Abstract

Title of Dissertation: Efficient Algorithms for Channel Routing and Placement Problems

Jau-Der Shih, Doctor of Philosophy, 1993

Dissertation directed by: Professor Ronald I. Greenberg
Department of Electrical Engineering

This thesis is primarily concerned with channel routing, a key problem in automatic layout design. In the context of both single-layer and multilayer routing, efficient algorithms are given to optimize common cost functions.

In the single-layer context, this thesis gives algorithms to solve the optimal offset, offset range, and optimal placement problems. It is shown that as long as only one side of the channel contains multiple terminals of the same net, all the three problems can be solved in linear time. When the placement of terminals is entirely unrestricted, the feasible offset and optimal offset problems can both be solved in time $O(n^{1.5} \lg n)$ under the reasonable assumption that the number of columns is $O(n)$, where n is the number of terminals.

For multilayer channel routing, this thesis gives algorithms to minimize the density of channels with movable terminals and channels with interchangeable terminals. The main cases considered for channels with movable terminals are channels with linear order constraints, channels with linear order constraints and separation constraints, channels with movable modules containing fixed terminals, and channels with movable modules and terminals. In each case, the results in this thesis improve previous results for running time and space by a factor of $L/\lg n$ and L, respectively, where L is the channel length. For channels with interchangeable terminals, we consider the problem of minimizing channel density by interchanging terminals within cells on the channel sides. Efficient algorithms are given for channels with fixed cells and channels with movable cells.

The last part of the thesis presents an algorithm to find a maximum-density planar subset of n 2-pin nets in a channel, which can be helpful in the context of multilayer channel routing. It is established that a simple bound on the running time is $O(n \lg n + n(t+1)w)$, where t is the number of two-sided nets, and w is the number of nets in the output.

Efficient Algorithms for Channel Routing and Placement Problems

by

Jau-Der Shih

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
1993

Advisory Committee:

Professor Ronald I. Greenberg, Chairman/Advisor
Professor Joseph JáJá
Professor Linda Milor
Professor A. Yavuz Oruç
Professor David Mount

Dedication

To my parents

Acknowledgements

I would like to thank first to my dissertation advisor, Ronald I. Greenberg, for his enlightening guidance and professional advice throughout my stay as graduate student at UMCP. His diligent proofreading and insightful comments have also contributed greatly to the presentation of this dissertation. I would also like to thank Professors Joseph JáJá, Linda Milor, A. Yavuz Oruç, and David Mount on my dissertation committee for reading my dissertation and for offering helpful comments and suggestions.

Thanks also to the Electrical Engineering Department for the resources and stimulating research environment, and to all my friends who have made my stay at Maryland pleasent and enjoyable.

Finally, special thanks go to my parents. I owe them a great deal of gratitude and would like to dedicate this dissertation to them for their patience, understanding and encouragement.

Table of Contents

List of Figures

Chapter 1

Introduction

Layout systems for VLSI chips usually consist of two phases: the *placement* phase and the *routing* phase. In the placement phase, predesigned pieces of the circuit, which are called *modules*, are placed within the area of the chip. Each module has *terminals* located along its boundary, and the desired interconnections are specified by giving disjoint sets of terminals. Terminals in the same set are to be connected together. Then in the routing phase, the interconnections between these modules are completed using a specified number of layers. For the most part, wires on different layers are insulated from each other, but we can use *vias* to connect wires in adjacent layers. Though the number of layers varies from one technology to another, it may be treated as a fixed constant according to whichever technology has been selected. The prime objective is then to minimize the total area of the chip. Other important layout characteristics, such as total wire length and signal propagation delay, also should be minimized to the extent possible.

The practical realities of fabrication technology of VLSI chips impose many limitations on the chip design. There are constraints which the layout must satisfy [45], such as a minimum transistor size, minimum widths for wires of various types, and minimum separations between wires. These constraints are called *design rules*. The layout problem as described above under the design rules contains a huge number of variables and is very complicated. It is hard to optimize any of the above objectives. Even when all the modules are placed, the problem of minimizing the chip area is still hard to deal with.

Various design methodologies have been suggested to deal with the complexity of the layout problems, both in the placement phase and routing phase. However, since both the placement and routing problems are believed to be intractable on many accounts, it is necessary to consider restricted versions of the problems or heuristic algorithms.

This thesis is primarily concerned with the routing phase, yet we allow limited freedom for the modules. That is, the modules are not completely fixed, they can move around in a restricted sense. The next section describes the basic problem definition and VLSI models used throughout this thesis.

1.1 The Channel Routing Problem

The channel routing problem was introduced by Hashimoto and Stevens [27] in 1971, and became very popular for a broad variety of technologies since then. The idea is to partition the routing area into rectangular *channels* (see Figure 1.1), which are then routed individually. By breaking the original problem into a set of smaller, local problems, the difficulties of global routing are alleviated, and the complexity of the problem is reduced significantly.

One design methodology in which channel routing has been particularly suc-

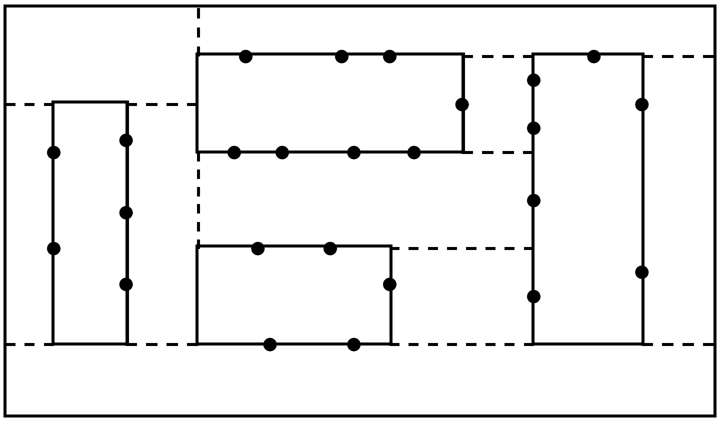

Figure 1.1: A placement of four modules on a chip. Terminals are represented by circles. The dashed lines show a possible partitioning into channels.

cessful is the standard-cell layout design. In standard-cell layout design, predesigned modules, which are called standard-cells, are placed in rows on the chip. These cells are all of the same height, which yields rectangular routing regions between the rows of cells, i.e., channel routing problems.

We now give the formal definition of the channel routing problem and the basic terminology used in this thesis.

A *channel* is the area between two parallel lines, which are refered to as *sides* of the channel. (We will henceforth draw these lines horizontally.) In a channel routing problem, there is a set of *nets*, each consisting of terminals lying on the two parallel sides. Channel routing is the problem of connecting all terminals of each net within the channel in such a way that no two nets cross each other on the same layer. Usually, the separation between the sides is unspecified, and the objective is to minimize the *width* or *separation* (the distance between the two sides) required to route the channel.

To simplify the discussion, we restrict attention to a rectilinear, grid-based model in which terminals lie on gridpoints and wires are disjoint paths through grid edges. This restriction often arises in practice, based on the language used to describe layouts or fabrication techniques involved. We refer to each of the horizontal grid lines (lines that are parallel to the channel sides) as a *row* or *track*, while the vertical grid lines are referred to as *columns*. Also define the *span* of a net to be the closed interval (in the horizontal direction) between the leftmost and the rightmost terminals that belong to the net.

In order to wire the terminals, several wiring models have been proposed. Each model has a set of rules that specifies how wires can be run on each layer. The rules significantly affect the routing algorithm to be employed as well as the routing area needed.

The primarily classification for wiring models is the number of conducting layers that are used. A secondary criterion deals with the ways in which these layers are allowed to interconnect and overlap. We now present some of the wiring models to which results in this thesis apply.

- **Single-Layer Model**

  In this model, only one layer is allowed for interconnection. Paths corresponding to different wires cannot overlap with or cross each other. Figure 1.2 shows an example of a routed single-layer channel. In general, the single-layer model can be used only when the interconnection pattern is planar. For example, if we have only 2-point nets going across the channel, then the order of the terminals' labels must be the same on both sides; this situation is called *river routing*.

- **The Manhattan Model**

  This is the most popular and well studied two-layer wiring model. It is also called the *directional model*, in which one layer is reserved for horizontal

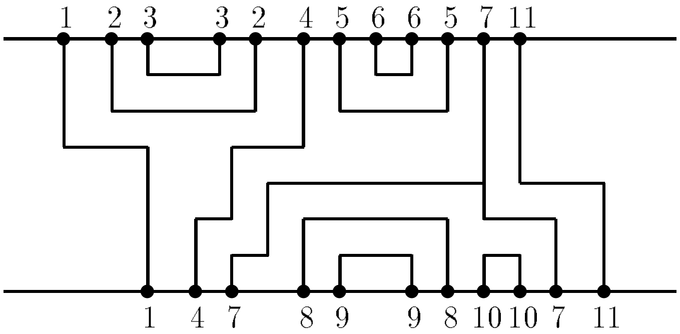

Figure 1.2: An example of a routed single-layer channel.

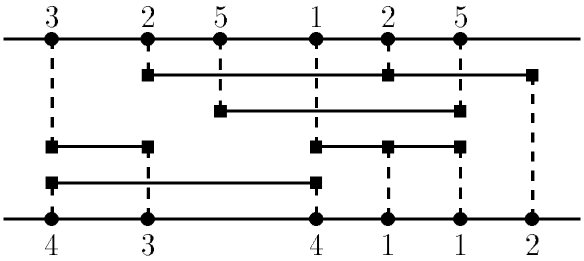

Figure 1.3: A representative channel routing problem in the Manhattan model. The horizontal wires (solid) are in one layer and the vertical (dashed) in the other layer. The vias are represented by squares and the terminals by circles.

wires and one layer for vertical wires. Nets can connect from one layer to another by way of a *contact cut* or *via*; nets cannot intersect one another on the same layer. Thus paths may cross over each other, but they cannot share any segments. Figure 1.3 shows a routing of an example problem.

- **Multi-Layer Models**

  In the *one layer per net* model, each net is routed in one layer. The advantage of this approach is that it does not need any vias; however, the use of layers may be excessive. General models for routing in more than two-layers involve complicated rules concerning contacts: A contact between layer i and layer j may not coincide with any wiring on the layers k in between, $i < k < j$. Often, some layers are reserved for only horizontal wires or only vertical wires.

- **The Knock Knee Model**

  This model was introduced by Thompson [57]. Here wires are disjoint paths, i.e., they are allowed to share corners, but no segment overlap is allowed. Layers must be assigned in such a way that whenever a corner is shared between two wires, they are in different layers. The advantage of this model is that it avoids undesirable electrical properties due to overlaps between different layers. However, the complexity of layer assignment increases significantly, since now we have the freedom to change layers at any "free" grid-point.

Given a channel routing problem and wiring model, there is a minimum separation that is required to route the channel. There are lower bounds on channel width that can serve as predictors for area requirements during the placement phase. They can also be used to justify the quality of routing algorithms. Therefore, it is important to know what the lower bounds are. We now discuss some of the most important bounds.

- **Density**

  The density at a column is defined to be the number of nets whose span crosses the column. The only exception is the *trivial nets* (two-terminal nets whose terminals are on same column), they do not contribute to the density at any column. The channel density d is the maximum over all column densities.

  For the Manhattan wiring model, d is an obvious lower bound on separation; since no wire overlap is allowed in the model, at least as many as d

tracks are necessary to carry the nets that must cross the column which has density d. When more than two interconnection layers are available, a trivial lower bound on separation is density divided by the number of layers on which horizontal routing is allowed. In the knock-knee model, which allows single point overlap between tracks, we can simply redefine spans of nets to be half-closed, half-open intervals (e.g., closed on the left and open on the right). Then with the same definition of density, d is again a lower bound for the two-layer knock-knee model since two nets can now share a track at a column.

- **Vertical Constraints**

  As the vertical segment of one net cannot overlap with that of another net on the same column, a constraint is introduced to avoid this situation. The constraints can be captured by the *vertical constraint graph* (VCG). The VCG is formed by using a node to represent each net and including an edge from net A to net B if a pin for net A appears above a pin B in the same column of the channel. In the Manhattan model, if each net is routed using only one track, then the length of the longest VCG path is a lower bound on the channel width.

- **Flux and Related Bounds**

  Another lower bound on channel width in the two-layer Manhattan model is the *flux*, denoted by f, which was introduced by Baker, Bhatt, and Leighton [1]. They showed that any two-layer channel routing problem can be solved in width $2d + O(f)$, and they note that flux appears to be bounded by a small constant for practical problems.

Of the above bounds, only density is universal. All others apply only to the Manhattan wiring model. In addition, many practical algorithms can achieve widths that are usually within one of the density for Manhattan routing. Therefore, in this thesis, we concentrate on minimizing density for multilayer routing. For single-layer routing, though density is still a lower bound, the channel width may be much larger than the channel density. On the other hand, channel width for single-layer routing is one of the few tractable cases among channel routing problems. Thus, for single-layer routing, we will work with actual channel width.

The channel routing problem has been extensively studied in the literature, from both a theoretical and a practical point of view. In general, density and flux are used to find an upper bound on channel width in those algorithms from a theoretical point of view, while density and the VCG are used in practical algorithms. A recent survey has been provided by LaPaugh and Pinter [34]. They classify the existing channel routing algorithms into "provably good" algorithms and "practically good algorithms". Those algorithms that produce excellent results on real data may have a terrible worst-case performance in theory (both in terms of time complexity and quality of the results), but these cases are unlikely to happen in the real world. On the other hand, an algorithm which has a provably good worst-case performance may virtually always perform worse than a good practical algorithm on real data in practice. It is unlikely to have an algorithm that is both provably and practically good since most variations of the channel routing problem are generally believed to be NP-complete. In fact, rigorous NP-completeness proofs have been provided for various models [35, 61, 52, 56].

1.2 Thesis Overview

In the next two chapters of the thesis, we consider single-layer routing problems. Single-layer routing has become more relevant as technology evolves towards chips with increasing numbers of layers. With many layers, it becomes more likely that an individual layer can be dedicated to a coplanar subset of the original collection of nets. For example, the heuristic multi-layer channel router MulCh [23] improved upon previous multilayer channel routers by dividing the problem into essentially independent subproblems of one, two, or three layers.

In seeking to minimize the separation, we allow the two rows of terminals to be shifted relative to one another by an amount referred to as the *offset*. We consider three specific problems relating to single-layer channel routing. The *optimal offset problem* involves finding the offset that minimizes the amount of separation necessary to route the channel. The *offset range problem* involves finding all offsets that give enough room to route at a given separation. Finally, the *optimal placement problem* considers a scenario in which the terminals on each of the two sides are grouped into several chunks. Within each chunk, the positions of the terminals are fixed, but, on each side of the channel, the chunks can slide back and forth as long as their order does not change. The goal in this problem is to minimize the channel length given a channel width.

When only one side of the channel contains multiple terminals of the same net, Chapter 2 shows that the feasible offset problem, the optimal offset problem, and the optimal placement problem can all be solved in $O(n)$ time, where n is the number of terminals. If the placement of terminals is entirely unrestricted, it is impossible to avoid a worst-case solution time of $\Omega(n^2)$. However, if the number of columns c is $O(n)$, Chapter 3 shows that the feasible offset and optimal offset problems can both be solved in time $O(n^{1.5} \lg n)$, which improves upon a "naive" $O(cn)$ approach. Better running times are obtained when there are no two-sided nets. In case c is large, both of the problems can be solved independent of c in $O(n^2 \lg n)$ time. In addition, the optimal placement problem can be solved in $O(n^3)$ time.

In multi-layer channel routing, determining the width required to route a channel for fixed terminals is NP-complete. However, channel density is a good estimate for channel width in practice. Chapters 4 and 5 develop efficient algorithms to minimize the channel density under the condition that the terminals have some flexibility to move around or to be interchanged.

In Chapter 4, we minimize the density for a channel of fixed length L under the following constraints on the terminal positions. For channels with only *linear order constraints* (the orderings of the terminals on each side of the channel are fixed), we obtain an algorithm with $O(pq \lg(p+q))$ running time and $O(pq)$ space, where p is the number of top terminals, and q is the number of bottom terminals. If we add *separation constraints* (the distance between each pair of consecutive terminals is within a certain range), the running time and space become $O(pqL^2 \lg(p+q))$ and $O(pqL)$, respectively. With multiple modules and fixed terminals within each module, we obtain $O(L^2 \lg(p+q))$ time and $O(L^2)$space. If the terminals within the modules are also movable, then the running time becomes $O(pqL^2 \lg(p+q))$, and the space becomes $O(pqL^2)$. In each of these cases, we improve previous results for running time and space by a factor of $L/\lg n$ and L, respectively.

In Chapter 5, we consider the problem of minimizing channel density by interchanging terminals within cells on the channel sides. In each cell, we are given a set of available positions and a set of terminals, the number of positions may be greater than or equal to the number of terminals. Every terminal belongs to some net and is to be assigned to one of these available positions. Our model differs from the existing models in that most existing models assuming every position within the cell is available for the placement of the terminals. By considering a limited set of positions, we model the assignment of signals to terminal positions on standard cells or gate-array logic blocks once an appropriate sequence of functional blocks has been selected on each side of the channel. Though the problem is NP-hard in general, we can obtain efficient algorithms with the reasonable restriction that at most one terminal per net is to be assigned to each cell. For channels with fixed cells, our running time is $O(n \lg n)$, where n is the number of available positions. In the case when there are multiple components, and cells are fixed within the components while the components are movable, our running time is $O(CLn^2 \lg n)$, where C is the number of components. If the number of available positions is equal to the number of terminals in each cell, then the running time becomes $O(n^3 \lg n)$.

The final part of the thesis, in Chapter 6, provides an algorithm to find a maximum-density planar subset of n 2-pin nets in a channel, which can be helpful in the context of multilayer channel routing. Let t be the number of two-sided nets and w be the number of nets in the output. A simple bound on the running time is shown to be $O(n \lg n + n(t+1)w)$, which improves upon a "naive" $O(n^3)$ approach. In the special case when there are no two-sided nets, or there are no single-sided nets, the bound stated above becomes $O(n \lg n + nw)$.

Chapter 2

Single-Layer Channel Routing and Placement with Single-Sided Nets on One Side

Much attention has been given to planar or single-layer wire routing for VLSI chips. Most popular has been river routing in the restricted sense of the term, the connection of two parallel rows of corresponding points[1], e.g., [19, 38, 46, 47, 54, 58]. Other works have considered routing within a rectangle [10], placement and routing within a ring of pads [2], or routing between very general arrangements of modules [13, 37, 42].

In this chapter, we consider three specific problems relating to single-layer channel routing. In seeking to minimize the separation, we allow the two rows of terminals to be shifted relative to one another by an amount referred to as the *offset* as illustrated in Figure 2.1. (The offset may be positive or negative.) The *optimal offset problem* involves finding an offset that minimizes the amount

[1]This is the only usage of the term "river routing" in this thesis; we refer to more complicated variations of the problem as "single-layer" or "planar" routing.

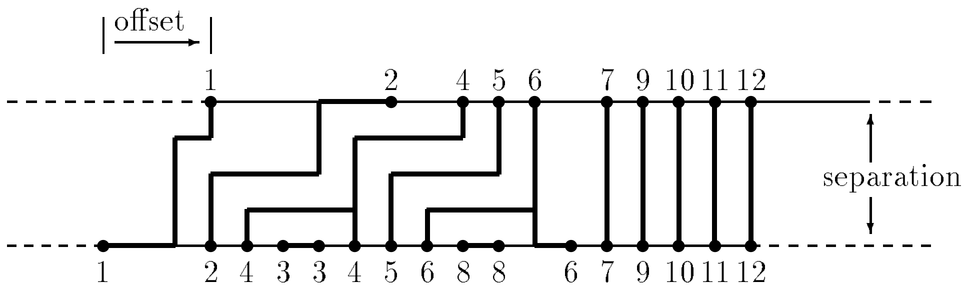

Figure 2.1: A single-layer channel and a routing that achieves the minimum separation. The number next to each terminal identifies the net to which it belongs.

of separation necessary to route the channel. The *offset range problem* involves finding all offsets that give enough room to route at a given separation. Finally, the *optimal placement problem* considers a scenario in which the terminals on each of the two sides are grouped into several chunks. Within each chunk, the positions of the terminals are fixed, but, on each side of the channel, the chunks can slide back and forth as long as their order does not change. The goal in this problem is to minimize the channel length given a channel width. Figure 2.1 shows an optimal offset and a corresponding routing that achieves minimum separation.

Prior work on these problems has been restricted to the *river routing* framework, in which each net has exactly one terminal on each side of the channel [38, 46], but real channels may include nets with many terminals, including multiple connections on the same side of the channel (e.g. some of the examples in [23]). (Such multiple connections are even more likely in the optimal placement problem with multiple modules.) There is no loss of generality in restricting attention to nets that have just two terminals (by a reduction described in [24] that derives from "folklore"), but the river routing rule that the two terminals must be on opposite sides of the channel is overly strict. We refer

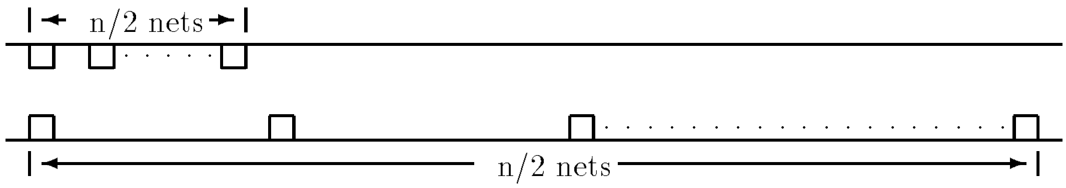

Figure 2.2: For small separation, the number of disjoint intervals of feasible offsets of the channel above is $\Omega(n^2)$

to the type of net allowed in river routing as a *two-sided net*, whereas a net with its two terminals on the same side of the channel is a *single-sided net*.

