also deserves credit for the basic approach used in Algorithm 1. I would like to thank Professor Yoji Kajitani who visited us from the Toykyo Institute of Technology and provided valuable comments on our work as well as suggesting the curve coloring problem in Chapter III. I am also appreciative of the valuable discussions I had with researchers at AT&T Bell Labs, Westinghouse, and the National Bureau of Standards. I would especially like to thank Dr Ken Galloway and Loren Linholm of the National Bureau of Standards for their guidance and support throughout most of my tenure at Maryland. Of course, I must thank all of the organizations which provided me with financial support including the National Bureau of Standards, Westinghouse, the Systems Research Center of the University of Maryland, the National Science Foundation through grants MIP-84-51510 and CDR-85-00108 and AT&T. Lastly, I would like to thank all of my friends at the Electrical Engineering Department, the Systems Research Center, and the National Bureau of Standards, who made my experience there enjoyable and my beautiful wife Jackie, whose patience, support, and encouragement helped me accomplish all that I have.

Table of Contents

Page

List of Figures vi

Chapter I. Introduction 1

1. Motivation 1
2. Outline 2
3. Routing Terminology 3
4. Via Minimization Problems 8
5. Previous Work 12
6. Graph Theoretical Terminology and Notation 13

Chapter II. Complexity 16

1. Overview 16
2. The General 2-CVM Problem 17
3. The Restricted 2-CVM$_6$ Problem 23
4. Summary 29

Chapter III. Algorithms 30

1. Overview 30
2. The 2-CVM$_3$ Problem for Topological Layouts 30
    2.1 The basic 2-CVM$_3$ algorithm 30
    2.2 Complexity 42
3. The 2-CVM$_3$ Problem for Geometric Layouts 44
    3.1 Application of the basic algorithm 44
    3.2 Improved 2-CVM$_3$ algorithm 45
    3.3 Essential vias 57

4. The 2-CVM Problem 59

5. The Curve Coloring Problem 65

6. Summary 67

Chapter IV. Implementation 69

1. Overview 69

2. Minimizer 70

2.1 The program 70

2.2 Results 74

3. Efficiency Considerations 80

4. Practical Considerations 81

4.1 Critical nets 81

4.2 Crux zones 82

4.3 Unequal layers 82

4.4 Fixed layer terminals 84

5. Heuristics 86

6. Summary 88

Chapter V. Conclusion 90

1. Summary 90

2. Recent Results 92

3. Future Work 92

Bibliography 94

List of Figures

Number Page

1.1 Example layout 4

1.2 Routing segments and subsegments 5

1.3 Functionally equivalent sets of routing segments 7

1.4 An optimum 2-UVM solution 9

1.5 Input layout and two "optimum" solutions 11

2.1 Sublayout H 18

2.2 Type 1 independent sublayout 19

2.3 Type 2 independent sublayouts 20

2.4 Straight line embedding $\overline{G_a}$ and the layout $\mathcal{L}_{G_a}$ 22

2.5 Sublayout H^* 25

2.6 Planar grid embedding $\hat{G}_a$ and the layout $\mathcal{L}^*_{G_a}$ 26

3.1 Example layout $\mathcal{L}_2$ and the derived graph $G_{\mathcal{L}_2}$ 32

3.2 Equivalent via locations 33

3.3 Inserting a via in $\mathcal{L}$ 35

3.4 An MOFC for $\overline{G_{\mathcal{L}_2}}$ 36

3.5 An MONP for $G^d_{\mathcal{L}_2}$ 37

3.6 The graph M for $G^d_{\mathcal{L}_2}$ 38

3.7 A maximum weight matching for the graph M and the resulting layout $\mathcal{L}'_2$ 40

3.8 Clusters and via candidate zones in layout $\mathcal{L}_3$ 46

3.9 The graph $G'_{\mathcal{L}_3}$ 48

3.10 An MOCNC for the graph $G'_{\mathcal{L}_3}$ and the layout $\mathcal{L}'_3$ 53

3.11 Equivalent via sites 56

3.12 Essential vias 58

3.13 Paths in $G'^{d}_{\mathcal{L}}$ 62

3.14 Curve coloring problem and optimum solution 66

4.1 Layout example 1 74

4.2 MIN1 run time for channel routing examples 75

4.3 Channel routing example 77

4.4 MIN2 run time for channel routing examples 78

4.5 Channel routing examples with via candidate zones 79

4.6 Comparison of MIN2 to a heuristic algorithm 80

4.7 Example with crux zones 83

4.8 Example with fixed layer terminals 85

4.9 Histogram of path lengths 87

Abstract

Title of Dissertation: Via Minimization for IC and PCB Layouts

Nicholas Joseph Naclerio, Doctor of Philosophy, 1987

Dissertation directed by: Kazuo Nakajima, Associate Professor, Electrical Engineering Department

In the design of integrated circuits (ICs), it is important to minimize the number of vias between conductors on different layers since excess vias lead to decreased yield and degraded circuit performance. Similarly, in the design of printed circuit boards (PCBs), it is important to minimize the number of contact holes used to connect copper strips on opposite sides of the board. Excess contact holes increase manufacturing cost and decrease the board's reliability.

Given a particular design of an IC (or PCB), the Constrained Via Minimization Problem is to find a layer assignment that requires the fewest possible vias (or contact holes). It is shown that this problem is NP-hard for two-layer layouts and remains so when the layout is restricted to be grid-based, vias are restricted to lie at previously existing junctions, and the maximum number of wires which are joined at any particular junction does not exceed six.

A new graph-theoretic formulation of the two-layer problem is then presented along with an algorithm which yields optimum results when the maximum junction degree is limited to three. The worst-case time complexity of the algorithm is $O(n^3)$ where n is the number of routing segments in the given layout. Unlike previous algorithms, this algorithm does not require the layout to be grid based and places no constraints on the location of vias or the number of wires that may be joined at a single junction. If there are junctions with degree exceeding three, then our solution may be slightly less then optimum. If vias are limited to a subset of all possible via sites such as those allowable by a particular set of geometric design rules, then a further speedup is possible.

This algorithm has been fully implemented. When tested on examples from the literature, it was found to be faster than the existing heuristic algorithms. A new heuristic is also suggested that is based on properties of the optimum solutions which were generated by this algorithm.

Chapter I

Introduction

1. Motivation

During integrated circuit fabrication, interconnections between circuit modules are formed by patterning conductive paths on one of two or more layers. Because the layers are separated by a thin insulator, it is necessary to etch holes, called *vias*, through the insulator in order to electrically connect paths on different layers. Minimizing the number of these vias increases the performance of the electrical circuit and the yield of the manufacturing process and decreases the amount of area required for interconnections. In the design of printed circuit boards, copper paths are patterned on two sides of an insulating board. In order to electrically connect paths on opposite sides of the board, contact holes must be drilled through the insulator and plated with solder. Minimizing the number of contact holes, which we will henceforth refer to as vias, decreases manufacturing cost and increases the reliability of the board.

2. Outline

In the remainder of this chapter, we will define the terminology associated with via minimization and give some formal problem definitions. We will then summarize the previous work done on via minimization and review the graph theoretical terminology that will be used throughout the remainder of this work.

Chapter 2 deals with the complexity of the so-called Constrained Via Minimization problem. Given an already routed circuit, the problem is to find a minimum cardinality set of vias for which a valid layer assignment exists using only two layers. The associated decision problem is first proven to be NP-complete. It is then shown that the problem remains NP-complete even if one or more of the following restrictions are imposed.

1) The input layout must conform to a rectangular grid.
2) Vias are restricted to lie at the endpoints of routing segments in the input layout.
3) The maximum junction degree is limited to six or more.

Chapter 3 discusses polynomial time algorithms for the Constrained Via Minimization problem. The basic algorithm presented gives optimum results when there are no junctions in the input layout having degree greater than three. Unlike previous algorithms, the algorithm does not require the layout to be grid based and places no constraints on the location of vias. The algorithm produces an approximate solution when there are junctions having degree greater than three. Modifications are described which greatly reduce the average time complexity for most problems.

Chapter 4 discusses how the algorithms presented in Chapter 3 have been implemented and gives the results of applying them to real problems. Run times of different optimum and heuristic algorithms are compared. Methods of increasing the average efficiency of the implemented algorithms are also described. Based on empirical data, a heuristic is suggested and demonstrated which further reduces the necessary computation time.

Chapter 5 summarizes the contributions of this work and mentions some very recent results by other researchers. Future directions for research in this area are also mentioned.

3. Routing Terminology

Often the physical design of printed circuit boards (PCB's) or integrated circuits (IC's) is broken down into two steps called *placement* and *routing.* In the first step, the location of circuit elements called *modules* are determined according to certain criteria. In the second step, interconnections are made between the modules. A module is defined as a region of the horizontal plane bounded by a closed curve. Electrical connections to the modules are made at interface points called *terminals* located on their boundary. A group of terminals which need to be electrically connected are called a *net.* A set of nets is called a *netlist.* In the routing step, a set of wires must be specified which interconnect the groups of terminals defined by the netlist. These wires cannot pass through the modules and must lie within a region called the *routing area.* The wires are implemented as a number of straight line segments which are called *routing segments* and which are

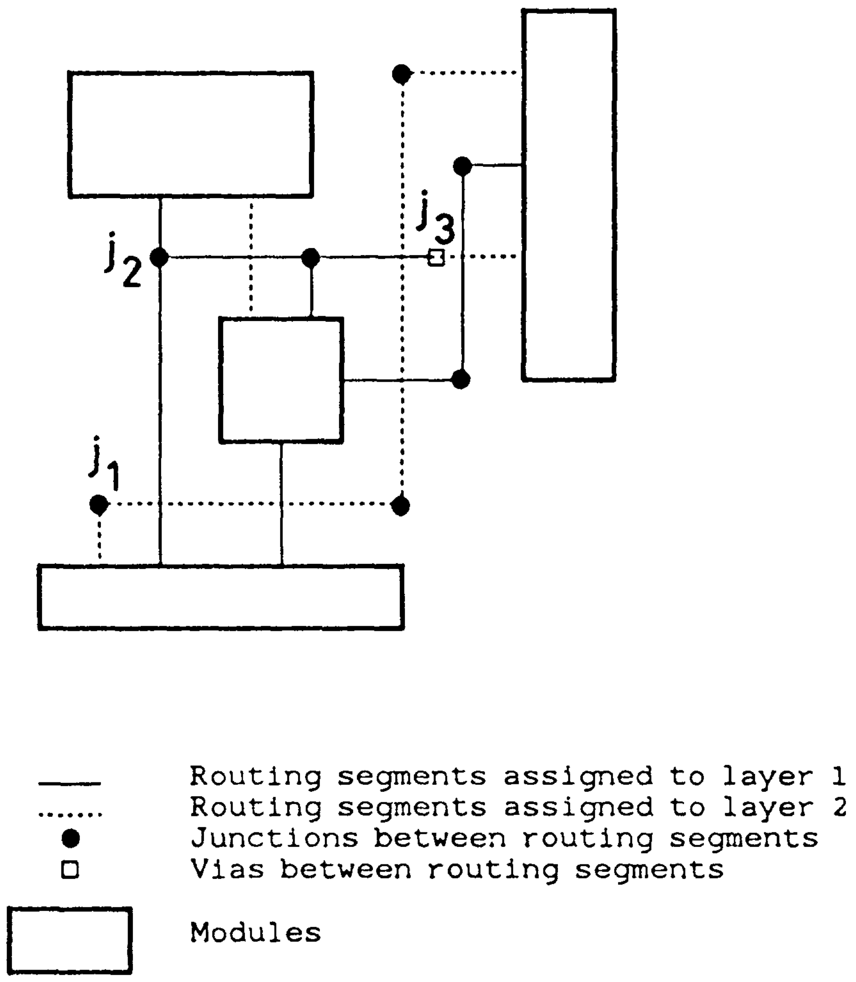

j_3
j_2
j_1
Routing segments assigned to layer 1
Routing segments assigned to layer 2
Junctions between routing segments
Vias between routing segments
Modules

Figure 1.1. Example Layout

defined by unordered pairs of endpoints. Each endpoint consists of a pair of x-y coordinates in the horizontal plane. A routing segment may be assigned to one of ℓ horizontal planes called *layers* where ℓ is some positive integer. We assume that all terminals and modules are available on all layers.

If two or more routing segments share an endpoint, then they are assumed to be electrically connected. If the point that they share is not a

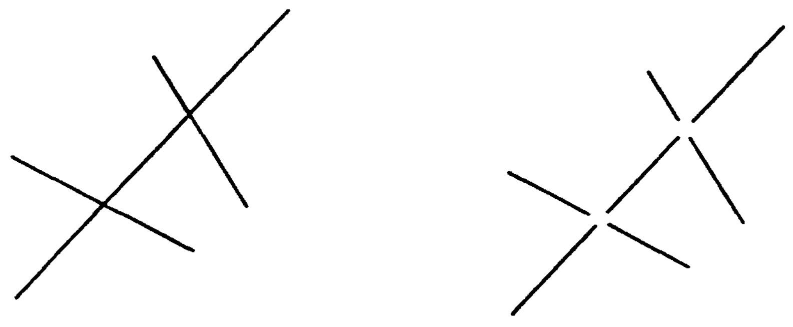


Figure 1.2. Routing segments and subsegments

terminal, it is called a *junction*. If any two routing segments meeting at a particular junction are assigned to different layers, the junction is called a *via*. By definition, a routing segment never contains any junctions except at its endpoints which are always either a terminal or a junction. The number of routing segments joined at a particular junction is called the *junction degree*. In Figure 1.1, the degree of junctions j_1 and j_3 is two while the degree of j_4 is three. A point where the vertical projections of two routing segments not in the same net intersect is called a *crossing*. A mapping $\alpha\colon \mathcal{R} \rightarrow \{1, 2, \ldots, \ell\}$ which assigns each routing segment to a layer in such a manner that no two segments assigned to the same layer cross and no two routing segments assigned to different layers form a junction without a via is called a *valid layer assignment*. A maximal length portion of a routing segment which does not contain any crossings is called a *routing subsegment*. Figure 1.2 shows a set of routing segments and the associated routing subsegments. Since it would be redundant to change layers more than once along any routing subsegment, we will only allow one via to be placed on each routing subsegment. We say that one set of routing segments $\mathcal{R}$ is *functionally equivalent* to another set of routing segments $\mathcal{R}'$, denoted by $\mathcal{R} \approx \mathcal{R}'$, if the vertical projections of the segments in $\mathcal{R}$ and $\mathcal{R}'$ cover the same set of points in the plane and realize the same netlist. Figure 1.3 shows two sets of routing segments which are functionally equivalent.

A *valid topological layout* is defined by a six-tuple $\mathcal{L} = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}, \alpha)$ where $\mathcal{M}$, $\mathcal{T}$, and $\mathcal{W}$ are the sets of modules, terminals, and nets, respectively. $\mathcal{R}$ is a set of routing segments which realizes the connections specified in $\mathcal{W}$. And, α is a mapping $\alpha: \mathcal{R} \rightarrow \{1, 2, \ldots, \ell\}$ which defines a valid layer assignment for $\mathcal{R}$ given the set of vias $\mathcal{V}$. $\mathcal{M}$ includes one special module m_ϕ defined by the boundary of the routing area with the infinite area outside of it as the module's interior. Given a three-tuple $\mathcal{P} = (\mathcal{M}, \mathcal{T}, \mathcal{W})$ and a positive integer ℓ, the *topological routing problem* is to find a valid topological layout $\mathcal{L} = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}, \alpha)$ which realizes the netlist W using at most ℓ layers.

A *valid geometric layout* differs from a topological layout in the fact that the routing segments, vias, and modules all have finite size and must meet additional constraints called *geometric design rules.* Given a three-tuple $\mathcal{P} = (\mathcal{M}, \mathcal{T}, \mathcal{W})$, an integer ℓ, and a set of geometric design rules GDR, the *geometric routing problem* is to find a valid geometric layout $\mathcal{L}_g = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}, \alpha)$ which realizes the netlist $\mathcal{W}$ using at most ℓ layers and which satisfies the rules in GDR. If $\ell \geq 2$, any topological routing problem

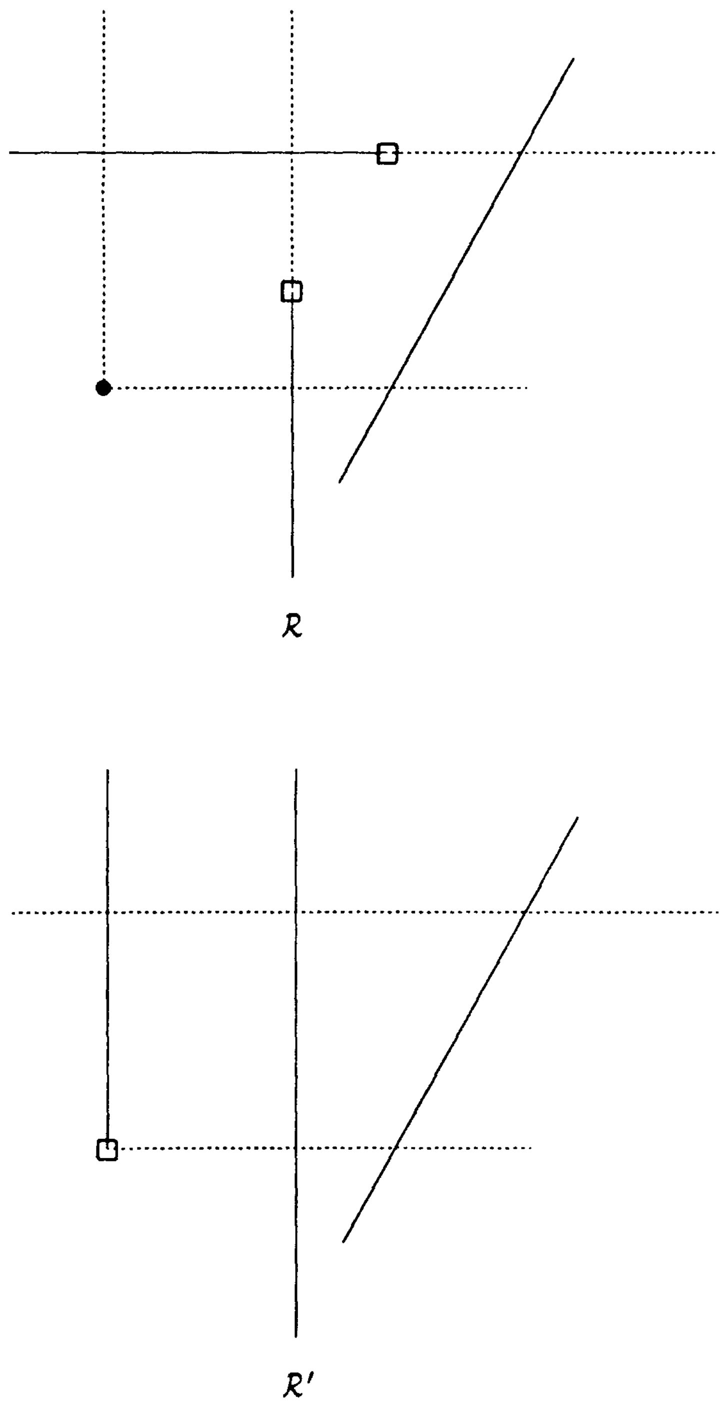


Figure 1.3. Functionally equivalent sets of routing segments

will have a solution, but a geometric routing problem may not. In the remainder of this work, the term layout will mean a valid topological layout unless otherwise noted. A layout which has all of its routing segments parallel to one of two perpendicular axes in the horizontal plane is called a *grid based* layout. In the case of grid based geometric layouts, all parallel routing segments are sometimes assumed to be separated by integer multiples of a value called the *grid pitch.* In the case of topological layouts, that pitch is assumed to be arbitrarily small.

4. Via Minimization Problems

There have been two approaches to minimizing the number of vias used in a particular layout. In the first approach, via minimization is accomplished during the routing step as the main optimization criterion. We call this *Unconstrained Via Minimization* and define the optimization problem, when $\ell \geq 1$ layers are available for routing, as follows:

(ℓ-UVM) Given a routing problem $\mathcal{P} = (\mathcal{M}, \mathcal{T}, \mathcal{W})$, generate a layout $\mathcal{L} = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}, \alpha)$ which uses at most ℓ layers and requires the minimum number of vias.

The result of using this method is the fewest possible vias at any cost. Not only is this a computationally difficult problem to solve but, the optimum ℓ-UVM solution is usually a layout which is undesirable because of long wire lengths and large area. Figure 1.4 shows an example of an optimum 2-UVM solution from [24].

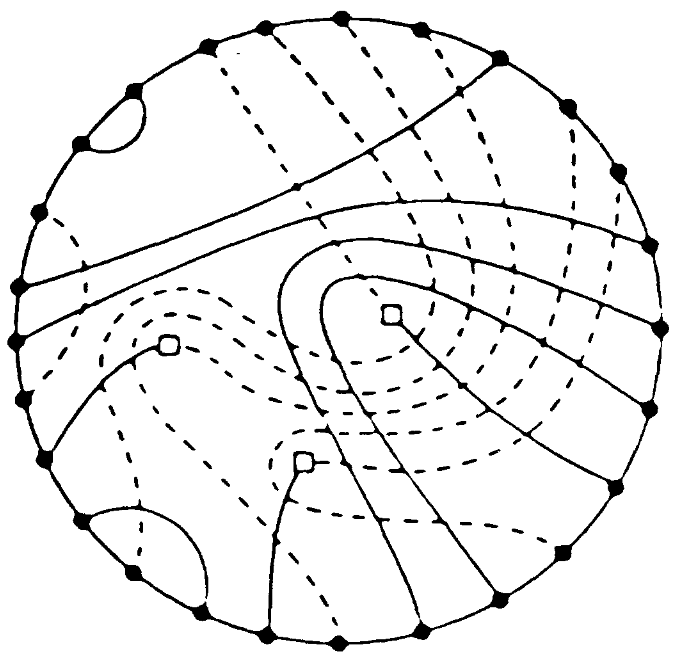


Figure 1.4. An optimum 2-UVM solution

In the second approach, via minimization is accomplished after the routing step has been completed and without significantly altering the layout generated by the router. We call this approach *Constrained Via Minimization* and define the optimization problem for $\ell \geq 1$ layer layouts as follows:

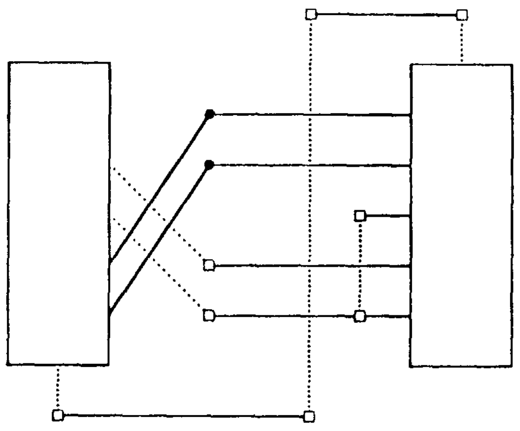
(ℓ-CVM) Given an ℓ-layer layout $\mathcal{L} = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}, \alpha)$, generate an ℓ-layer layout $\mathcal{L}' = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}', \mathcal{V}', \alpha')$ such that $\mathcal{R} \approx \mathcal{R}'$ and $\mathcal{V}'$ is a minimum cardinality set of vias.