We show in this chapter that the optimal offset, offset range, and optimal placement problems can all be solved in $O(n)$ time as long as all single-sided nets are on one side of the channel, where n is the number of terminals. Unfortunately, if there are single-sided nets on both sides of the channel, it is impossible to solve all these problems in linear time; there may be $\Omega(n^2)$ disjoint intervals of offsets that allow routing to be completed at a given separation, which is illustrated in Figure 2.2.

Although we restrict attention in this chapter to a rectilinear, grid-based model, the treatment is easily extended to gridless but rectilinear layouts as in [24]. Also for convenience, we allow routing on channel boundaries.

The remainder of this chapter is organized as follows. In Section 2.1, we conveniently express the routability conditions for a channel with all single-sided nets on one side. These conditions are used in Sections 2.2, 2.3, and 2.4 to solve the offset range, optimal offset, and optimal placement problems, respectively. In each of these problems, the running time is $O(n)$, where n is the number of nets.

2.1 Cut Conditions

In this section we use the theory of single-layer routing developed in [42] to derive a routability test for channels with single-sided nets on one side. We justify this routability test carefully, since the literature contains *erroneous* proposals of a similar test for general channels [48].

Without loss of generality, we assume that all single-sided nets are on the bottom side. The *contour* of these single-sided nets is the upper boundary of the routing region consumed in the routing of these nets that minimizes total wire length. That is, when the nets are routed as tightly as possible against the bottom of the channel, the contour is formed by the uppermost nets and portions of the channel boundary.

Our first step for all the problems treated in this paper is to determine the contour of the single-sided nets, which Pinter [48] shows can be done in linear-time:

Lemma 1 (Pinter) *The bendpoints in the contour of a set of n single-sided nets can be found in $O(n)$ time.* ■

Once we find the contour of the single-sided nets, we can replace those nets with the contour. At each column, we define the *extension* of the contour to be the distance that the contour extends into the channel at that column. Let $b_0, b_1, \ldots, b_{t-1}$ denote the x-coordinates of the top terminals of two-sided nets (in sorted order) and $a_0, a_1, \ldots, a_{t-1}$ denote the x-coordinates of the bottom terminals of two-sided nets; a_i is to be connected to b_i, for $0 \leq i \leq t-1$. Also let $c_0, c_1, \ldots, c_{k-1}$ denote the x-coordinates where the contour of single-sided nets bends.

We need a few more definitions in order to invoke Maley's theory. We define a *cut* χ to be a line segment that connects a top and bottom terminal or travels at 45° from a terminal to the opposite side of the channel; these correspond to the pivotal cuts in [42]. The *flow* across χ is the number of nets that must cross χ, namely those nets having terminals on both sides of χ and those having an endpoint of χ as a terminal. The *capacity* of χ is one greater than the maximum of the horizontal and vertical separations of its endpoints; if χ is the line segment from (x_1, y_1) to (x_2, y_2), then

$$capacity(\chi) = max\{|x_1 - x_2|, |y_1 - y_2|\} + 1 \ .$$

The cut χ is *safe* if $flow(\chi) \leq capacity(\chi)$, which means that there is enough space along χ for the wires to get through.

Lemma 2 *A channel is routable if and only if every cut is safe.* ■

This lemma follows from the corresponding result in [42, §2.1]. (Our slightly different definitions of flow and capacity allow Maley's formulation in terms of cuts emanating from "feature" endpoints to correspond to cuts emanating from terminals. Since we allow routing on the channel boundaries, the only "features" are terminals and two routing obstacles (horizontal lines) located one unit outside of what we have been referring to as the channel boundaries.)

We now show that many cuts can be removed from consideration. First, we need not check the cuts emanating from a terminal a that are outside of the "cone" formed by the two 45° cuts emanating from a. This is because pivoting a 45° cut around a so that the other endpoint moves further away increases the capacity at least as much as the flow, so the cuts outside the cone are safe if the 45° cuts are. Now we say that a cut χ is *dominated* by cuts α and β if all

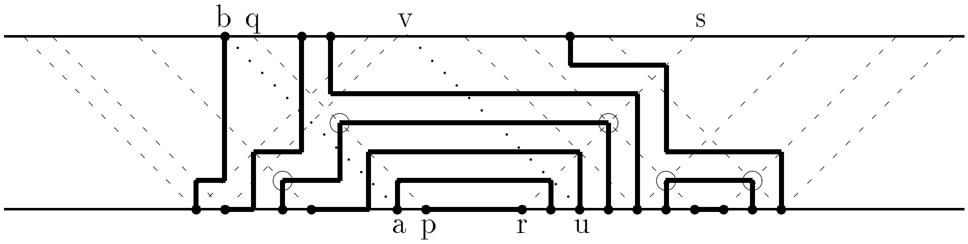

Figure 2.3: Only those 45° cuts from bottom terminals shown with dashed lines need be checked. The hollow circles mark the convex corners of the contour of single-sided nets.

these cuts have the same capacity, and the nets that must cross χ must all cross α or all cross β. In this case, χ need not be checked because it is unsafe if and only if α or β is unsafe. Since all the single-sided nets are on the bottom, many cuts are likely to be dominated by others. In particular, any cut χ connecting to a terminal only at the top of the channel can be eliminated because it is dominated by the two parallel cuts emanating from bottom terminals, one to the left and one to the right, that are nearest to χ. Additionally, any remaining cut that is not a 45° cut is unnecessary because it is dominated by the two 45° cuts emanating from its bottom endpoint. Finally, the cuts from terminals of single-sided nets are unnecessary unless they cross the contour of single-sided nets at a convex corner. For example, in Figure 2.3, cut $\overline{ab}$ has no greater flow than $\overline{pq}$, and $\overline{uv}$ is dominated by $\overline{pq}$ and $\overline{rs}$.

From the above reasoning, we have the following theorem.

Theorem 3 *A channel with all single-sided nets on the bottom is routable if and only if all* 45° *cuts from bottom terminals of two-sided nets and all* 45° *cuts crossing the contour of single-sided nets at a convex corner are safe.* ■

Using this theorem, we can derive conditions for feasibility of separation s and offset d. First define e_i to be the extension of the single-sided contour at

c_i, and t_i to be the number of two-sided nets whose bottom terminals are to the left of c_i. To simplify our presentation, we assume through the rest of this paper that a separate check is performed to ensure that $e_i \leq s$ for all i.

Now we express the constraints related to each b_i by considering separately the cuts from terminals of two-sided nets and then the cuts that cross convex corners of the single-sided contour. In each case, we first consider the cuts that go towards the left from a bottom terminal and then those that go right. For the left-going cuts from terminals of two-sided nets, the condition on b_i is that $b_i + d < a_j - s$ whenever $j \geq i + s + 1$, from which the tightest constraint is clearly $b_i + d < a_{i+s+1} - s$. Similarly, for the right-going cuts, the constraint is $a_{i-s-1} + s < b_i + d$. (At this point, our constraints differ from the river routing constraints in [46] only because we have allowed routing on both channel boundaries.) For the left-going cuts that go through convex corners of the contour, the constraints on b_i are that $b_i + d < c_j + e_j - s$ whenever $t_j + e_j + 1 \geq i + s + 1$. (See Figure 2.4.) By defining c_{r_i} to be the nearest bendpoint to the right of a_i such that $t_{r_i} - i + e_{r_i} \geq s$, we can pick out the tightest constraint on b_i, that $b_i + d < c_{r_i} + e_{r_i} - s$. Finally, by analogously defining c_{l_i} to be the nearest bendpoint to the left of a_i such that $i - t_{l_i} + e_{l_i} > s$, the complete set of constraints on b_i based on Theorem 3 is

$$max\{a_{i-s-1}, c_{l_i} - e_{l_i}\} + s < b_i + d < min\{a_{i+s+1}, c_{r_i} + e_{r_i}\} - s \quad 0 \leq i < t \tag{2.1}$$

Here we define $a_j = -\infty$ if $j < 0$ and $a_j = \infty$ if $j \geq t$; also c_{l_i} (c_{r_i}) is defined to be $-\infty$ (∞) if there is no bendpoint satisfying the necessary conditions.

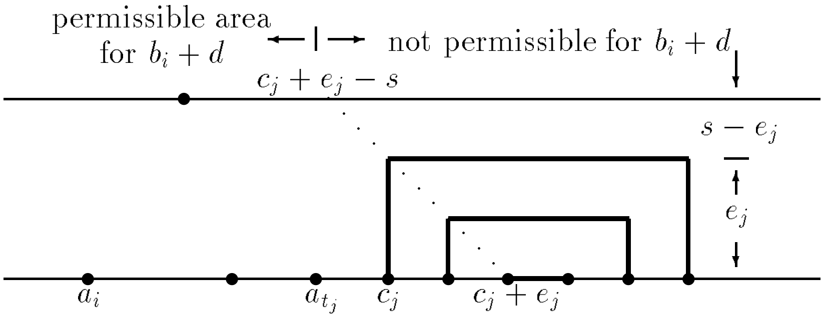

Figure 2.4: Because of the cut shown, $b_i + d$ can only be positioned in the area shown as permissible when $t_j + e_j \geq i + s$.

2.2 The Offset Range Problem

If we let

$$l(s) = \max_{0 \leq i < t} \{a_{i-s-1}, c_{l_i} - e_{l_i}\} + s - b_i$$

and

$$u(s) = \min_{0 \leq i < t} \{a_{i+s+1}, c_{r_i} + e_{r_i}\} - s - b_i \ ,$$

then we know from Condition (2.1) that a pair (s, d) is feasible if and only if $l(s) < d < u(s)$.

Theorem 4 *The offset range problem can be solved in $O(n)$ time.*

Proof. All we need to do is to compute $l(s)$ and $u(s)$. We can find all t_i's, for $0 \leq i < k$, by a linear scan. Furthermore, since l_i is nondecreasing as i increases, we can find all the l_i's in $O(n)$ time. Thus, we can compute $l(s)$ in $O(n)$ time. Similarly, $u(s)$ is computable in $O(n)$ time. ■

2.3 The Optimal Offset Problem

In this section, we use a halving technique as in [46] to solve the optimal offset problem in $O(n)$ time. Let T be the original set of two-sided nets and let L be the original contour of single-sided nets. We define T^e to be the even-numbered two-sided nets, and L^e to be the new contour with the extensions of L divided by two (and rounded down to integral values). Also define optsep(A) to be the minimum separation attainable with an optimal offset for a channel with the set A of nets. (Once optsep(A) is determined, the solution of the offset range problem can be used to determine the optimal offsets.) The following lemma states the relationship between optsep($T \cup L$) and optsep($T^e \cup L^e$).

Lemma 5 *Let* $s = \text{optsep}(T \cup L)$ *and* $s^e = \text{optsep}(T^e \cup L^e)$. *Then* $s \geq 2s^e - 1$ *and* $s \leq 2s^e + 2$.

Proof. Consider any cut crossing p single-sided nets and q two-sided nets of $T \cup L$. Then $p + q \leq s + 1$, and the flow of the cut for $T^e \cup L^e$ is at most $(p+1)/2 + (q+1)/2 \leq \frac{s+1}{2} + 1$. So $s^e + 1 \leq \frac{s+1}{2} + 1$, or $2s^e \leq s + 1$. Similarly, the flow of the cut for $T^e \cup L^e$ is at least $p/2 + (q-1)/2$. Since there exists some cut with $p + q = s + 1$, we have $s^e + 1 \geq \frac{s}{2}$, or $2s^e \geq s - 2$. ■

Theorem 6 *The optimal offset problem can be computed in* $O(n)$ *time.*

Proof. To solve optsep($T \cup L$), we recursively solve optsep($T^e \cup L^e$). From Lemma 5, once we know optsep($T^e \cup L^e$), we only have to check 4 possible separations to achieve the optimal offset. Each separation can be checked in $O(n)$ time using Theorem 4. Thus, the time to solve optsep($T \cup L$) is equal the time to solve optsep($T^e \cup L^e$) plus $O(n)$. Let $T(n)$ be the time complexity to solve optsep($T \cup L$). Then the recursive call optsep($T^e \cup L^e$) has time complexity at most $T(\lceil n/2 \rceil)$. Therefore, $T(n) \leq T(\lceil n/2 \rceil) + O(n)$, which yields $T(n) = O(n)$.

■

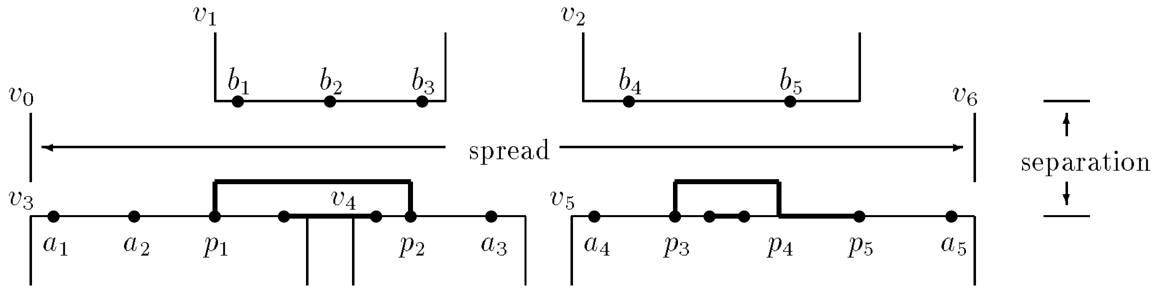

Figure 2.5: Two sets of chunks on either side of a channel. Variables are assigned to the horizontal position of each chunk's left edge.

2.4 The Optimal Placement Problem

The optimal placement problem is defined in [38] as follows. The terminals on each side of the channel are grouped into chunks which must be placed as a unit. On each side of the channel, the order of the chunks is fixed, but their positions are not. As shown in Figure 2.5, the separation is the vertical distance between the two lines of terminals, and the spread is the horizontal dimension of the channel. Given a separation s, the goal is to find a placement which achieves the minimum spread. For simplicity, we assume that terminals can not sit on the corners of the chunks; removing this restriction forces only slight modification to the algorithm.

We first find the contour of the single-sided nets when all the chunks are abutted. From Section 2.1, the condition for the channel to be routable is

$$\alpha_i + s + 1 \leq b_i \leq \beta_i - s - 1, \qquad 0 \leq i < t \ , \tag{2.2}$$

where $\alpha_i = max\{a_{i-s-1}, c_{l_i} - e_{l_i}\}$, and $\beta_i = min\{a_{i+s+1}, c_{r_i} + e_{r_i}\}$. Then the α_i and β_i values all correspond to the position of a terminal on some chunk and can all be found in linear time. The cuts that our routability conditions correspond to all emanate from these terminals, and this set of cuts that must be checked does not change as the placement of the modules is varied.

It now suffices to translate our cut conditions (2.2) into constraints on the module placement in the same fashion as in [38]. Let us number the chunks from 1 to k on the top and $k+1$ to m on the bottom. For each chunk i, let the variable v_i represent the horizontal position of its left edge. Any placement can therefore be specified by an assignment of values of these variables. Also add two variables, v_0 and v_{m+1}, to the set of variables to represent the left and right boundaries of the channel. The spread is thus $v_{m+1} - v_0$.

Now, if terminals α_i, b_i, and β_i lie on chunks f, g, and h, respectively, the constraint $\alpha_i + s + 1 \leq b_i \leq \beta_i - s - 1$ can be rewritten as $v_g - v_f \geq r_{gf}$ and $v_h - v_g \geq r_{hg}$, where r_{gf} and r_{hg} incorporate $s+1$ and the offsets of the terminals from the left edge of their respective chunks. For each pair of adjacent chunks i and $i+1$ on the same side, there is an additional constraint $v_{i+1} - v_i \geq w_i$, where w_i is the width of chunk i. We also have constraints $v_1 - v_0 \geq 0, v_{k+1} - v_0 \geq 0, v_{m+1} - v_k \geq w_k$, and $v_{m+1} - v_m \geq w_m$ for boundary conditions.

Now define a *placement graph* $G(V, E)$ to be a directed graph such that each vertex represents a variable v_i, and a directed edge with weight λ_{fg} goes from v_f to v_g if there is a constraint of the form $v_g - v_f \geq \lambda_{fg}$. The goal of minimizing the spread $v_{m+1} - v_0$ subject to the routability constraints can be achieved by solving a single-source-longest-paths problem in the placement graph. Furthermore, It is shown in [38] that if a placement graph satisfies Lemma 7 below, then linear time suffices to solve the necessary longest-paths problem. Thus, we need only show Lemma 7 holds despite having generated our constraints from a more general arrangement of nets than in [38].

To state Lemma 7, we define a partial order $\prec$ on the vertices so that $u \prec v$ when the chunks corresponding to u and v lie on the same side of the channel and u's chunk is to the left of v's chunk. The left boundary vertex v_0 precedes all other vertices, and all vertices precede the right boundary vertex v_{m+1}. The partial order $\preceq$ is the natural extension to $\prec$ that includes equality. Also define a *cross* edge to be an edge corresponding to chunks on opposite sides of the channel.

Lemma 7 *Any placement graph G=(V,E) has the following properties:*
(1) There do not exist cross edges (u,v) and (x,y) such that $u \prec x$ and $y \prec v$.
(2) There do not exist cross edges (u,v) and (x,y) such that $v \prec x$ and $y \prec u$.

Proof. Note first that α_i and β_i are nondecreasing as i increases. (To see this for α_i, just note that a_{i-s-1} and $c_{l_i} - e_{l_i}$ are nondecreasing.) Now we prove properties (1) and (2) by contradiction.
(1) There are two cases for a pair of edges violating property (1), each of which yields a contradiction:

Case I: The edge (u, v) is caused by the terminals b_i in v and α_i in u, and the edge (x, y) is caused by the terminals b_j in y and α_j in x. Since $y \prec v$, b_j is to the to the left of b_i, i.e., $j < i$. Then, since α_i is nondecreasing as i increases, we have $\alpha_j \leq \alpha_i$, which implies $x \preceq u$, a contradiction.

Case II: The edge (u, v) is caused by the terminals b_j in u and β_j in v, and the edge (x, y) is caused by the terminals b_i in x and β_i in y. By a similar argument to Case I, we can get a contradiction.

(2) There are also two cases for a pair of edges violating property(2); again each case yields a contradiction.

Case I: The edge (u, v) is caused by the terminals b_i in v and α_i in u, and the edge (x, y) is caused by the terminals b_j in x and β_j in y. Since $v \prec x$, b_i is to the left of b_j, i.e., $i < j$. Then, since β_i is nondecreasing as i increases, we have $\beta_i \leq \beta_j$. Also $\alpha_i \leq \beta_i$ by definition; therefore, $\alpha_i \leq \beta_i \leq \beta_j$, which implies $u \preceq y$, a contradiction.

Case II: The edge (u, v) is caused by the terminals b_j in u and β_j in v, and the edge (x, y) is caused by the terminals b_i in y and α_i in x. Using a similar argument as in Case I yields a contradiction. ■

As indicated above, the proof of Lemma 7 immediately yields the main result of this section:

Theorem 8 *The optimal placement problem can be solved in $O(n)$ time.*

■

Chapter 3

Feasible Offset and Optimal Offset for Single-Layer Channel Routing

This chapter considers the *feasible offset problem* and optimal offset problem for general single-layer routing. The feasible offset problem has the same definition as the offset range problem defined in Chapter 2, but the property that the feasible offsets always constitute a single continuous range does not hold for channels with single-sided nets on both sides. The existence of single-sided nets on both channel sides causes significant algorithmic complication. This chapter is mostly derived from a joint paper with Ronald Greenberg of University of Maryland [25].

One complication that arises when single-sided nets are included is that the solution time is no longer insensitive to the number of columns in the problem (at least for feasible offset). Let c be the total number of grid columns from the leftmost terminal to the rightmost terminal, and n be the number of terminals. As illustrated in Figure 2.2, if c is large, the number of disjoint intervals of feasible offsets may be $\Omega(n^2)$. But if $c = O(n)$, we show that feasible offset can be solved in $O(n^{1.5} \lg n)$ time. This improves on the naive $O(cn)$ time obtained by running the $O(n)$ algorithm for the minimum separation problem at each of the $2c$ offsets that may need to be checked. In the remainder of this chapter, we express our running times in terms of c as well as n where necessary, but we concentrate on obtaining a good running time when $c = O(n)$. Later, we give an algorithm that is less efficient for $c = O(n)$ but has a running time independent of c.

The remainder of this chapter is organized as follows. In Section 3.1, we introduce some additional terminology and notation, and we show how to solve the feasible offset problem for a channel in which all nets are single-sided. In this case, the running time with $c = O(n)$ is $O(n^{1.5}\sqrt{\lg n})$, which leads to an $O(n^{1.5}\sqrt{\lg n})$ algorithm for optimal offset. (The optimal offset problem as defined above is trivial when all nets are single-sided; large offset minimizes separation. But we can handle a nontrivial generalization of the problem in which certain offsets are disallowed.) In Section 3.2, we show how to combine ideas from Section 3.1 with some new ideas to obtain solutions for channels with both single-sided and two-sided nets. For the general channel, the running time to solve either feasible offset or optimal offset is $O(n^{1.5}\lg n)$. In Section 3.3, we show how to solve the problems in time $O(n^2 \lg n)$ independent of c. We also note that the optimal placement problem can be handled in $O(n^3)$ time.

3.1 Channels with Single-Sided Nets Only

In this section, we deal with the special case of channels with only single-sided nets. Much of the work we do here will help us in the next section where we consider channels that have both single-sided and two-sided nets.

We begin by explaining some notation and terminology that we use throughout this paper. First, we refer to single-sided nets that have their two terminals on the top (bottom) as *upper (lower) nets.* Also define L, U, and T to be the sets of lower, upper, and two-sided nets, respectively, and N to be the complete set of nets in the channel. In addition, we often use the same notation interchangeably for a set of nets or for a lower or upper contour. We also refer often to subsets of contours, which simply means restricting the contour to certain columns (even though there may be no set of nets that would generate the resulting contour). We use the notations FOP and OOP to refer to the feasible offset problem and optimal offset problem, respectively. We also use the more precise notation FOP(s, A) to represent the set of solutions to the feasible offset problem with separation s and the set A of nets (or contours or contour fragments). We also use analogous notations SSFOP and SSOOP for the corresponding problems when all nets are single-sided. (For optimal offset, we permit the problem specification to disallow some set of offsets, e.g., all offsets $\geq c/2$; otherwise SSOOP is trivial.)

Our first step in solving SSFOP is to find the contours of the upper and lower nets, which can be solved by Lemma 1 in $O(n)$ time. Once we find the contours of the upper and lower nets, SSFOP can be expressed simply in terms of these contours. Then we are simply seeking all offsets for which no vertical cut corresponds to extensions of the upper and lower contours that sum to more than the separation under consideration.

We begin with a general lemma allowing us to decompose SSFOP into smaller instances of the problem. In each of the smaller problems, we use only a portion of the lower contour, while retaining the entire upper contour. In fact, the lemma applies even when there are also two-sided nets. (Naturally, we also could switch the roles of the lower and upper contour.)

Lemma 9 *Let* $L_1, L_2, \ldots, L_k$ *be any subsets of the contour* L *of the lower nets such that* $L_1 \cup L_2 \cup \cdots \cup L_k = L$*, and let* A *be an additional set of nets, then* $\mathrm{FOP}(s, L \cup A) = \bigcap_{i=1}^{k} \mathrm{FOP}(s, L_i \cup A)$.

Sketch of proof. This follows from the fact that routing is possible if and only if each line segment connecting gridpoints is long enough (in the L_∞ metric) to accommodate the number of nets that must cross it. More detail on the theory of single-layer routability can be found in [42]. ∎

We now proceed to decompose the lower contour into pieces that are easier to handle and are not too numerous. The next three lemmas are directed towards handling pieces of the contour that have large extension, and the following two lemmas handle portions of the contour in which there are not too many distinct extensions. Then we show how to put these two ideas together to solve the entire problem.

For the next lemma, we define a special type of contour fragment, such that if it comprises the entire lower contour, then SSFOP is particularly easy to solve. A *monotonic* subset of the lower contour L is a subset of L, such that the extensions within the selected columns are monotonically nondecreasing or monotonically nonincreasing as we move across the columns.

Lemma 10 *If* L_m *is a monotonic subset of the lower contour, and* U *is the upper contour, then we can solve* $\mathrm{SSFOP}(s, L_m \cup U)$ *in* $O(n)$ *time.*

Sketch of proof. Without loss of generality, assume the monotonic piece has (nondecreasing) extensions $l_a, l_{a+1}, \ldots, l_{a+k}$ from left to right, where subscripts indicate the columns covered, and denote the upper extensions by $u_1, u_2, \ldots, u_c$. For simplicity, we pad out the sequence of upper extensions with c 0's on both sides and denote the new sequence as $u_{-c+1}u_{-c+2}\cdots u_{2c}$. Then the procedure in Figure 3.1 solves the problem in $O(c)$ time. The idea of the procedure is to consider vertical cuts that intersect the monotonic piece of contour as it slides across the channel from a far left position (highly negative offset) to a far right position. As the monotonic lower contour moves left to right, we check the relevant extensions of the upper contour in a left to right fashion as well. We never have to recheck an upper extension that did not lead to infeasibility at a previous offset, since the lower contour is nondecreasing. Furthermore, each time that an upper extension does lead to infeasibility, we know that we should increment the offset, and we can only increment the offset $O(c)$ times. (Offsets beyond c columns in either direction are guaranteed to be feasible.) Though we have only shown a running time of $O(c)$, we can actually solve the problem in $O(n)$ time, because we really only need to look at columns where the upper contour bends. ■

In the next lemma, we show that not only are monotonic pieces of contour easy to handle, but that we don't have to check too many of them as long as we restrict attention to sections of contour with large extension. Here we define a monotonic subset to be *maximal* if no other monotonic subset contains it. Now we bound the number of maximal monotonic subsets in the portion of the contour with extension at least h.

Lemma 11 *Let L_g be the subset of the lower contour containing only extensions greater than or equal to h, then L_g contains at most $c/2h$ maximal monotonic pieces.*

```
q ← 0
for i ← −c to 2c do
    while q > 0 and l_{a+q} + u_i > s
        q ← q − 1
    endwhile
    if l_{a+q} + u_i ≤ s
    then
        q ← q + 1
    endif
    if q = k + 1
    then
        mark i − (a + k) as a feasible offset
        q ← q − 1
    endif
endfor
```

Figure 3.1: The pseudocode to solve SSFOP for a monotonic subset of L.

Proof. To have a maximal monotonic piece of the lower contour with extensions of at least h, there must be h lower nets nested one inside the next. Therefore, a maximal monotonic piece with extensions greater than or equal to h must span at least $2h$ columns, so L_g contains at most $c/2h$ maximal monotonic pieces. ■

We can now put together Lemmas 9, 10 and 11 to solve SSFOP efficiently for any piece of lower contour in which all extensions are large enough:

Lemma 12 *If L_g is a subset of the lower contour containing only extensions greater than or equal to h, and U is the upper contour, then we can solve* SSFOP$(s, L_g \cup U)$ *in $O(cn/h)$ time.*

Proof. By Lemma 9, we know that it suffices to solve the problem independently for each of the maximal monotonic pieces of L_g. By Lemma 11 there are $O(c/h)$ such pieces, and by Lemma 10, $O(n)$ time suffices for each piece. ■

What remains is to solve the SSFOP for the portion of the lower contour with small extensions. (Later, we'll show how to choose h, the dividing point between large and small extensions.) The next lemma handles the simplified case in which all extensions on the lower contour are 0 or 1. The following lemma goes on to handle h distinct extensions.

Lemma 13 *If all extensions are* 0 *or* 1*, we can solve* SSFOP *in $O(c \lg c)$ time.*

Proof. The only interesting case is separation 1, and the feasible offsets correspond to the zero entries in the convolution of the upper and lower extensions. The convolution can be computed in $O(c \lg c)$ time by using the Fast Fourier Transform method. (See [17], for example.) ■

Lemma 14 *If all extensions of the lower contour are at most h, we can solve* SSFOP *in $O(hc \lg c)$ time.*

Proof. From Lemma 9, $\mathrm{SSFOP}(s, L \cup U) = \bigcap_{i=1}^{h} \mathrm{SSFOP}(s, L_i \cup U)$, where L_i is the subset of the lower contour with extension i. We can now solve $\mathrm{SSFOP}(s, L_i \cup U)$ using Lemma 13 after assigning 1 to the lower extensions in L_i and those upper extensions exceeding $s - i$, and 0 to the other extensions. Since we have a total of h problems, each solvable in $O(c \lg c)$ time, the total time is $O(hc \lg c)$. ■

Now we can provide an overall solution to SSFOP by combining our results for contours with large extensions and contours with small extensions.

Theorem 15 SSFOP *can be solved in $O(c\sqrt{n \lg c})$ time.*

Proof. $\mathrm{SSFOP}(s, L \cup U) = \mathrm{SSFOP}(s, L_l \cup U) \cap \mathrm{SSFOP}(s, L_g \cup U)$, where L_l is the subset of L with extensions less than h and L_g is the subset of L with extensions greater than or equal to h. Using Lemmas 14 and 12 with $h = \sqrt{n / \lg c}$, the solution time is $O(c\sqrt{n \lg c})$. ■

Corollary 16 SSFOP *can be solved in $O(n^{1.5}\sqrt{\lg n})$ time for $c = O(n)$.*

■

We solve SSOOP in the same time as SSFOP by again adapting the halving technique used in Section 2.3. Let U^e be the contour obtained from U by dividing each extension by two (and rounding down to integral values), while L^e and the optsep function are as in Section 2.3. Now if we let $s = \mathrm{optsep}(L \cup U)$ and $s^e = \mathrm{optsep}(L^e \cup U^e)$, the relationship $2s^e \leq s \leq 2s^e + 2$ can be derived by a similar technique to Lemma 5. This relationship allows us to solve SSOOP:

Theorem 17 SSOOP *can be solved in* $O(n^{1.5}\sqrt{\lg n})$ *time when* $c = O(n)$.

Proof. To find optsep$(L \cup U)$, we recursively find $s^e = \text{optsep}(L^e \cup U^e)$. Then we need only determine which of the separations $2s^e$, $2s^e + 1$, and $2s^e + 2$ have feasible offsets for $L \cup U$. Letting $T(n)$ be the solution time for optsep with maximum extension n, and using Theorem 15, we have $T(n) \leq T(\lfloor n/2 \rfloor) + O(c\sqrt{n \lg c})$, which yields $T(n) = O(c\sqrt{n \lg c})$. By letting $c = O(n)$, we have $T(n) = O(n^{1.5}\sqrt{\lg n})$. ∎

3.2 General Single-Layer Channel

In this section, we use the ideas of section 3.1 to solve FOP and OOP when there are two-sided as well as single-sided nets. As before, we begin by computing the contours of the upper and lower nets. Also as before, we consider separately the portions of the lower contour with large extensions and the portions with small extensions and then show how to put these ideas together.

To deal with the extra complications of two-sided nets, we also must introduce two new definitions.

First, let L_0 be a subset of the contour of the lower nets and T a set of two-sided nets whose lower terminals are to the left of L_0. Define T/L_0 as the upper contour obtained by pulling up the lower terminals of the nets in T and reconnecting them to the upper side to the left of preexisting terminals. That is, we convert the nets in T to single-sided nets without violating planarity and without moving what were their lower terminals to the wrong side of L_0. This notation is also used analogously for any set A of upper and two-sided nets as long as the lower terminal of each two-sided net is to the left or right of all nonzero

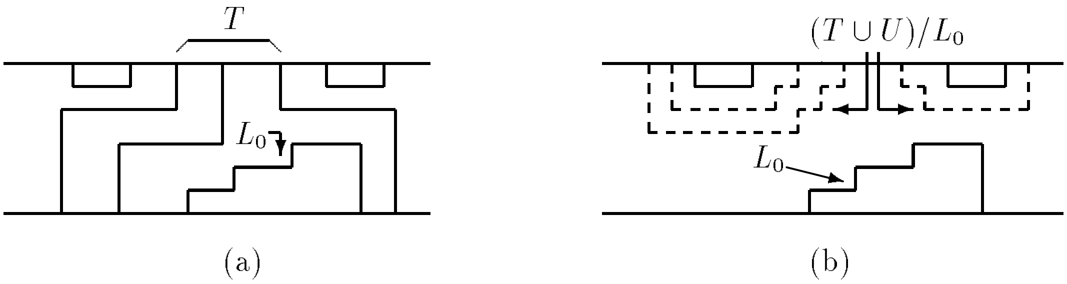

Figure 3.2: The effect of two-sided nets in (a) is incorporated into the top contour in (b). In this figure, L_0 is a monotonic *portion* of the lower contour.

extensions in L_0. In all cases, A/L_0 is the upper contour formed by moving lower terminals in T away from L_0 and to the upper side. Finally, the notation can also be used with a portion of the upper contour, in which case "upper" and "lower" are reversed throughout the definition. Figure 3.2 illustrates the definition of $(T \cup U)/L_0$.

For the second definition, let M be a subset of the contour of the upper or lower nets. We define $M|_s$ to be a new contour in which we replace all extensions exceeding s with extension s.

We now proceed in the next two lemmas to handle a portion of lower contour with only large extensions. As before, the first lemma shows how to handle a monotonic piece of lower contour, and the second lemma handles a contour portion with large extensions by dividing it into maximal monotonic pieces.

Lemma 18 *Let A be a set of upper and two-sided nets. Then we can solve* FOP$(s, L_m \cup A)$ *in $O(n)$ time, where L_m is a monotonic portion of the lower contour.*

Sketch of proof. The solution is the intersection of the feasible offsets from (1) solving FOP without L_m (using Lemma 4) and (2) solving FOP by considering (at each offset) only those columns L_m intersects. The idea for subproblem (2) is to reroute the two-sided nets in the fashion shown in Figure 3.2. By doing so we incorporate into the upper contour any effect of two-sided nets that can come into play with L_m. But we may be building up the upper contour excessively in columns beyond the range of L_m, which effect is eliminated by truncating the new upper contour to extension s. (This truncation does not hurt when columns of the upper contour fall into the range of L_m, since any upper extension of s or greater will be enough to flag infeasibility when matched to any (nonzero) extension of L_m.) By this reasoning, the solution to subproblem (2) is given by $\mathrm{SSFOP}(s, L_m \cup (U \cup T/L_m)|_s)$. Using lemma 1 and lemma 10, we can solve $\mathrm{SSFOP}(s, L_m \cup (U \cup T/L_m)|_s)$ in $O(n)$ time. ■

Now we combine Lemmas 9, 18 and 11 to solve FOP for a subset of L with large extensions. As before, we define large as exceeding h and specify the value of h later.

Lemma 19 *If L_g is a subset of L containing only extensions greater than or equal to h, then we can solve* $\mathrm{FOP}(s, L_g \cup U \cup T)$ *in $O(cn/h)$ time.* ■

Proof. By lemma 9, L_0 can be decomposed into maximal monotonic subsets $L_1, L_2, \ldots, L_k$. Using lemma 18, we have

$$\begin{aligned}
\mathrm{FOP}(s, L_g \cup U \cup T) &= \bigcap_{i=1}^{k} \mathrm{FOP}(s, L_i \cup U \cup T) \\
&= \bigcap_{i=1}^{k} [\mathrm{FOP}(s, U \cup T) \cap \mathrm{FOP}(s, L_i \cup (U \cup T/L_i)|_s)] \\
&= \mathrm{FOP}(s, U \cup T) \cap \bigcap_{i=1}^{k} \mathrm{FOP}(s, L_i \cup (U \cup T/L_i)|_s) \ .
\end{aligned}$$

Now, k is $O(c/h)$ by lemma 11. $\mathrm{FOP}(s, U \cup T)$ and each $\mathrm{FOP}(s, L_i \cup (U \cup T/L_i)|_s)$ can be solved in $O(n)$ time by Lemma 18. Thus, the total running time is $O(cn/h)$. ■

Now that we have taken care of FOP with large extensions, we use the next two lemmas to deal with small extensions. The next lemma tells us how to transform certain instances of FOP into SSFOP, and will be used in handling general instances of FOP with small extensions.

Lemma 20 *Let T_l and T_r be two sets of two-sided nets such that all the nets in T_l are to the left of those in T_r. (That is, the upper terminals in T_l are to the left of those in T_r and similarly for the lower terminals.) Also let U_l be a set of upper nets in which all terminals are to the left of (the upper terminals of) T_r, and L_r a set of lower nets in which all terminals are to the right of T_l. (See Figure 3.3.) Then,*

$$\begin{aligned}\mathrm{FOP}(s, U_l \cup T_l \cup T_r \cup L_r) &= \mathrm{FOP}(s, U_l \cup T_l \cup T_r) \cap \mathrm{FOP}(s, L_r \cup T_l \cup T_r) \cap \\ &\quad \mathrm{FOP}(s, ((U_l \cup T_l)/L_r)|_s \cup ((L_r \cup T_r)/U_l)|_s)\ .\end{aligned}$$

Proof. The argument is similar to the one for lemma 18. At any given offset that is infeasible, either there is a vertical cut demonstrating infeasibility that goes through both U_l and L_r, or there is not. In the former case, we know that we can incorporate the effect of the two-sided nets into the upper and lower contours; i.e., solving $\mathrm{FOP}(s, ((U_l \cup T_l)/L_r)|_s \cup ((L_r \cup T_r)/U_l)|_s)$, as illustrated in Figure 3.3, will rule out the infeasible offsets of the first type. On the other hand, if the infeasibility does not result from interaction between U_l and L_r, it suffices to solve $\mathrm{FOP}(s, U_l \cup T_l \cup T_r)$ and $\mathrm{FOP}(s, L_r \cup T_l \cup T_r)$. Thus, intersection of the feasible offsets from these three problems provides the feasible offsets for the original problem. ■

We can now solve FOP with small extensions:

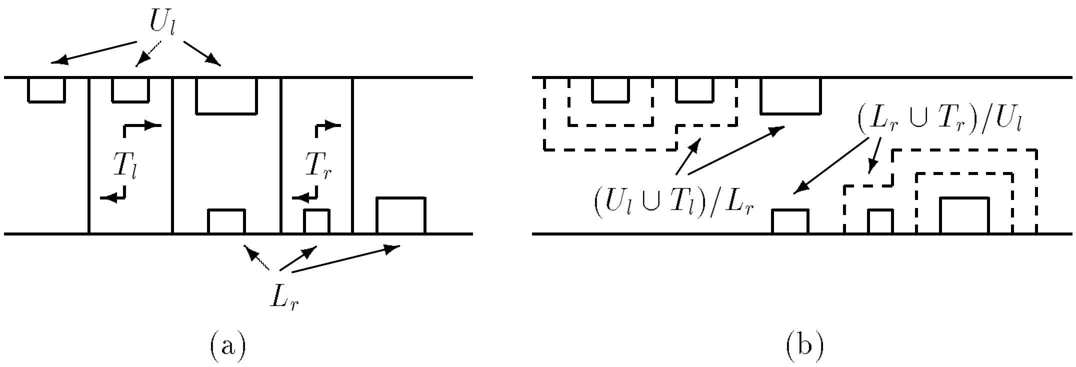

Figure 3.3: The effect of two-sided nets in (a) is incorporated into the upper and lower contours in (b).

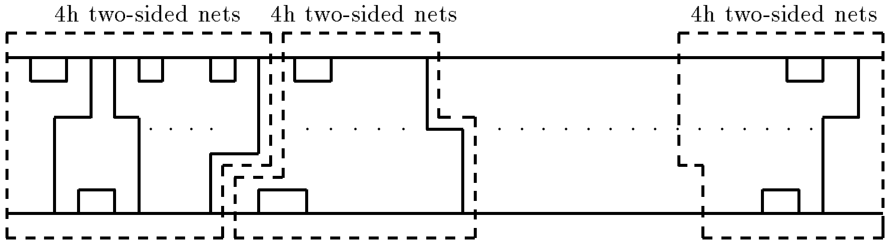

Figure 3.4: The partition for $s < 4h$.

Lemma 21 *If the extensions of the upper and lower contours are all less than h, then we can solve* FOP *in $O(hc\lg^2 c)$ time.*

Proof. Let t be the number of two-sided nets. We first consider the case when $s < 4h$. Divide the channel into $t/4h$ blocks $B_1, B_2, \ldots, B_{t/4h}$, each spanning $4h$ two-sided nets as shown in Figure 3.4. Let L_i, U_i, and T_i denote the lower nets, upper nets, and two-sided nets in block i. (Single-sided nets at a boundary between blocks of two-sided nets are assigned to exactly one of those blocks.) Since the upper side and lower side of a block may not be of the same length, we define c_i to be the sum of the number of columns the upper side spans and the number of columns the lower side spans.

From lemma 9, $\mathrm{FOP}(s, N) = \bigcap_{i=1}^{t/4h} \mathrm{FOP}(s, L_i \cup T \cup U)$. Since $s < 4h$, there must be fewer than $4h$ two-sided nets through any vertical cut at any feasible offset. Therefore, any offset with vertical cuts through L_i and U_j for $j > i+1$ or $j < i-1$ would be an unfeasible offset, because such a cut would be crossed by all the nets in T_{i+1} or T_{i-1}. Thus we can write

$$\mathrm{FOP}(s, N) = \bigcap_{i=1}^{t/4h} [\mathrm{FOP}(s, L_i \cup T \cup U_i) \cap \mathrm{FOP}(s, L_i \cup T \cup U_{i+1}) \cap \\ \mathrm{FOP}(s, L_i \cup T \cup U_{i-1})] \ .$$

Note also that no vertical cuts through both L_i and U_j can cut any two-sided nets outside blocks i through j, so we can rewrite $\mathrm{FOP}(s, N)$ as

$$\mathrm{FOP}(s, N) = \bigcap_{i=1}^{t/4h} [\mathrm{FOP}(s, L_i \cup T_i \cup U_i) \cap \mathrm{FOP}(s, L_i \cup T_i \cup T_{i+1} \cup U_{i+1}) \cap \\ \mathrm{FOP}(s, L_i \cup T_{i-1} \cup T_i \cup U_{i-1})] \ .$$

Now we can solve each of $\mathrm{FOP}(s, L_i \cup T_i \cup T_{i+1} \cup U_{i+1})$ and $\mathrm{FOP}(s, L_i \cup T_{i-1} \cup T_i \cup U_{i-1})$ in time $O(h(c_{i-1} + c_i + c_{i+1}) \lg(c_{i-1} + c_i + c_{i+1}))$ as follows. We use Lemma 20 to further decompose the problem, Lemma 1 for the computation of new contours (which will still have $O(h)$ extensions), and Lemmas 4 and 14 to solve the subproblems.