For brevity, the 2-CVM problem will sometimes be referred to as simply

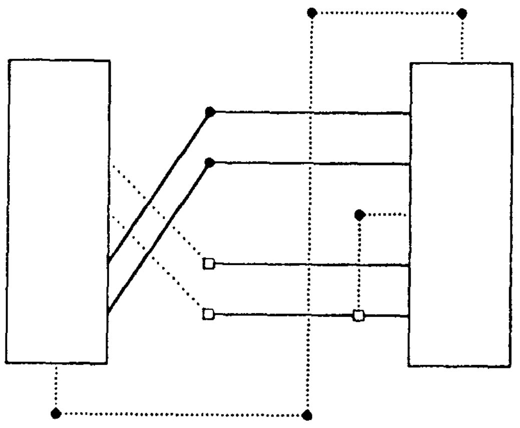
CVM. Note that unless otherwise specified, an ℓ-layer layout may utilize less than ℓ layers. Unlike the ℓ-UVM problem, in the ℓ-CVM problem minimizing the number of vias is a secondary consideration since the placement of the routing segments has already been determined by some other method. This allows more important criteria such as area or path length to be given preference. We will also consider the case when the maximum junction degree in a layout is limited. The ℓ-CVM$_k$ problem will be defined for positive integers l and k as follows:

(ℓ-CVM$_k$) Given an ℓ-layer layout $\mathcal{L} = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}, \alpha)$ with no junction having degree greater than k, generate an ℓ-layer layout $\mathcal{L}' = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}', \mathcal{V}', \alpha')$ such that $\mathcal{R} \approx \mathcal{R}'$ and $\mathcal{V}'$ is a minimum cardinality set of vias.

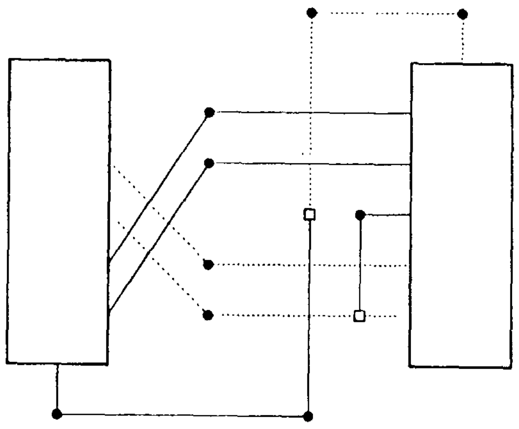
When comparing different approaches to a particular ℓ-CVM$_k$ problem, it is important to note that different authors define the set of possible via sites, called *via candidates*, differently. For example, one common set of via candidates consists of the set of endpoints of the given routing segments in $\mathcal{R}$. However, a complete set of via candidates would include one via site on every routing subsegment in $\mathcal{L}$ for a total of $O(|\mathcal{R}|^2)$ via candidates. It should be clear that this simplification may result in a significant speedup at the expense of optimality. Figure 1.5a shows a simple layout and Figures 1.5b and 1.5c show two different "optimum" 2-CVM$_3$ solutions. In Figure 1.5b, vias have been restricted to the endpoints of the given routing segments. In Figure 1.5c vias are placed at any geometrically allowable site. In the case of geometric layouts, the number of via sites allowed by the design rules may be significantly less than $O(|\mathcal{R}|^2)$.



a) 8 Vias



b) 3 Vias



c) 2 Vias

Figure 1.5. Input layout and two "optimum" solutions

5. Previous Work

Hashimoto and Stevens [17] first suggested the 2-CVM problem in 1971 as a way to reduce the number of vias generated by their channel routing algorithm. In their formulation, they assume that the layout is grid based, all nets contain exactly two terminals, and that vias are placed only at the endpoints of the given routing segments. Their problem was thought to be NP-hard† for nearly a decade, leading other researchers to develop heuristic algorithms [29] or use integer programming with branch and bound techniques to obtain an optimum solution [7]. In 1980, Kajitani [20] showed that the problem posed by Hashimoto and Stevens [17] can be solved in polynomial time using Hadlock's maximum cut algorithm for planar graphs [16]. This encouraged other researchers to look for a polynomial time solution for the more general case. In 1982, Pinter [27] obtained optimum results for the case when the maximum junction degree is limited to three, but his algorithm can not be applied to layouts which contain junctions with higher degree. For reasons that will become clearer in Chapter 3, this is a serious limitation even for grid based layouts.

In 1983, Chen, Kajitani, and Chan [4] presented a polynomial time algorithm for grid based layouts which gives optimum results when the maximum junction degree is limited to three and approximate results otherwise. However, they restrict vias to lie at the endpoints of the given set of routing segments. Recently, Chang and Du [3] developed a heuristic algorithm which

† See [14] for complete definitions of NP-completeness and NP-hardness.

can handle junctions of any degree, but which does not guarantee optimum results for any case.

Hsu [18] first considered the 2-UVM problem in 1983 with the added constraints that all terminals be placed on the boundary of the routing area, all nets consist of exactly two terminals, and no two wires cross each other more than once. Suspecting that the problem was NP-hard, he suggested a heuristic algorithm. Later Marek-Sadowska [24] considered the same problem without the constraint on crossings and showed how it could be solved optimally if a "maximum bipartite subgraph" of a particular graph could be found. Although her particular node-deletion problem is likely to be NP-hard, Marek-Sadowska failed to give a convincing proof.

6. Graph Theoretical Terminology and Notation

A *graph* is a pair $G = (N, E)$, where N is a set of nodes and E is a set of edges. Each edge consists of an unordered pair of nodes in N. Two nodes x and y are said to be *adjacent* if $(x, y) \in E$. The edge $(x, y) \in E$ is said to be *incident* upon each of the nodes x and y which are called its *endpoints*. If two edges share the same pair of endpoints, they are called *parallel* edges. If both endpoints of an edge are the same, the edge is called a *self-loop*. The number of edges incident upon a particular node is the *degree* of the node. A *path* is a sequence of not necessarily distinct edges $(n_{i_0}, n_{i_1}), (n_{i_1}, n_{i_2}), (n_{i_2}, n_{i_3}), \ldots, (n_{i_{m-1}}, n_{i_m})$. The length of the path between n_{i_0} and n_{i_m} is m. A path is called a *cycle* if $n_{i_0} = n_{i_m}$. A cycle is called a *simple cycle* if every node in the cycle is the endpoint of exactly two edges in the cycle. A graph is called *connected* if there is a path between every two nodes. A graph is called *complete* if every two nodes are adjacent. A graph is called *bipartite* if N can be partitioned into two subsets N_1 and N_2 such that no two nodes in the same subset are adjacent. A graph $H = (N', E')$ is called a *subgraph* of G if $N' \subseteq N$ and $E' \subseteq E$.

We will sometimes perform certain operations on a graph in order to generate another graph. We will call the procedure of removing a given set of nodes from N and all of their incident edges from E, *node deletion.* We will call the procedure of removing a given set of edges from E, *edge deletion.* If $N' \subseteq N$ for some graph $G = (N, E)$, then the subgraph (N', E') where $E' = \{(x, y) \in E \mid x, y \in N'\}$ is said to be *induced* by N' and will be denoted by $G(N')$. If we partition the set of nodes N into a family of subsets $N_1, N_2, N_3, \ldots, N_m$ such that two nodes are in the same subset N_i if and only if there is a path between them in G, then we call the induced subgraphs $G(N_1), G(N_2), G(N_3), \ldots, G(N_m)$ *connected components.* We will define the procedure of *contracting* an edge (x, y) as follows. First (x, y) is deleted from E. Second all edges $(y, z) \in E$ are replaced by (x, z). Finally, y is deleted from N. Given a connected graph $G = (N, E)$, a *cutset* is a set of edges whose deletion partitions N into two nonempty subsets N_1 and N_2 such that no node in N_1 is adjacent to any node in N_2.

A representation of a connected graph $G = (N, E)$ consisting of a set of points corresponding to the nodes in N and a set of curves connecting exactly those pairs of points which correspond to adjacent nodes is called an *embedding.* A particular graph can have many different embeddings. If all of the points lie in the plane and no two curves intersect except at their endpoints, then we call the embedding a *planar embedding.* We will denote a planar embedding of G by $\tilde{G}$. Any graph that has at least one planar embedding is called a *planar graph.* A planar embedding of G in which each edge is represented by a straight line segment is called a *straight line planar embedding* and will be denoted by $\overline{G}$. Any planar graph can be represented by a straight line planar embedding [11]. In a planar embedding of a graph, any area which does not contain any edges and which is bounded by a cycle is called a *face.* The infinite area surrounding the planar embedding is called the *exterior face.* A cycle which defines a face in the embedding is called a *fundamental cycle.*

For a particular planar embedding $\tilde{G}$ of a graph $G = (N, E)$, we define a unique graph $G^d = (N^d, E^d)$ called a *dual graph.* There is a one-to-one correspondence between faces in $\tilde{G}$ and nodes in G^d . There is also a one-to-one mapping $\rho: E^d \rightarrow E$ such that a set of edges S is a simple cycle in E^d if and only if $\rho(S)$ is a cutset for G. An embedding of G^d can be superimposed on $\tilde{G}$ so that nodes lie on the faces they correspond to and each edge $(x, y) \in E$ is intersected by $\rho(x, y)$. Two faces are said to be *adjacent faces* if their corresponding nodes in the dual graph are adjacent.

Chapter II

Complexity

1. Overview

In this chapter, we will show for the first time that without restrictions on the maximum junction degree, the Constrained Via Minimization problem is NP-hard. For clarity, we begin by showing a polynomial transformation from the NP-complete Planar Node Cover problem [15] to the general 2-CVM problem. We then refine our method to show a transformation from a restricted version of the Planar Node Cover problem [13] to the 2-CVM$_6$ problem with the additional restrictions that the layout be grid based and vias be placed at the endpoints of the given routing segments. In this manner we prove that the 2-CVM problem is NP-hard even when one or more of the following restrictions are made.

1) The input layout must be grid based.
2) Only endpoints of the given routing segments are considered via candidates.
3) The maximum junction degree is limited to six or more.

2. The General 2-CVM Problem

Let us begin by considering the general 2-CVM problem. The input to this problem is a layout $\mathcal{L} = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}, \alpha)$ such that $\mathcal{M}$, $\mathcal{T}$, $\mathcal{W}$, $\mathcal{R}$, and $\mathcal{V}$ are the sets of modules, terminals, nets, routing segments, and vias, respectively. α is a mapping $\alpha: \mathcal{R} \rightarrow \{1, 2, \ldots, \ell\}$ which defines the layer assignment of each routing segment in $\mathcal{R}$. The layout is not required to be grid based and may take any form. There are no restrictions on the choice of via candidates, or on the maximum junction degree. The decision problem to be considered is defined as follows:

(*cvm*) Given a layout $\mathcal{L} = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}, \alpha)$ and a positive integer k, does there exist a layout $\mathcal{L}' = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}', \mathcal{V}', \alpha')$ such that $\mathcal{R} \approx \mathcal{R}'$ and $|\mathcal{V}'| \leq k$?

Note that k may be bounded by $|\mathcal{R}|^2$, since no more than one via along any routing subsegment is ever required. Thus, a solution with k or fewer vias could easily be guessed non-deterministically in polynomial time. Since it is easy to test if that solution is valid, we know that $cvm \in NP$. In order to show that the problem is NP-complete, we will show a polynomial transformation from the Planar Node Cover problem(PNC) [15] whose decision problem is defined as follows:

(*pnc*) Given a planar graph $G = (N, E)$ and a positive integer k, does there exist a node cover N' for G satisfying $|N'| \leq k$?

A *node cover* for a graph $G = (N, E)$ is a subset $N' \subseteq N$ such that for each edge $(n_i, n_j) \in E$, $n_i \in N'$ or $n_j \in N'$. The *pnc* problem was shown to be NP-complete in Theorem 2.7 of [15].

In order to construct the transformation, we will use the sublayout H shown in Figure 2.1. Note that H requires at least one via if it is to be realized using only two layers. If that via is placed at either the junction labeled v_s or the junction labeled v_t, then a valid layer assignment will exist for H.

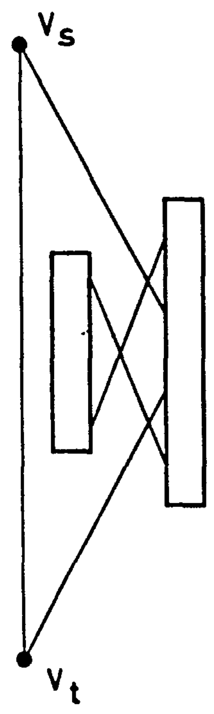


Figure 2.1. **Sublayout** H

Lemma 2.1. *If we create a layout L by joining one or more nonoverlapping sublayouts identical to H together at either v_s or v_t, and replacing either v_s or v_t from each of the H's with a via, then L will have a valid layer assignment.*

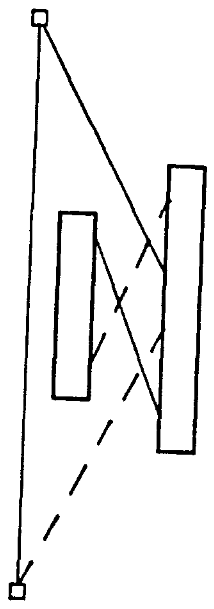


Figure 2.2. Type 1 independent sublayout

Proof: We will define an *independent sublayout* I as a portion of a layout whose routing segments intersect routing segments not belonging to I only at vias. Because of this property, routing segments within a particular independent sublayout I can be assigned to layers without considering the layer assignment of any segments which do not belong to I. Any layout L constructed in the manner described above can be partitioned into two types of independent sublayouts which are shown in Figures 2.2 and 2.3. The independent sublayouts of Type 1 consist of a single H and have a via at both v_s and v_t. The independent sublayouts of Type 2 are made up of one or more H's joined at a single junction j and have a via at whichever of v_s or v_t is not coincident with j. As shown in Figures 2.2 and 2.3, valid layer assignments exist for both types of independent sublayouts. Since L is made up of only two types of independent sublayouts and both types of independent

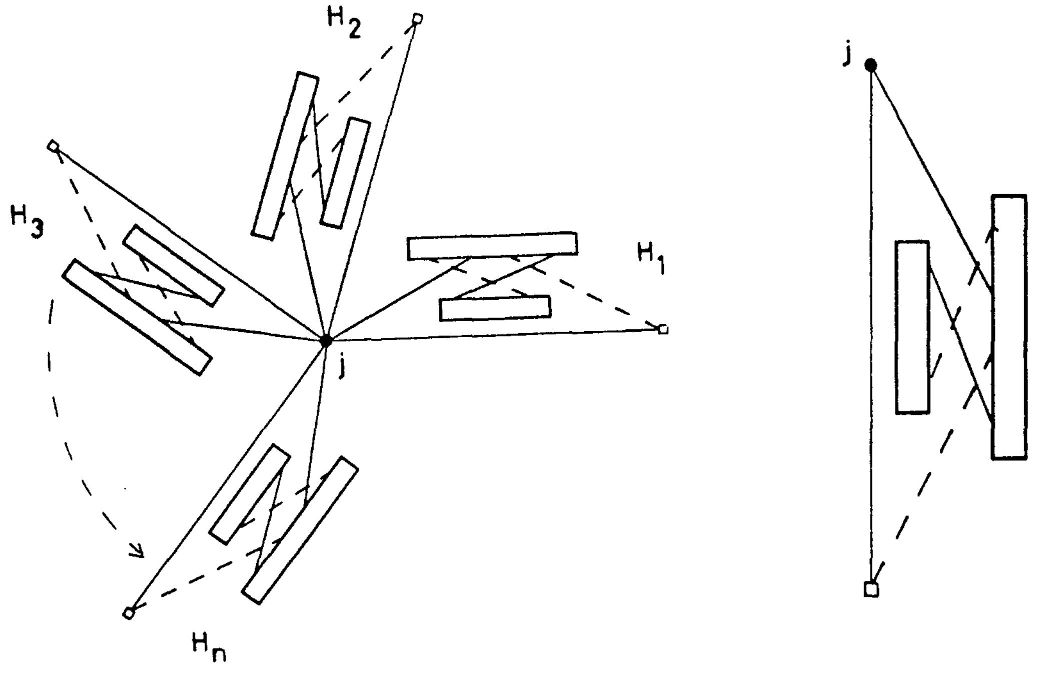


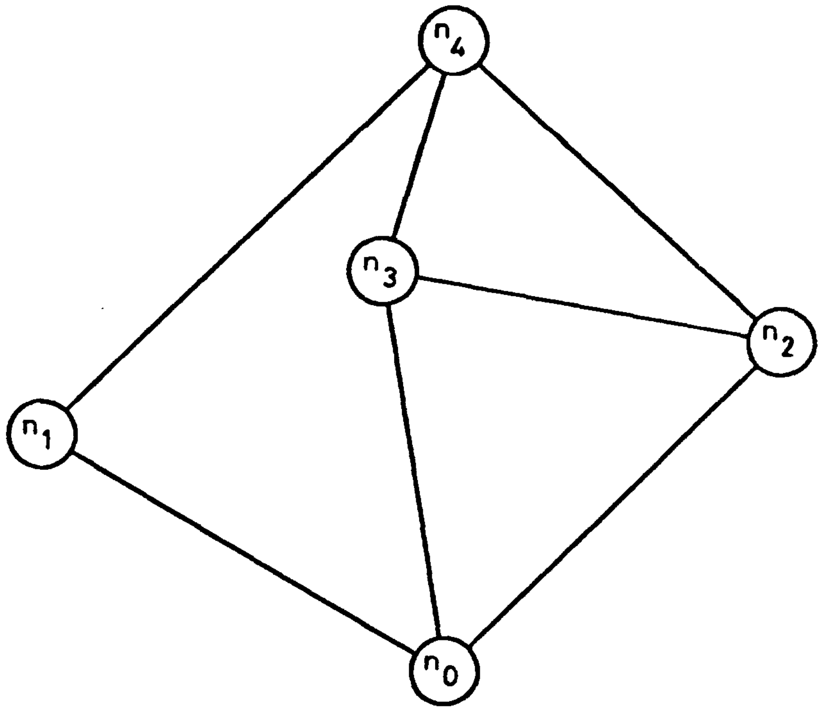
Figure 2.3. Type 2 independent sublayouts

sublayouts have valid layer assignments, we know that the entire layout has a valid layer assignment. ∎

We are now ready to construct a polynomial transformation from *pnc* to *cvm*. Given an instance of *pnc* consisting of a planar graph $G = (N, E)$ and a positive integer k, we will generate an instance of *cvm* consisting of a layout $\mathcal{L}_G$ and the same integer k. Let $\overline{G}$ be a straight line planar embedding of the graph G. There are several well known algorithms for obtaining a planar embedding of a graph in polynomial time (for example [5]). Fary [11] showed that any such embedding can be converted to a straight line planar embedding in polynomial time. Rather than introducing any additional notation, we will simply refer to the line segments and points in $\overline{G}$ by the edges and nodes that they correspond to in G. We will form a small region around each edge (s,t) in $\overline{G}$ in such a manner that no two regions overlap. This will always be possible since no two edges intersect in the planar embedding. Next, we will replace each edge (s,t) with a sublayout H in such a manner that H is completely within the region surrounding the edge (s,t) and that v_s and v_t coincide with s and t, respectively. We call the resulting layout $\mathcal{L}_G$. Figure 2.4 shows a straight line planar embedding for a graph G_a and the layout $\mathcal{L}_{G_a}$ obtained from it.

Assume that G has a node cover N_1 such that $|N_1| \leq k$. By definition, N_1 must include at least one endpoint from each edge in E. Let $\mathcal{V}_1$ be the set of vias which is made up of exactly the junctions that correspond to the nodes in N_1. $\mathcal{V}_1$ will include either v_s or v_t (or both) from every H and $|\mathcal{V}_1| = |N_1|$. Thus, by Lemma 2.1, $\mathcal{V}_1$ is a set of vias of size k or less for which a valid layer assignment for $\mathcal{L}_G$ exists.

Now assume that there is a set of vias $\mathcal{V}_2$, of size k or less, for $\mathcal{L}_G$ for which a valid layer assignment exists. From our discussion, we know that $\mathcal{V}_2$ must include at least one via located in every H. Although there are no restrictions in *cvm* on the location of vias, we will assume that all vias are placed at junctions which originally corresponded to either v_s or v_t for some H. This does not affect the generality of the solution, since a via placed elsewhere could easily be moved to either v_s or v_t, in the same H without increasing the total number of vias or affecting the existence of a valid layer assignment. Since the sublayouts overlap only at v_s and v_t, there would be no reason to place a via elsewhere. Thus, $\mathcal{V}_2$ will include at least



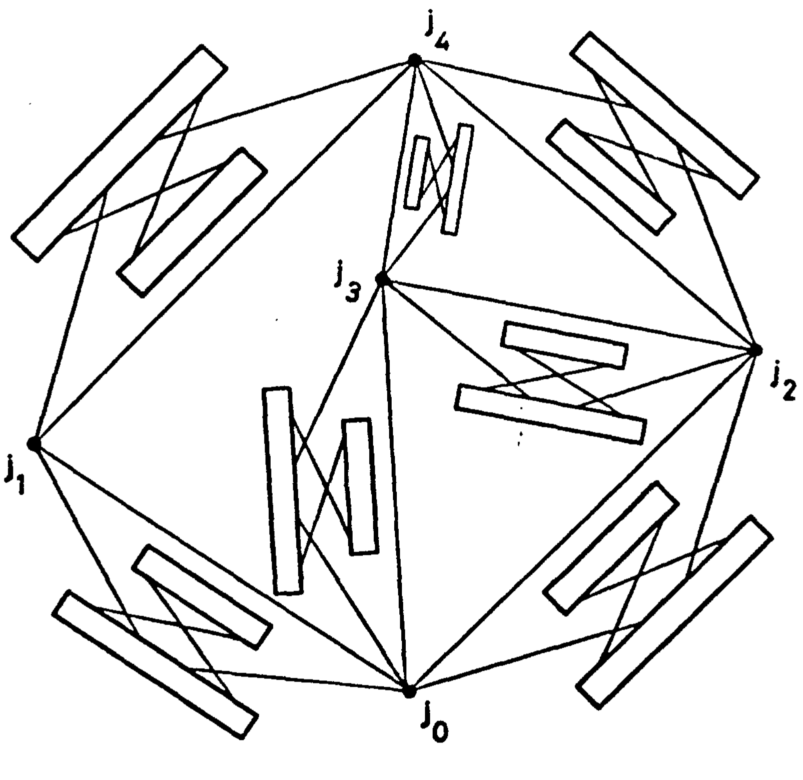


Figure 2.4. Straight line embedding $\overline{G}_a$ and the layout $\mathcal{L}_{G_a}$

one via placed at either v_s or v_t (or both) for each H. Let N_2 be the set of nodes which correspond to the vias in $\mathcal{V}_2$. N_2 will include at least one endpoint of every edge in E and thus constitutes a node cover for G with $|N_2| = |\mathcal{V}_2| \leq k$.