To solve $\mathrm{FOP}(s, L_i \cup T_i \cup U_i)$, we use a recursive method, for which we consider the general problem of solving $\mathrm{FOP}(s, L^* \cup T^* \cup U^*)$ with $|T^*| = t^* \leq 4h$. We decompose such a problem into left and right blocks, each having half as many two-sided nets as the original. Using subscripts l and r to denote the portions of L^*, T^*, and U^* falling in the left and right blocks, we know from Lemma 9 that

$$\begin{aligned} \mathrm{FOP}(s, L^* \cup T^* \cup U^*) &= \mathrm{FOP}(s, L_l^* \cup T^* \cup U^*) \cap \mathrm{FOP}(s, L_r^* \cup T^* \cup U^*) \\ &= \mathrm{FOP}(s, L_l^* \cup T_l^* \cup U_l^*) \cap \mathrm{FOP}(s, L_l^* \cup T^* \cup U_r^*) \cap \\ &\quad\ \mathrm{FOP}(s, L_r^* \cup T^* \cup U_l^*) \cap \mathrm{FOP}(s, L_r^* \cup T_r^* \cup U_r^*) \ . \end{aligned}$$

The restrictions from T^* to T_l^* and T_r^* are by reasoning similar to that used above. Also as above, we use Lemma 20, Lemma 1, Lemma 4, and Lemma 14 to solve FOP$(s, L_l^* \cup T^* \cup U_r^*)$ and FOP$(s, L_r^* \cup T^* \cup U_l^*)$ in time $O(hc^* \lg c^*)$, where c^* is the sum of the number of top and bottom columns spanned by L^*, T^*, and U^*. FOP$(s, L_l^* \cup T_l^* \cup U_l^*)$ and FOP$(s, L_r^* \cup T_r^* \cup U_r^*)$ are just recursive calls; we denote the sum of the number of top and bottom columns in these subproblems as c_l^* and c_r^*. Then, the time $T(t^*, c^*)$ to compute FOP$(s, L^* \cup T^* \cup U^*)$ can be written as

$$T(t^*, c^*) = T(t^*/2, c_l^*) + T(t^*/2, c_r^*) + O(hc^* \lg c^*) \qquad t^* \leq 4h \ ,$$

where $c_l^* + c_r^* = c^*$, and $T(1, c) = O(hc \lg c)$. There are $O(\lg t^*)$ stages in the recursion, and the total work on all subproblems at any given stage is $O(hc^* \lg c^*)$ time. Thus, $T(t^*, c^*) = O(hc^* \lg c^* \lg t^*)$, and, in particular,

$$T(4h, c_i) = O(hc_i \lg c_i \lg h) = O(hc_i \lg^2 c) \ .$$

Putting everything together, the time to solve FOP(s, N) is

$$\begin{aligned}
T(t, c) &= \sum_{i=1}^{t/4h} T(4h, c_i) + \sum_{i=1}^{t/4h} O((c_{i-1} + c_i + c_{i+1}) h \lg c) \\
&= \sum_{i=1}^{t/4h} O(hc_i \lg^2 c) + O(hc \lg c) \\
&= O(hc \lg^2 c) + O(hc \lg c) \\
&= O(hc \lg^2 c).
\end{aligned}$$

Finally, we must return to solving the original problem in the case that $s \geq 4h$. We divide the channel into $t/2h$ blocks, each spanning $2h$ two-sided nets. From Lemma 9, FOP$(s, N) = \bigcap_{i=1}^{t/2h}$ FOP$(s, L_i \cup T \cup U)$. Furthermore, at any offset, we need not consider any vertical cut for which the number of two-sided nets crossing the cut is less than $s - 2h$ or greater than s. In the former case, we know the cut cannot provide evidence of infeasibility; in the latter case infeasibility is guaranteed. Thus, we can write FOP(s, N) as

$$\bigcap_{i=1}^{t/2h}[\mathrm{FOP}(s, L_i \cup T_i \cup T_{i+1} \cup \cdots \cup T_{i+s/2h} \cup U_{i+s/2h-1} \cup U_{i+s/2h}) \cap \\ \mathrm{FOP}(s, L_i \cup T_i \cup T_{i-1} \cup \cdots \cup T_{i-s/2h} \cup U_{i-s/2h+1} \cup U_{i-s/2h})].$$

At this point, we could proceed as when $s < 4h$ by incorporating two-sided nets into the upper and lower contours, but there are too many two-sided nets to get the desired running time; we might end up with more than $O(h)$ distinct extensions in the contours. Instead, we take out the $s-4h$ two-sided nets between L_i and $U_{i+s/2h-1}$, and the $s - 4h$ two-sided nets between L_i and $U_{i-s/2h+1}$, and we decrease s by $s - 4h$. Each unfeasible offset in this new problem actually denotes the center of a range of $2(s - 4h) + 1$ unfeasible offsets of the original problem, but with this proviso, the task is to solve

$$\bigcap_{i=1}^{t/2h}[\mathrm{FOP}(4h, L_i \cup T_i \cup T_{i+s/2h-1} \cup T_{i+s/2h} \cup U_{i+s/2h-1} \cup U_{i+s/2h}) \cap \\ \mathrm{FOP}(4h, L_i \cup T_i \cup T_{i-s/2h+1} \cup T_{i-s/2h} \cup U_{i-s/2h+1} \cup U_{i-s/2h})] \ .$$

We can solve these subproblems using Lemmas 20, 1, 4, and 14 as before. Also, with a similar analysis for the combined running time of the subproblems, we get a total time of $O(hc \lg c)$. ■

Theorem 22 FOP *can be solved in* $O(c\sqrt{n} \lg c)$ *time.*

Proof. We can use Lemma 9 to write $FOP(s, N) = FOP(s, L_l \cup T \cup U_l) \cap FOP(s, L_g \cup T \cup U) \cap FOP(s, U_g \cup T \cup L)$, where L_l is the subset of L with extensions less than h, and L_g is the subset of L with extensions greater than or equal to h, and similarly for U_l and U_g. The first subproblem can be solved in $O(hc \lg^2 c)$ time using Lemma 21, and the latter two subproblems can be solved in $O(cn/h)$ time using Lemma 19. By letting $h = \sqrt{n}/\lg c$, FOP can be solved in $O(c\sqrt{n}\lg c)$ time. ∎

Corollary 23 FOP *can be solved in* $O(n^{1.5}\lg n)$ *time for* $c = O(n)$. ∎

Corollary 24 OOP *can be solved in* $O(n^{1.5}\lg n)$ *time when* $c = O(n)$.

Sketch of proof. The proof is similar to the proof of Theorem 17. ∎

3.3 Further Results

We have shown how to solve the feasible offset and optimal offset problems for single-layer channel routing with $c = O(n)$ in time $O(n^{1.5}\lg n)$. (In addition, the time is $O(n^{1.5}\sqrt{\lg n})$ when all nets are single-sided and $O(n)$ when all single-sided nets are on one side of the channel.) Furthermore, we have shown how to solve the feasible offset problem in $O(c\sqrt{n}\lg c)$ time for any value of c, which improves upon the naive bound of $O(cn)$ except for very large values of c. But we can do even better than $O(c\sqrt{n}\lg c)$ for moderately large values of c, i.e., larger than about $n^{1.5}$. Essentially all the necessary machinery is already in place for this result that FOP can be solved in $O(n^2\lg n)$ time independent of the number of columns:

Theorem 25 FOP *can be solved in* $O(n^2\lg n)$ *time.*

Proof. Using Lemma 9, we decompose the lower contour into maximal monotonic subsets. Since there are only n nets, we have at most n monotonic subsets. We can find the feasible offsets for each subset in $O(n)$ time using Lemma 18. The total time to find the feasible offsets for all of the subsets of the lower contour is $O(n^2)$. Furthermore, for each subset, the set of feasible offsets can be output as a list of at most n nonoverlapping intervals with all the interval endpoints in sorted order. Two sets of nonoverlapping intervals with endpoints in sorted order can be intersected in time proportional to the total number of intervals, which is an upper bound on the output size. We intersect the $O(n)$ sets of intervals in a tournament style, i.e., we go from n sets with n intervals in each set, to $n/2$ sets with $2n$ intervals in each sets,..., to 1 set with n^2 intervals. There are $\lg n$ stages, with $O(n^2)$ work at each stage, yielding a total time of $O(n^2 \lg n)$. ■

Finally, it is also desirable to solve the optimal placement problem (as defined in Chapter 2) for general single-layer routing. By adapting ideas used by Chao and LaPaugh [11] for density minimization, the problem can be solved in $O(n^3)$ time. This is explained further in Chapter 5.

Chapter 4

Minimizing Channel Density with Movable Terminals

This chapter describes algorithms for various problems which are helpful in reducing the channel width for routing of two-layer or multilayer channels. Our model deviates from the conventional model in that we assume terminals are movable subject to certain constraints, a situation that arises in practice when designing chips via the standard-cell or gate-array methodology. This chapter is mostly derived from a joint paper with Ronald Greenberg of University of Maryland [26].

Traditionally, channel routers have assumed that the positions of terminals on each side are fixed. However, it is quite typical in practice that some of the terminals are movable. By taking advantage of these moving terminals, we can reduce the channel width. It is well known that determining the width required to route a channel for fixed terminals is NP-complete [56]. Thus in many cases, the goal is instead to minimize the channel *density*, the maximum over all columns of the number of nets that must cross the column. Density can be computed quickly, and it provides a fairly good estimate for channel width.

In fact, many existing channel routers achieve widths that are usually within one of the density, e.g., [6, 18, 21, 40, 50, 53, 62]. (Focusing on density may also be appropriate when more than two interconnection layers are available, in which case the lower bound on separation becomes density divided by the number of layers allowing horizontal routing; e.g., see [3, 4, 12, 16, 20, 23, 29, 49] for multilayer channel routing.)

In this chapter we consider the situation in which the orderings of the terminals along each side of the channel are fixed, but the exact positions may vary. There are a number of practical situations in which such flexibility arises [8], and it can lead to substantial reduction in channel density and separation [8, 22]. When only the ordering of terminals on each side is fixed, Gopal, Coppersmith, and Wong [22] give an $O(n^2)$ algorithm to minimize the separation[1], where n is the number of terminals. LaPaugh and Pinter [33] presented an $O(n^2 \lg n)$ algorithm to minimize the channel density with the additional constraint that the relative positions of the terminals on each side are fixed. That is, the terminals lie on a single top module and a single bottom module, and the only freedom is to shift the modules relative to each other. More recently, Johnson, LaPaugh, and Pinter [31] provided an $O(n^3)$ algorithm to minimize density when there are multiple modules and terminal positions are fixed within each module, but the only other constraint is a fixed order for the modules on each side.

In the above works, however, the resulting channel length may be as large as $p + q$, where p is the number of top terminals and q is the number of bottom terminals (or as large as the sum of the module lengths). In contrast, Cai and

[1]This does not contradict the NP-completeness result, due to the use of a model in which there is complete freedom to choose the amount of space between adjacent terminals.

Wong [7, 8] minimize density for a channel of fixed length L under a wide variety of constraints on the terminal positions. For channels with only *linear order constraints* (the orderings of the terminals on each side of the channel are fixed), they proposed an $O(pqL)$ algorithm to minimize the channel density. If we add *separation constraints* (the distance between each pair of consecutive terminals is within a certain range), their running time and space become $O(pqL^3)$ and $O(pqL^2)$, respectively. With multiple modules and fixed terminals within each module, they obtain $O(L^3)$ time and space. If the terminals within the modules are also movable, then the running time and space become $O(pqL^3)$.

In this chapter we provide more efficient algorithms for these four problems of Cai and Wong [7, 8]. In each case, we improve the running time by a factor of $L/\lg(p+q)$ and the space by a factor of L. (It is easy to also incorporate "position constraints", which specify a set of allowable columns for each terminal, as do Cai and Wong; we will omit further discussion of such constraints in this thesis.)

The remainder of this chapter is organized as follows. In Section 4.1, we introduce some additional terminology and notation which will be used throughout this chapter. Section 4.2 describes an algorithm to find the minimum channel density for channels with linear order constraints by using a dynamic programming approach. The algorithm is then extended in Sections 4.3, 4.4, and 4.5 to handle channels with separation constraints, channels with movable modules, and channels with movable modules and movable terminals, respectively. Finally, in Section 4.6, we provide some extensions and conclusion remarks.

4.1 Preliminaries

We begin by giving a more formal problem definition and some notation. We define $t_1, t_2, \ldots, t_p$ and $b_1, b_2, \ldots, b_q$ to be the terminals on the top and bottom side of the channel, which are ordered from left to right. We are given L column positions in which to place the terminals while retaining the given ordering on each side. The goal is to find the positions of the terminals such that the channel density is minimized.

Note that the density at any given column depends only on the fixed order of the terminals on each side and the position of that column within those orderings. Then let $d_1(i,j)$ be the density at the column of t_i when t_i is placed between b_j and b_{j+1}, let $d_2(i,j)$ be the density at the column of b_j when b_j is placed between t_i and t_{i+1}, and let $d_3(i,j)$ be the density at the column of t_i and b_j when they are aligned. These density functions can be computed in $O(pq)$ time for all possible i,j. The computation is a simple double loop over i and j; for example, $d_1(i+1,j)$ can be computed in constant time from $d_1(i,j)$ by looking at which terminals are connected to t_i and t_{i+1}. (If there are many terminals per net, we can perform a preprocessing step that removes all but the leftmost and rightmost terminal of each net on the top and bottom of the channel.) We assume throughout this paper that the d_1, d_2, and d_3 values have been computed and saved. Also define $\delta_1^d(i,j)$ to be

$$\delta_1^d(i,j) = \begin{cases} 1 & \text{if } d_1(i,j) \le d \\ \infty & \text{if } d_1(i,j) > d \end{cases},$$

and define $\delta_2^d(i,j)$ and $\delta_3^d(i,j)$ analogously.

The main idea of our algorithms is as follows. Given a target density d, we compute the minimum channel length required to achieve the density. Based on the computed channel length and L, we increase or decrease the target density. By using a binary search on all the possible channel densities, we can find the minimum density achievable in length L.

4.2 Channels with Linear Order Constraints

In this section, we give an algorithm to minimize the channel density for channels with linear order constraints. We begin by showing how to find the minimum channel length at a given target density d. To do that, we introduce some subproblems used as the basis for a solution by dynamic programming. (We show in detail only how to find the minimum channel length, but one can readily retrace the computations leading to this result to determine the corresponding terminal placement.)

The length function $L^d(i,j)$ is defined to be the minimum number of columns spanned by top terminals $t_1, \ldots, t_i$ and bottom terminals $b_1, \ldots, b_j$, with the restriction that each of those columns has density at most d when all the other terminals are placed to the right of both t_i and b_j. If the target density d is unachievable, then $L^d(i,j)$ is defined to be ∞. We define $L_1^d(i,j)$ the same way as $L^d(i,j)$ but with the constraint that t_i is to the right of b_j. $L_2^d(i,j)$ and $L_3^d(i,j)$ are defined similarly but with the constraint that t_i is to the left of b_j, and t_i is aligned with b_j, respectively. We now show how to compute these functions recursively using the shorthand

$$L^d(i,j) = \min\{L_1^d(i,j), L_2^d(i,j), L_3^d(i,j)\} \ .$$

The final answer to our problem is $L^d(p,q)$.

Consider first the computation of $L_1^d(i,j)$. By the definition of $L_1^d(i,j)$, t_i

must be to the right of b_j. Thus we require one column more than are spanned by $t_1, t_2, \ldots, t_{i-1}$ and $b_1, b_2, \ldots, b_j$, and we must check the density constraint in this new column:

$$L_1^d(i,j) = (L^d(i-1,j)+1)\delta_1^d(i,j) \ .$$

Similarly, we can express $L_2^d(i,j)$ and $L_3^d(i,j)$ as

$$L_2^d(i,j) = (L^d(i,j-1)+1)\delta_2^d(i,j)$$

and

$$L_3^d(i,j) = (L^d(i-1,j-1)+1)\delta_3^d(i,j) \ .$$

For initial conditions, we have, for $c = 1,2,3$,

$$L_c^d(0,j) = j\prod_{k=1}^{j}\delta_c^d(0,k), \qquad j = 0,1,\ldots,q \ ,$$

and

$$L_c^d(i,0) = i\prod_{k=1}^{i}\delta_c^d(k,0), \qquad i = 0,1,\ldots,p \ ,$$

where we think of t_0 and b_0 as dummy terminals at the left of their respective sides that do not contribute to density.

Theorem 26 *Given a target density d, the minimum channel length subject to linear order constraints can be computed in $O(pq)$ time and space.*

Proof. We have already noted that the δ values can be computed in $O(pq)$ time, and an additional $O(p+q)$ time suffices to determine the initial conditions. Then we compute the values of the three length functions together in order of increasing i and j using the recurrences above. There is a total of $O(pq)$ values to compute, and each can be computed in $O(1)$ time from previously computed values. ■

Corollary 27 *The minimum density of a channel subject to linear order constraints can be found in $O(pq \lg(p+q))$ time and $O(pq)$ space.*

Proof. The minimum density problem can be solved by binary search on density, which is at most $p+q$. ■

4.3 Channels with Linear Order Constraints and Separation Constraints

In this section, we extend the algorithm of Section 4.2 to handle channels with linear order constraints and separation constraints. Let the separation constraints have the following form: the distance s_i between t_i and t_{i+1} must satisfy $l_i \le s_i \le r_i$, and the distance s'_j between b_j and b_{j+1} must satisfy $l'_j \le s'_j \le r'_j$.

To handle the distance constraints, we have to modify the length functions. Let $L_1^d(i,j,k)$ and $L_2^d(i,j,k)$ be defined as in Section 4.2 but with the restriction that the horizontal distance between t_i and b_j equals k (in absolute value). We define $L_3^d(i,j)$ exactly as before. The constraints for the three length functions are illustrated in Figure 4.1. Then, $L^d(i,j)$ is obtained by minimizing over the three types of length functions and all possible k's.

Consider $L_1^d(i,j,k)$ first. There are three cases: (1) t_{i-1} is to the right of b_j, (2) t_{i-1} is to the left of b_j, and (3) t_{i-1} is aligned with b_j. And the minimum among the three cases is the minimum channel length. In the first case,

$$L_1^d(i,j,k) = \min_{k'}\{L_1^d(i-1,j,k') + k - k'\}\delta_1^d(i,j) \ ,$$

with $l_{i-1} \le k - k' \le r_{i-1}$. Figure 4.2(a) illustrates the restriction on k'. The

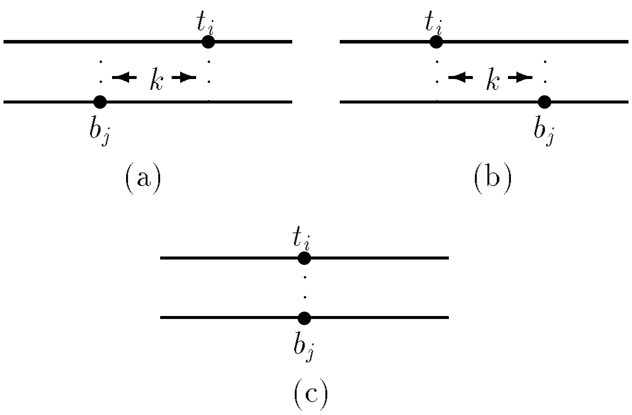

Figure 4.1: Three types of length functions: (a) $L_1^d(i,j,k)$ (b) $L_2^d(i,j,k)$ (c) $L_3^d(i,j)$

second case can be analyzed similarly, and we have

$$L_1^d(i,j,k) = \min_{k'}\{L_2^d(i-1,j,k') + k\}\delta_1^d(i,j) \ ,$$

with $l_{i-1} \le k + k' \le r_{i-1}$. In the third case, which is possible only when $l_{i-1} \le k \le r_{i-1}$, we find

$$L_1^d(i,j,k) = (L_3^d(i-1,j) + k)\delta_1^d(i,j) \ .$$

The three cases are shown in Figure 4.2. In all cases, we have $0 < k < L$, and we assign a length function value of ∞ for values of k that are impossible given the other constraints.

From the above argument, $L_1^d(i,j,k)$ can be expressed as

$$L_1^d(i,j,k) = \begin{cases} (\min A_1)\delta_1^d(i,j) & \text{if } l_{i-1} \le k \le r_{i-1} \\ (\min A_2)\delta_1^d(i,j) & \text{otherwise} \end{cases}$$

where

$$A_1 = \{L_3^d(i-1,j) + k\} \cup A_2 \ ,$$

and

$$A_2 = \{\min_{l_{i-1} \le k-k' \le r_{i-1}}\{L_1^d(i-1,j,k') + k - k'\}, \\ \min_{l_{i-1} \le k+k' \le r_{i-1}}\{L_2^d(i-1,j,k') + k\}\} \ .$$

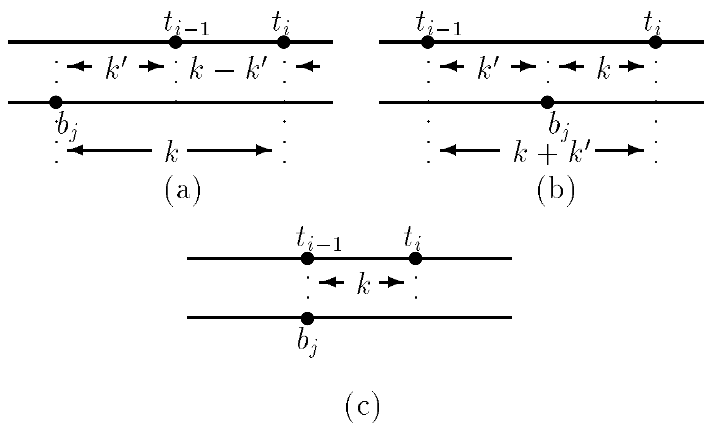

Figure 4.2: Three possibilities of $L_1^d(i,j,k)$: (a) t_{i-1} is to the right of b_j. (b) t_{i-1} is to the left of b_j. (c) t_{i-1} is aligned with b_j.