Since the transformation from *pnc* on G and k to *cvm* on $\mathcal{L}_G$ and k described above only requires replacing each edge in the straight line planar embedding of G with the sublayout H, it can easily be accomplished in polynomial time with respect to the size of the input. Thus, we have the following theorem:

Theorem 2.1. *cvm is NP-complete .* ■

3. The Restricted 2-CVM$_6$ Problem

In this section, we show that the decision problem associated with the restricted version of the 2-CVM problem described by Chen, et al. [4] with the further restriction that the maximum junction degree be limited to six is NP-complete. This result will then be extended to show that for any combination of the restrictions ennumerated below, the problem remains NP-complete.

1) The input layout must be grid based.
2) Only endpoints of the given routing segments are considered via candidates.
3) The maximum junction degree is limited to six or more.

We will distinguish between the various decision problems by using a single subscript to denote the maximum junction degree allowed for an instance and superscripts to denote any other restrictions imposed. Thus, we will denote the most restricted decision problem as $cvm_6^{1,2}$ which is defined as follows:

($cvm_6^{1,2}$) Given a grid based layout $\mathcal{L} = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}, \alpha)$ with no junction having degree exceeding six and a positive integer k, does there exist a layout $\mathcal{L}' = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}', \mathcal{V}', \alpha')$ such that $\mathcal{R} \approx \mathcal{R}'$, all of the vias in $\mathcal{V}'$ are located at endpoints of segments in $\mathcal{R}$, and $|\mathcal{V}'| \leq k$.

By the same reasoning that was used for *cvm*, we know that $cvm_6^{1,2} \in NP$. In order to prove that it is also NP-complete, we will show a polynomial transformation from a restricted version of the Planar Node Cover problem which is described below.

(pnc_3) Given a planar graph $G = (N, E)$ with no node having degree greater than three and a positive integer k, does there exist a node cover N' for G satisfying $|N'| \leq k$?

pnc_3 was shown to be NP-complete in Lemma 2.1 of [13].

Suppose that we have a planar graph $G = (N, E)$ and a positive integer k as an instance of pnc_3. We will generate an instance of $cvm_6^{1,2}$ consisting of a layout $\mathcal{L}_G^*$ and the same integer k. In order to accomplish this, we will first embed the graph G on a planar rectangular grid. Tamassia [30] recently presented a polynomial time algorithm to do this which generates $O(|E|)$ straight line segments. We will denote such a planar grid embedding of G

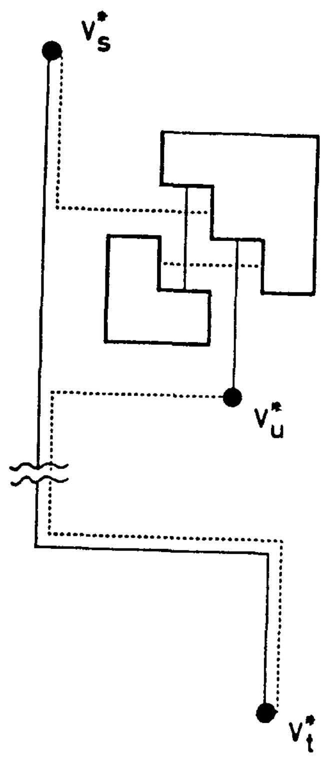
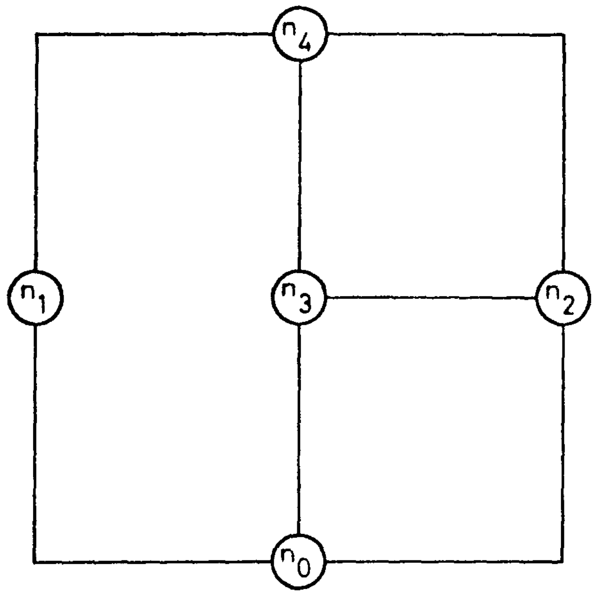


Figure 2.5. Sublayout H^*

by $\hat{G}$. We then proceed in a manner similar to that used in the previous section using a sublayout H^* shown in Figure 2.5. H^* also requires at least one via if it is to have a valid layer assignment. Because H^* is topologically equivalent to H, Lemma 2.1 will hold for H^*. We will replace each edge (s,t) in $\hat{G}$ with an H^* in such a manner that v_s^* and v_t^* coincide with s and t, respectively, to obtain the layout $\mathcal{L}_G^*$. This is done by replacing the first vertical or horizontal segment of each edge with the portion of H^* nearest v_s^* and then adapting the remaining portion to follow the rest of the line segments which make up the edge. If the H^*'s are sufficiently narrow, then no two H^*'s will intersect except at v_s^* or v_t^*. Figure 2.6 shows a planar grid embedding $\hat{G}_a$ of G_a and the corresponding layout $\mathcal{L}_{G_a}^*$. Since no node in



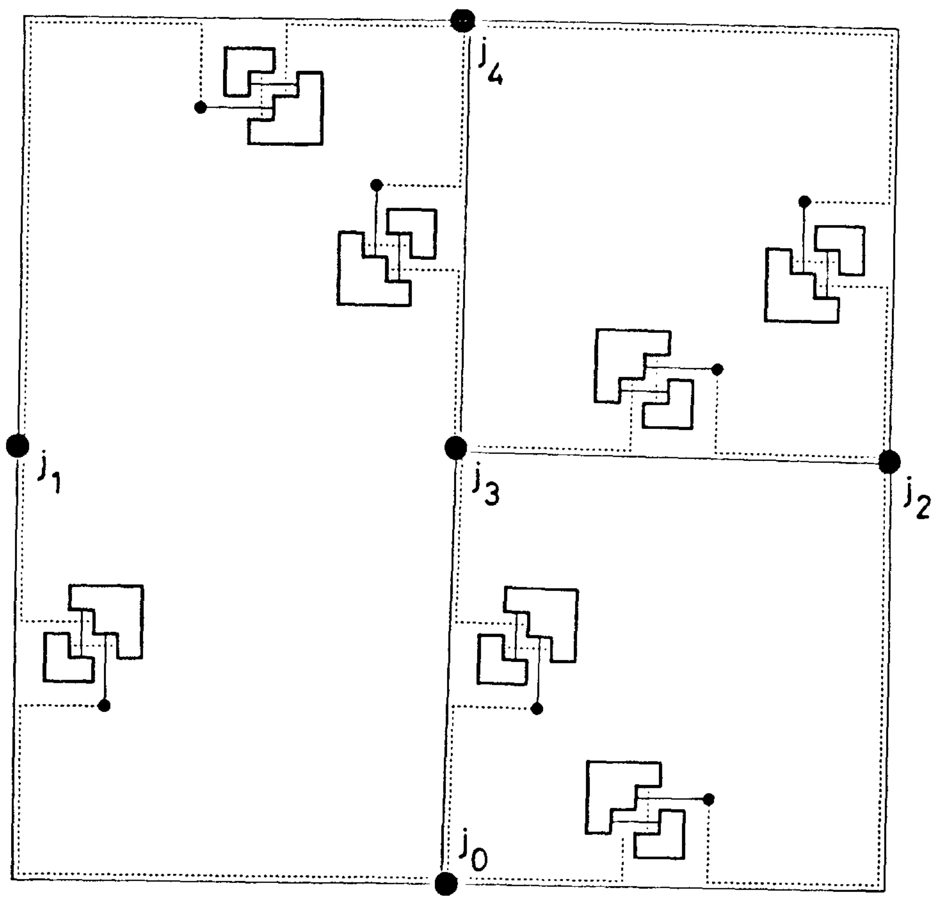


Figure 2.6. Planar grid embedding $\hat{G}_a$ and the layout $\mathcal{L}^*_{G_a}$

G can have degree exceeding three and the junctions v_s^* and v_t^* in H^* have degree two, no junction in $\mathcal{L}_G^*$ will have degree exceeding six.

$\mathcal{L}_{G_a}^*$

Assume that G has a node cover N_1 such that $|N_1| \leq k$. Let $\mathcal{V}_1$ be defined as the set of junctions in $\mathcal{L}_G^*$ which correspond to the nodes in N_1. We can show using Lemma 2.1 that if the junctions in $\mathcal{V}_1$ are made vias, then a layout is obtained for $\mathcal{L}_G^*$ which requires k or fewer vias.

Now assume that there is a set of k or fewer vias $\mathcal{V}_2$ which are all located at the endpoints of the routing segments in $\mathcal{L}_G^*$ and for which a valid layer assignment exists. We know that $\mathcal{V}_2$ must include at least one via located in every H^*. Furthermore, we can assume that all of the vias are placed at junctions corresponding to either v_s^* or v_t^* since a via placed at a junction corresponding to v_u^* in some H^*, can be moved to either v_s or v_t in the same H^*. Thus, $\mathcal{V}_2$ will include at least one via placed at either v_s^* or v_t^* (or both) for each H. Let N_2 be the set of nodes which correspond to the vias in $\mathcal{V}_2$. Using the same argument as in the proof of Theorem 2.1, N_2 will be a node cover for G with k or fewer nodes.

As mentioned earlier, we can construct, in polynomial time, a planar grid embedding $\hat{G}$ of G which has $O(|E|)$ straight line segments [30]. We then generate $\mathcal{L}_G^*$ by replacing each line segment with a portion of the sublayout H^*. Thus, the entire transformation can be accomplished in polynomial time and we have the following theorem:

Theorem 2.2. *$cvm_6^{1,2}$ is NP-complete .* ∎

It should be noted that the proof of this theorem does not have to take advantage of Restriction 2 on via location. In fact, as stated in the proof of Theorem 2.1, vias placed anywhere can always be moved to either v_s^* or v_t^* in any H^* without affecting the results. Thus, the NP-completeness result will hold without Restriction 2 and we have:

Theorem 2.3. *cvm_6^1 is NP-complete .* ■

Because Restrictions 1 and 3 affect only the class of valid instances for the *cvm* problem, relaxing either or both of these constraints does not affect our NP-completeness results. Thus, it follows from Theorem 2.2 that cvm_6^2, $cvm_j^{1,2}$, and cvm_j^2 are all NP-complete for any j greater than six. From Theorem 2.3, we can also deduce that cvm_j^1, cvm_6, and cvm_j are also NP-complete for all j greater than six. Thus, we have our strongest theorem.

Theorem 2.4. *The cvm problem is NP-complete given one or more of the following restrictions:*

1) The input layout must be grid based.

2) Only endpoints of the given routing segments are considered via candidates.

3) The maximum junction degree is limited to six or more.

■

4. Summary

In this chapter, we have shown that the general Constrained Via Minimization decision problem is NP-complete. We have also shown that this result holds when the input layout is constrained to lie on a rectangular grid and/or when vias are restricted to lie at junctions which existed in the input layout. In addition, our result holds when the maximum junction degree is limited to six or more with or without any of the other constraints listed earlier.

Chapter III

Algorithms

1. Overview

This chapter will describe a unified algorithmic approach to the 2-CVM problem. Section 2 describes the basic 2-CVM$_3$ algorithm for topological layouts. Section 3 describes how to decrease the average time complexity for the 2-CVM$_3$ algorithm when it is applied to geometric layouts. Section 4 describes how to extend the algorithm to the general 2-CVM problem. Section 5 discusses the applicability of the basic 2-CVM$_3$ algorithm to a particular curve coloring problem. Finally, Section 6 summarizes the results of this chapter.

2. The 2-CVM$_3$ Problem for Topological Layouts

2.1. The basic 2-CVM$_3$ algorithm

We begin by considering the case when two layers are available for routing, vias may be placed anywhere, and no junction is permitted to have degree greater than three. We assume, without loss of generality, that each terminal is the endpoint of exactly one routing segment. If a terminal was actually the endpoint of more than one routing segment, it could be split into several terminals each separated by some very small distance. We do not place any other constraints on the topology of the layout except that it can be represented by a set of straight line segments. We define a graph $G_{\mathcal{L}} = (N, E)$ based on a given layout $\mathcal{L}$ as follows:

$$N = C \cup J \cup T$$

where

$C = \{c_x \mid c_x$ corresponds to a unique crossing in $\mathcal{L}$ located at $SITE(c_x)\}$,

$J = \{j_x \mid j_x$ corresponds to a unique junction in $\mathcal{L}$ located at $SITE(j_x)\}$,

$T = \{t_x \mid t_x$ corresponds to a unique terminal in $\mathcal{L}$ located at $SITE(t_x)\}$,

and

$E = \{(n_x, n_y) \mid$ there is a unique subsegment $SEG(n_x, n_y)$ in $\mathcal{L}$ with endpoints $SITE(n_x)$ and $SITE(n_y)\}$.

$SITE(x)$ denotes the point in $\mathcal{L}$ where the feature corresponding to node x is located. $SEG(x, y)$ denotes the subsegment in $\mathcal{L}$ corresponding to the edge (x, y) in $G_{\mathcal{L}}$. Clearly, $G_{\mathcal{L}}$ cannot contain any self-loops or parallel edges. Figure 3.1 shows an example layout $\mathcal{L}_2$ and the derived graph $G_{\mathcal{L}_2}$. We also define $VIA(x, y)$ to be some legal via site on $SEG(x, y)$. In this section, we assume that $VIA(x, y)$ is never an endpoint of $SEG(x, y)$ and thus all

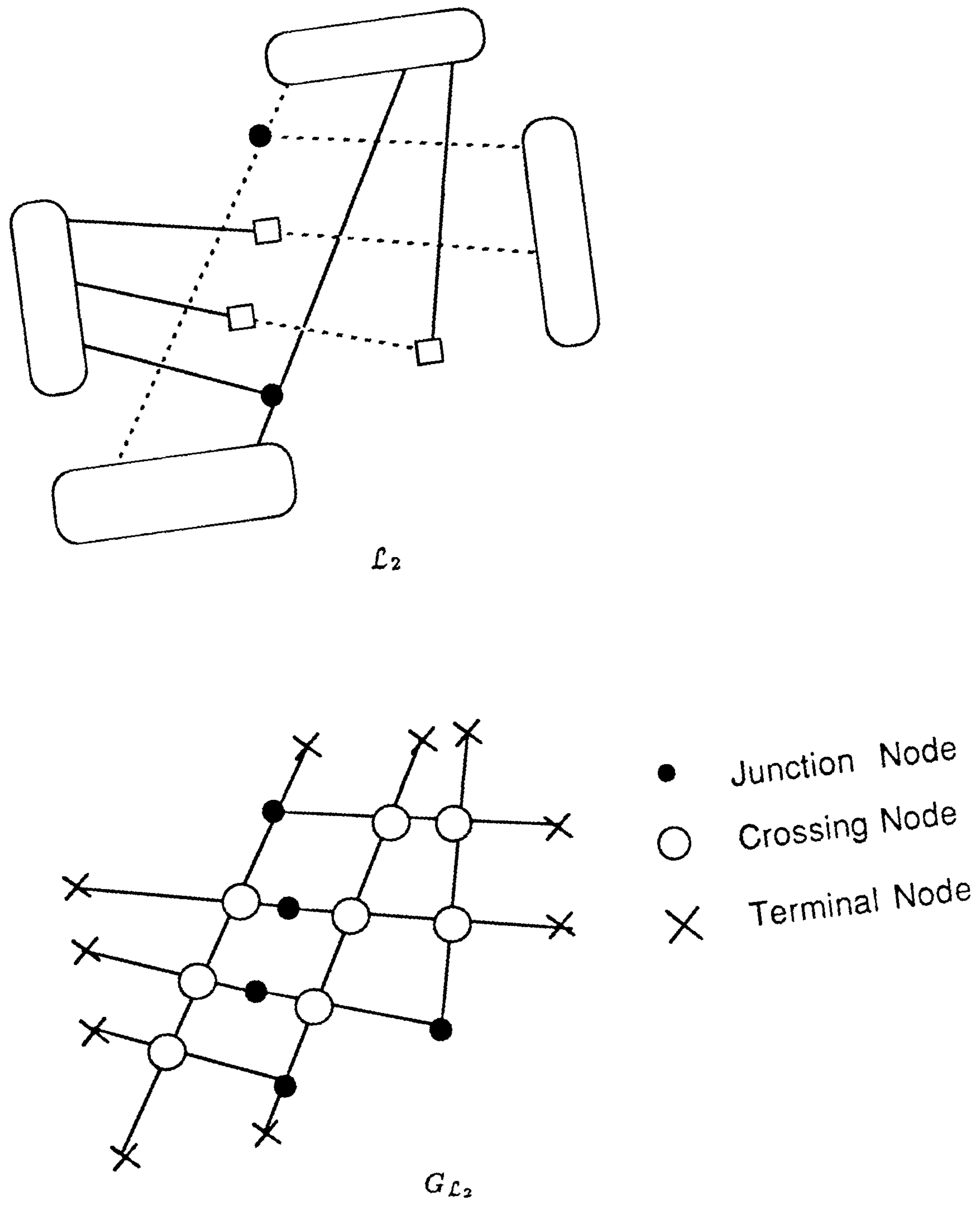


Figure 3.1. Example Layout $\mathcal{L}_2$ and the derived graph $G_{\mathcal{L}_2}$

via candidates are distinct. This is possible because we limit the maximum junction degree to three so that any via placed at a junction $SITE(j)$ can always be shifted slightly onto some subsegment $SEG(x, j)$ [26] as shown in Figure 3.2.

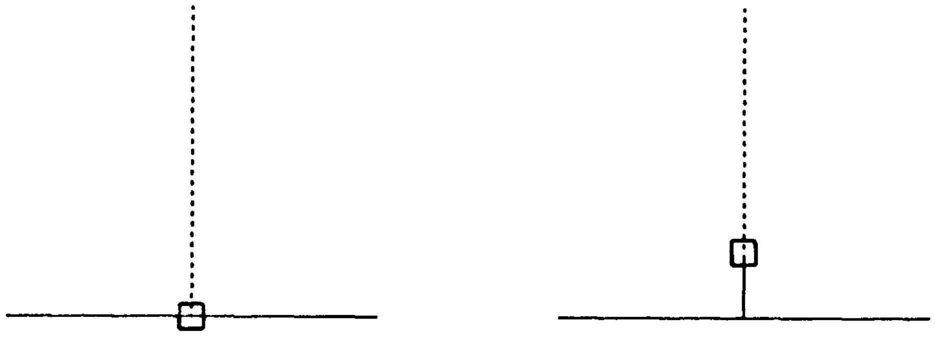


Figure 3.2. Equivalent Via Locations

For any $G_{\mathcal{L}}$, we define a straight line planar embedding, denoted by $\overline{G_{\mathcal{L}}}$, in which each node $n \in N$ is located at $SITE(n)$. If a face in $\overline{G_{\mathcal{L}}}$ is bounded by a cycle $(n_{i_0}, n_{i_1}), (n_{i_1}, n_{i_2}), \ldots, (n_{i_{m-1}}, n_{i_0})$ such that

$$\left|\left\{n_{i_k} \in C \mid 0 \leq k \leq m-1\right\}\right|$$

is odd, then we call it an *odd face*; otherwise we call it an *even face*.