Similarly, $L_2^d(i,j,k)$ and $L_3^d(i,j)$ can be expressed as follows:

$$L_2^d(i,j,k) = \begin{cases} (\min B_1)\delta_2^d(i,j) & \text{if } l'_{j-1} \leq k \leq r'_{j-1} \\ (\min B_2)\delta_2^d(i,j) & \text{otherwise} \end{cases}$$

and

$$L_3^d(i,j) = \begin{cases} (\min C_1)\delta_3^d(i,j) & \text{if } [l_{i-1}, r_{i-1}] \cap [l'_{j-1}, r'_{j-1}] \neq \emptyset \\ (\min C_2)\delta_3^d(i,j) & \text{otherwise} \end{cases}$$

where

$$B_1 = \{L_3^d(i,j-1) + k\} \cup B_2 \ ,$$

$$\begin{aligned} B_2 = \{&\min_{l'_{j-1} \leq k+k' \leq r'_{j-1}} \{L_1^d(i,j-1,k') + k\}, \\ &\min_{l'_{j-1} \leq k-k' \leq r'_{j-1}} \{L_2^d(i,j-1,k') + k - k'\}\} \ , \end{aligned}$$

$$C_1 = \{L_3^d(i-1,j-1) + max\{l_{i-1}, l'_{j-1}\}\} \cup C_2 \ ,$$

$$\begin{aligned} C_2 = \{&\min_{(k'',k') \in S_{i,j}} \{L_1^d(i-1,j-1,k') + k''\}, \\ &\min_{(k'',k') \in T_{i,j}} \{L_2^d(i-1,j-1,k') + k''\}\} \ , \end{aligned}$$

$$S_{i,j} = \{(k'',k') | l_{i-1} \leq k'' \leq r_{i-1} \text{ and } l'_{j-1} \leq k'' + k' \leq r'_{j-1}\} \ ,$$

and

$$T_{i,j} = \{(k'',k') | l'_{j-1} \leq k'' \leq r'_{j-1} \text{ and } l_{i-1} \leq k'' + k' \leq r_{i-1}\} \ .$$

Theorem 28 *Given a target density d, the minimum channel length subject to linear order constraints and separation constraints can be computed in $O(pqL^2)$ time and $O(pqL)$ space.*

Proof. We compute values of the length functions in order of increasing i, j and k, and then the minimum channel length is

$$\min\left\{\min_{0<k<L} L_1^d(p,q,k), \min_{0<k<L} L_2^d(p,q,k), L_3^d(p,q)\right\} .$$

There are $O(pqL)$ values of L_1^d and L_2^d to be computed, and each can be computed from previously computed values in $O(L)$ time. In addition, there are $O(pq)$ values of L_3^d to be computed, each in time $O(L^2)$. ∎

Corollary 29 *The minimum density of a channel subject to linear order constraints and separation constraints can be found in $O(pqL^2 \lg(p+q))$ time and $O(pqL)$ space.* ∎

4.4 Channels with Movable Modules

This section considers the problem of channels with movable modules but with the terminals at fixed positions within their modules. We first augment the set of terminals to include the endpoints of the modules. Then we insert pseudo-terminals on the modules until every column in the modules contains a terminal or a pseudo-terminal as in [8]. As a result, the separation constraints between terminals inside a top module have the form $l_i = r_i = 1$ (an adjacency constraint), and the separation constraints between the right endpoint of a top module and the left endpoint of the module immediately to its right are $l_i = 1$, and $r_i = \infty$.

(The constraints on the bottom are similar.) Now we can see this problem as a channel subject to linear order constraints and special separation constraints.

The length functions used in this section are as defined in Section 4.2. The approach to calculate these length functions is the same except for a modification to handle adjacency constraints. Using the notational shorthand

$$L^d_{x,y}(i,j) = \min\left\{L^d_x(i,j), L^d_y(i,j)\right\} ,$$

we have:

$$L^d_1(i,j) = \begin{cases} (L^d(i-1,j)+1)\delta^d_1(i,j) & \text{if } r_{i-1} = \infty \\ (L^d_{1,3}(i-1,j)+1)\delta^d_1(i,j) & \text{if } r_{i-1} = 1 \end{cases}$$

$$L^d_2(i,j) = \begin{cases} (L^d(i,j-1)+1)\delta^d_2(i,j) & \text{if } r'_{j-1} = \infty \\ (L^d_{2,3}(i,j-1)+1)\delta^d_2(i,j) & \text{if } r'_{j-1} = 1 \end{cases}$$

$$L^d_3(i,j) = \begin{cases} (L^d(i-1,j-1)+1)\delta^d_3(i,j) & \text{if } r_{i-1} = r'_{j-1} = \infty \\ (L^d_{1,3}(i-1,j-1)+1)\delta^d_3(i,j) & \text{if } r_{i-1} = 1 \text{ and } r'_{j-1} = \infty \\ (L^d_{2,3}(i-1,j-1)+1)\delta^d_3(i,j) & \text{if } r_{i-1} = \infty \text{ and } r'_{j-1} = 1 \\ (L^d_3(i-1,j-1)+1)\delta^d_3(i,j) & \text{if } r_{i-1} = r'_{j-1} = 1 \end{cases}$$

and

$$L^d(i,j) = \min\{L^d_1(i,j), L^d_2(i,j), L^d_3(i,j)\} .$$

Theorem 30 *Given a target density d, the minimum channel length for channels with movable modules can be computed in $O(L^2)$ time and space.*

Proof. We can compute $L^d_1(i,j)$, $L^d_2(i,j)$, and $L^d_3(i,j)$ from previously computed values in $O(1)$ time. Including the pseudo-terminals, there are $O(L)$ terminals on each side of the channel, which yields $O(L^2)$ length function values to be computed. ∎

Corollary 31 *The minimum density with movable modules can be found in* $O(L^2 \lg(p+q))$ *time and* $O(L^2)$ *space.* ■

4.5 Channels with Movable Modules and Terminals

In this section, we consider channels with movable terminals and modules. That is, the modules on each side of the channel are movable as in Section 4.4, and we also allow the terminals to move within their modules. To handle this situation, we have to introduce new definitions and length functions.

Define a *left terminal* to be the leftmost terminal of a module. Also define $M(p)$ to be the module where terminal p is located, v_i to be the length of $M(t_i)$, and w_j to be the length of $M(b_j)$. The length functions used here have four variables i, j, k, and l as illustrated in Figure 4.3; here k and l represent the distance from the rightmost of t_i and b_j to the left edges of their modules. The length function $L^d(i,j)$ is equal to the minimum of the three types of length functions for all possible k's and l's (where each length function accounts for the lengths of the modules containing $t_1, t_2, \ldots, t_i$ and $b_1, b_2, \ldots, b_j$).

For many values of k and l, we can immediately set length function values to ∞. For example, if terminal t_i is the mth terminal in its module, then $L_1^d(i,j,k,l) = \infty$ for any $k < m-1$. In what follows we give recurrences for the length functions under the assumption that such restrictions have already been taken into account.

To simplify the presentation, we define notational shorthand as in Section 4.4:

$$L_{x,y}^d(i,j,k,l) = \min\left\{L_x^d(i,j,k,l), L_y^d(i,j,k,l)\right\}$$

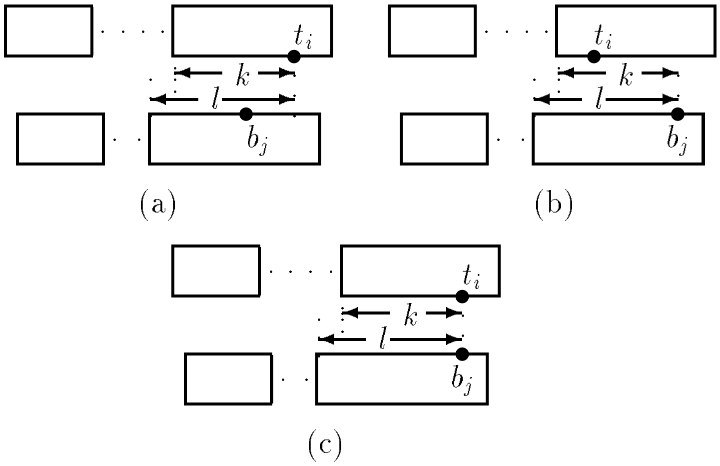

Figure 4.3: Three types of length functions: (a) $L_1^d(i,j,k,l)$ (b) $L_2^d(i,j,k,l)$ (c) $L_3^d(i,j,k,l)$

and

$$L^d(i,j,k,l) = \min\{L_1^d(i,j,k,l), L_2^d(i,j,k,l), L_3^d(i,j,k,l)\} \ .$$

We first consider $L_1^d(i,j,k,l)$. There are two cases according to whether t_i is a left terminal or not. We seek the minimum among the channel lengths obtained in the following three subcases: (1) t_{i-1} is to the right of b_j, (2) t_{i-1} is to the left of b_j, and (3) t_{i-1} is aligned with b_j. Note that if the relative position of $M(t_i)$ and $M(b_j)$ is fixed, then the actual positions of the terminals on the two modules have no effect on the value of the length functions as long as the density is less than or equal to d.

Case (A): t_i is not a left terminal.

(1): In the subcase where t_{i-1} is to the right of b_j, we know that we can place t_{i-1} in the column just before t_i, since t_{i-1} and t_i are on the same module, and the definition of $L_1^d(i,j,k,l)$ implies that there are no bottom terminals between b_j and t_i. Thus we have

$$L_1^d(i,j,k,l) = L_1^d(i-1,j,k-1,l-1)\delta_1^d(i,j) \ .$$

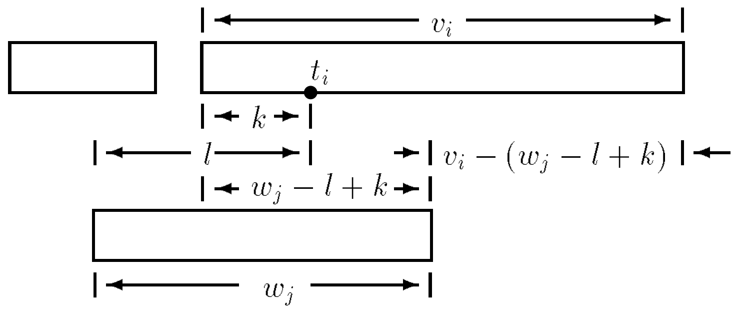

Figure 4.4: Calculating the channel length when t_i is a left terminal.

(2) and (3): In the subcases where t_{i-1} aligned with or to the left of b_j, we know that we can place b_j in the column just before t_i if $w_j \geq l-1$; otherwise, we can place b_j at the right end of its module.

Putting the subcases together, we have

$$L_1^d(i,j,k,l) = \begin{cases} L^d(i-1,j,k-1,l-1)\delta_1^d(i,j) & \text{if } w_j \geq l-1 \\ \min\{L_1^d(i-1,j,k-1,l-1), & \\ \qquad L_{2,3}^d(i-1,j,w_j+k-l,w_j)\}\delta_1^d(i,j) & \text{if } w_j < l-1 \end{cases}$$

Case (B): t_i **is a left terminal.**

(1): In the subcase where t_{i-1} is to the right of b_j, we know that we can push t_{i-1} to the right edge of its module, giving us

$$L_1^d(i,j,k,l) = \min_{l'<l-k}\{L_1^d(i-1,j,v_{i-1},l')\}\delta_1^d(i,j) + \max\{0, v_i - k - (w_j - l)\} .$$

The term added at the end accounts for the possible increase in channel length when module $M(t_i)$ is included, as shown in Figure 4.4.

(2): In the subcase where t_{i-1} is to the left of b_j, we know that we can place b_j in the column just before t_i if $w_j \geq l-1$; otherwise we can push b_j to the right edge of its module.

(3) In the subcase where t_{i-1} is aligned with b_j, we can push t_{i-1} to the right edge of its module if $l - w_j \leq k$; otherwise we can push b_j to the right edge of

its module.

Putting the subcases together gives:

$$\begin{aligned} L_1^d(i,j,k,l) &= \min_{l',k',k'',l''} \{L_1^d(i-1,j,v_{i-1},l'), L_2^d(i-1,j,k',\min\{w_j,l-1\}), \\ &\qquad L_3^d(i-1,j,k'',l'')\}\delta_1^d(i,j) + \max\{0, v_i - k - (w_j - l)\} \ , \end{aligned}$$

where $l' < l-k$, $k' > v_{i-1} + k + \min\{w_j - l, -1\}$, and k'' and l'' are defined as follows. If $l - w_j > k$, then $l'' = w_j$ and $k'' > v_{i-1} + w_j + k - l$. If $l - w_j \le k$, then $k'' = v_{i-1}$ and $l'' < l - k$.

We can write recurrences for L_2 in a fashion similar to L_1. When b_j is not a left terminal,

$$L_2^d(i,j,k,l) = \begin{cases} L^d(i,j-1,k-1,l-1)\delta_2^d(i,j) & \text{if } v_i \ge k-1 \\ \min\{L_2^d(i,j-1,k-1,l-1), & \\ \qquad L_{1,3}^d(i,j-1,v_i,v_i+l-k)\}\delta_2^d(i,j) & \text{if } v_i < k-1 \end{cases}$$

When b_j is a left terminal,

$$\begin{aligned} L_2^d(i,j,k,l) &= \min_{k',l',l'',k''} \{L_2^d(i,j-1,k',w_{j-1}), L_1^d(i,j-1,\min\{v_i,k-1\},l'), \\ &\qquad L_3^d(i,j-1,k'',l'')\}\delta_2^d(i,j) + \max\{0, w_j - l - (v_i - k)\} \ , \end{aligned}$$

where $k' < k - l$, $l' > w_{j-1} + l + \min\{v_i - k, -1\}$, and l'' and k'' are defined as follows. If $k - v_i > l$, then $k'' = v_i$ and $l'' > w_{j-1} + v_i + l - k$. If $k - v_i \le l$, then $l'' = w_{j-1}$ and $k'' < k - l$.

Finally, we consider L_3. It is easy to see that when t_i is not a left terminal,

$$L_3^d(i,j,k,l) = L_2^d(i-1,j,k,l)\delta_3(i,j) \ .$$

Similarly, when b_j is not a left terminal,

$$L_3^d(i,j,k,l) = L_1^d(i,j-1,k,l)\delta_3(i,j) \ .$$

Finally, if t_i and b_j are both left terminals,

$$L_3^d(i,j,k,l) \;=\; \min_{w_{j-1}+l<l'} L_1^d(i,j-1,k,l') + \max\{0, w_j - l - (v_i - k)\}\ .$$

Theorem 32 *Given a target density d, the minimum channel length problem for channels with movable modules and terminals can be computed in $O(pqL^2)$ time and space.*

Proof. All the length functions $L_1^d(i,j,k,l)$, $L_2^d(i,j,k,l)$, and $L_3^d(i,j,k,l)$ can be computed from the previously computed values in $O(1)$ time because all the minimizations appearing in our recurrences can be performed on the fly. In fact the minimizations never depend on the values of both k and l; for example the minimization over $l' < l-k$ needs only be performed for each value of $l-k$, and there is no need for more than $O(1)$ extra storage as long as these minimizations are peformed in order of the value of $l-k$. There is a total of $O(pqL^2)$ length functions, which yields the stated running time and space. ■

Corollary 33 *The minimum density of a channel with movable modules and terminals can be solved in $O(pqL^2 \lg(p+q))$ time and $O(pqL^2)$ space.* ■

4.6 Conclusion and Extensions

We have presented algorithms to minimize the channel density for a variety of problems. These algorithms improve the previous known results by $O(L/\lg(p+q))$ in running time and $O(L)$ in space. These algorithms can also easily be extended to channels with exits or channels with irregular boundaries as in [7] without increasing the complexity.

In the process of minimizing density for a fixed channel length, we have provided even more efficient algorithms to minimize length at a fixed density. By running the latter type of algorithm $O(p+q)$ times, we can also minimize more complex cost measures, such as area (where density is treated as width) in a channel of length at most L. We can also improve the space bound for our algorithms to find minimum channel length or minimum density if we are not worried about recovering the actual terminal placement. Since the length function values for a given sum of i and j depend only on values with a lesser sum of i and j, we need only store the values for one previous sum at a time. Thus all the space requirements decrease by a factor of $\max\{p, q\}$ (or L for the case of movable modules with fixed terminals).

Chapter 5

Minimizing Channel Density with Interchangeable Terminals

This chapter gives algorithms to minimize channel density by interchanging terminals within cells on the channel sides. These algorithms have important applications in hierarchical layout design. In standard cells or gate-array logic blocks, for example, some terminals on the cells are allowed to be interchanged. As shown in Figure 5.1, there are two AND gates in the top module. We can model the top module as two fixed cells, with the two input terminals of an AND gate in each cell, and the terminals within each cell are interchangeable. By interchanging the terminals in the cells, we can reduce the channel density.

We consider the situation in which there is a set of positions available in each cell and a set of permutable terminals to be assigned to the cell. The number of available positions may be greater than or equal to the number of terminals. In [14] and [15], Cong and Khoo considered the density minimization problem when all the terminals are interchangeable on the same channel side, i.e., there is only one cell on each side, and terminals can sit anywhere on the channel boundary. When there are only two-terminal nets, the algorithm in [15]

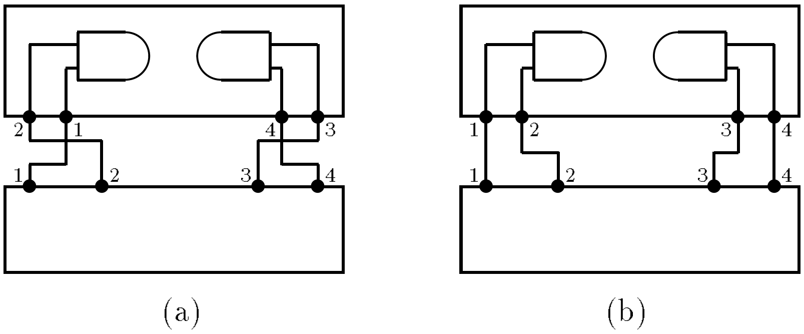

Figure 5.1: Interchanging terminals in the same cell can lead to a reduction in the channel width: (a) Before interchange, the channel density is 2. (b) After interchange, the channel density is 1.

can minimize the density in $O(n)$ time, where n is the number of terminals. If there are multi-terminal nets, the algorithm in [14] can find a density within one of the optimal density in $O(n \lg n)$ time. Later, Tragoudas [59] and Cai and Wong [9] improved their algorithm by minimizing the density for channels with multi-terminal nets in $O(n)$ time. If, however, there are multiple cells on each side, Tragoudas and Tollis [60] showed that the problem of minimizing channel density is NP-hard; still, they developed linear time algorithms for the special cases of channels with only two-terminal nets and channels with at most one terminal per net on each cell.

All the above works have assumed that every position within the cell is available for the placement of the terminals. In practice, however, not every position of the cells is available for assignment of terminals. In gate-array or standard-cell layout design, for example, the positions of the terminals on each cell are fixed, and we can only use these positions to permute logically equivalent terminals on each cell.

This chapter considers the situation when only a set of positions in each cell is available for the placement of the terminals. We use the practical assumption that each net has at most one terminal in each cell. Our goal is to minimize the channel density. If the cells are fixed, then our running time is $O(n \lg n)$, where n is the number of positions available in the cells. We also consider the case when there are multiple modules, with fixed cells within the modules, on each of the channel sides. The modules can be moved along the side of the channel as long as their order is not changed. The running time for this problem is $O(CLn^2 \lg n)$, where C is the number of modules, and L is a bound on the allowed channel length. If the number of available positions is equal to the number of terminals in each cell, then the running time becomes $O(n^3 \lg n)$.

Several other issues related to channel routing with permutable terminals can also be found in the literature. In [30, 32, 39], heuristics are given to reduce the number of tracks required for routing. Lin and Sahni [41] present a linear-time algorithm that obtains maximum terminal alignment. Sahni and Wu [51] show that it is NP-hard to find whether there exists an interchange of terminals that results in an acyclic vertical constraint graph or to minimize the longest path of the vertical constraint graph.

The remainder of this chapter is organized as follows. In Section 5.1, we describe an algorithm to find the minimum channel density for channels with fixed cells. In Sections 5.2, we use the algorithm as a subroutine to handle channels with movable components. Finally, in Section 5.3, we provide some concluding remarks.

5.1 Channels with Fixed Cells

In this section, we first give an algorithm to minimize the channel density for channels with only one cell on each side. We then use the algorithm as a subroutine to minimize the channel density for multiple cells on each side. The approach of the algorithm is as follows. Given a target density d, we solve the decision problem of whether there exists a terminal assignment such that the resulting channel has density less than or equal to d. By using a binary search on all the possible channel densities based on the answer of the decision problem, we can find the minimum density achievable in the channel. Also we assume the available positions on each cell are sorted according to their x-coordinates. If the positions are not in sorted order, $O(n \lg n)$ preprocessing time suffices to do the job.