Lemma 3.1. *Given a valid topological layout* $\mathcal{L} = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}, \alpha)$, *there exists a valid topological layout* $\mathcal{L}' = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}', \alpha')$ *such that* $\mathcal{V}' = \phi$ *if and only if* $\overline{G_{\mathcal{L}}}$ *is free of odd faces.*

Proof: Assume that there exists a valid topological layout $\mathcal{L}' = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}', \alpha')$ such that $\mathcal{V}' = \phi$. Each face is bounded by a sequence of edges $(n_{i_0}, n_{i_1}), (n_{i_1}, n_{i_2}), \ldots, (n_{i_{m-1}}, n_{i_0})$. For each pair of edges $(n_{i_{k-1}}, n_{i_k})$ and $(n_{i_k}, n_{i_{k+1}})$ such that $n_{i_k} \in C$, $SEG(n_{i_{k-1}}, n_{i_k})$ and $SEG(n_{i_k}, n_{i_{k+1}})$ cross at $SITE(n_{i_k})$ in $\mathcal{L}$ and thus must be assigned to different layers. For each pair of edges $(n_{i_{k-1}}, n_{i_k})$ and $(n_{i_k}, n_{i_{k+1}})$ such that $n_{i_k} \in J$, $SEG(n_{i_{k-1}}, n_{i_k})$ and $SEG(n_{i_k}, n_{i_{k+1}})$ form a junction at $SITE(n_{i_k})$ in $\mathcal{L}$. Since $SITE(n_{i_k})$ cannot be a via in $\mathcal{L}'$, $SEG(n_{i_{k-1}}, n_{i_k})$ and $SEG(n_{i_k}, n_{i_{k+1}})$ must be assigned to the same layer. Thus, layer changes occur only at crossings in $\mathcal{L}'$ and since the number of such changes must be even for any cycle, all faces in $\overline{G_{\mathcal{L}}}$ are even. Conversely, if we assume that all faces in $\overline{G_{\mathcal{L}}}$ are even, then all layer changes can take place at crossings in $\mathcal{L}$ and there exists a valid topological layout $\mathcal{L}' = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}', \alpha')$ such that $\mathcal{V}' = \phi$. ∎

If we insert a via v on a particular subsegment $SEG(x, y)$, then we can "cut" the corresponding edge (x, y) in $G_{\mathcal{L}}$ by deleting (x, y) and adding two new nodes a and b to J and two new edges (x, a) and (b, y) to E as shown in Figure 3.3. Both $SITE(a)$ and $SITE(b)$ refer to v. The routing segment in $\mathcal{R}$ which includes the subsegment $SEG(x, y)$ is now partitioned into two routing segments at v. The two fundamental cycles which shared (x, y) now become one cycle with a length equal to the sum of the lengths of the two constituent cycles. A set of edges whose cutting leaves $\overline{G_{\mathcal{L}}}$ free of odd faces is called an *Odd Face Cover* (OFC). An OFC of minimum cardinality is called a *Minimum Odd Face Cover* (MOFC). Figure 3.4 shows an MOFC for $\overline{G_{\mathcal{L}_2}}$. Since there is a one-to-one correspondence between edges in $G_{\mathcal{L}}$ and via candidates in $\mathcal{L}$, we have the following lemma:

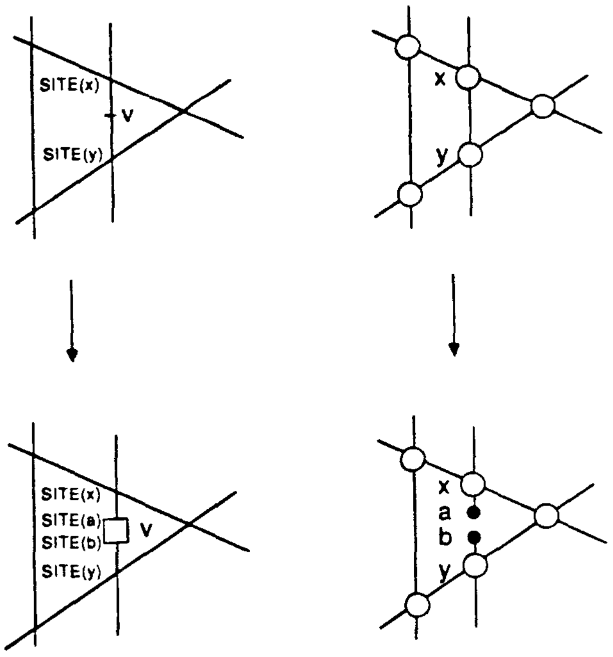


Figure 3.3. Inserting a via in $\mathcal{L}$

Lemma 3.2. *Given some valid topological layout $\mathcal{L} = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}, \alpha)$, if P is an MOFC for $\overline{G_{\mathcal{L}}}$, then there exists a valid topological layout $\mathcal{L}' = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}', \mathcal{V}', \alpha')$ such that $\mathcal{R}' \approx \mathcal{R}$ and $\mathcal{V}' = \{VIA(x,y) \mid (x,y) \in P\}$ is of minimum cardinality.*

∎

We use a method similar to that of Hadlock [1,16] to obtain an MOFC for the planar graph $G_{\mathcal{L}}$. Without loss of generality, we assume that $G_{\mathcal{L}}$ is connected. It is not difficult to see that the number of odd faces in $\overline{G_{\mathcal{L}}}$ must always be even. Furthermore, eliminating an odd face can only be accomplished by merging it with another odd face. If a particular odd face does not share an edge with another odd face, it may be necessary to first

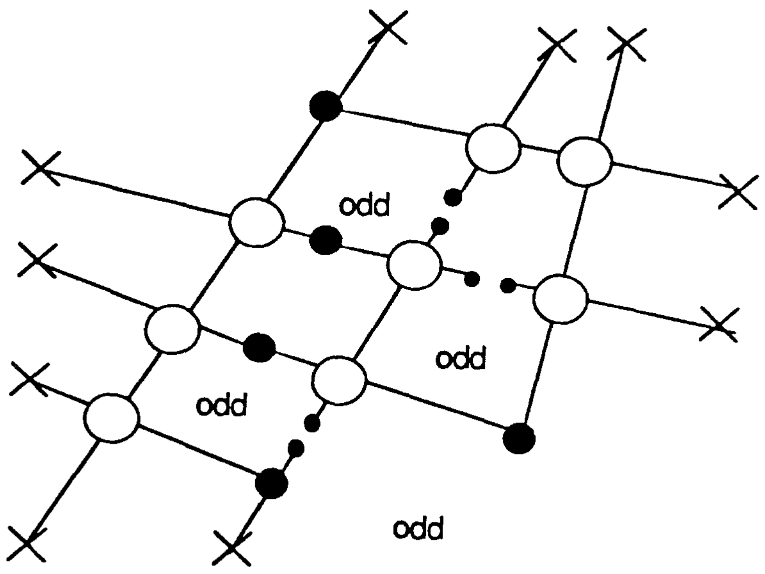


Figure 3.4. An MOFC for $\overline{G_{\mathcal{L}_2}}$

merge it with several even faces. Thus, the problem of finding an MOFC reduces the problem of finding a pairing of odd faces such that the total number of edges which must be cut to merge all pairs is minimized. To simplify the bookkeeping necessary to do this, we generate the dual of $\overline{G_{\mathcal{L}}}$, denoted by $G^d_{\mathcal{L}} = (N^d, E^d)$. In $G^d_{\mathcal{L}}$, one node corresponds to each face in $\overline{G_{\mathcal{L}}}$ and one edge $(x, y) \in E^d$ to each edge $(r, s) \in E$. Those edges which form self loops in $G^d_{\mathcal{L}}$ can be deleted since they correspond to edges in $G_{\mathcal{L}}$ which are not in any cycle. We define a mapping $\rho: E^d \rightarrow E$ such that $\rho(x, y) = (r, s)$. Those nodes which correspond to odd faces, as we have defined them, are called *odd nodes*. Contracting an edge in $G^d_{\mathcal{L}}$ corresponds to cutting an edge in $G_{\mathcal{L}}$. When two nodes in $G^d_{\mathcal{L}}$ are merged by contracting the edge between

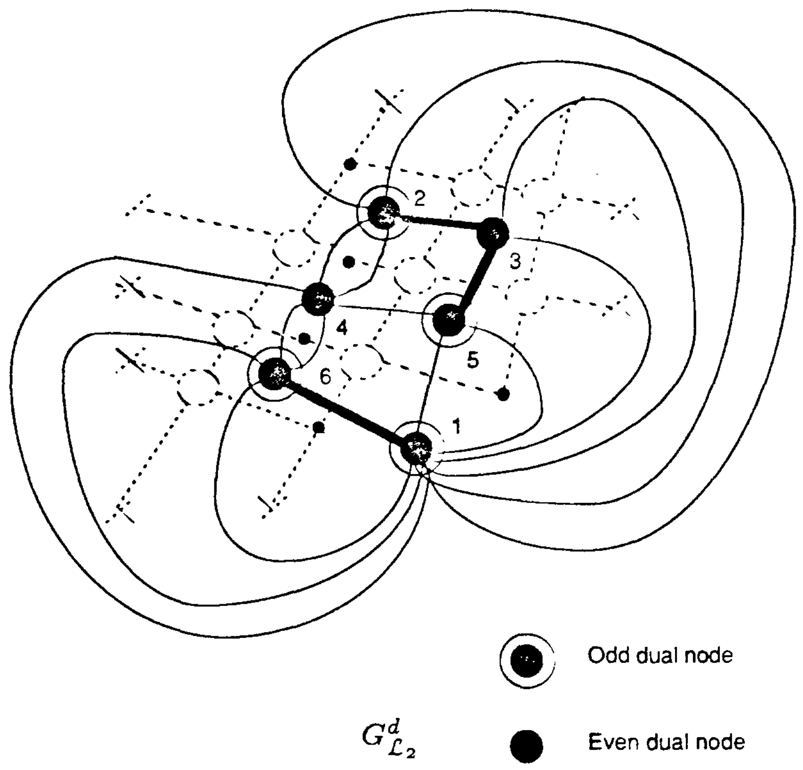


Figure 3.5. An MONP for $G^d_{\mathcal{L}_2}$

them, the new node is an odd node if and only if an odd and an even node are merged. The problem of finding an MOFC for $\overline{G_{\mathcal{L}}}$ can now be solved by finding a minimum cardinality set of edges in $G^d_{\mathcal{L}}$ whose contraction leaves $G^d_{\mathcal{L}}$ free of odd nodes. We call such a set of edges a *Minimum Odd Node Pairing* (MONP). Figure 3.5 shows the MONP for $G^d_{\mathcal{L}_2}$ corresponding to the MOFC for $\overline{G_{\mathcal{L}_2}}$ shown in Figure 3.4. It has been shown that an MONP is made up of a set of edge disjoint paths between pairs of odd nodes with each node being the endpoint of exactly one such path [16]. Because of this property, an MONP can be obtained by finding a maximum weight matching on the complete graph $M = (N_{odd}, E_M)$, where N_{odd} is the set of odd nodes in $G^d_{\mathcal{L}}$ and $E_M = \{(x,y) \,|\, x,\ y \in N_{odd} \text{ and } x \neq y\}$.

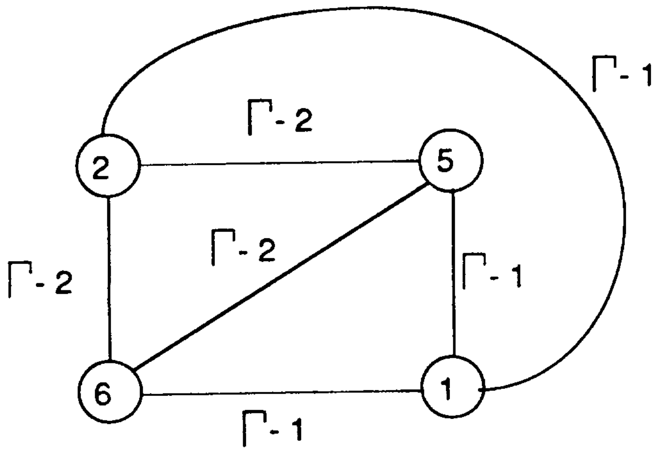


Figure 3.6. The graph M for $G^d_{\mathcal{L}_2}$

For each pair of nodes x, $y \in N_{odd}$, we define a path p_{x-y} between x and y such that

$$p_{x-y} = \{(r,s) \mid (r,s) \text{ is an edge along the shortest path between } x \text{ and } y \text{ in } G^d_{\mathcal{L}} \}.$$

We also define a set of via candidates

$$\mathcal{V}_{x-y} = \{VIA(\rho(r,s)) \mid (r,s) \in p_{x-y}\}.$$

For each edge in M we assign a weight $WEIGHT(x,y) = \Gamma - |\mathcal{V}_{x-y}|$ where Γ is some large constant. Figure 3.6 shows the graph M for $G^d_{\mathcal{L}_2}$. A maximum weight matching for M is a set of edges $X \subseteq E_M$ such that no node is the endpoint of more than one edge in X and

$$\sum_{(x,y)\in X} WEIGHT(x,y)$$

is a maximum. Since M is complete and $|N_{odd}|$ is always even, each node in M is the endpoint of exactly one edge in X. The maximum weight matching problem can be solved using Edmonds' algorithm [8] which can be implemented in $O(|N_{odd}|^3)$ [12,22]. Once X has been found, an MONP is uniquely determined by

$$P = \bigcup_{(x,y)\in X} p_{x-y}$$

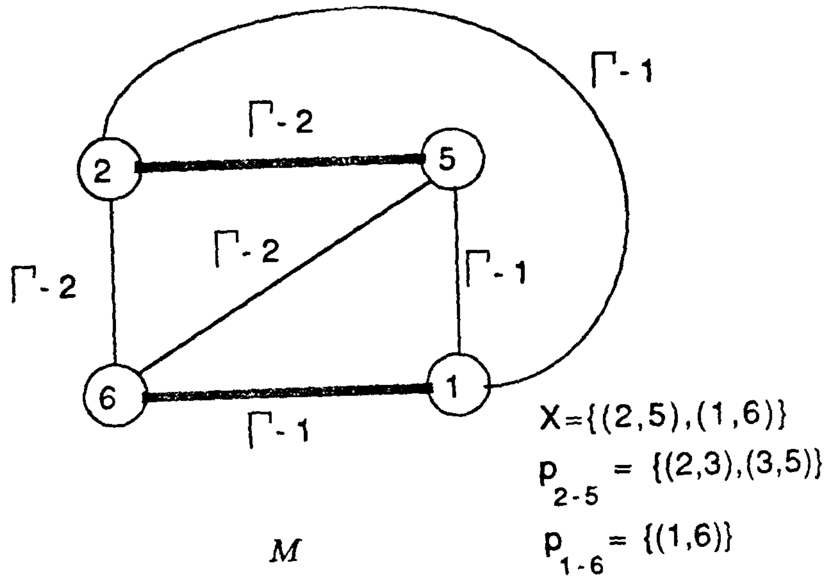
and a minimum cardinality set of vias is given by

$$\mathcal{V}' = \bigcup_{(x,y)\in X} \mathcal{V}_{x-y}.$$

Figure 3.7a shows a maximum weight matching for the graph M, shown in Figure 3.6, which corresponds to the MONP for $G^d_{\mathcal{L}_2}$, shown in Figure 3.5, and the MOFC for $\overline{G_{\mathcal{L}_2}}$ shown in Figure 3.4. The remaining problem is to find a layer assignment α' for $\mathcal{L}' = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}', \mathcal{V}', \alpha')$ given $\mathcal{R}'$ and $\mathcal{V}'$. This can easily be accomplished in linear time with respect to the number of routing subsegments. Figure 3.7b shows a solution layout $\mathcal{L}'_2$ resulting from the maximum matching in Figure 3.7a. Summarizing our discussion, we have the following algorithm:

Algorithm 1 - The Basic 2-CVM$_3$ Algorithm

Step 1: Generate the graph $G_{\mathcal{L}} = (N, E)$. Label each edge $(x,y) \in E$ with a pointer $VIA(x,y)$ to a via site on $SEG(x,y)$.



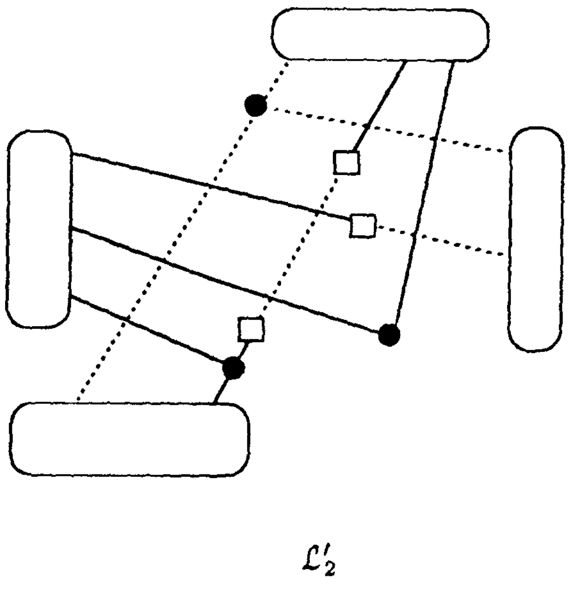


Figure 3.7 A maximum weight matching for the graph M and the resulting layout $\mathcal{L}'_3$

Step 2: Find an MOFC P for $G_{\mathcal{L}}$.

a: Generate the dual graph $G^d_{\mathcal{L}} = (N^d, E^d)$. Identify the set of odd nodes $N_{odd} \subseteq N^d$.

b: Generate the complete graph $M = (N_{odd}, E_M)$ where

$$E_M = \{(x,y) \mid x,\ y \in N_{odd} \textit{ and } x \neq y\}.$$

Assign a weight $WEIGHT(x,y)$ to each edge $(x,y) \in E_M$ such that

$$WEIGHT(x,y) = \Gamma - |\mathcal{V}_{x-y}|$$

where Γ is some very large constant and $\mathcal{V}_{x-y}$ is the set of vias associated with merging the two odd faces corresponding to x and y.

c: Find a maximum weight matching X on M. This gives an MONP P for $G'^d_{\mathcal{L}}$ which is given by:

$$P = \bigcup_{(x,y)\in X} p_{x-y}$$

and an MOFC P' for $\overline{G'_{\mathcal{L}}}$ given by

$$P' = \{\rho(x,y) \mid (x,y) \in P\}.$$

Step 3: Determine a minimum cardinality set of vias from

$$\mathcal{V}' = \bigcup_{(x,y)\in P'} VIA(x,y)$$

or directly from

$$\mathcal{V}' = \bigcup_{(x,y)\in X} \mathcal{V}_{x-y}.$$

Generate the set of routing segments $\mathcal{R}'$ from $\mathcal{R}$ by introducing a junction at each site in $\mathcal{V}'$.

Step 4: Find a valid layer assignment α' for the layout $\mathcal{L}' = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}', \mathcal{V}', \alpha')$.

■

The following theorem follows directly from our discussion:

Theorem 3.1. *Algorithm 1 finds an optimum solution for the 2-CVM$_3$ problem.*

■

2.2. Complexity

All of the steps in Algorithm 1 can be carried out in polynomial time. The first step requires locating all of the crossings between routing segments and can easily be implemented in $O(|\mathcal{R}|^2)$ time where $\mathcal{R}$ is the set of routing segments in $\mathcal{L}$. In Step 2a, the dual graph can be found in linear time with respect to the number of edges in $G_{\mathcal{L}}$ which is at most $O(|\mathcal{R}|^2)$. Step 3 is linear in time with respect to the number of vias determined by the algorithm, which cannot exceed the number of junctions in the given layout and is thus bounded by $|\mathcal{R}|$. In Step 4, layer assignment can be carried out in $O(|\mathcal{R}|^2)$ using a Breadth-First-Search [10] type approach. The only steps that are computationally more difficult are Steps 2b and 2c.

The computation of edge weights in Step 2b, requires finding a shortest path from each odd node in $G^d_{\mathcal{L}}$ to all other odd nodes in $G^d_{\mathcal{L}}$. This can be done using Breadth-First-Search [10] in $O(|E^d| \cdot |N_{odd}|)$ time where E^d and N_{odd} are the sets of edges and odd nodes in $G^d_{\mathcal{L}}$, respectively, and $|E^d| = O(|\mathcal{R}|^2)$.

The second computationally expensive step is finding a maximum weight matching on the graph M. Several authors [12,22] have shown that Edmonds' algorithm can be implemented in as little as $O(n^3)$ time †, where in this case $n = |N_{odd}|$.

Since, at first glance, it may appear that $|N_{odd}|$ is only bounded by the number of faces in $G_{\mathcal{L}}$, it would seem that increasing the number of edges in $G_{\mathcal{L}}$ from $O(|\mathcal{R}|)$ to $O(|\mathcal{R}|^2)$ by considering all possible via sites, instead of only one per segment as other authors have done [4,17,20], would increase the complexity of Steps 2b and 2c to $O(|\mathcal{R}|^4)$ and $O(|\mathcal{R}|^6)$, respectively. However, this is not the case.

Lemma 3.3. *For any valid topological layout* $\mathcal{L} = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}, \alpha)$, $|N_{odd}| \leq 2 \cdot |\mathcal{R}|$.

Proof: Let $\mathcal{V}'$ be the set of vias found by Algorithm 1. Since each path between two odd nodes must have a length of at least one and all such paths are disjoint, $|\mathcal{V}'| \geq |N_{odd}|/2$. Since Algorithm 1 finds the minimum number of vias considering all possible sites, $|\mathcal{V}'| \leq |\mathcal{V}|$ for any layout $\mathcal{L}$ and hence $|N_{odd}| \leq 2 \cdot |\mathcal{V}|$. Since any via is, by definition, the endpoint of at least two routing segments, $|\mathcal{V}| \leq |\mathcal{R}|$ and thus $|N_{odd}| \leq 2 \cdot |\mathcal{R}|$. ∎

† Note that Pinter [27] incorrectly states that an $O(n^{2.5})$ maximum weight matching algorithm has been reported by Micali and Vazirani [25]. That algorithm is in fact a maximum cardinality matching algorithm.

Thus, we have the following theorem:

Theorem 3.2. *Algorithm 1 generates an optimum solution for the 2-CVM$_3$ problem with input layout $(\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}, \alpha)$ in $O(|\mathcal{R}|^3)$ time.*

■

3. The 2-CVM$_3$ Problem for Geometric Layouts

3.1. Application of the basic algorithm

Algorithm 1 can be applied to geometric layouts with only one slight modification. In the geometric case, there are likely to be routing subsegments along which there is no allowable via site. To handle this situation, we define $VIA(x,y) = \Lambda$ if there is no allowable via site on $SEG(x,y)$. The definition of p_{x-y} must also be modified as follows to reflect this change:

$$p_{x-y} = \big\{(r,s) \bigm| (r,s) \text{ is an edge along the shortest path between } x \text{ and } y \text{ in } G^d_{\mathcal{L}} \text{ not containing any edges } (r',s') \text{ for which } VIA(\rho(r',s')) = \Lambda\big\}.$$

This is necessary to assure that all of the edges in

$$P = \bigcup_{(x,y)\in X} p_{x-y}$$

actually correspond to allowable via sites. If all of the edges $(x,y) \in E^d$ for which $VIA(\rho(x,y)) = \Lambda$ are deleted, then finding an MONP, say P, on $G^d_{\mathcal{L}}$ yields an optimum 2-CVM$_3$ solution for $\mathcal{L}$ given by $\{VIA(\rho(x,y)) \,|\, (x,y) \in P\}$. A solution always exists under any set of design rules which are met by the input layout $\mathcal{L}$.

3.2. Improved 2-CVM$_3$ algorithm

Algorithm 1 produces optimum results and is very efficient when considering via candidates on all or most routing segments. However, it is not very well suited for dense geometric layouts where the number of allowable via candidates is much smaller than $|\mathcal{R}|^2$. For that case we will extend the basic approach to take advantage of the limited number of via sites. To simplify our discussion and illustrations, we assume that input layouts are grid based and that vias are allowed at any intersection of two grid lines where a crossing is not located.

A *cluster* is a set of crossings in $\mathcal{L}$ which are mutually connected by routing subsegments not containing any via sites. The assignment of any one routing subsegment which has an endpoint in a particular cluster to a particular layer determines the layer assignment of the remaining routing subsegments which have at least one endpoint in that cluster. A cluster which is not included in any other cluster will be called a *maximal cluster.* In the case of topological layouts, each maximal cluster will consist of a single crossing since every routing subsegment contains a possible via site. In the case of geometric layouts, however, an entire layout may be composed of only a few maximal clusters. In that case, a significant speedup can be attained by identifying those clusters. Figure 3.8 shows a grid based layout $\mathcal{L}_3$. The hashed curves show the borders of all maximal clusters. Note that our definition of clusters differs from that of Chen, et al. [4] because of our less restrictive problem definition.

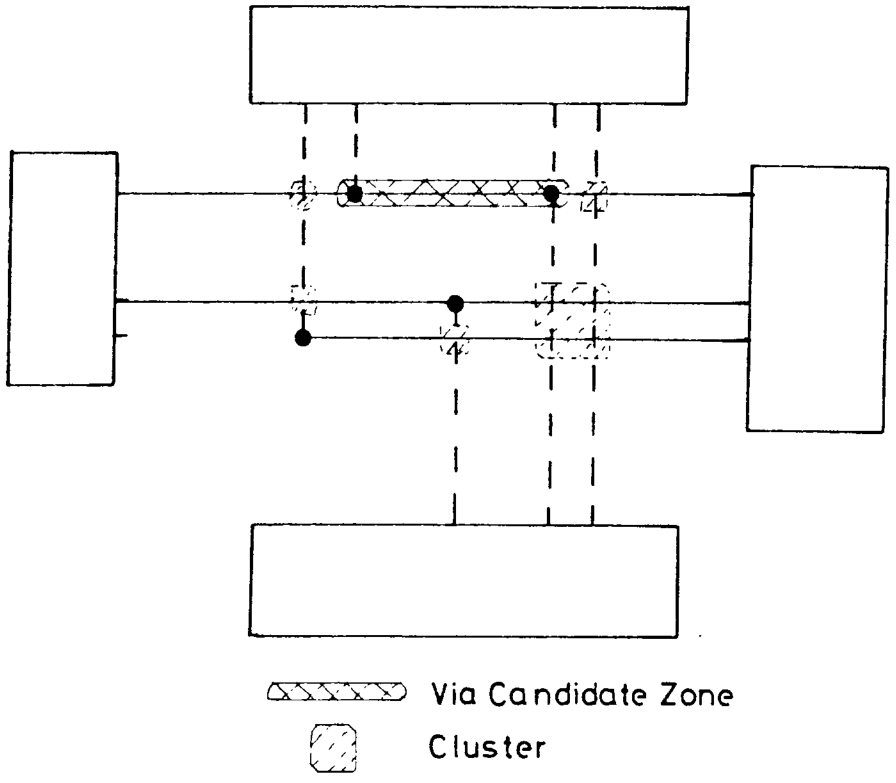


Figure 3.8. Clusters and via candidate zones in layout $\mathcal{L}_3$

A group of one or more via candidates in $\mathcal{L}$ which are mutually connected by routing subsegments which do not contain any crossings will be called a *via candidate zone.* As we have defined it, a via candidate zone may consist of only a single via candidate. We say that a via candidate zone z is a *maximal via candidate zone* if there is no other via candidate zone which includes all of the via candidates in z. Chen, et al. [4] suggested that at most one via is ever needed in a via candidate zone if all of the routing subsegments in the zone can be assigned to both layers. If this is allowed, the minimum number of vias required for a particular layout may be reduced. Some slight speedup may also be obtained since fewer possible via sites must be considered by the algorithm. Figure 3.8 shows all via candidate zones in $\mathcal{L}_3$ consisting of more than one via candidate.

Since merging via candidates into via candidate zones can change the number of vias in an optimum CVM solution and may require assigning some routing segments to both layers, we define a slightly different via minimization problem which allows both of these things. The new class of problems are denoted by ℓ-CVM$_k^*$ or simply ℓ-CVM* when $k = \infty$. The ℓ-CVM$_k^*$ problem is defined for positive integers l and k as follows:

(ℓ-CVM$_k^*$) Given an ℓ-layer layout $\mathcal{L} = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}, \alpha)$with no junction having degree greater than k, generate an ℓ-layer layout $\mathcal{L}' = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}', \mathcal{V}', \beta)$ such that $\mathcal{R} \approx \mathcal{R}'$ and $\mathcal{V}'$ is a minimum cardinality set of vias,

where β is a one-to-many mapping $\beta \colon \mathcal{R} \rightarrow \{1, 2, \ldots, \ell\}$. It should be pointed out that an optimum solution to the 2-CVM$_3^*$ problem may require fewer vias than an optimum solution to the 2-CVM$_3$ problem, but will never require more.

In order to modify Algorithm 1 to take advantage of clusters and via candidate zones, we need to change the definition of $G_{\mathcal{L}}$ slightly. We add a label denoted by $TYPE(x,y)$ to each edge (x,y) in $G_{\mathcal{L}}$ where $TYPE(x,y)$ is given by the layer assignment of the routing segment which included $SEG(x,y)$ in $\mathcal{L}$. We also define a graph $G'_{\mathcal{L}} = (N', E')$ based on the layout $\mathcal{L}$ as follows:

$$N' = C' \cup J'$$

where

$$C' = \{c_x \mid c_x \text{ corresponds to a unique maximal cluster in } \mathcal{L}\}$$

$$J' = \{j_x \mid j_x \text{ corresponds to a unique maximal via candidate zone in } \mathcal{L}\}$$

and

$$E' = \{(c_x, j_y) \mid c_x \in C',\ j_y \in J',\ \text{and there is a unique subsegment } SEG(c_x, j_y) \text{ in } \mathcal{L} \text{ with endpoints in the cluster corresponding to } c_x \text{ and the via candidate zone corresponding to } j_y\}.$$

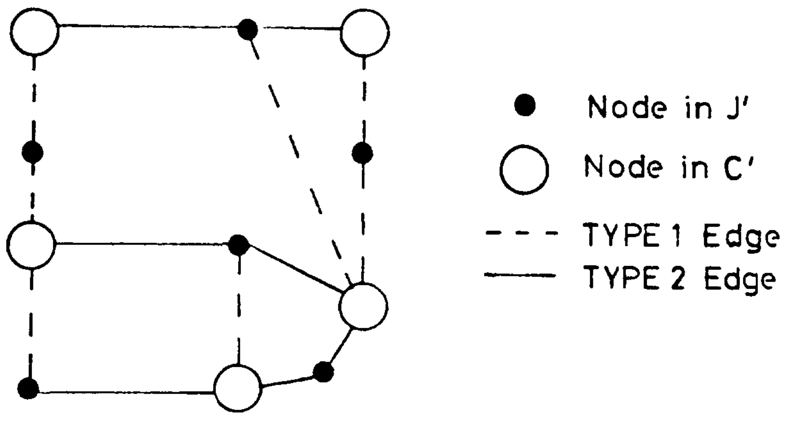


Figure 3.9. The graph $G'_{\mathcal{L}_3}$

Figure 3.9 shows the graph $G'_{\mathcal{L}_3}$. It is easy to see that $G'_{\mathcal{L}}$ is always planar and is bipartite such that each edge $(c, j) \in E'$ has endpoints such that $c \in C'$ and $j \in J'$. We specify that for each edge $(c, j) \in E'$

$$VIA(c, j) = SITE(j)$$

where $SITE(j)$ now corresponds to some junction in the via candidate zone corresponding to j. Likewise $SITE(c)$ for any $c \in C'$ corresponds to a particular crossing in the cluster corresponding to c. Note that there may be two edges (c_1, j) and (c_2, j) in E' such that $VIA(c_1, j) = VIA(c_2, j)$. However, there is a one-to-one correspondence between the nodes in J' and the maximal via candidate zones in $\mathcal{L}$. $G'_{\mathcal{L}} = (N', E')$ can be derived from $G_{\mathcal{L}} = (N, E)$ as follows:

```
procedure main
    $J' = J$
    $C' = C$
    $E' = E$
    for each edge $(x, y) \in E'$
      if $x \in J'$ and $y \in J'$ then contract $(x, y)$
      if $x \in T$ then $E' = E' - (x, y)$
      if $x \in C'$ and $y \in C'$ then
        if $VIA(x, y) = \Lambda$ then
          contract $(x, y)$
        else
          $E' = E' - \{(x, y)\} \cup \{(x, w), (w, y)\}$
            where $w$ is a new node and $SITE(w)$
            is a legal via site along $SEG(x, y)$
          $J' = J' \cup \{w\}$
    $N' = C' \cup J'$
end
```

The procedure **contract** (x, y) is defined for an ordered pair of nodes x and y in $G_{\mathcal{L}}$ as follows:

```
contract $(x, y)$
  for each edge $(a, y) \in E'$
    $E' = E' - \{(a, y)\} \cup \{(a, x)\}$
  if $y \in J'$ then $J' = J' - \{y\}$
    else if $y \in C'$ then $C' = C' - \{y\}$
return
```

Since each node in C' no longer corresponds to a single node as it did in C, we must reconsider the conditions for a valid layer assignment.

Lemma 3.4. *For any two edges $(c, j_1), (c, j_2) \in E'$ such that $c \in C'$ and $j_1, j_2 \in J'$, $SEG(c, j_1)$ should be assigned to the same layer as $SEG(c, j_2)$ if and only if $TYPE(c, j_1) = TYPE(c, j_2)$.*

Proof: Since (c, j_1) and (c, j_2) are both incident on c, $SEG(c, j_1)$ and $SEG(c, j_2)$ are incident upon the same maximal cluster. If $TYPE(c, j_1) = TYPE(c, j_2)$, they were assigned to the same layer in $\mathcal{L}$. Thus, they must be assigned to the same layer in any layout. Likewise, if $TYPE(c, j_1) \neq TYPE(c, j_2)$, then $SEG(c, j_1)$ and $SEG(c, j_2)$ were assigned to different layers in $\mathcal{L}$. Thus, they must be assigned to different layers in any layout. ∎

We say a cycle $(n_{i_0}, n_{i_1}), (n_{i_1}, n_{i_2}), \ldots, (n_{i_{m-1}}, n_{i_0})$ in $G'_{\mathcal{L}}$ is *odd* if

$$\left|\left\{n_{i_k} \in C' \;\middle|\; TYPE(n_{i_{k-1}}, n_{i_k}) \neq TYPE(n_{i_k}, n_{i_{k+1}}),\ 0 \leq k \leq m-1\right\}\right|$$

is odd; otherwise it is *even*. We say a face f in $\overline{G'_{\mathcal{L}}}$ is *odd* if and only if the fundamental cycle in $G'_{\mathcal{L}}$ which defines that face is odd; otherwise the face is *even*.

Lemma 3.5. *Given a valid geometric layout* $\mathcal{L} = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}, \alpha)$, *there exists a valid geometric layout* $\mathcal{L}' = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}', \alpha')$ *such that* $\mathcal{V}' = \phi$ *if and only if* $G'_{\mathcal{L}}$ *is free of odd cycles.*

Proof: Assume that there exists a valid geometric layout $\mathcal{L}' = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}', \alpha')$ such that $\mathcal{V}' = \phi$. Consider some cycle $(n_{i_0}, n_{i_1}), (n_{i_1}, n_{i_2}), \ldots, (n_{i_{m-1}}, n_{i_0})$ in $G'_{\mathcal{L}}$. For each pair of edges $(n_{i_{k-1}}, n_{i_k})$ and $(n_{i_k}, n_{i_{k+1}})$ such that $n_{i_k} \in C'$, $SEG(n_{i_{k-1}}, n_{i_k})$ and $SEG(n_{i_k}, n_{i_{k+1}})$ must be assigned to different layers if and only if $TYPE(n_{i_{k-1}}, n_{i_k}) \neq TYPE(n_{i_k}, n_{i_{k+1}})$ by Lemma 3.4. For each pair of edges $(n_{i_{k-1}}, n_{i_k})$ and $(n_{i_k}, n_{i_{k+1}})$ such that $n_{i_k} \in J'$, $SEG(n_{i_{k-1}}, n_{i_k})$ and $SEG(n_{i_k}, n_{i_{k+1}})$ meet at a via candidate zone in $\mathcal{L}$. Since there are no vias in $\mathcal{L}'$, $SEG(n_{i_{k-1}}, n_{i_k})$ and $SEG(n_{i_k}, n_{i_{k+1}})$ must be assigned to the same layer. Thus, layer changes in $\mathcal{L}$ occur only when $n_{i_k} \in C'$ and $TYPE(n_{i_{k-1}}, n_{i_k}) \neq TYPE(n_{i_k}, n_{i_{k+1}})$. Since the number of layer changes in $\mathcal{L}$ must be even around any cycle, all cycles in $G'_{\mathcal{L}}$ are even. Conversely, if we assume that all cycles in $G'_{\mathcal{L}}$ are even, then all layer changes can take place when $n_{i_k} \in C'$ and $TYPE(n_{i_{k-1}}, n_{i_k}) \neq TYPE(n_{i_k}, n_{i_{k+1}})$, and there exists a valid geometric layout $\mathcal{L}' = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}', \alpha')$ such that $\mathcal{V}' = \phi$. ■

If we insert a via v at a particular junction $SITE(j)$, then we can delete j from $G'_{\mathcal{L}}$ since there is no longer a requirement that the routing segments which share an endpoint at v be assigned to the same layer. A set of nodes whose deletion leaves $G'_{\mathcal{L}}$ free of odd cycles is called an *Odd Cycle Node Cover* (OCNC). An OCNC of minimum cardinality will be called a *Minimum Odd Cycle Node Cover*(MOCNC). Since there is a one-to-one correspondence between the nodes in J' and the via candidate zones in $\mathcal{L}$, we have the following lemma:

Lemma 3.6. *Given some valid geometric layout $\mathcal{L} = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}, \alpha)$, if Q is an MOCNC for $G'_{\mathcal{L}}$, then there exists a valid geometric layout $\mathcal{L}' = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}', \mathcal{V}', \beta)$ such that $\beta: \mathcal{R}' \rightarrow \{1, 2, both\}$, $\mathcal{R}' \approx \mathcal{R}$ and $\mathcal{V}' = \{SITE(j) \mid j \in Q\}$ is of minimum cardinality.*

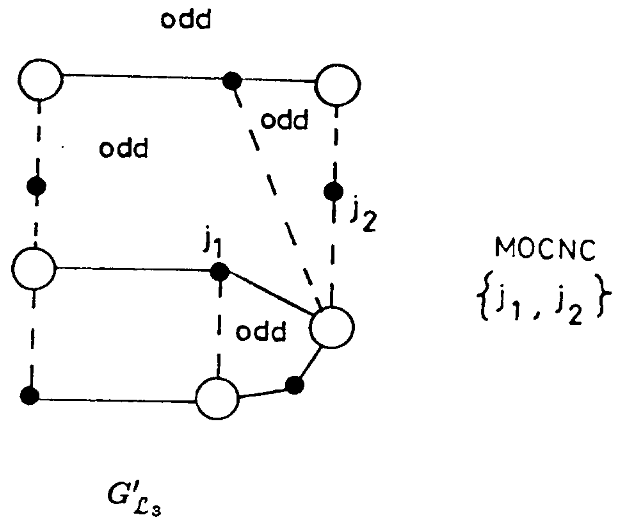
■

Figure 3.10 shows an MONC for the graph $G'_{\mathcal{L}_3}$ and the corresponding solution layout $\mathcal{L}'$.

In general, the problem of deleting the minimum number of nodes which leave a graph free of odd cycles is NP-hard even when restricted to planar [23] or bipartite graphs [31]. However, if we delete nodes only from J' and no node in J' has a degree greater than three, then the problem is solvable in polynomial time as stated in the following lemma:

Lemma 3.7. *If P is an MOFC for $G'_{\mathcal{L}}$ and no node in J' has degree exceeding three, then $Q = \{j \mid (c, j) \in P\}$ is an MOCNC for $G'_{\mathcal{L}}$.*

Proof: As shown earlier, it is never necessary to locate a via on a junction of degree less than or equal to three since that junction could always be moved



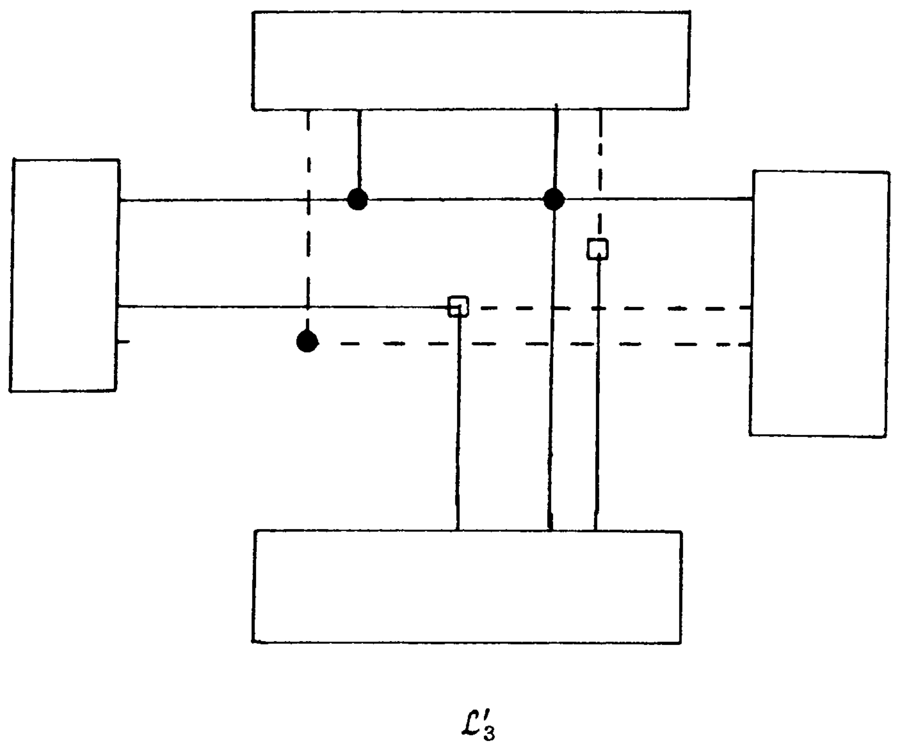


Figure 3.10. An MOCNC for the graph $G'_{\mathcal{L}_3}$ and the layout $\mathcal{L}'_3$

slightly onto one of the routing subsegments which shares an endpoint with that junction. The same is true of locating a via in a via candidate zone z when there are three or less routing subsegments incident upon z. Thus, a minimum cardinality set of vias located at sites in via candidate zones, such as $SITE(j)$, corresponding to an MOCNC for $G'_{\mathcal{L}}$, can always be replaced by a set of vias located on routing subsegments, such as $SEG(c, j)$, which correspond to an MOFC for $\overline{G'_{\mathcal{L}}}$. Clearly, the converse is also true. Therefore, if P is an MOFC for $G'_{\mathcal{L}}$ and no node in J' has degree exceeding three, then $Q = \{j \mid (c, j) \in P\}$ is an MOCNC for $G'_{\mathcal{L}}$. Lemma 3.9 given in the following section can also be used to prove this lemma in a more direct manner. ∎

If we assume that all of the nodes in J' have degree three or less, then by Lemma 3.7 we can find an MOCNC in polynomial time and obtain an optimum 2-CVM$_3^*$ solution. Given a layout $\mathcal{L} = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}, \alpha)$, an outline of the algorithm would be as follows:

Algorithm 2 - The Improved Algorithm

Step 1: Generate the graph $G_{\mathcal{L}} = (N, E)$. Label each edge (x, y) with a pointer $VIA(x, y)$ to a via site on $SEG(x, y)$. If none exists, $VIA(x, y) = \Lambda$.

Step 2: Generate the graph $G'_{\mathcal{L}} = (N', E')$ from $G_{\mathcal{L}} = (N, E)$. $\mathcal{R}'$ is obtained from $\mathcal{R}$ by adding the junctions $\{SITE(j) \mid j \in J' - J\}$.

Step 3: Compute an MOCNC Q.

a: Generate the dual graph $G^d_{\mathcal{L}} = (N^d, E^d)$. Identify the set of odd nodes $N_{odd} \subseteq N^d$.

b: Generate the complete graph $M = (N_{odd}, E_M)$. Compute the weights for the edges in E_M as described.

c: Find a maximum weight matching X on M. This gives an MONP P for $G'^{d}_{\mathcal{L}}$ which is given by:

$$P = \bigcup_{(x,y)\in X} p_{x-y}$$

and an MOFC P' for $\overline{G'_{\mathcal{L}}}$ given by

$$P' = \{\rho(x,y) \mid (x,y) \in P\}$$

and an MOCNC

$$Q = \{j \in J' \mid (c,j) \in P'\}.$$

Step 4: Mark the vias given by $\mathcal{V}' = \{SITE(j) \mid j \in Q\}$. Note that $SITE(j)$ may be only one of several via candidates in a via candidate zone.

Step 5: Find a layer assignment β for the layout $\mathcal{L}' = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}', \mathcal{V}', \beta)$. All routing subsegments contained in a via candidate zone which has been assigned a via are assigned to both layers. ∎

This algorithm has only one more step than Algorithm 1, namely Step 2. Step 2 can easily be accomplished in $O(|E|)$ time. Thus, the total time required for Algorithm 2 is still $O(|\mathcal{R}|^3)$ although the average time will be much less as demonstrated in Chapter IV.

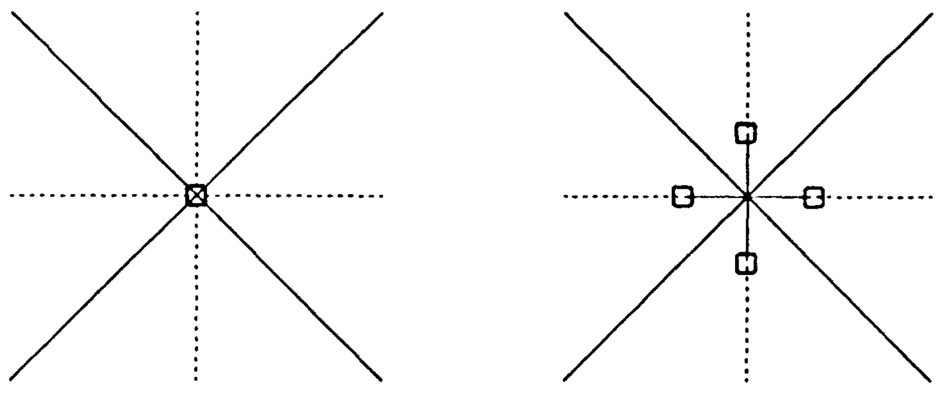


Figure 3.11. Equivalent via sites.