We start by giving some definitions. A net with *left exit* is a net that connects to at least one terminal outside and to the left of the channel. A net with *right exit* and a net with *two exits* can be defined analogously. Now define a position to be *free* if there is no terminal assigned to that position, and define a position to be *occupied* if there is a terminal assigned to that position. Also define r_i and l_i to be the x-coordinates of the right and left endpoints of net N_i, respectively. Two positions are *aligned* if they have the same x-coordinate. A two-terminal net is called a *trivial net* if its terminals are assigned to the same column. Note that terminals belonging to nets with two exits can be placed at any positions because these positions of do not affect the channel density. By subtracting the number of nets with two exits from the target density, we can take these nets out of consideration. From now on we assume there are no nets with two exits.

We assign terminals belonging to nets with exits first. A net with right

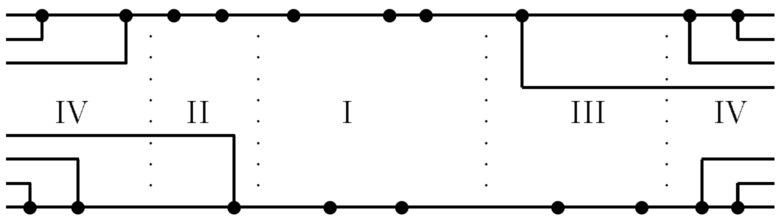

Figure 5.2: The channel is divided into four areas: area I, II, III, and IV

(left) exit contributes one unit of density to every column to the right (left) of the column where its terminal is located; intuitively, then, terminals belonging to nets with right (left) exits should be assigned to the rightmost (leftmost) available positions on the cell. In actuality, this is a good first step; we will see that some adjustment may be necessary as we proceed to place the remaining nets, each of which has one terminal on each cell. We divide the channel into four areas. Area IV is composed of the rightmost and leftmost regions in the channel in which all terminals on top and bottom are occupied by nets with exits. Area III is the remaining rightward block of the channel that contains nets with right exits. Similarly, area II is the remaining block containing nets with left exits. Finally, area I is the remaining region. (Areas II and III may overlap making area I nonexistent.) These areas are shown in Figure 5.2. Note that the terminal assignment in area IV is optimal, since assigning terminals belonging to other nets to this area will not reduce the density at any column of the channel. Therefore, we only have to consider the terminal assignment in areas I, II, and III. The following is a sketch of the procedure to solve the decision problem.

Procedure ONECELL

1. Assign terminals belonging to nets with right exits to the rightmost posi-

tions of the cells. Assign terminals belonging to nets with left exits to the leftmost positions of the cells.

We now proceed with steps 2 through 5 as long as there remain unassigned two-terminal nets.

2. For every pair of aligned positions in area I, assign a two-terminal net to the positions.

3. For every pair of aligned positions in area II, starting from the leftmost aligned positions, free the position occupied in step 1 and assign a two-terminal net to the aligned pair. Assign the leftmost free position to the terminal whose position was "stolen". If this assignment does not cause the channel to have density greater than d in area III, then proceed to the next aligned pair. Otherwise reassign the freed position to the terminal which occupied it in step 1 and go to the next step.

4. For every pair of aligned positions in area III, proceed analogously to step 3.

5. Repeat assigning a two-terminal net to the leftmost possible free positions such that the resulting channel density does not exceed the target density d.

6. If every net has been assigned successfully, and the channel density in area IV is no greater than d, return "yes"; otherwise return "no".

We now show procedure ONECELL answers the decision problem correctly.

Lemma 34 *Given a target density d, procedure ONECELL answers the decision problem correctly.*

Proof. We prove the lemma by showing that we can always transform a valid terminal assignment to the assignment resulting from procedure ONECELL. First, there is a way to interchange terminals to achieve the definition of area IV without increasing the density, since the terminal assignment in area IV is optimal. For any terminal x belonging to a net with right exit, if there is a terminal y that does not belong to a net with right exit or a trivial net, and is assigned to the right of terminal x, we can interchange x and y without increasing the channel density. Similar arguments hold for terminals belonging to nets with left exits. Once these interchanges are performed, we can define areas I through IV as before.

Now, for any aligned pair of positions in area I, if the two terminals assigned to the positions belong to different nets, we interchange the top terminals of the two nets. As shown in Figure 5.3, the interchange reduces the density by 1 at the column where the aligned positions are located, while it retains the same density at other columns. Then, for any two nontrivial nets N_i and N_j, if $l_i < l_j$, we interchange terminals if necessary to satisfy $r_i < r_j$. This interchange again does not increase the density at any column. Similarly, we can interchange terminals of different nets assigned to aligned positions of areas II and III without increasing the channel density. Finally, by shifting the nontrivial two-terminal nets to the leftmost possible positions, the resulting assignment is the same as from procedure ONECELL. ■

Lemma 35 *Given a target density d, the decision problem of whether there exists a terminal assignment without exceeding channel density d for channels with one cell on each side can be solved in $O(n)$ time.*

Proof. Step 1 and step 2 can be done easily in $O(n)$ time. In step 3, for every

Figure 5.3: Interchanging terminals belonging to different nets in an aligned pair of positions does not increase channel density: (a) Before interchange, the density is 2. (b) After interchange, the density is 1.

new position assigned in area III, we have to check whether the density in this area exceeds d. Assume that we have spent $O(n)$ time to compute the density in each column containing a terminal position and that the density at the right boundary of area III is no greater than d. (If it is greater than d, then the target density is infeasible and we do nothing in step 3.) Now, if a terminal is assigned to a position in area III, then the density increases by 1 in every column between that position and the left boundary of area III. By storing the maximum density within the region from the left boundary of area III to the position in area III most recently occupied in step 3, we can check the density restriction in $O(1)$ time; when a new position is assigned, the new maximum density is equal to the maximum of the old maximum density + 1 and the density at the column of the new position. Thus, step 3 can be completed in $O(n)$ time. Similar arguments hold for step 4. A similar $O(1)$ density check can be obtained in step 5 by checking the density at newly assigned positions and by keeping track of the maximum density in the overlap of regions II and III if such overlap exists.

■

For channels with multiple cells on each side, we scan the cells from left to right and use a modification of procedure ONECELL as a subroutine to answer the decision problem. Let the channel have a cells $T_1, \ldots, T_a$ on top and b cells

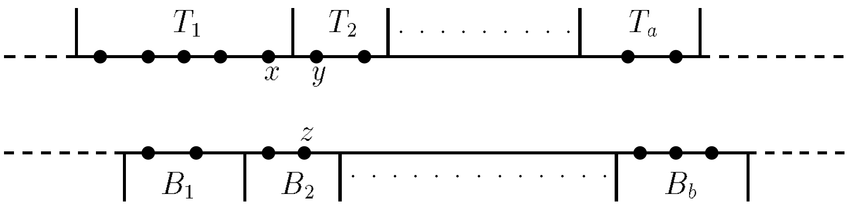

Figure 5.4: A channel with multiple cells on each side

$B_1, \ldots, B_b$ on bottom as shown in Figure 5.4. In each cell, we say a terminal belongs to a net with right exit if the net has a terminal to the right of the cell and there is no overlap between the two cells to which the terminals belong. Nets with left exits and two exits are defined analogously. Note that we only have to consider at most four terminals per net, the leftmost and rightmost terminals on top and bottom, for terminal assignment. The placement of other terminals does not affect the channel density. From now on we assume that each net has at most four terminals.

The terminals of four terminal nets or two terminal nets on any two overlapped cells can be assigned by using procedure ONECELL, but there exists a kind of three terminal net that needs special treatment. As shown in Figure 5.4, suppose we are now processing cells T_1 and B_2, and terminals x, y, and z belong to the same net. Then terminal z is not assigned by procedure ONECELL because z is not a terminal of a two-terminal net, nor a terminal belonging to a net with an exit. (However, we can still say that x belongs to a net with a right exit.) In this case, we leave terminal z unassigned until we process cells T_2 and B_2. We call this kind of net a *special net*. A special net is a three terminal net such that one of the three cells where the terminals are located is overlapped with the other two cells. The terminals of special nets can be properly assigned by a slight modification of ONECELL as follows: In step 5, we assign the terminals to the leftmost possible free positions without violating the density constraint according to the following order: terminals belonging to special nets with one terminal in area II, and then two terminal nets. We call the modified procedure ONECELL1. Note that during the assignment of the terminals of special nets, the order of these terminals is not important. As long as there exists a valid assignment, any permutation of these terminals will be a valid assignment. The correctness of the procedure can be proved by a similar argument as for procedure ONECELL.

Let Q_1 be the set of cells that are not overlapped with any other cells, and let Q_2 be the set of pairs of overlapped cells. Also define an overlapped pair of cells (T_i, B_j) to be to the left of (T_k, B_l) if $i \leq k$ and $j \leq l$. Figure 5.5 shows the algorithm for channels with multiple cells on each side.

We use Figure 5.4 as an example to explain the algorithm. Since each of the cells is overlapped with some other cells, we do nothing in lines 1–5. Now starting with T_1 and B_1, we use procedure ONECELL1 to assign terminals to the two cells. Since cell T_1 also overlaps with B_2, we then again use procedure ONECELL1 to process T_1 and B_2 (with previously assigned terminals fixed where they were assigned). The process continues until all terminals are assigned or the channel density exceeds d at some point of the process.

The correctness of the algorithm can be proved using a similar argument as in procedure ONECELL. The running time for procedure MULTICELL is also $O(n)$ since every terminal is processed only once in one of the while loops. We have the following theorem:

Theorem 36 *The minimum density of a channel with interchangeable terminals and fixed cells on each side can be found in $O(n \lg n)$ time.* ■

procedure MULTICELL

while $Q_1 \neq \emptyset$ **do**
    Remove a cell from Q_1
    Assign terminals of nets with right exits to the rightmost positions of the cell.
    Assign terminals of nets with left exits to the leftmost positions of the cell.
endwhile
while $Q_2 \neq \emptyset$ **do**
    Remove the leftmost pair of cells from Q_2
    Use procedure ONECELL1 to assign terminals to the two cells.
endwhile
if every pair of cells from Q_2 has a feasible assignment and the channel density in the region of nonoverlapped cells is no greater than d
then return “yes”
else return “no”
endif

Figure 5.5: This procedure answers the decision problem for channels with multiple cells on each side.

Proof. The minimum density problem can be solved by binary search on density, which is at most n. ■

5.2 Channels with Movable Components

This section considers a scenario in which the terminals on each of the two sides are grouped into several components. Within each component, there are multiple fixed cells as in Section 5.1, but the components on each side of the channel can slide back and forth as long as their relative order does not change. This situation happens in standard-cell layout design. Also we are given a bound L on the allowed channel length.

To solve the problem, we first compute the minimum channel length required at a target density d. Then by using a binary search on all the possible densities, we can find the minimum density achievable in length L. We now show how to compute the minimum channel length at a given density d by a dynamic programming approach.

The approach we use here is similar to the approach used to minimize channel length at a given density in [11], with terminals fixed on the components. We first give some definitions. Suppose we have c_t top components $T_1, \ldots, T_{c_t}$ and c_b bottom components $B_1, \ldots, B_{c_b}$, with $c_t + c_b = C$. Let $l(M)$ be the length of component M, and let $lp(M)$ and $rp(M)$ be the x-coordinates of the left and right endpoints of component M, respectively. The space between components is called a *gap*. The gap between top components T_i and T_{i+1} is defined to be top gap $i + \triangle$. We consider the top gap as a *top gap component* $T_{i+\triangle}$ with adjustable length. The endpoints of $T_{i+\triangle}$ are defined to be $lp(T_{i+\triangle}) = rp(T_i)$ and

$rp(T_{i+\triangle}) = lp(T_{i+1})$. The gap to the left of T_1 is referred to as gap component $T_{0+\triangle}$, and $lp(T_{0+\triangle})$ equals $\min\{lp(T_1), lp(B_1)\}$, which is the leftmost end of the channel. Similarly, the rightmost top gap component $T_{c_t+\triangle}$ has $rp(T_{c_t+\triangle}) = \max\{rp(T_{c_t}), rp(B_{c_b})\}$. The bottom gap components are defined analogously. Also define *outside free positions* as follows: If T_i overlaps with B_j and $rp(T_i)$ is less than $rp(B_j)$, then the outside free positions on B_j are the positions to the right of $rp(T_i)$ such that all terminals on B_j connecting to the right of T_i are assigned to the right of these positions. The outside free positions on T_i when $rp(T_i)$ is greater than $rp(B_j)$ are defined analogously. Figure 5.6 shows the situation when $rp(T_i)$ is less than $rp(B_j)$, and the number of outside free positions on B_j is l.

We now define length functions that measure the minimum channel length of partial placements. The length function $L^d(i,j)$ is defined to be the minimum number of columns spanned by top components $T_1, \ldots, T_i$ and bottom components $B_1, \ldots, B_j$, with the restriction that each of the columns to the left of $\min\{rp(T_i), rp(B_j)\}$ has density at most d when all the other components are placed to the right of both T_i and B_j. If the target density d is unachievable, then $L^d(i,j)$ is defined to be ∞. We define $L^d(i,j,k,l)$ the same way as $L^d(i,j)$ but with the constraint that $rp(B_j) - rp(T_i) = k$, and l satisfies the following condition: If k is greater than (less than) 0, then l is the number of outside free positions on B_j (T_i) that can be used for processing B_j and T_i; if k is equal to 0, then l is equal to 0. Note that it is possible that there are no outside free positions on B_j (T_i); in this case, we define l to be negative, and its absolute value is equal to the number of terminals connecting to the right of B_j (T_i) that must assigned to the left of $rp(T_i)$ ($rp(B_j)$). Given a value k, the range of possible l

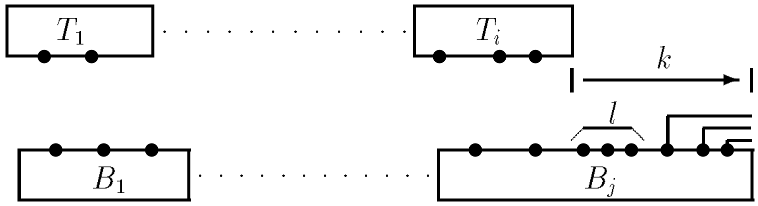

Figure 5.6: Length function $L(i, j, k, l)$

can be determined easily. The length functions involving gap components are defined similarly. Figure 5.6 shows the definition of length function $L^d(i, j, k, l)$. The one-sided arrow labeled with k in the figure represents that the value of k equals the x-coordinate of the head minus the x-coordinate of the tail. Now $L^d(i, j)$ is obtained by minimizing over all possible k's and l's, and the final answer to our problem is $\min\{L^d(c_t + \triangle, c_b), L^d(c_t, c_b + \triangle)\}$. From now on, we omit the superscript d if the meaning is clear.

In order to compute the length functions, we modify procedure ONECELL1 and MULTICELL. Procedure ONECELL2 is similar to procedure ONECELL1 but it assigns terminals to the rightmost possible positions. Without loss of generality, assume that the left endpoint of the bottom cell is to the right of the left endpoint of the top cell. Procedure ONECELL2 has 7 steps and has the same steps through step 4 as in procedure ONECELL1. But in step 5, ONECELL2 assigns the remaining terminals to the rightmost possible positions without violating the density constraint in the order of terminals belonging to special nets with one terminal in area III, and then two terminal nets. In step 6, if the density is at most d in areas I, II, and III after all nets are assigned, and the top position that was assigned to the last terminal is p, then do the following: Starting from the rightmost trivial net to the left of p, reassign the rightmost possible positions to terminals of the trivial nets in areas III and I. Then starting from the leftmost trivial net in area II, interchange the bottom terminal with the rightmost terminal belonging to a net with left exit and reassign the two terminal net to the rightmost possible positions. This process continues until no terminals can be reassigned without causing infeasibility. Finally the procedure checks the feasibility of the channel. The procedure ONECELL2 can be done in $O(n)$ time.

Procedure MULTICELL2 is similar to MULTICELL, but it uses ONECELL2 as a subroutine and marches across pairs of cells from right to left. Also in the last step, MULTICELL2 checks the feasibility only in the overlapped region of the components.

To derive length functions, we need one more definition. Define $D(i,j,k,l)$ to be 1 if there exists a valid terminal assignment of T_i and B_j using procedure MULTICELL2, where k and l are defined as in $L(i,j,k,l)$, and to be ∞ otherwise.

We now derive the relationship between length functions. We first express length function $L(i,j,k,l)$ in terms of gap length functions with smaller indices. Consider the case when k is greater than or equal to 0. Since the relative positions of T_i and B_j are fixed, assigning terminals to the rightmost possible positions of T_i without violating the density constraint will give the most freedom to assign terminals on those components before T_i and B_j. Therefore, we can use procedure MULTICELL2 to assign the terminals. Suppose $l(B_j)$ is less than or equal to $l(T_i)+k$ (see Figure 5.7(a)), and let l'_1 be the number of outside free positions on T_i (relative to $B_{j-1+\triangle}$) such that assigning terminals of nets with left exits to these positions will not cause the density of the overlapped region of T_i and B_j to exceed d. Then

$$L(i,j,k,l) = (L(i,j-1+\triangle,k-l(B_j),l'_1)+k)D(i,j,k,l) \ .$$

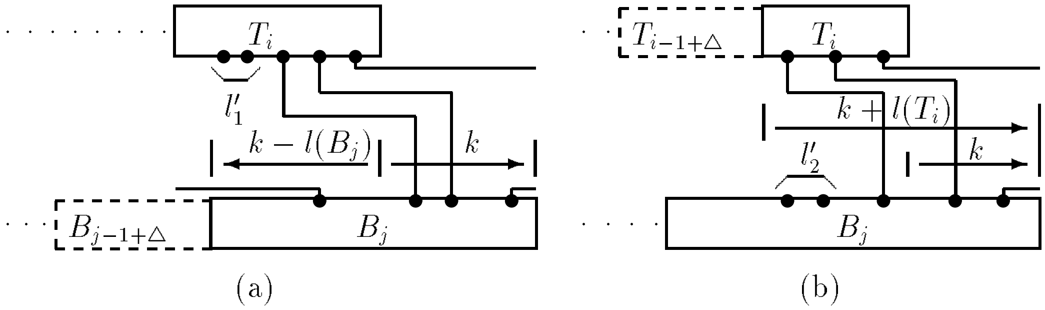

Figure 5.7: $L(i, j, k, l)$ is expressed in terms of $L(i, j-1+\triangle, k-l(B_j), l'_1)$ and $L(i-1+\triangle, j, k+l(T_i), l'_2)$: (a) $l(B_j) \leq l(T_i)+k$ (b) $l(B_j) > l(T_i)+k$

l'_1 can be easily determined after the execution of procedure MULTICELL2 on T_i and B_j. Similarly, if $l(B_j)$ is greater than $l(T_i) + k$, then by defining l'_2 analogously, we have

$$L(i, j, k, l) = (L(i-1+\triangle, j, k+l(T_i), l'_2))D(i, j, k, l) \; .$$

Figure 5.7(b) shows the configuration. The function $D(i, j, k, l)$ checks the feasibility of the overlapped region of the components. The case when k is less than 0 can be handled similarly.

From the above argument and by defining l'_i, $i = 1, \ldots, 4$, as the outside free positions in each of the following cases, we have

$$L(i,j,k,l) = \begin{cases} (L(i, j-1+\triangle, k-l(B_j), l'_1) + k)D(i,j,k,l) & \\ \qquad\qquad \text{if } k \geq 0 \text{ and } l(B_j) \leq l(T_i) + k \\ (L(i-1+\triangle, j, k+l(T_i), l'_2))D(i,j,k,l) & \\ \qquad\qquad \text{if } k \geq 0 \text{ and } l(B_j) > l(T_i) + k \\ (L(i-1+\triangle, j, k+l(T_i), l'_3) - k)D(i,j,k,l) & \\ \qquad\qquad \text{if } k < 0 \text{ and } l(B_j) > l(T_i) + k \\ (L(i, j-1+\triangle, k-l(B_j), l'_4))D(i,j,k,l) & \\ \qquad\qquad \text{if } k < 0 \text{ and } l(B_j) \leq l(T_i) + k \end{cases}$$

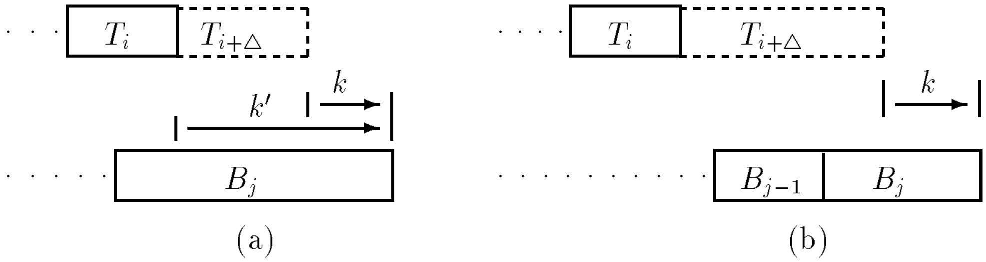

Figure 5.8: Computation of $L(i+\triangle, j, k, l)$: (a) T_i is overlapped with B_j. (b) T_i is not overlapped with B_j.

Now we need to express the length functions of $L(i+\triangle, j, k, l)$ and $L(i, j+\triangle, k, l)$ using smaller indices. We show how to solve $L(i+\triangle, j, k, l)$; $L(i, j+\triangle, k, l)$ can be computed similarly. As shown in Figure 5.8, there are two possibilities: T_i is overlapped with B_j, or T_i is not overlapped with B_j. If T_i is not overlapped with B_j, then B_{j-1} must be abutted with B_j. We only have to consider the case when k is greater than or equal to 0; the case when k is less than 0 has the same length as when k is 0.