It is important to recall that although the maximum degree of nodes in J was three, the maximum degree of the nodes in J' may be more. If that is the case, then an MOFC may not lead to an MOCNC for $G'_{\mathcal{L}}$. In terms of the 2-CVM problem, this is because a via placed at a junction cannot always be replaced by a single via not on the junction. In fact, a via at a junction with degree d may be equivalent to as many as $d/2$ vias placed on routing subsegments which meet at that junction as shown in Figure 3.11. In terms of the graph $G'_{\mathcal{L}}$, an MOFC P_1 will lead to a less than optimum MOCNC whenever there exists another MOFC P_2 such that $|P_1| = |P_2|$ and

$$\left|\left\{j \mid (c,j) \in P_1\right\}\right| > \left|\left\{j \mid (c,j) \in P_2\right\}\right|$$

Thus, the solution generated by Algorithm 2 may not be optimal for the 2-CVM$_3^*$ problem, however:

Lemma 3.8. *The number of vias produced by Algorithm 2 for the 2-CVM$_3^*$ problem on some layout $\mathcal{L}$ will be no greater the number of vias produced by Algorithm 1 for the same layout.*

Proof: An MOFC for $\overline{G_{\mathcal{L}}}$ will have the same cardinality as an MOFC for $\overline{G'_{\mathcal{L}}}$ since both give a minimum cardinality set of vias for $\mathcal{L}'$ without locating any vias at junctions. An OCNC for $G'_{\mathcal{L}}$ may have smaller cardinality than an MOFC for $\overline{G'_{\mathcal{L}}}$ since one via placed at a junction may be equivalent to several on routing segments, but never greater cardinality. Thus, Algorithm 2 will produce at most the same number of vias as Algorithm 1 which does not consider via candidates. ■

Thus, since we have shown Algorithm 1 to be optimal, we have:

Theorem 3.3. *Algorithm 2 produces optimum results for the 2-CVM$_3$ problem with input layout $\mathcal{L} = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}, \alpha)$ in $O(|\mathcal{R}|^3)$ time.*

■

3.3. Essential vias

In the previous section, we showed that solving the 2-CVM$_3$ problem required breaking all odd cycles and gave an algorithm for doing so. However, in geometric layouts there may exist some odd cycles which pass through only one node $n_{ess} \in J'$ as shown in Figure 3.12. In this case, we call $SITE(n_{ess})$ an *essential via* since deleting n_{ess} is the only way to break that cycle. Chen, et al. [4] first suggested the idea of essential vias under their more restricted model. Later, Chang and Du [3] suggested a heuristic

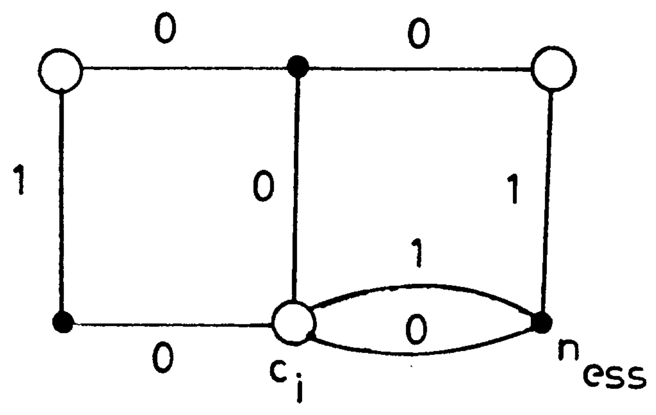


Figure 3.12. Essential Vias

algorithm which locates a restricted subset of all essential vias. Using our formulation, we can locate all essential vias by scanning the edge list of each node in J' to see if there are two parallel edges of different types. If there are, then each essential via is added to the set of vias and all of the edges incident upon n_{ess} deleted. It will be shown in the next chapter that essential vias account the vast majority of the necessary vias in certain types of layouts. In such cases, this step greatly reduces necessary computation time without changing the results. Given a layout $\mathcal{L} = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}, \mathcal{V}, \alpha)$, we modify our algorithm as follows:

Algorithm 2A

Step 1: Generate the graph $G_{\mathcal{L}} = (N, E)$. Label each edge (x, y) with a pointer $VIA(x, y)$ to a via site on $SEG(x, y)$. If none exists, $VIA(x, y) = \Lambda$.

Step 2: Generate the graph $G'_{\mathcal{L}} = (N', E')$ from $G_{\mathcal{L}} = (N, E)$. $\mathcal{R}'$ is obtained from $\mathcal{R}$ by adding the junctions in $\{SITE(j) \mid j \in J' - J\}$.

Step 2a Locate all essential vias in $\mathcal{V}_{ess}$ and delete the corresponding nodes and edges from $G'_{\mathcal{L}}$.

Step 3: Compute an MOCNC Q using the same method as in Algorithm 2

Step 4: Mark the vias given by $\mathcal{V}' = \{SITE(j) \mid j \in Q\}$.

Step 5: Find a layer assignment β for the layout $\mathcal{L}' = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}', \mathcal{V}' \cup \mathcal{V}_{ess}, \beta)$.

Again, the additional procedure, Step 2a, can be accomplished in $O(|E'|)$ time and hence the overall time complexity of the algorithm remains $O(|\mathcal{R}|^3)$.

4. The 2-CVM Problem

The algorithms described in the previous two sections always give optimum results for the 2-CVM$_3$ problem. In fact, Algorithm 2A gives a good first order approximation for the 2-CVM problem. There are several things which we can do, however, to improve on that approximation. Using our maximum matching approach, we choose a set of edges which lie on disjoint shortest paths between pairs of odd nodes in the dual graph $G'^d_{\mathcal{L}}$. When there are junctions with degree greater than three, it is possible to have two edges (n_{i_1}, n_{i_2}) and (n_{i_2}, n_{i_3}) along the same shortest path such that $VIA(n_{i_1}, n_{i_2}) = VIA(n_{i_2}, n_{i_3})$. Thus the cardinality of p_{x-y} might be greater than that of $\mathcal{V}_{x-y}$ for some shortest paths between x and y in $G'^d_{\mathcal{L}}$. Since we choose p_{x-y} as a path containing the fewest edges, we may not be getting the path which really corresponds to the fewest vias. To avoid this, we redefine p_{x-y} as the set of edges

$$p_{x-y} = \{(r,s) \mid (r,s) \text{ is an edge along a shortest path between } x \text{ and } y \text{ in } G^d_{\mathcal{L}} \}$$

where the length of a path is defined as the number of distinct via labels $VIA(\rho(r,s))$ on edges (r,s) making up the path. Although this assures that the shortest path corresponds to the fewest vias, it does not assure that the smallest sum of path lengths corresponds to the fewest vias. This is because the sets of via candidates $\mathcal{V}_{x_1-y_1}$ and $\mathcal{V}_{x_2-y_2}$ associated with two shortest path edge sets $p_{x_1-y_1}$ and $p_{x_2-y_2}$ may not be disjoint. Thus minimizing the sum of path lengths, may not always give us the minimum possible number of vias. To examine when this occurs, we consider the following lemma:

Lemma 3.9. *If S is an MONP for $G'^d_{\mathcal{L}}$ and $j \in J'$, then the maximum number of edges $(x,y) \in S$ for which $VIA(\rho(x,y)) = SITE(j)$ is $\lfloor deg(j)/2 \rfloor$, where $deg(j)$ denotes the degree of the node $j \in J'$ in $G'_{\mathcal{L}}$ and $\lfloor k \rfloor$ denotes the smallest integer not less than k.*

Proof: All of the edges $(x, y) \in E^d$ for which $VIA(\rho(x, y)) = SITE(j)$ form a fundamental cycle W in $G'^d_{\mathcal{L}}$ consisting of $deg(j)$ edges. Assume that there are two paths $p\langle a, b\rangle = [p\langle a, r\rangle, p\langle r, s\rangle, p\langle s, b\rangle]$ and $p\langle c, d\rangle = [p\langle c, t\rangle, p\langle t, u\rangle, p\langle u, c\rangle]$ whose edges are part of an MONP S where $p\langle a, b\rangle$ denotes some particular path between a and b. Furthermore assume that r, s, t, and u are all part of the cycle W. Thus, the shortest path length between any pair of them is defined to be one. We prove, by contradiction, that $s \neq t$.

Assume that $s = t$ as shown in Figure 3.13a. Notice that if we replace $p\langle r, s\rangle$ and $p\langle t, u\rangle$ with $p\langle r, u\rangle$, as shown in Figure 3.13b, we have a different pairing of the nodes a, b, c, and d given by the paths of edges $p\langle a, d\rangle = [p\langle a, r\rangle, p\langle r, u\rangle, p\langle u, d\rangle]$ and $p\langle b, c\rangle = [p\langle b, s\rangle, p\langle s, c\rangle]$. If we replaced the edges in S which occur in $p\langle a, d\rangle$ and $p\langle b, c\rangle$ with the edges in $p\langle a, b\rangle$ and $p\langle c, d\rangle$, then we would have an ONP for $G'^d_{\mathcal{L}}$ with cardinality of one less than S. Thus, $s \neq t$.

Since there are exactly $deg(j)$ faces in $\overline{G_{\mathcal{L}}}$ adjacent to node j, there can be at most $\lfloor deg(j)/2 \rfloor$ edges (x, y) in any MONP for $G'^d_{\mathcal{L}}$ for which $VIA(\rho(x, y)) = SITE(j)$. ∎

From Lemma 3.9 it is easy to see why the MONP for $G'^d_{\mathcal{L}}$ always leads to an MOCNC for $G'_{\mathcal{L}}$ whenever there are no junctions with degree greater than three. Since an MONP for $G'^d_{\mathcal{L}}$ gives an MOFC for $\overline{G'_{\mathcal{L}}}$, Lemma 3.7 must be true. When junction degree exceeds three, we would like to choose an MONP for $G'^d_{\mathcal{L}}$ in which as many edges as possible correspond to common via candidate zones. Since that would correspond to finding an MOCNC, we know that we cannot do it optimally. However, as a final refinement of

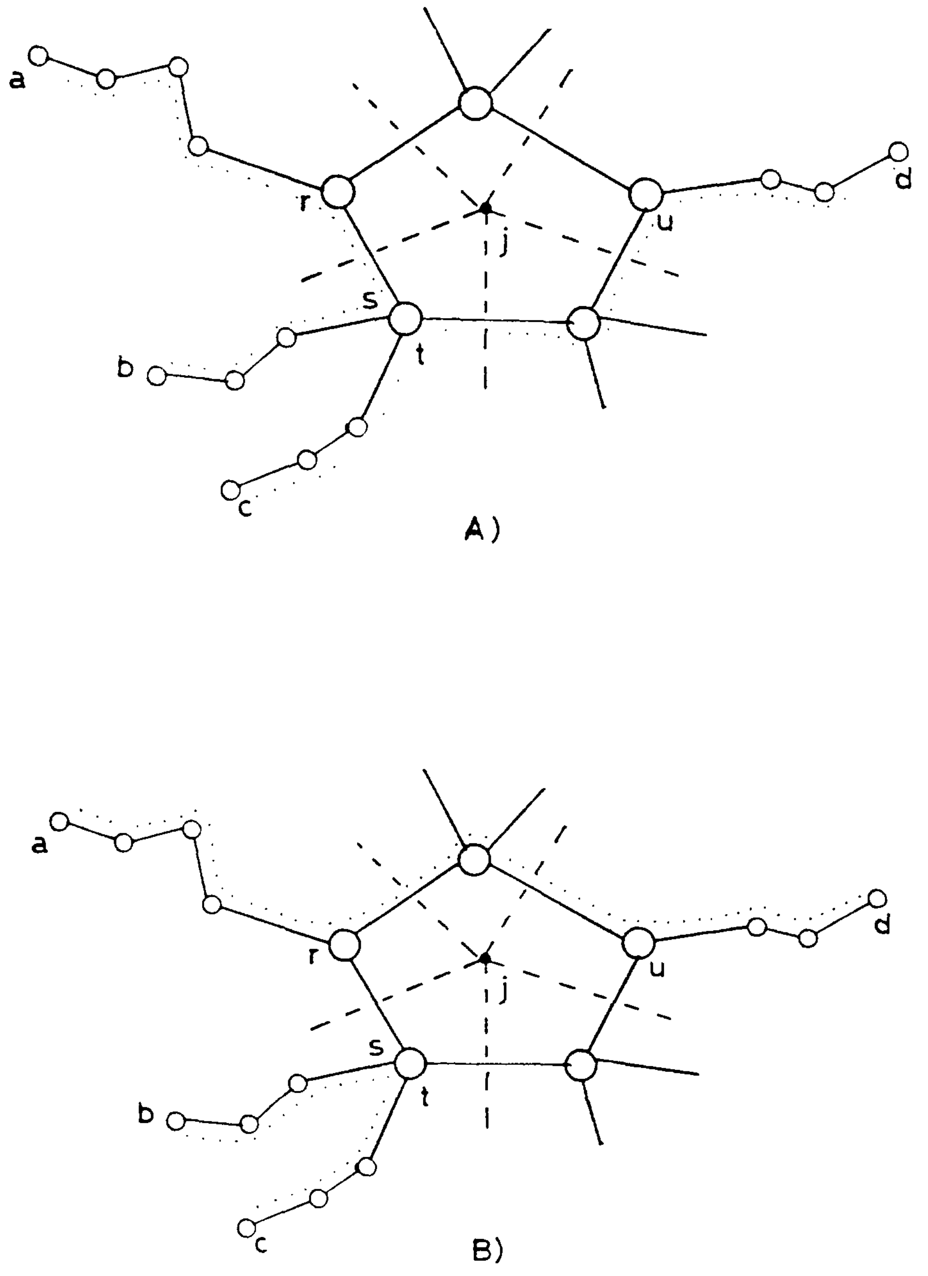


Figure 3.13. Paths in $G'_{\mathcal{L}}$

our approximation method, we will choose an MONP S which maximizes

$$\sum_{(x,y)\in S} deg(j) \text{ where } VIA(\rho(x,y)) = SITE(j).$$

This can be accomplished by adding a weight to each node $j \in J'$ defined by:

$$W(j) = 1.0 - \epsilon \cdot deg(j)$$

where ϵ is some very small constant. Among all possible sets of edges p_{x-y} in $G'^{d}_{\mathcal{L}}$, we choose one which minimizes

$$\sum_{SITE(j)\in \mathcal{V}_{x-y}} W(j)$$

and weights on the edges in M would be defined as

$$WEIGHT(x,y) = \Gamma - \sum_{SITE(j)\in \mathcal{V}_{x-y}} W(j)$$

By choosing such an MONC, we maximize the probability that the OCNC we derive be minimal without affecting the time complexity of the algorithm. Thus, we have our final algorithm which differs from Algorithm 2A in the extra bookkeeping that goes on in Step 4.

Algorithm 3 - The Most General Algorithm

Step 1: Generate the graph $G_{\mathcal{L}} = (N, E)$.

Step 2: Generate the graph $G'_{\mathcal{L}} = (N', E')$ from $G_{\mathcal{L}} = (N, E)$. Assign weights to nodes according to their degree.

Step 3: Locate all essential vias in $\mathcal{V}_{ess}$.

Step 4: Compute an MOCNC Q.

a: Generate the dual graph $G^{d}_{\mathcal{L}} = (N^{d}, E^{d})$.

b: Generate the complete graph $M = (N_{odd}, E_M)$. Find all sets of edges

$$p_{x-y} = \{(r,s) \mid (r,s) \text{ is an edge along a path between } x \text{ and } y \text{ in } G^d_{\mathcal{L}} \text{ containing the fewest distinct via labels.}\}$$

such that

$$\sum_{SITE(j) \in \mathcal{V}_{x-y}} W(j)$$

is minimized. Assign a weight

$$WEIGHT(x,y) = \Gamma - \sum_{SITE(j) \in \mathcal{V}_{x-y}} W(j)$$

to each edge $(x,y) \in E_M$.

c: Find a maximum weight matching X on M. This gives an MONP P for $G'^d_{\mathcal{L}}$ which maximizes

$$\sum_{(x,y) \in P} deg(j) \text{ where } VIA(\rho(x,y)) = SITE(j)$$

and an MOFC P' for $\overline{G'_{\mathcal{L}}}$ given by

$$P' = \{\rho(x,y) \mid (x,y) \in P\}$$

and an *approximate* MOCNC

$$Q = \{j \in J' \mid (c,j) \in P'\}.$$

Step 5: Mark the vias given by $\mathcal{V}' = \{SITE(j) \mid j \in Q\}$.

Step 6: Find a layer assignment β for the layout $\mathcal{L}' = (\mathcal{M}, \mathcal{T}, \mathcal{W}, \mathcal{R}', \mathcal{V}' \cup \mathcal{V}_{ess}, \beta)$.

The maximum amount that this algorithm may differ from the optimum solution is given by:

Theorem 3.4. *The solution obtained by Algorithm 3 does not differ from the optimum 2-CVM* solution by more than:*

$$\sum_{\substack{deg(j)>3 \\ j\in J}} \lfloor (deg(j)-2)/2 \rfloor .$$

Proof: It follows directly from Lemma 3.9 that this is the maximum difference between the cardinality of an MOFC for $\overline{G'_{\mathcal{L}}}$ and an MOCNC for $G'_{\mathcal{L}}$.

■

Since the number of via candidate zones with degree significantly greater than three is likely to be quite small, this approximate algorithm is likely to yield very good results.

5. The Curve Coloring Problem

We defined the input to the 2-CVM problem as a set of straight line segments called routing segments. This seemed logical since layouts are generally made up of straight line segments and in fact any layout has a piecewise linear approximation. However, it is of some theoretical interest to remove this restriction and consider the following curve coloring problem [21].

(CC) Given a set of curves, find the minimum number of cuts which must be made so that the remaining curves are two-colorable.

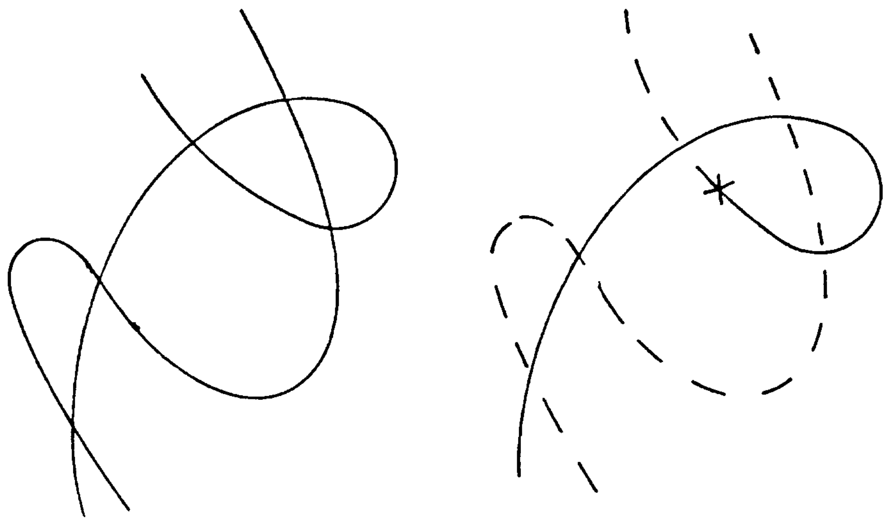


Figure 3.14. Curve coloring problem and optimum solution

In this context, we say that a family of curves are two colorable if and only if each continuous curve is assigned to one of two colors and no two curves assigned to the same color intersect each other. Representing an arbitrary set of curves efficiently and determining all of their intersections is not a simple task. If however, we assume that we are given a set of curves and a list of intersections, then we can apply our algorithm directly to determine the locations of the required cuts. In this case, there are no junctions, so we are guaranteed to get optimum results. Figure 3.14 shows an example problem and an optimum solution.

6. Summary

In this chapter we have presented three algorithms for the 2-CVM problem. First we presented Algorithm 1 which optimally solves the 2-CVM$_3$ problem in $O(|\mathcal{R}|^3)$ time. It overcomes the limitations of previous 2-CVM$_3$ algorithms by allowing vias to be placed anywhere along routing segments and not restricting the input layout to be grid based. It has the same time complexity as the previous algorithms [4,27] and is simpler to implement. Second, we showed how to improve the efficiency of the first algorithm when it is applied to geometric layouts. Without adding any additional restrictions, we modified the algorithm to take advantage of the fact that geometric design rules greatly limit the number of possible solutions. We also described the notion of via candidate zones and show how they can be used to further reduce the number of vias required for a particular layout. Although, in the worst case, Algorithm 2A is no faster than Algorithm 1, we will show in the next chapter that it is, on average, $O(|\mathcal{R}|)$ times faster for certain problems. In the case when the geometric design rules do not limit the placement of vias, Algorithm 2A degenerates to Algorithm 1. Third, we extended Algorithm 2A to handle the general 2-CVM problem. Although this problem is NP-hard, we described an algorithm which is optimum to within a known amount. If there are no junctions with degree greater than three in the input layout, then this algorithm is equivalent to Algorithm 2A. Finally, we discussed the possible application of Algorithm 1 to a curve coloring problem.

Thus, we have a family of algorithms that can be applied to a wide variety of problems and which produce optimum results in polynomial time whenever possible.

Chapter IV

Implementation

1. Overview

This chapter will deal with the implementation of the algorithms described in Chapter III. Section 2 describes a system which implements those algorithms and compares its performance with existing algorithms. Section 3 discusses how the average efficiency of the implementation might be improved using techniques from Computational Geometry. Section 4 describes practical extensions which could be made to adapt to real-world situations. Section 5 describes a heuristic derived from our experimentation with real examples. Finally, Section 6 summarizes the results of this chapter.

2. MINIMIZER

2.1. The program

The combined algorithms of Chapter III have been fully implemented in a via minimization system called MINIMIZER. It is the first such system that has been reported using an optimum algorithm. MINIMIZER consists of two main programs called MIN1 and MIN2 and several ancillary programs. All have been written in the C programming language and are residing on a Sun-3/160 minicomputer running the UNIX operating system. Together they consist of over 5000 lines of C-code. A brief outline of the programs included in MINIMIZER is as follows:

MIN1 - Implements Algorithm 1 described in Chapter III. It takes as input a file containing a layout description and generates as output another file containing a description of a via-minimized layout.

MIN2 - Same as MIN1, but with the extensions of Algorithms 2 and 3 described in Chapter III.

DRAW - Displays a layout file on either a color or black-and-white Sun.

PLOT - Converts a layout file to a format suitable for printing on an IMAGEN laser printer.

CONV - Converts a file generated by Rim and Nakajima's channel router [28] to a layout file compatible with this system.

INPUT - A simple graphics utility which allows the user to create a layout on the Sun using mouse input.

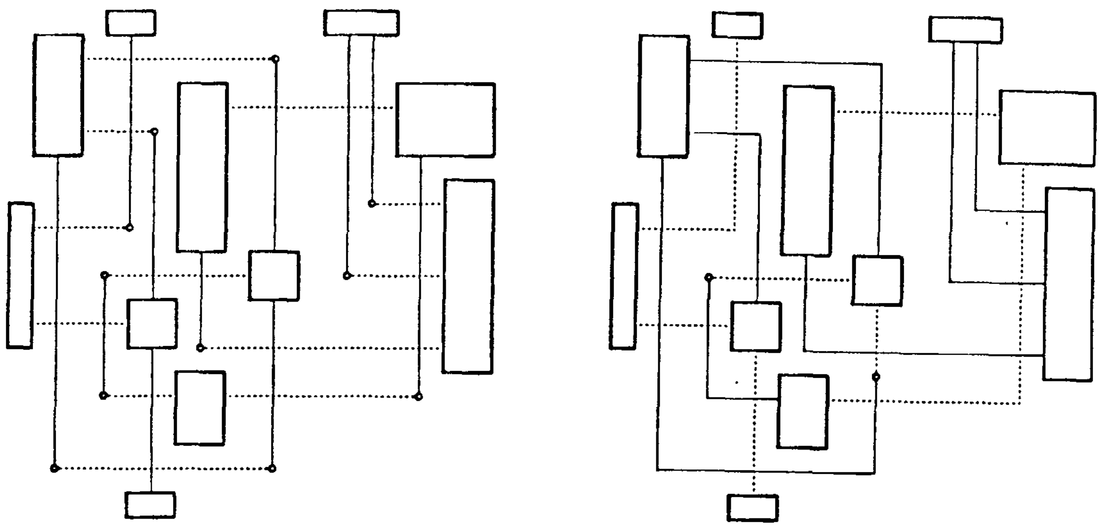
The layout files are simple, human readable ASCII files. A layout is specified in terms of lines, boxes, and points which correspond to routing segments, modules, and vias, respectively. Limited text annotation is also supported. All of the programs except DRAW and INPUT are portable to any other UNIX system without modification and to any other C-language environment with only minor modifications. DRAW and INPUT utilize the SunCGI graphics libraries and may be portable to other systems supporting the still developing ANSI CGI standard. The remainder of this section will describe MIN2.

When executed, MIN2 is given the name of an input file and several options. These options determine, among other things, what locations in the input layout will be considered via candidates. The default assumption is that the layout is based on a unity grid and that vias should be placed only at open grid points. The other options which are available are to treat the input as a topological layout in which vias may be placed anywhere or to restrict vias to the endpoints of the given routing segments. The later can be used to speed up the algorithm or to obtain results comparable to what the previous optimum algorithms [4,20] might have generated. Other options select whether via candidate zones should be allowed or whether any heuristics (Section 4.5) should be employed. A list of the major modules in MIN2 is as follows:

main -	interprets the command options, opens and closes the necessary files, calls the routines below in the order that they are listed and calculates run statistics.
getwlist -	reads the input file and generates an internal list of routing segments.
graph_gen -	generates the derived graph $G_{\mathcal{L}}$ as described in Section 2.1 of Chapter III. from the list of routing segments. Crossings are located using the straight-forward method of comparing all pairs of routing segments.
merge_junctions -	merges junction nodes in $G_{\mathcal{L}}$ into nodes corresponding to maximal via candidate zones as described in Section 3.2 of Chapter III.
merge_crossings -	merges crossing nodes in $G_{\mathcal{L}}$ into nodes corresponding to maximal clusters as described in Section 3.2 of Chapter III.
find_essential_vias -	locates all essential vias as described in Section 3.3 of Chapter III and simplifies the graph accordingly.

dual_graph -	generates the dual graph $G'^{d}_{\mathcal{L}}$ from the graph $G'_{\mathcal{L}}$ which was created by the last three routines.
remove_self_loops -	simplifies $G'^{d}_{\mathcal{L}}$ by removing all edges corresponding to edges in $G'_{\mathcal{L}}$ which are not in any cycles.
graph_M -	generates the graph M. This includes finding the shortest path between every pair of odd nodes as described in Section 3.4 of Chapter III. This is accomplished using a modified Breadth-First-Search which takes into account via labels.
matching_N4 **matching_N3 -**	implement the $O(n^4)$ and $O(n^3)$ maximum matching algorithms described in Lawler [22] which are both based on Edmonds' method [8,9]. Both have been modified because of mistakes or omissions in Lawler's description. Each is over 1000 lines of code although many of their routines are very similar. Only one of them is linked to MIN2 at a time. For the problems we considered, matching_N3 was not any faster than matching_N4.
mark_vias -	modifies G' to reflect the location of the vias identified by the maximum matching algorithm.
assign_layers -	traverses the edges in G' assigning the corresponding subsegments to layers.
output_graphics -	generates the final layout file.

MIN1 consists of all of the same modules as MIN2 except merge_junctions, merge_crossings, and find_essential_vias.



```
Input file = kajitani.in
Number of Wire Segments = 23
Number of Original Vias = 11
Number of Vias = 2

user time: 0.10 sec
sys time: 0.10 sec
```

Figure 4.1. Layout example 1

2.2. Results

The MINIMIZER system has been tested on a number of examples from the literature, generated by hand, and generated using a channel router [28]. Figure 4.1 shows the input layout for a problem taken from [20] and the

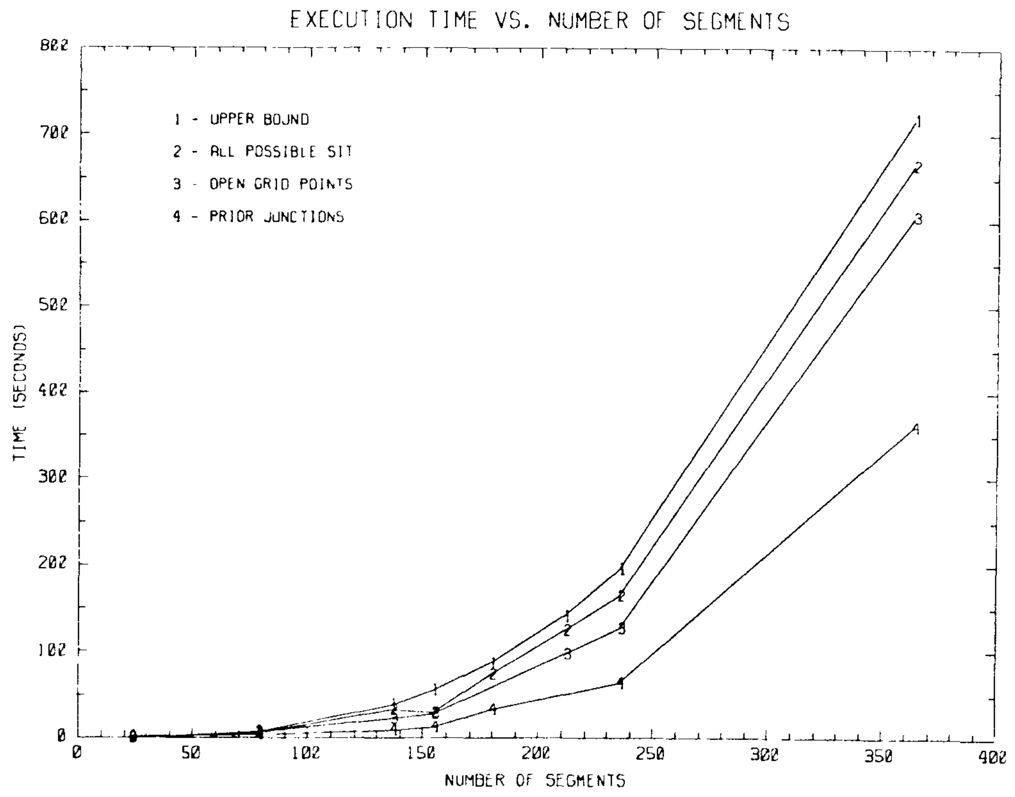


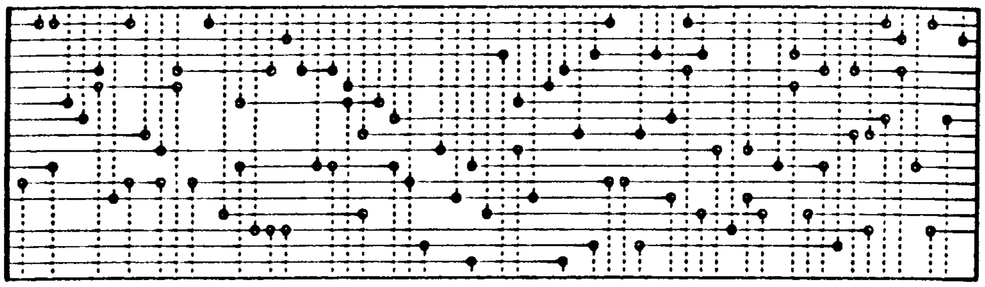
Figure 4.2. MIN1 run time for channel routing examples

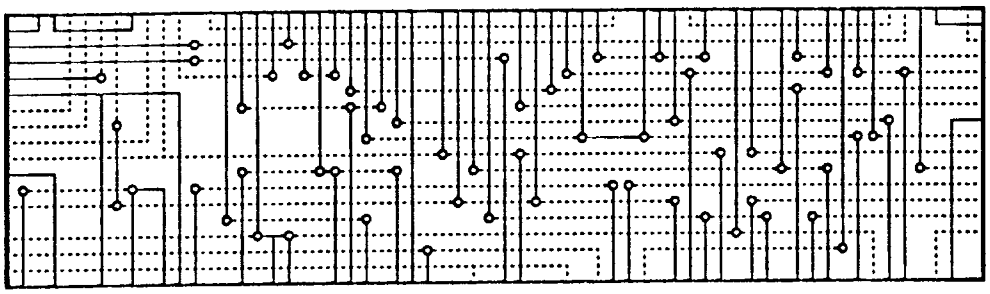
corresponding computer generated output. Figure 4.2 shows the time used by MIN1 to obtain optimum layer assignments for a series of real channel routing problems that were first routed using the efficient channel routing algorithm described by Rim and Nakajima [28]. The three lower curves correspond to considering only the endpoints of routing segments, only open grid points, and all possible via locations as via candidates. The top curve represents the worst case bound of cn^3, where c is found empirically to be approximately 10^{-5} seconds when the program is executed on a Sun3/160 computer. It is easy to see that in all cases the time required is less then the upper bound of $O(n^3)$ and that increasing the maximum number of via candidates from $O(n)$ to $O(n^2)$ only increases the execution time by a constant factor.

The number of vias required for these examples were reduced by between 14% and 41% from the number required by the original layer assignment which placed all vertical routing segments on one layer and all horizontal routing segments on the other. In several cases, the number of tracks could be reduced by overlapping horizontal segments that were assigned to different layers by this algorithm. Figure 4.3 shows the actual input and graphical output generated by this program for one of the channel routing problems mentioned.

Figure 4.4 shows the amount of computer time required by MIN2 to solve the same problems. Not only has the time been drastically reduced, but the actual time dependency has been reduced from $O(n^3)$ to $O(n^2)$ indicating that most of the time is being consumed by the steps preceding the maximum matching step. Figure 4.5 shows the output generated by MIN2 for the same example as shown in Figure 4.3. Notice that the number of vias has been further reduced from 66 to 63 through the introduction of via candidate zones. Subsegments assigned to both layers are indicated by bold lines.

Chang and Du [3] recently published a heuristic via minimization algorithm and claimed some speedup over a previous method. Although their algorithm is much simpler than ours, it has a worst-case time complexity





```
Input file = 3aYK2t.in
Number of Wire Segments = 136
Number of Original Vias = 91
Number of Vias = 66

user time: 22.00 sec
sys time: 1.36 sec
```

Figure 4.3. Channel routing example

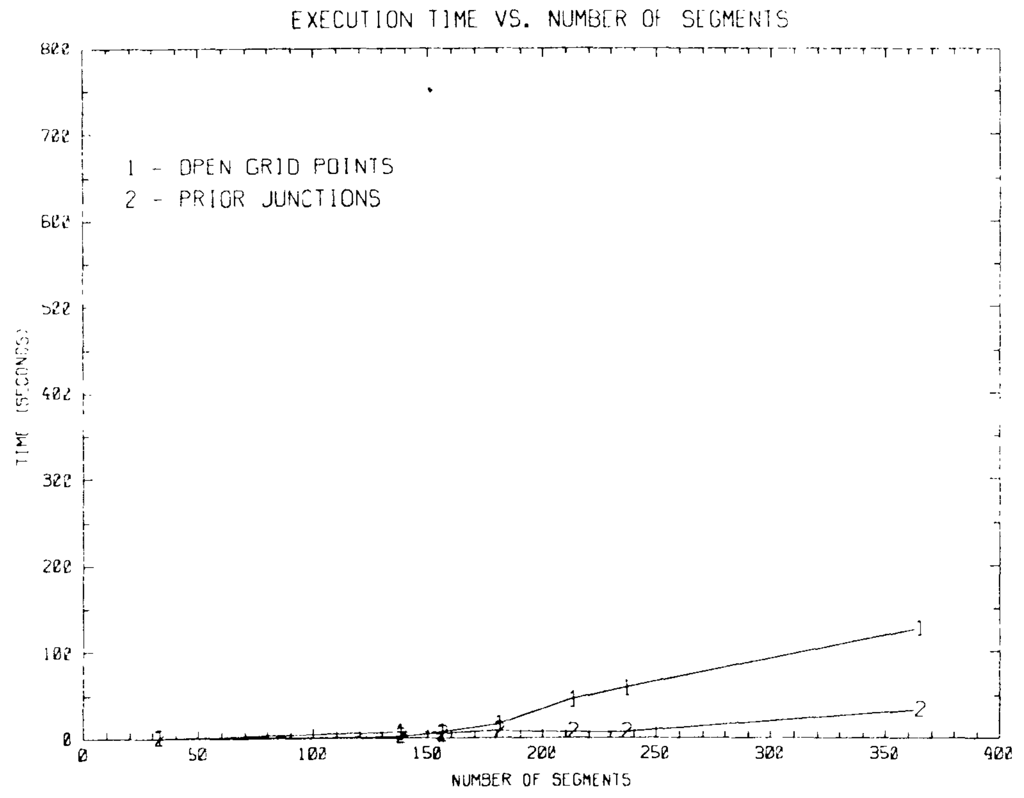
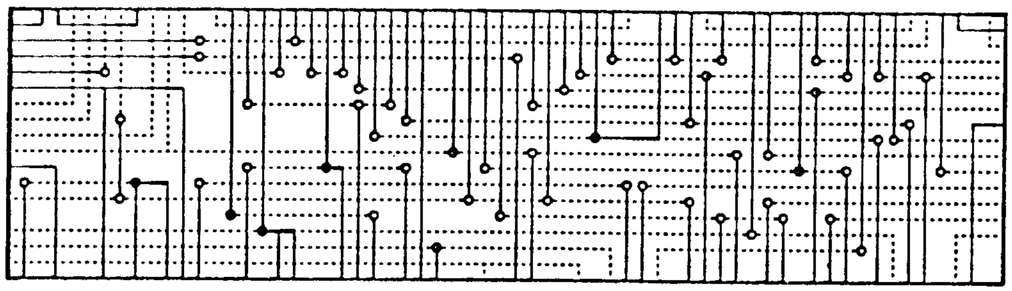


Figure 4.4. MIN2 run time for channel routing examples

which is much worse. Although they did not publish any run times for large examples, we were able to compare our algorithm with theirs on a number of small examples taken from [32]. Figure 4.6 shows a comparison of our run times on a Sun-3/160 with theirs on a VAX 11/780. Curve 1 is their main heuristic algorithm. Curve 2 is their algorithm for identifying a subset of the essential vias. Curve 3 is our optimum algorithm MIN2. Note that MIN2 is significantly faster than their main algorithm for all of the problems considered. MIN2 is also faster than their essential via algorithm for all but



Input file = 3aYK2t.in
Number of Wire Segments = 136
Number of Original Vias = 91
Number of Vias = 63

user time: 2.70 sec
sys time: 0.62 sec

Figure 4.5. Channel routing example with via candidate zones

the biggest problem considered. For that case, however, their essential via algorithm does not locate all of the necessary vias so their slower algorithm would also have to be applied to generate a complete solution.

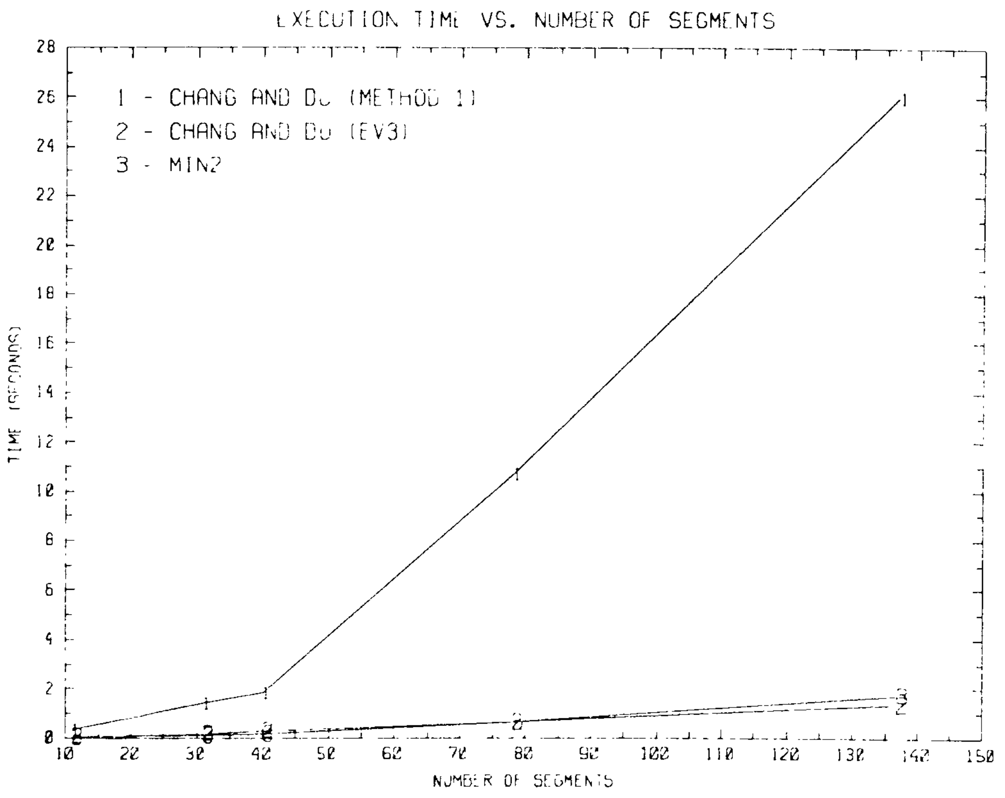


Figure 4.6. Comparison of MIN2 to a heuristic algorithm

3. Efficiency Considerations

When using MIN2 on the channel routing examples mentioned above, it was found that the number of odd nodes in $G^d_{\mathcal{L}}$ was reduced so drastically that the time required to generate $G_{\mathcal{L}}$ and $G'_{\mathcal{L}}$ and to locate all essential vias became significant. In this section, we will consider how to implement those first three steps of the algorithm more efficiently. To begin with, we will consider the problem of generating the graph $G_{\mathcal{L}}$ from a set of routing segments $\mathcal{R}$. Using the brute force approach, this requires $O(|\mathcal{R}|^2)$ time to compare each routing segment with all of the others. Bently and Ottoman [2] showed how to find all crossings between pairs of line segments in $O(|\mathcal{R}| \cdot \log|\mathcal{R}| + c \cdot \log|\mathcal{R}|)$ time or in $O(|\mathcal{R}| \cdot \log|\mathcal{R}| + c)$ time if all of the segments are parallel to one of two perpendicular axes, where c denotes the number of crossings located. Using the algorithm described in Section 3.2 of Chapter III, $G'_{\mathcal{L}}$ can be obtained from $G_{\mathcal{L}}$ in $O(|E|) = O(|\mathcal{R}| + c)$ time. Once we have $G'_{\mathcal{L}}$, the essential vias can be found in $O(|E'|) = O(|\mathcal{R}| + c)$ time. Thus when $c \ll |\mathcal{R}|^2$, locating all of the crossings may be the most time consuming part of the first three steps.

If vias are restricted to the endpoints of routing segments in $\mathcal{R}$, then it is not necessary to partition the routing segments into subsegments and we can avoid locating all crossings between pairs of routing segments. Instead we can locate all maximal clusters directly from $\mathcal{R}$ and generate $G'_{\mathcal{L}}$ without using $G_{\mathcal{L}}$. If the layout is grid based, then we can use the geometric graph search algorithm of Imai and Asano [19] to locate all clusters in $O(|\mathcal{R}| \cdot \log|\mathcal{R}|)$ time.

4. Practical Considerations

4.1. Critical nets

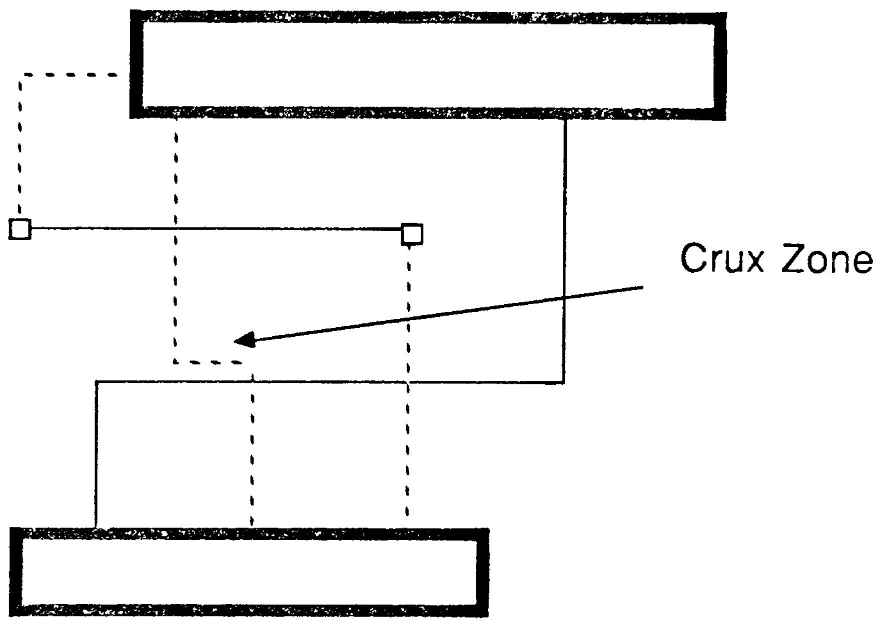
Often, a designer may want to designate that certain nets not be assigned any vias. These could be power and ground nets or nets that have very critical timing requirements. This is easily accommodated by the same mechanism as used for the geometric constraints. In this case, $VIA(x,y) = \Lambda$ for all edges (x,y) in $G_{\mathcal{L}}$ for which $SEG(x,y)$ is part of a critical net. This will not affect the time complexity of the algorithm or the existence of an optimum solution if the input layout meets the same constraints.

4.2. Crux zones

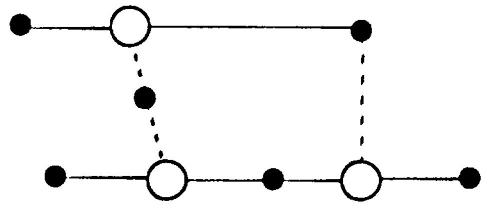
In printed circuit boards and in some integrated circuit processes, it is allowable to have two parallel routing segments overlap for some distance. Clearly, the two overlapping segments must be assigned to different layers and additionally, no vias can be placed on either segment where they are overlapping. Chen, et al. [4] named such overlap regions *crux zones* and gave them special attention in their algorithm. In our formulation, crux zones may simply be treated as clusters. Figure 4.7 shows a layout with crux zones and the resulting derived graph.