To express the length functions of $L(i+\triangle, j, k, l)$ and $L(i, j+\triangle, k, l)$, we need some more definitions. Define $d(i+\triangle, j, k, k', l)$ to be 1 if there exists a valid terminal assignment of $T_{i+\triangle}$ and B_j using procedure MULTICELL2 and under the restriction that the assignment can use at most l outside free positions, and to be ∞ otherwise, where k and l are defined as in $L(i+\triangle, j, k, l)$, and k' is defined to be $rp(B_j) - rp(T_i)$. If k or l is infeasible, we define $d(i+\triangle, j, k, k', l)$ to be ∞. $d(i, j+\triangle, k, k', l)$ can be defined analogously. Also define $n(M, a, b)$ to be the number of positions on component M in the interval $[a, b]$, and $n'(M, a)$ to be the number of positions to the right of a on M such that all terminals to the right of a connecting to terminals to the right of M are assigned to the right of these positions.

We can now partition $L(i+\triangle, j, k, l)$ into two classes: OVERLAP and

NONOVERLAP. We have

$$L(i+\triangle, j, k, l) = \min\{OVERLAP(k', l'), NONOVERLAP\} \ ,$$

where

$$OVERLAP(k', l') = \min_{(k',l') \in I_{k',l'}} \{L(i, j, k', l')\} d(i+\triangle, j, k, k', l) \ ,$$

$$NONOVERLAP = (L(i+\triangle, j-1, 0, 0) + l(B_j)) d(i+\triangle, j, k, l(B_j), l) \ ,$$

$$I_{k',l'} = \{(k', l') | k \leq k' \leq l(B_j) \text{ and}$$

$$l' = \min\{l + n(B_j, rp(T_i), rp(T_{i+\triangle})), n'(B_j, rp(T_i))\}\} \ .$$

Note that in the OVERLAP case, l' is dependent on k'; that is, if we are given a k', then l' is fixed. Now, suppose for any fixed i and j, all $L(i, j, k, l)$ are known for all possible k and l, and the values are stored in a two-dimensional table. Then the minimum value in $OVERLAP(k', l')$ is only a suffix minimum in some diagonal line of the table. We can find all the suffix minima for all the diagonal lines in $O(nL)$ time. By storing these values, any suffix minimum can be found in $O(1)$ time.

Similarly, $L(i, j+\triangle, k, l)$ can be expressed as

$$L(i, j+\triangle, k, l) = \min\{OVERLAP'(k', l'), NONOVERLAP'\} \ ,$$

where

$$OVERLAP'(k', l') = \min_{(k',l') \in J_{k',l'}} \{L(i, j, k', l')\} d(i, j+\triangle, k, k', l) \ ,$$

$$NONOVERLAP' = (L(i-1, j+\triangle, 0, 0) + l(T_i)) d(i, j+\triangle, k, l(T_i), l) \ ,$$

$$J_{k',l'} = \{(k', l') | -l(T_i) \leq k' \leq k \text{ and}$$

$$l' = \min\{l + n(T_i, rp(B_j), rp(B_{j+\triangle})), n'(T_i, rp(B_j))\}\} \ .$$

We now compute the initial length functions $L(0+\triangle, j, k, l)$ and $L(i, 0+\triangle, k, l)$ for all possible i, j, k, l. We only show how to compute $L(0+\triangle, j, k, l)$; $L(i, 0+\triangle, k, l)$ can be done similarly. As explained in the NONOVERLAP case, in the calculation of $L(0+\triangle, j, k, l)$, we only have to find those k's whose value is in the range of $0 \leq k \leq l(B_j)$. We have

$$L(0+\triangle, 1, k, l) = l(B_1)d(0+\triangle, 1, k, l(B_1), l)$$

and

$$L(0+\triangle, j, k, l) = (L(0+\triangle, j-1, 0, 0) + l(B_j))d(0+\triangle, j, k, l(B_j), l) \ .$$

What remains is to compute the functions $D(i, j, k, l)$, $d(i+\triangle, j, k, k', l)$, and $d(i, j+\triangle, k, k', l)$. Suppose we want to compute $D(i, j, k, l)$ for a fixed i, and for all possible j, k, and l. We place the components as shown in Figure 5.9 and move T_i from left to right. Suppose that T_i is at some position overlapped with B_j and $rp(B_j) - rp(T_i) = k$, then for every possible number of outside free positions l, we use procedure MULTICELL2 to find $D(i, j, k, l)$. Note that after we execute procedure MULTICELL2, the number of outside free positions on T_i or B_j that can be used for connecting to B_{j-1} or T_{i-1} can also be found. The running time for procedure MULTICELL2 is proportional to the number of positions on the two cells. Also we have at most the same number of positions as the number of different values of l. Therefore, the running time to find all $D(i, j, k, l)$ for all possible i, j, k, and l is $O(CLn^2)$. To compute $d(i+\triangle, j, k, k', l)$, note that if we fix i, j, k, and l, then $d(i+\triangle, j, k, k', l)$ is a step function with only one discontinuity. Therefore, by using a similar process, $d(i+\triangle, j, k, k', l)$ and $d(i, j+\triangle, k, k', l)$ can be computed in time $O(CLn^2)$.

Figure 5.9: Computation of $D(i, j, k, l)$

Theorem 37 *Given a target density d, the minimum channel length for channels with interchangeable terminals and movable components can be computed in $O(CLn^2)$ time and space.*

Proof. We have already noted that the D functions can be computed in $O(CLn^2)$ time, and an additional $O(CLn)$ time suffices to determine the initial conditions. Also we need $O(c_t c_b)$ tables to store the suffix minima to compute the OVERLAP functions. Each table can be computed and stored in $O(nL)$ time and space, which yields $O(c_t c_b nL)$ running time and space. Now every length function can be computed from previously computed values in $O(1)$ time. By putting all the running time and space together yields the stated running time and space.

■

Corollary 38 *The minimum density of a channel with interchangeable terminals and movable components can be found in $O(CLn^2 \lg n)$ time.* ■

5.3 Conclusion

We have presented algorithms to minimize the channel density for a variety of problems. The model we used is more general than previously considered. These algorithms can also easily be extended to handle channels with irregular boundaries without increasing the complexity.

For the case of movable components with the same number of terminals and available positions, our algorithm can be easily modified to achieve $O(n^3 \lg n)$ running time and space. This can be done by just dropping the variable l and use a similar analysis for running time as in [11]. Our algorithms can also be used as a subroutine for area minimization as in Section 4.6. The argument for improvement of the space requirements also holds here.

Finally, we can also use similar techniques to solve the optimal placement problem for single-layer routing in Chapter 3. Given a width w, we want to minimize the channel length. We define $D(i,j,k)$ to be 0 if the overlapped interval of T_i and B_j is feasible under width w, and to be 1 otherwise, where k is as defined before. Also define $d(i+\triangle, j, s)$ to be 0 if the point with distance s from the right endpoint of B_j is feasible under width w when $T_1, T_2, \ldots, T_i$ are put to the left of B_j and $T_{i+1}, \ldots, T_{c_t}$ are put to the right of B_j, and to be 1 otherwise, where $0 \le s \le l(B_j)$. $D(i, j+\triangle, s)$ is defined analogously. Both D and d are step functions, and there are at most $O(n^2)$ discontinuities for all these step functions. If we can show all the D and d values can be found in $O(n^3)$ time, then by applying the techniques in [11], the optimal placement problem for single-layer channel routing can be solved in $O(n^3)$.

Lemma 39 *All the D and d functions can be found in $O(n^2 \lg n)$ time.*

Proof. We show how to compute the D values; the d values can be computed similarly. To compute $D(i,j,\cdot)$, for $1 \le j \le c_b$, we place the modules as in Figure 5.9. Now we can use Theorem 25 to compute $D(i,j,\cdot)$ by moving module T_i from left to right. There is a total of n terminals for all top and bottom modules, which yields $O(n^2 \lg n)$ running time. ∎

Corollary 40 *The optimal placement problem for single-layer routing can be computed in $O(n^3)$ time.*

Chapter 6

Finding a Maximum-Density Planar Subset of a Set of Nets in a Channel

This chapter describes an algorithm to find a maximum-density planar subset of n 2-pin nets in a channel, which can be helpful in the context of multilayer channel routing. Previous work has demonstrated that as increasing numbers of chip layers become available, it may be beneficial to subdivide a channel routing problem into smaller problems with fewer layers, some of them single-layer problems. Methods to extract planar collections of nets with large density help to ensure that offloading nets to single-layer subproblems will substantially ease the problem of routing the remaining nets.

This work is motivated by the idea of partitioning the nets in a different fashion than MulCh [23] in order to make more effective use of the planar layers. Specifically, we seek to fill each planar layer with a good collection of nets before proceeding with the remainder of the assignment of nets to layer groups. One approach for obtaining an appropriate collection of coplanar nets is to extract the largest possible number of coplanar nets, which is equivalent to finding a maximum-cardinality independent set in a circle graph [55]. This approach, however, does not consider reducing the density of the remaining nets of the channel, which is, in practice, a good estimate of the difficulty of routing the remaining nets. We therefore propose to extract a set of coplanar nets of maximum density, which we refer to as an MDPS (maximum-density planar set). Then, since the MDPS may include nets from only a limited range of the channel, one can extract additional coplanar nets by continuing to work on the set of nets obtained by removing the original MDPS and the nets intersected by nets in the MDPS. After repeated application of an MDPS algorithm, one is likely to have extracted a coplanar collection of nets that does a good job of reducing the density of the remaining set of nets. We therefore focus on finding an efficient MDPS algorithm. (Similar problems are considered by Lee and Sarrafzadeh [36], but they seek planar subsets with small density rather than large density and do not allow single-sided nets.)

We use the following notation in this chapter. For net i, the column containing the leftmost terminal is denoted l_i, and the column containing the rightmost terminal is denoted r_i. We will assume henceforth that all of the l_i and r_i values are in the range from 1 to $2n$, but we will charge $n \lg n$ time for this assumption, the time required to sort and renumber an arbitrary set of l_i and r_i values. We use t to denote the number of two-sided nets and w to denote the number of nets in the MDPS.

A naive approach to finding an MDPS is to repeatedly apply the algorithm of Supowit [55] or Buckingham [5, pp. 119–124], which find a maximum-cardinality planar subset. Let $S|i$ denote those nets from the set S that cross column i, and let N be the complete set of nets in the input. Then, we can find an MDPS in $O(n^3)$ time by finding the largest among the maximum-cardinality planar subsets of $N|1$, $N|2$, $\ldots$, $N|2n$. In fact, a somewhat better bound on the running time for the approach using Buckingham's algorithm can be obtained based on the number of containment relationships among the nets considered. But the algorithm we present in the next section greatly expands the class of input problems that can be solved in better than time proportional to n^3. A simple bound on the running time is $O(n \lg n + n(t+1)w)$. In the special case when there are no two-sided nets, the bound stated above becomes $O(n \lg n + nw)$; this bound can also be achieved in the more likely case of no single-sided nets. In addition, the bounds stated so far can be strengthened by incorporating into the running time the number of edges in certain interval overlap and interval containment graphs.

6.1 Overlap Graphs, Containment Graphs, and Maximum Cliques

We begin with the definition of interval overlap graphs and interval containment graphs, since our MDPS algorithm makes extensive use of techniques for finding maximum cliques in such graphs. Let $F = \{I_1, I_2, \ldots, I_n\}$ be a family of closed intervals on the real line. A graph $G_c = (V, E)$ is called an interval containment graph for F if there is a one-to-one correspondence between V and F such that two vertices in V are adjacent to each other if and only if one of the corresponding intervals in F contains the other. (The interval $I_1 = [l_1, r_1]$ contains the interval $I_2 = [l_2, r_2]$ if $l_1 \leq l_2 \leq r_2 \leq r_1$.) A graph $G_o = (V, E)$ is called an interval overlap graph for F if there is a one-to-one correspondence between V and F such that two vertices in V are adjacent to each other if and only if the corresponding intervals in F overlap each other but neither one contains the other. (Two intervals $I_1 = [l_1, r_1]$ and $I_2 = [l_2, r_2]$ overlap each other if $l_1 \leq l_2 \leq r_1 \leq r_2$ or $l_2 \leq l_1 \leq r_2 \leq r_1$.) It is convenient also for us to be able to refer to the interval graph G_i, which is the union of G_c and G_o.

The connection between these graphs and the nets in a channel is obtained by simply viewing each net as an interval extending from its left terminal to its right terminal. Hence, we will use sets of nets and sets of intervals interchangeably in most notations. For example, $G_o(F)$ will represent the interval overlap graph corresponding to a set of intervals F or a set of nets F; similarly $G_c(F)$ is the corresponding interval containment graph.

Our MDPS algorithm incorporates an algorithm by Masuda, et. al. [44] for finding maximum cliques in interval overlap graphs. (Actually, the algorithm finds the size of the maximum clique in each of the subgraphs of the interval overlap graph obtained by restricting attention to nets crossing a particular column, and this is the capability we require.) The algorithm can also be readily adapted to work with interval containment graphs, which is an additional capability we require. In the remaining of this section, we provide some background on the algorithm of Masuda, et. al.

The algorithm of Masuda et. al. finds, for each column c in the channel, the size of the maximum clique of $G_o(F|c)$. The running time given in [44] is $O(n_F \lg n_F + \min\{m_F, n_F w_F\})$, where n_F is the number of intervals in F, m_F is the number of edges in $G_o(F)$, and w_F is the size of the maximum clique of $G_o(F)$. But, under the assumption that no two intervals have a common right endpoint or a common left endpoint and all endpoints are between 1 and $2n_F$, the algorithm runs in time $O(n_F + \min\{m_F, n_F w_F\})$. The restriction on common endpoints will be met because of the particular classes of intervals to which we will apply the algorithm of Masuda et. al. in Section 6.2. Also the limitation on the range of the terminal positions can easily be enforced given the assumption that the terminal positions in the entire channel were sorted up front as mentioned before. In our further description of the algorithm of Masuda et. al. and our adaptation of it, it is also important to realize that given the limited range of terminal positions, we can do such tasks as sorting a set of nets in order of left endpoint in time linear in the number of nets (by bin sorting).

A key observation of Masuda et. al. is that finding maximum cliques is equivalent to solving certain instances of the longest increasing subsequence (LIS) problem, which we review here. Let $\pi = [\pi(1), \pi(2), \ldots, \pi(s)]$ be a sequence of distinct numbers. If a subsequence $Y = [\pi(i_1), \pi(i_2), \ldots, \pi(i_t)]$ of π satisfies $\pi(i_1) < \pi(i_2) < \cdots < \pi(i_t)$, then it is called an increasing subsequence (IS). A longest increasing subsequence (LIS) of π is an IS with maximum length among all IS's of π.

The correspondence between maximum cliques and longest increasing subsequences is obtained as follows. First sort the set of intervals $F = \{I_1, I_2, \ldots, I_n\}$ in order of ascending right endpoint and renumber the intervals so that $r_1 \leq r_2 \leq \cdots \leq r_n$. Next, create for each c, a sequence τ_c of the intervals numbers from $F|c$ sorted in ascending order of their left endpoints. Now there is a one-to-one correspondence between IS's of τ_c and cliques of $G_o(F|c)$.

The results for interval overlap graphs can be applied to interval containment graphs by simply considering longest decreasing subsequences instead of longest increasing subsequences. (Finding the maximum clique of an interval containment graph $G_c(F)$ can be done in $O(n_F \lg n_F)$ time [43], but that algorithm does not find the size of each $G_c(F|c)$, a capability we require in Section 6.2.) The algorithm of Masuda et. al. that finds all the requisite longest increasing subsequences will serve equally well to find the longest decreasing subsequences for the interval containment graph in $O(n_F + \min\{m_F, n_F w_F\})$ time once the sorting is accounted for up front. Again, n_F is the number of nets, m_F is the number of edges in the graph under consideration, and w_F is the size of the maximum clique.

6.2 The MDPS Algorithm

In this section we explain how to use the maximum clique techniques to find an MDPS of a set of nets in a channel. For simplicity, we show in detail only how to find the size of an MDPS. By keeping track of decisions made in determining the size, one can readily determine the actual set of nets in the MDPS.

Given a set of 2-pin nets in a channel, we divide the nets into four subsets ST (single-sided nets connected to top), SB (single-sided nets connected to bottom), DR (two-sided nets with left endpoint on top) and DL (two-sided nets with left endpoint on bottom).

In accordance with our earlier comments, we assume that the nets in each of the four subsets are sorted in ascending order of right endpoints. Also, define $MC(F)$ to be a maximum clique of the interval containment graph $G_c(F)$, if $F \subset ST$ or $F \subset SB$; $MC(F)$ to be a maximum clique of the interval overlap graph $G_o(F)$, if $F \subset DL$ or $F \subset DR$. In addition, define a clique of an MDPS to be a subset of an MDPS for which there exists some column that is crossed by every net in the subset.

Remark 41 *SB and ST are mutually coplanar. Furthermore, the problem of finding a maximum-density planar subset of SB or ST is equivalent to finding a maximum clique of the interval containment graph for SB or ST, respectively.*

Remark 42 *If $x \in DR$, $y \in DL$, and there exists some column c such that $l_x \le c \le r_x$, and $l_y \le c \le r_y$, then x and y must intersect each other. Thus, a maximum clique of an MDPS may contain either elements of DR or elements of DL, but not both.*

Remark 43 *The problem of finding an MDPS of DL or DR is equivalent to finding a maximum clique of the interval overlap graph for DL or DR, respectively.*

Let us consider a channel with only 2-sided nets first. From Remarks 42 and 43, we only have to find a maximum clique of the interval overlap graphs for DR and DL, and the larger one is an MDPS. To find either maximum clique, we need only $O(n + \min\{m, nw\})$ time, where n, m, and w represent nets, overlap edges, and output size for the corresponding graph. For simplicity, we express the overall running time (including the sorting up front) using n, m, and w as the overall nets, overlap edges, and output size, yielding $O(n \lg n + \min\{m, nw\})$ time to find an MDPS.

Now let us consider a channel with only single-sided nets. In this case, we find the maximum-density planar subsets column by column. From Remark 41, it is equivalent to finding maximum cliques in the interval containment graphs for ST and SB at every column. So the algorithm works as follows: For $c =$ 1 to $2n$, find $|MC(ST|c)|$ and $|MC(SB|c)|$, and then the size of the MDPS is $\max_{1 \le c \le 2n}\{|MC(ST|c)| + |MC(SB|c)|\}$. Since it takes $O(n + \min\{m, nw\})$ time to find $|MC(ST|c)|$'s and $|MC(SB|c)|$'s, the algorithm has $O(n \lg n + \min\{m, nw\})$ time complexity, with n, m, and w as before except that we are now dealing with interval containment edges.

Finally, let us consider a channel with both single-sided and two-sided nets. Because a clique of an MDPS cannot contain both elements of DL and DR, we first compute an MDPS containing no elements of DR; then compute an MDPS containing no elements of DL, the larger one is an MDPS.

Let $DL = \{\nu_1, \ldots, \nu_{n_{DL}}\}$. In what follows, we will establish some relationship among DL, ST, and SB.

Remark 44 *If ν_i is in the maximum clique of an MDPS, then for any element x of ST in the maximum clique of the MDPS, $r_x < r_{\nu_i}$, and for any element y of SB in the maximum clique of the MDPS, $l_y > l_{\nu_i}$. Moreover, if $\nu_i, \nu_{i+1}, \ldots, \nu_j$ are all in the maximum clique of an MDPS, then $r_x < \min\{r_{\nu_i}, \ldots, r_{\nu_j}\}$, and $l_y > \max\{l_{\nu_i}, \ldots, l_{\nu_j}\}$.*

Now define ST_{ν_i}, SB_{ν_i}, and DL_{ν_i} in the following way:

$$
\begin{aligned}
ST_{\nu_i} &= \{a | a \in ST, r_a < r_{\nu_i}\} \\
SB_{\nu_i} &= \{a | a \in SB, l_a > l_{\nu_i}\} \\
DL_{\nu_i} &= \{\nu_x | \nu_x \in DL, x \geq i, l_{\nu_i} \leq l_{\nu_x} < r_{\nu_i}\}
\end{aligned}
$$

Also define P_i to be the maximum clique of an MDPS containing ν_i such that if there exists other $\nu_x \in MDPS$, then $i < x$. If the MDPS contains two-sided nets, then $|MDPS| = \max_{1 \leq j \leq n_{DL}}\{P_j\}$.

With these definitions, we have the following two results obtained from Remark 44:

Lemma 45 *If ν_i is in the maximum clique of an MDPS, then no elements of $ST - ST_{\nu_i}$ or $SB - SB_{\nu_i}$ are in that clique.* ■

Lemma 46 *Let $C_{i,j}$ be the maximum clique of an MDPS containing $MC(DL_{\nu_i}|j)$ and no other elements of DL. Then, for $l_{\nu_i} \leq j \leq r_{\nu_i}$,*

$$|C_{i,j}| = |MC(DL_{\nu_i}|j)| + \max_{l_\nu \leq k \leq r_{\nu_i}} \{|MC(ST_{\nu_i}|k)| + |MC(SB_\nu|k)|\} ,$$

where l_ν is the rightmost left endpoint of $MC(DL_{\nu_i}|j)$, and $|P_i|$ can be expressed as $|P_i| = \max_{l_{\nu_i} \leq j \leq r_{\nu_i}} \{|C_{i,j}|\}$. ■

Proof. By the construction of DL_{ν_i}, the right endpoint of ν_i is the leftmost right endpoint of all elements in DL_{ν_i}, and the left endpoint of ν_i is the leftmost left endpoint of all elements in DL_{ν_i}. Thus, ν_i is included in every $MC(DL_{\nu_i}|j)$ for $l_{\nu_i} \leq j \leq r_{\nu_i}$. (See Figure 6.1.) From Remark 44, only elements of ST_{ν_i} and SB_ν can be included with $MC(DL_{\nu_i}|j)$ in $C_{i,j}$. Since any two nets from different sets among DL_{ν_i}, ST_{ν_i}, and SB_ν are coplanar, we can find an MDPS at column j by finding maximum cliques of the three sets separately, which yields the expression for $|C_{i,j}|$. Finally, the expression for P_i follows because it must contain one of the $MC(DL_{\nu_i}|j)$ sets. ■

From Lemmas 45 and 46, we can compute (the size of) an MDPS, using the algorithm shown in Figure 6.2. After we find an MDPS of the nets in SB, ST, and DL, an MDPS of the nets in SB, ST, and DR can be computed similarly. The larger one is an MDPS of all the nets.

For X being any of the four sets SB, ST, DL, or DR, let n_X, m_X, and w_X denote the number of nets in X, the number of edges in $G_i(X)$, and the maximum clique size for $G_i(X)$. Also let n, m, and w be the maximum over the choices for X of the corresponding parameters.

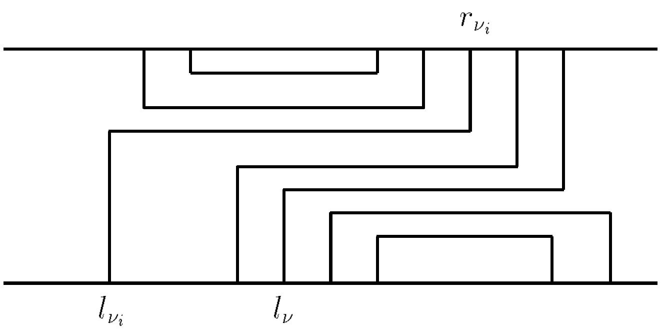

Figure 6.1: In this figure, $C_{i,j}$ contains 3 two-sided nets, r_{ν_i} and l_ν are shown as above.

We can now analyze the running time of the algorithm. Lines 1–2 can be completed in time $O(n+\min\{m_{ST}, n_{ST}w_{ST}\}+\min\{m_{SB}, n_{SB}w_{SB}\})$, and lines 3–6 can be completed in that amount of time multiplied by n_{DL}. In the remainder of the algorithm, the running time of most steps is bounded by nn_{DL}. The n_{DL} executions of line 9 requires an additional $O(n_{DL}\min\{m_{DL}, n_{DL}w_{DL}\})$ time, but this is dominated by the time required by line 15. The test in line 12 ensures that line 15 is executed at most $\min\{m_{DL}, n_{DL}w_{DL}\}$ times; each execution requires $O(n)$ time. Putting all the terms together, the running time of the procedure in Figure 6.2 is

$$\begin{aligned} O(\ & (n_{DL}+1)(\min\{m_{SB}, n_{SB}w_{SB}\} + \min\{m_{ST}, n_{ST}w_{ST}\}) \\ & +n(n_{DL} + \min\{m_{DL}, n_{DL}w_{DL}\}))\ . \end{aligned}$$

Finally, the overall time to find an MDPS can be obtained by incorporating terms based on DR and the $n \lg n$ sorting time performed up front.

Under a variety of conditions in which the sets of the four types of nets have small numbers of edges in the corresponding interval graph or small clique sizes, the running time is substantially superior to the naive $O(n^3)$ approach. Even if we simplify the bound greatly, we can see substantial opportunity for the find $|MC(ST|1)|, \ldots, |MC(ST|2n)|, |MC(SB|1)|, \ldots, |MC(SB|2n)|$

$P_0 \leftarrow \max_{1 \le c \le 2n}\{|MC(ST|c) + |MC(SB|c)|\}$

for $i \leftarrow 1$ **to** n_{DL} **do**

  find ST_{ν_i}, $|MC(ST_{\nu_i}|1)|, \ldots, |MC(ST_{\nu_i}|2n)|$

  find SB_{ν_i}, $|MC(SB_{\nu_i}|1)|, \ldots, |MC(SB_{\nu_i}|2n)|$

endfor

for $i \leftarrow 1$ **to** n_{DL} **do**

  find DL_{ν_i}

  find $|MC(DL_{\nu_i}|1)|, \ldots, |MC(DL_{\nu_i}|2n)|$

  $|C_{i,l_{\nu_i}}| \leftarrow 1 + \max_{l_{\nu_i} \le k \le r_{\nu_i}}\{|MC(ST_{\nu_i}|k)| + |MC(SB_{\nu_i}|k)|\}\}$

  for $j \leftarrow l_{\nu_i} + 1$ **to** r_{ν_i} **do**

    if $|MC(DL_{\nu_i}|j)| > |MC(DL_{\nu_i}|j-1)|$

    then

      Let $\nu \in DL$ be such that $l_\nu = j$.

      $|C_{i,j}| \leftarrow \max\{|C_{i,j-1}|, |MC(DL_{\nu_i}|j)|$

        $+\max_{l_\nu \le k \le r_{\nu_i}}\{|MC(ST_{\nu_i}|k)| + |MC(SB_\nu|k)|\}\}$

    else

      $|C_{i,j}| \leftarrow |C_{i,j-1}|$

    endif

  endfor

  $P_i \leftarrow |C_{i,j}|$

endfor

$|MDPS| \leftarrow \max\{P_0, \max_{1 \le i \le n_{DL}}\{P_i\}\}$

Figure 6.2: This algorithm uses interval containment graphs and interval overlap graphs to compute an MDPS of the nets in SB, ST, and DL.

running time to be $o(n^3)$. In particular, we can express the overall running time to find an MDPS as $O(n(\lg n + t + \min\{m, tw\}) + \min\{m, nw\})$, where t is the number of two-sided nets in the channel. If we simplify further by removing the minimization with respect to m and note that w is always at least 1, we obtain a bound of $O(n(\lg n + tw + w)) = O(n \lg n + n(t+1)w)$.

Chapter 7

Conclusion

The major contribution of this thesis is to provide efficient algorithms to optimize different cost functions for single-layer and multi-layer channel routing and placement problems in VLSI layout design. This section summarizes the key findings and suggests further areas of investigation.

In the single-layer context, we have shown that as long as only one side of the channel contains multiple terminals of the same net, linear time suffices to solve the optimal offset, offset range, and optimal placement problems. When the placement of terminals is entirely unrestricted, we show that if the number of columns c is $O(n)$, we can solve both the optimal offset and offset range problems in time $O(n^{1.5} \lg n)$.

One direction for further research is to improve the time for feasible offset when the number of columns is large. We know that $\Omega(n^2)$ is a lower bound on the worst case running time, but we suspect that it may not be difficult to obtain an $O(n^2)$ upper bound as well. Another remaining open question is whether our upper bounds for feasible offset with smaller c can be improved. We know of no nontrivial lower bounds, i.e., better than $\Omega(\min\{c, n^2\})$. It also might be possible to improve the time to solve optimal offset without making further progress on feasible offset. Though it seems unlikely that optimal offset would be much easier than feasible offset, optimal offset has a much smaller output size, and output size is the only basis for our lower bounds on feasible offset. In the optimal placement problem, we can adapt ideas used by Chao and LaPaugh [11] for density minimization to solve this problem in $O(n^3)$ time. A further direction for research is to improve this $O(n^3)$ time when there is a reasonable bound on the number of columns.

In the context of multilayer channel routing, since minimizing channel width is NP-complete in many cases, we use density instead as our cost function. When the order of the terminals is fixed but the terminals are movable, we improve previous results for both running time and space in a variety of cases. When the terminals are interchangeable, we provide a new model which is suitable for standard cells or gate-array layout design. Using this new model, we obtain efficient algorithms to minimize the channel density for different problems.

For the case of movable modules with fixed terminals, density can be minimized in a channel of length L in $O(n^3 \lg n)$ time independent of L (which is an improvement for large L) using the method of Chao and LaPaugh [11]. However, their method can not be extended to handle channels with movable terminals or channels with interchangeable terminals. An interesting open question is to solve other variations of the problem in time polynominal in n only. Also in Chapter 5, we have assumed that at most one terminal per net is to be assigned to each cell. For the case when more than two terminals belonging to the same net are allowed on the same cell, it is impossible to find an optimal solution due to the NP-completeness. A useful area of investigation would be to find heuristic algorithms for this problem.

Our goal is to minimize the channel density for a general layout. To achieve this, the next step is to handle the case of multiple parallel channels as occurred in the standard-cell layout design. In other words, parallel routing channels are stacked on top of one another and adjacent channels share a row of components. Unfortunately, even when each row contains only one component, the problem is NP-complete [28]. Therefore, heuristic algorithms will be needed for general layout problems.

The last part of this thesis presented an algorithm to find a maximum-density planar subset of nets in a channel. This algorithm runs in time $o(n^3)$ for many types of input problems. For example, superior running time is obtained if the number of two-sided nets or the number of nets in the output is $o(n)$. An open area for investigation is to obtain a simpler algorithm that runs in $o(n^3)$ running time on all n-net problems.

Bibliography

[1] Brenda S. Baker, Sandeep N. Bhatt, and Tom Leighton. An approximation algorithm for manhattan routing. In Franco P. Preparata, editor, *VLSI Theory.* Volume 2 of *Advances in Computing Research*, pages 205–229. JAI Press, 1984.

[2] Brenda S. Baker and Ron Y. Pinter. An algorithm for the optimal placement and routing of a circuit within a ring of pads. In *24th Annual Symposium on Foundations of Computer Science*, pages 360–370. IEEE Computer Society Press, 1983.

[3] Douglas Braun, Jeffrey L. Burns, Fabio Romeo, Alberto Sangiovanni-Vincentelli, Kartikeya Mayaram, Srinivas Devadas, and Hi-Keung Tony Ma. Techniques for multi-layer channel routing. *IEEE Trans. Computer-Aided Design of Integrated Circuits*, 7(6):698–712, June 1988.

[4] Peter Bruell and Paul Sun. A "greedy" three layer channel router. In *IEEE International Conference on Computer-Aided Design (ICCAD-85)*, pages 298–300. IEEE Computer Society Press, 1985.

[5] Mark A. Buckingham. Circle graphs. Technical Report NSO-21, Courant Institute of Mathematical Sciences, Computer Science Department, New

York University, October 1980.

[6] M. Burstein and R. Pelavin. Hierarchical channel router. *INTEGRATION, the VLSI journal*, 1(1):21–38, January 1983.

[7] Yang Cai and D. F. Wong. Minimizing channel density by shifting blocks and terminals. In *IEEE International Conference on Computer-Aided Design (ICCAD-91)*, pages 524–527. IEEE Computer Society Press, 1991.

[8] Yang Cai and D. F. Wong. Optimal channel pin assignment. *IEEE Trans. Computer-Aided Design of Integrated Circuits*, 10(11):1413–1424, November 1991.

[9] Yang Cai and D. F. Wong. Channel density minimization by pin permutation. In *Proceedings IEEE International Conference on Computer Design*, pages 378–382. IEEE Computer Society Press, 1992.

[10] Shing-Chong Chang, Joseph JáJá, and Kwan Woo Ryu. Optimal parallel algorithms for one-layer routing. Technical Report UMIACS TR 89-46, University of Maryland Institute for Advanced Computer Studies, April 1989.

[11] Liang-Fang Chao and Andrea S. LaPaugh. Finding all minimal shapes in a routing channel. Technical Report CS-TR-384-92, Princeton University Department of Computer Science, August 1992.

[12] Yun Kang Chen and Mei Lun Liu. Three-layer channel routing. *IEEE Trans. Computer-Aided Design of Integrated Circuits*, CAD-3(2):156–163, April 1984.

[13] Richard Cole and Alan Siegel. River routing every which way, but loose. In *25th Annual Symposium on Foundations of Computer Science*, pages 65–73. IEEE Computer Society Press, 1984.

[14] J. Cong and K. Khoo. A provable near-optimal algorithm for the channel pin-assignment problem. In *Proceedings IEEE International Conference on Computer Design*, pages 319–321. IEEE Computer Society Press, 1991.

[15] Jingsheng Cong. Pin assignment with global routing for general cell designs. *IEEE Trans. Computer-Aided Design of Integrated Circuits*, 10(11):1401–1412, November 1991.

[16] Jingsheng Cong, D. F. Wong, and C. L. Liu. A new approach to three- or four-layer channel routing. *IEEE Trans. Computer-Aided Design of Integrated Circuits*, 7(10):1094–1104, October 1988.

[17] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* McGraw-Hill, 1990.

[18] D. Deutsch. A 'dogleg' channel router. In *Proceedings of the 13th ACM/IEEE Design Automation Conference*, pages 425–433. IEEE Computer Society Press, 1976.

[19] Danny Dolev, Kevin Karplus, Alan Siegel, Alex Strong, and Jeffrey D. Ullman. Optimal wiring between rectangles. In *Proceedings of the 13th ACM Symposium on Theory of Computing*, pages 312–317. ACM Press, 1981.

[20] R. J. Enbody and H. C. Du. Near-optimal n-layer channel routing. In *Proceedings of the 23rd ACM/IEEE Design Automation Conference*, pages 708–714. IEEE Computer Society Press, 1986.

[21] Charles M. Fiduccia and Ronald L. Rivest. A "greedy" channel router. In *Proceedings of the 19th ACM/IEEE Design Automation Conference*, pages 418–424. IEEE Computer Society Press, 1982.

[22] Inder S. Gopal, Don Coppersmith, and C.K. Wong. Optimal wiring of movable terminals. *IEEE Trans. Computers*, C-32(9):845–858, September 1983.

[23] Ronald I. Greenberg, Alexander T. Ishii, and Alberto L. Sangiovanni-Vincentelli. MulCh: A multi-layer channel router using one, two, and three layer partitions. In *IEEE International Conference on Computer-Aided Design (ICCAD-88)*, pages 88–91. IEEE Computer Society Press, 1988.

[24] Ronald I. Greenberg and F. Miller Maley. Minimum separation for single-layer channel routing. *Information Processing Letters*, 43(4):201–205, September 1992.

[25] Ronald I. Greenberg and Jau-Der Shih. Feasible offset and optimal offset for single-layer channel routing. In *Second Israel Symposium on Theory of Computing and Systems*, pages 193–201, June 1993.

[26] Ronald I. Greenberg and Jau-Der Shih. Minimizing channel density with movable terminals. In *Third Great Lakes Symposium on VLSI*, pages 1–4, March 1993.

[27] Akihiro Hashimoto and James Stevens. Wire routing by optimizing channel assignment within large apertures. In *Proceedings of the 8th ACM/IEEE Design Automation Conference*, pages 155–169. IEEE Computer Society Press, 1971.

[28] F. L. Heng. *Two-Dimensional Compaction: Computing the Shape Function of a Slicing Layout.* PhD thesis, Department of Computer Science, Princeton University, October 1988.

[29] W. Heyns. The 1-2-3 routing algorithm or the single channel 2-step router on 3 interconnection layers. In *Proceedings of the 19th ACM/IEEE Design Automation Conference*, pages 113–120. IEEE Computer Society Press, 1982.

[30] Cliff Yungchin Hou and C. Y. Roger Chen. A hierarchical methodology to improve channel routing by pin permutation. In *IEEE International Conference on Computer-Aided Design (ICCAD-91)*, pages 440–443. IEEE Computer Society Press, 1991.

[31] David S. Johnson, Andrea S. LaPaugh, and Ron Y. Pinter. Minimizing channel density by lateral shifting of components. Manuscript, 1992.

[32] H. Kobayashi and C. E. Drozd. Efficient algorithms for routing interchangeable terminals. *IEEE Trans. Computer-Aided Design of Integrated Circuits*, 4(7):204–207, July 1985.

[33] Andrea S. LaPaugh and Ron Y. Pinter. On minimizing channel density by lateral shifting. In *IEEE International Conference on Computer-Aided Design (ICCAD-83)*, pages 123–124. IEEE Computer Society Press, 1983.

[34] Andrea S. LaPaugh and Ron Y. Pinter. Channel routing for integrated circuits. *Annual Review of Computer Science*, 4:307–363, 1990.

[35] Andrea Suzanne LaPaugh. *Algorithms for Integrated Circuit Layout: An Analytic Approach*. PhD thesis, Department of Electrical Engineering & Computer Science, Massachusetts Institute of Technology, November 1980. MIT/LCS/TR-248.

[36] D. T. Lee and Majid Sarrafzadeh. Maximum independent set of a permutation graph in k tracks. In *Proceedings of the Second Annual International Symposium on Algorithms*, 1991.

[37] Charles E. Leiserson and F. Miller Maley. Algorithms for routing and testing routability of planar VLSI layouts. In *Proceedings of the 17th ACM Symposium on Theory of Computing*, pages 69–78. ACM Press, 1985.

[38] Charles E. Leiserson and Ron Y. Pinter. Optimal placement for river routing. *SIAM Journal on Computing*, 12(3):447–462, August 1983.

[39] H. W. Leong and C. L. Liu. Permutation channel routing. In *Proceedings IEEE International Conference on Computer Design*, pages 579–584. IEEE Computer Society Press, 1985.

[40] H. W. Leong, D. F. Wong, and C. L. Liu. A simulated-annealing channel router. In *IEEE International Conference on Computer-Aided Design (ICCAD-85)*, pages 226–228. IEEE Computer Society Press, 1985.

[41] Li-Shin Lin and Sartaj Sahni. Maximum alignment of interchangeable terminals. *IEEE Trans. Computers*, 37(10):1166–1177, October 1988.

[42] F. Miller Maley. *Single-Layer Wire Routing and Compaction.* MIT Press, 1990.

[43] S. Masuda and K. Nakajima. Optimal algorithms for interval containment graphs. In *Proc. 29th Midwest Symposium on Circuits and Systems*, pages 431–434, Lincoln, NE, August 1986.

[44] S. Masuda, K. Nakajima, T. Kashiwabara, and T. Fujisawa. Efficient algorithms for finding maximum cliques of an overlap graph. Technical Report UMIACS-TR-86-31.1, University of Maryland Institute for Advanced Computer Studies, College Park, January 1988.

[45] Carver A. Mead and Lynn A. Conway. *Introduction to VLSI Systems.* Addison-Wesley, Reading, MA, 1980.

[46] Andranik Mirzaian. River routing in VLSI. *Journal of Computer and System Sciences*, 34:43–54, 1987.

[47] Andranik Mirzaian. A minimum separation algorithm for river routing with bounded number of jogs. In *IEEE International Conference on Computer-Aided Design (ICCAD-89)*, pages 10–13. IEEE Computer Society Press, 1989.

[48] Ron Yair Pinter. *The Impact of Layer Assignment Methods on Layout Algorithms for Integrated Circuits.* PhD thesis, Department of Electrical Engineering & Computer Science, Massachusetts Institute of Technology, August 1982. MIT/LCS/TR-291.

[49] Vijay Pitchumani and Qisui Zhang. A mixed HVH-VHV algorithm for three-layer channel routing. *IEEE Trans. Computer-Aided Design of Integrated Circuits*, CAD-6(4):497–502, July 1987.

[50] James Reed, Alberto Sangiovanni-Vincentelli, and Mauro Santomauro. A new symbolic channel router: YACR2. *IEEE Trans. Computer-Aided Design of Integrated Circuits*, CAD-4(3):208–219, July 1985.

[51] Sartaj Sahni and San-Yuan Wu. Two NP-hard interchangeable terminal problems. *IEEE Trans. Computer-Aided Design of Integrated Circuits*, 7(4):467–472, July 1988.

[52] M. Sarrafzadeh. Channel-routing problem in the knock-knee mode is NP-complete. *IEEE Trans. Computer-Aided Design of Integrated Circuits*, 6(4):503–506, July 1987.

[53] H. Shin and A. Sangiovanni-Vincentelli. A detailed router based on incremental routing modifications: Mighty. *IEEE Trans. Computer-Aided Design of Integrated Circuits*, CAD-6(6):942–955, November 1987.

[54] Alan Siegel and Danny Dolev. Some geometry for general river routing. *SIAM Journal on Computing*, 17(3):583–605, June 1988.

[55] Kenneth J. Supowit. Finding a maximum planar subset of a set of nets in a channel. *IEEE Trans. Computer-Aided Design of Integrated Circuits*, CAD-6(1):93–94, January 1987.

[56] Thomas G. Szymanski. Dogleg channel routing is NP-complete. *IEEE Trans. Computer-Aided Design of Integrated Circuits*, CAD-4(1):31–41, January 1985.

[57] C. D. Thompson. *A Complexity Theory for VLSI.* PhD thesis, Department of Computer Science, Carnegie-Mellon University, 1980.

[58] Martin Tompa. An optimal solution to a wire-routing problem. *Journal of Computer and System Sciences*, 23:127–150, 1981.

[59] Spyros Tragoudas. Algorithms for channel routing with interchangeable terminals. Technical Report 92-10, Southern Illinois University Department of Computer Science, 1992.

[60] Spyros Tragoudas and Ioannis G. Tollis. River routing and density minimization for channels with interchangeable terminals. Technical Report 92-11, Southern Illinois University Department of Computer Science, 1992.

[61] Jr. W. Lipski. On the structure of three-layer wirable layouts. In Franco P. Preparata, editor, *VLSI Theory.* Volume 2 of *Advances in Computing Research*, pages 231–243. JAI Press, 1984.

[62] Takeshi Yoshimura and Ernest S. Kuh. Efficient algorithms for channel routing. *IEEE Trans. Computer-Aided Design of Integrated Circuits*, CAD-1(1):25–35, January 1982.