4.3. Unequal layers

In most technologies, some layers are more preferable for routing than others. This may be due to the fact that routing segments assigned to a particular layer may have lower resistance, greater thickness, or take up less area than routing segments assigned to other layers. In the 2-CVM algorithm we have described and implemented, layer assignment is done independently of via placement and, as such, can only be given secondary consideration. However, given a particular set of vias, it is easy to choose a layer assignment which maximizes the number of routing segments assigned to a particular layer. This is due to the fact that each maximal cluster has only two possible layer



$\mathcal{L}_4$



$G'_{\mathcal{L}_4}$.

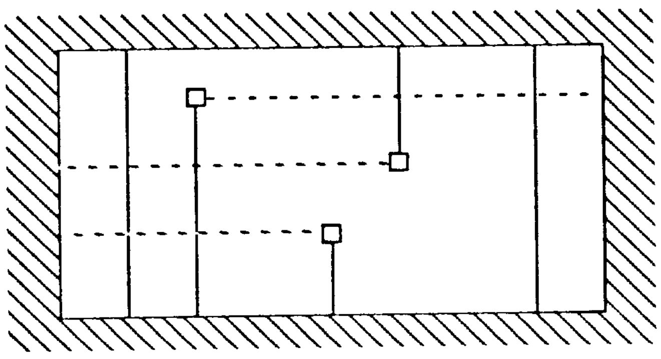
Figure 4.7. Example with crux zones

assignments. After via placement, those clusters which intersect at a junction that has not been assigned a via can be merged into a ***supercluster.*** After merging all possible clusters, each supercluster is given the layer assignment that maximizes the number of routing segments assigned to the desired layer. This can all be accomplished without increasing the time complexity of the layer assignment step.

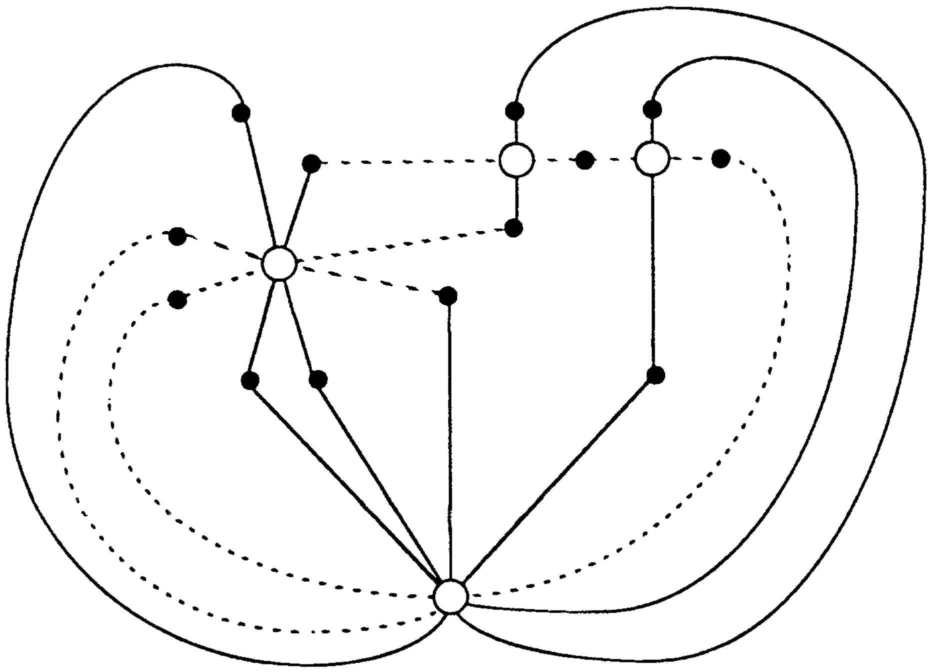
A second consideration related to layer assignment is that the minimum spacing between wires assigned to different layers may be different. For example, in some technology routing segments assigned to Layer 1 are realized as metal strips 3λ wide separated by at least 3λ, while routing segments assigned to Layer 2 are of the same width, but must be separated by at least 4λ. Thus, if two routing segments are separated by only 3λ, they cannot both be assigned to Layer 2. Such constraints can be taken into consideration when forming clusters. If this results in certain clusters having only one possible layer assignment, then the remainder of the algorithm will also have to be modified. A simpler solution would be to use a router which spaces all wires equally. After via minimization, the number of wires on the narrower layer could be maximized and then the layout compacted.

4.4. Fixed layer terminals

It is very common to assume that all terminals are available on both layers in theoretical discussions of routing and via minimization. In practice, however, it is at least as common for terminals to only be available on one layer. In such cases, all terminals may have to be vias to satisfy the assumption of the router. In order to avoid adding an excessive number of vias at the terminals,



$\mathcal{L}_5$



$G'_{\mathcal{L}_5}$

Figure 4.8. Example with fixed layer terminals

we can adapt our algorithm to handle fixed layer terminals. Consider the case when all terminals lie on the boundary of the routing area as in the case of channel and switchbox routing. When we form $G'_{\mathcal{L}}$, we simply treat the area outside the routing area, denoted by m_{ϕ}, as a single cluster as shown in Figure 4.8. This will assure that all of the segments incident upon m_{ϕ} which have the same type will be assigned to the same layer. When we assign segments to layers, we should begin with segments that have terminals as endpoints.

For the more general case when all terminals are not located on m_{ϕ}, we can extend this basic approach. We begin by generating a cluster in $G'_{\mathcal{L}}$ for each module m_i to express the relative layer assignments between segments incident upon that module. However, in order to assure that terminals on different modules are assigned to the correct relative layer, it is necessary to link the nodes corresponding to modules together by introducing some dummy edges and nodes in $G'_{\mathcal{L}}$.

5. Heuristics

Despite all of our improvements in the basic algorithm, for very large problems the most time consuming steps will be the computation of the edge weights for the graph $M = (N_{odd}, E_M)$ and the maximum weight matching step. In order to reduce the time required for those two steps we suggest a heuristic method based on empirical data gathered from real examples. Recall that computing the edge weights for the complete graph M requires finding a shortest path between each odd vertex in $G^d_{\mathcal{L}}$ and every other odd

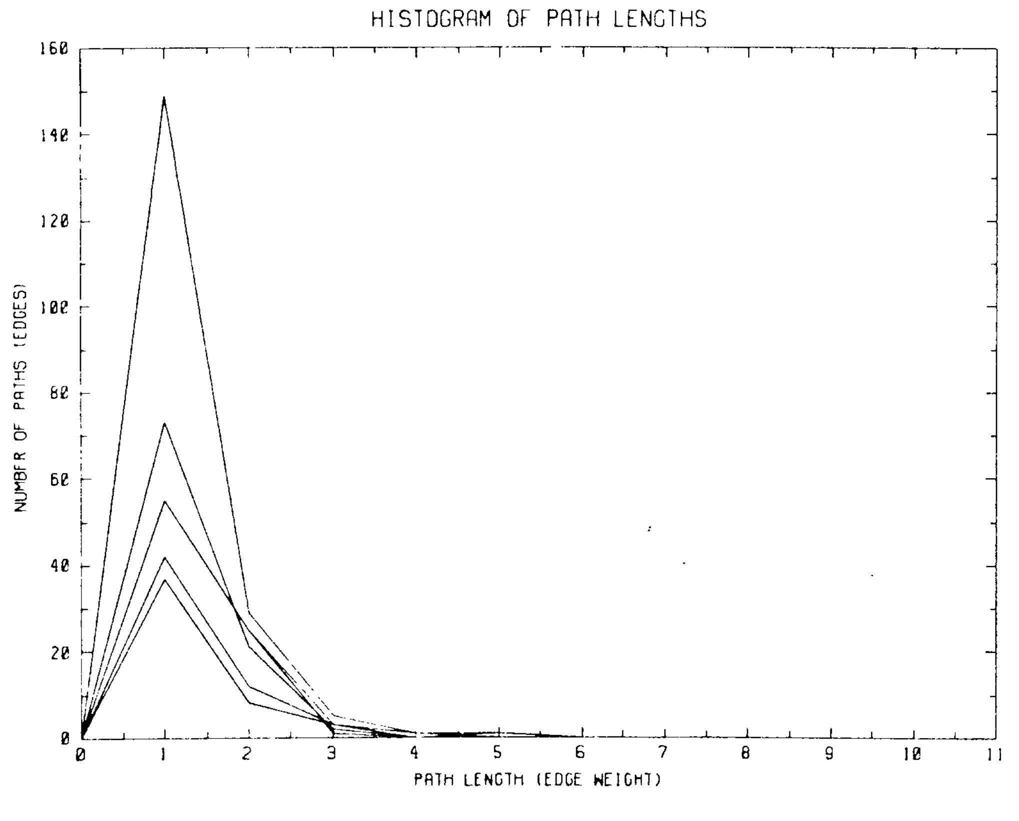


Figure 4.9. Histogram of path lengths

vertex. Since all of the edges have a weight of approximately one, this step requires $O(|E^d|)$ time for each of the vertices in N_{odd}. The weight on each edge in M corresponds to the cost, in vias, of merging a particular pair of odd faces in $G_{\mathcal{L}}$. Figure 4.9 shows a histogram of the weights on the edges in M actually chosen by Algorithm 1 as part of a maximum weight matching for a sequence of channel routing problems of various sizes. Note that most of the path lengths are very small and that the median path length does not change significantly with increasing problem size. In fact, for all of the examples we tested, with sizes of up to 1000 routing segments, we never encountered a path in a solution with length greater than 10. More than 95% had length of 2 or less. This is very significant, since it means that had we limited our search to paths of length 10 or less, we would still have obtained an optimum solution for all of the examples we tested. We call this limit on search depth the *maximum search depth* and denoted it by sd_{max}. Edges in M which would have weights greater than $\Gamma - sd_{max}$ are omitted. Some care must be taken to assure that this does not result in any maximal connected components in M which contain an odd number of nodes. If this occurs, sd_{max} may have to be increased for vertices in those components.

As mentioned, these results were obtained using Algorithm 1. When using Algorithm 2A or 3, the number of nodes in M is reduced so drastically that the computation of the edge weights and the maximum matching for M are not the most time consuming steps for problems of the size considered thus far. However, if the problem size were increased in size by an order of magnitude or more, or the efficiency of the first steps was improved, similar results could be expected.

6. Summary

In this chapter, we have described the first implementation of an optimum polynomial time algorithm for via minimization. Furthermore, we have shown that it is faster than the existing heuristic algorithm. We have discussed how to reduce the average time complexity even further using techniques from computational geometry. We have also discussed practical considerations for applying this algorithm to real-world problems and describe some extensions to handle them. For very large problems our system may not be fast enough. For that case we suggest a heuristic approach which is based on a common characteristic of the optimum solutions we generated with MIN2.

Chapter V

Conclusion

1. Summary

In this paper, we have discussed the constrained via minimization problem in great detail. In Chapter II, we showed that the general 2-CVM problem is NP-hard and it remains so even when the maximum junction degree is limited to six or more, the input layout is constrained to be grid based, and vias are placed only at the endpoints of the given routing segments. In Chapter III, we presented a family of polynomial time algorithms which do not place any constraints on the input layout or the placement of vias and which give optimum results when the maximum junction degree is limited to three. If there are junctions with degree greater than three, then all of the algorithms give approximate results.

Algorithm 1 is the basic algorithm. It is the simplest to implement, yet gives optimum results to the 2-CVM$_3$ problem and near optimum results for the 2-CVM$_k$ problem. The time complexity of Algorithm 1 is $O(|\mathcal{R}|^3)$, where $|\mathcal{R}|$ is the number of routing segments in the input layout. Algorithm 2 is designed for geometric layouts. It adds a step in which the basic graph derived from the input layout is simplified to take advantage of the limited number of geometrically allowable via sites. In the worst case Algorithm 2 has the same time complexity as Algorithm 1. Algorithm 2 also has the capability to merge adjacent via sites into a single via candidate zone by assigning some routing segments to both layers. Essential vias are defined to be those vias in the input layout which must be present in any output layout. Algorithm 2A adds a step in which all such vias are efficiently located, thus reducing the number of via candidates which must be considered by the remainder of the algorithm. Algorithm 3 adds a weighting function which increases the statistical likelihood of obtaining an optimum solution when the maximum junction degree exceeds three. For topological layouts with no junction degree exceeding three, all of the algorithms reduce to Algorithm 1.

In Chapter IV, we described how we have implemented the algorithms of Chapter III. Ours is the first implementation of an optimum 2-CVM$_3$ algorithm reported. We have proven that it will generate at least as few vias as any other algorithm and surprisingly found that it does so significantly faster than the existing heuristic algorithms. Traditionally, optimum algorithms are difficult to adapt to other constraints. However, we described how to adapt our algorithm to handle a wide variety of real life constraints such as critical nets, fixed layer terminals, crux zones, and unequal layers. In addition, we suggest a heuristic based on data gathered from the solutions generated by our optimum algorithm. Using the heuristic, it should be possible to significantly reduce the computation time needed for very large layouts and still generate optimum results most of the time.

2. Recent Results

We were the first to show that the 2-CVM_k problem is NP-hard for all $k \geq 6$. Very recently, Choi, Nakajima, and Rim [6] extended this result to layouts with no junction degree exceeding four. This leaves open only those cases for which we have developed polynomial time algorithms. Molitor [26] recently showed that the ℓ-CVM problem is NP-hard for all $\ell \geq 3$ even when the maximum junction degree is limited to four or more.

Molitor [26] also suggested an approach similar to our Algorithm 1 for layouts with no junction having degree exceeding three.

3. Future Work

The ℓ-CVM_k problem has been shown to be NP-hard for all $\ell \geq 2$ when $k \geq 4$. We have developed and implemented several efficient optimum algorithms for the case when $\ell = 2$ and $k \leq 3$ and approximate algorithms for the case when $\ell = 2$ and $k \geq 4$. This leaves open the cases when $\ell \geq 3$ and $k \leq 3$. Since integrated circuit manufacturing technology already permits the use of three or more layers, this would be an important area of research.

Another future direction might be to consider some more restricted cases of the 2-CVM problem such as channel routing where all of the terminals lie on the boundary of the routing area. For that particular case, the polynomial transformation used to show that *cvm* is NP-complete cannot be applied and and it is not known if the problem is polynomially solvable. A more interesting direction would be to consider a less constrained via minimization technique where minor changes to the layout would be allowed.

As mentioned in Chapter I, it is not known whether a polynomial time algorithm exists for the UVM problem, although it seems likely that the general case is NP-hard. It would be primarily of theoretical interest to see that proven. As seen in Figure 1.4, a UVM solution is not likely to be a practical layout for a real circuit because of excess wire length and area. Instead, more attention should probably be focused on developing practical routers which can optimize a weighted sum of factors such as wire length, number of vias, and total area while working within geometric and electrical constraints.

Bibliography

[1] K. Aoshima and M. Iri, "Comment on F. Hadlock's paper: Finding a maximum cut of a planar graph in polynomial time," *SIAM J. Comput.*, vol. 6, no. 1, pp. 86–87 , Mar. 1977. .

[2] J. L. Bentley and T. A. Ottmann, "Algorithms for reporting and counting geometric intersections," *IEEE Trans. Comput.*, vol. C-28, no. 9, pp. 643–647, Sept. 1979.

[3] K. C. Chang and D. H-C. Du, "Efficient algorithms for layer assignment problem," *IEEE Trans. Computer-Aided Design*, vol. CAD-6, no. 1, pp. 67–78, Jan. 1987.

[4] R-W. Chen, Y. Kajitani and S-P. Chan, "A graph-theoretic via minimization algorithm for two-layer printed circuit boards," *IEEE Trans. Circuits and Systems*, vol. CAS-30, no. 5, pp. 284–299, May 1983.

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

[6] H-A. Choi, K. Nakajima and C. S. Rim, "Complexity results for vertex-deletion graph bipartization and via minimization problems," to appear in *Proc. 25th Annual Allerton Conf. on Computing, Communication, and Control*, Monticello, IL, Sept. 1987.

[7] M. J. Cielielski and E. Kinnen, "An optimum layer assignment for routing in ICs and PCBs," in *Proc. 18th Design Automation Conf.*, Nashville, TN, June 1981, pp. 733–737.

[8] J. Edmonds, "Maximum matching and a polyhedron with 0-1 vertices," *J. Res. Nat. Bur. Standards*, vol. 69B, no. 1-2, pp. 125–130, June 1965.

[9] ———, "Paths, trees, and flowers," *Canad. J. Math.*, vol. 17, pp. 449–467, 1965..

[10] S. Even, *Graph Algorithms.* Rockville, MD: Computer Science Press, 1979.

[11] I. Fary, "On straight line representation of planar graphs," *Acta Sci. Math. (Szeged)*, vol. 11, pp. 229–233, 1948.

[12] H. N. Gabow, "Implementation of algorithms for maximum matching on nonbipartite graphs," Department of Computer Science, Stanford University, Stanford, CA, Ph.D. Dissertation, 1974.

[13] M. R. Garey and D. S. Johnson, "The rectilinear Steiner tree problem is NP-complete," *SIAM J. Appl. Math.*, vol. 32, no. 4, pp. 826–834, June 1977.

[14] ———, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco, CA: W. H. Freeman and Company, 1979.

[15] M. R. Garey, D. S. Johnson and L. Stockmeyer, "Some simplified NP-complete graph problems," *Theoret. Comput. Sci.*, vol. 1, pp. 237–267, 1976.

[16] F. Hadlock, "Finding a maximum cut of a planar graph in polynomial time," *SIAM J. Comput.*, vol. 4, no. 5, pp. 221–225, Sept. 1975.

[16] F. Hadlock, "Finding a maximum cut of a planar graph in polynomial time," *SIAM J. Comput.*, vol. 4, no. 5, pp. 221–225, Sept. 1975.

[17] A. Hashimoto and J. Stevens, "Wire routing by optimizing channel assignment within large apertures," in *Proc. 8th Design Automation Workshop*, Atlantic City, NJ, June 1971, pp. 155–169.

[18] C. -P. Hsu, "Minimum via topological routing," *IEEE Trans. Computer-Aided Design*, vol. CAD-2, no. 4, pp. 235–246 , Oct. 1983.

[19] H. Imai and T. Asano, "Efficient algorithms for geometric graph search problems," *SIAM J. Comput.*, vol. 15, no. 2, pp. 478–494, May 1986.

[20] Y. Kajitani, "On via hole minimization of routing on a 2-layer board," in *Proc. 1980 IEEE Int. Conf. on Circuits and Computers*, Oct. 1980, pp. 295–298.

[21] Y. Kajitani, "The orthogonal representation of curves with the minimum number of bends," Systems Research Center Systems Seminar, University of Maryland, College Park, May 1987.

[22] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids.* New York: Holt, Rinehart, and Winston, 1976..

[23] J. M. Lewis and M. Yannakakis, "The node-deletion problem for hereditary properties is NP-complete," *J. Comput. System Sci.*, vol. 20, no. , pp. 219–230, 1980.

[24] M. Marek-Sadowska, "An unconstrained topological via minimization problem for two-layer routing," *IEEE Trans. Computer-Aided Design*, vol. CAD-3, no. 3, pp. 184–190, July 1984.

[25] S. Micali and V. Vazirani, "An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs ," in *Proc. Twenty-first Annual Symposium on Foundations of Computer Science*, Oct. 1980, pp. 17–27.

[26] P. Molitor, "On the contact minimization problem," in *Proc. 4th Annual Symp. on Theoretical Aspects of Computer Science*, Passau, West Germany, Feb. 1987, pp. 420–431.

[27] R. Y. Pinter, "Optimal layer assignment for interconnect," in *Proc. 1982 IEEE Int. Conf. on Circuits and Computers*, New York, NY, Sept. 1982, pp. 398–401.

[28] C. S. Rim and K. Nakajima, "An efficient channel routing algorithm for two and three layers," unpublished report, Systems Research Center, University of Maryland, College Park, MD, 1987.

[29] K. R. Stevens and W. M. VanCleemput, "Global via elimination in generalized routing environment," in *Proc. IEEE 1979 Int. Symp. on Circuits and Systems*, Tokyo, Japan, June 1979, pp. 689–692.

[30] R. Tamassia, "On embedding a graph in the grid with the minimum number of bends," *SIAM J. Comput.*, vol. 16, no. 3, pp. 421–444, June 1987.

[31] M. Yannakakis, "Node-deletion problems on bipartite graphs," *SIAM J. Comput.*, vol. 10, no. 2, pp. 310–327, May 1981.

[32] T. Yoshimura and E. Kuh, "Efficient Algorithms for Channel Routing," *IEEE Trans. Computer-Aided Design*, vol. 1, no. 1, pp. 25–35, Jan. 1982.

Curriculum Vitae

Name:	Nicholas Joseph Naclerio.
Permanent address:	4407 Evergreen Drive Woodbridge, Virginia 22193.
Degree and date to be conferred:	Ph.D., 1987.
Date of birth:	July 26, 1961.
Place of birth:	New York, New York.
Secondary Education:	Gar-Field Senior High School Woodbridge, Virginia 1979.

Collegiate institutions attended:	Dates	Degree	Date of Degree
Duke University	9/79 - 5/83	B.S.E.	5/83
University of Cambridge	10/83 - 8/84	C.P.G.S.	8/84
University of Maryland	8/84 - 9/87	Ph.D.	12/87

Major: Electrical Engineering.

Professional publications:

[1] N. J. Naclerio, "Superconducting Devices for Millimetre Wave Detection," University of Cambridge, Cambridge, England, CPGS Thesis, July 1984.

[2] N. J. Naclerio, J. M. Lumley, J. E. Evetts and R. E. Somekh, "Millimetre wavelength detection using Nb/NbO$_x$/Pb alloy tunnel junctions," *IEEE Trans. Magnetics*, vol. MAG-21, no. 2, pp. 903–905, Mar. 1985.

[3] N. J. Naclerio, S. Masuda and K. Nakajima, "The via minimization problem is NP-complete," Systems Research Center, University of Maryland, College Park, MD, TR87-42, Mar. 1987.

[4] ———, "Via Minimization for Gridless Routing," Systems Research Center, University of Maryland, College Park, MD, TR87-46, May 1987.

[5] ———, "Some NP-Completeness Results for Via Minimization," in *Proc. 1987 Conf. on Information Sciences and Systems*, Baltimore, MD, Mar. 1987, pp. 611–616.

[6] ———, "Via minimization for gridless layouts," in *Proc. 24th Design Automation Conf.*, Miami Beach, FL, June 1987, pp. 159–165.

[7] ———, "The via minimization problem is NP-complete," to appear in *IEEE Trans. Comput.*.

[8] C. S. Rim, N. J. Naclerio, S. Masuda and K. Nakajima, "A linear time algorithm for circular permutation layout," to appear in *Proc. 25th Annual Allerton Conf. on Computing, Communication, and Control*, Monticello, IL, Sept. 1987.

Professional positions held:

5/83-7/83	Research Intern	IBM Research Center, Yorktown Hts., NY.
5/82-9/82	Research Intern	IBM Research Center, Yorktown Hts., NY.