Yale University
Department of Computer Science

Channel Routing for Integrated Circuits (a survey)

Andrea S. LaPaugh and Ron Y. Pinter

YALEU/DCS/TR-713
June 1989

To appear in the *Annual Review of Computer Science, Vol. 4*, 1989.

Channel Routing for Integrated Circuits (a survey)

Andrea S. LaPaugh
Dept. of Computer Science
Princeton University
Princeton, NJ 08544

Ron Y. Pinter[*]
Dept. of Computer Science
Yale University
New Haven, CT 06520

1 Introduction

In this paper we review the current knowledge and major research directions concerning *channel routing*, a key problem in the automatic layout of electronic circuits. This problem was introduced by Hashimoto and Stevens in 1971 [40] as part of their layout method for printed circuit boards. It enjoys a continuing popularity for a broad variety of technologies (from printed circuit boards to integrated circuits), at all levels of complexity (from MSI, Medium Scale Integration, to VLSI, Very Large Scale Integration), and under various design methodologies[1].

In the channel routing framework, given a placement of circuit components (on the chip or the board), the routing area is divided into rectangular regions called channels (see Figure 1). The routing problem is then decomposed into two: deciding which channels each wire uses for interconnections, called global routing, and deciding where within each channel each wire runs, called channel routing. Variations on the framework allow for non-rectangular regions [56, 74].

Other, more general routing methods simply model the whole routing area as a plane containing obstacles. For example, the Lee-Moore routing algorithm [57], one of the earliest routing methods which is still popular today, routes one wire at a time in breadth-first fashion through this plane of obstacles. In contrast, the channel routing model decomposes the routing of each wire into the two stages: global and channel (*i.e.* local), and the interactions between wires is considered at each stage differently. The use of channel routing presupposes the validity of imposing the channel structure on a routing problem.

There are two types of channels: of fixed width and of variable width (the channel's width is the distance between its sides, as defined formally in Section 2). In the first, the channel

[*]On sabbatical leave from the IBM Israel Scientific Center.

[1]This is readily apparent upon examining both the routing algorithms used in available layout systems as well as the papers presented at such conferences as the International Conference on Computer Aided Design, the Design Automation Conference, and a number of other more specialized conference series. Many of these papers have subsequently appeared in journals such as the *IEEE Transactions on Computer-Aided Design for Circuits and Systems* and *Integration — the VLSI Journal.*

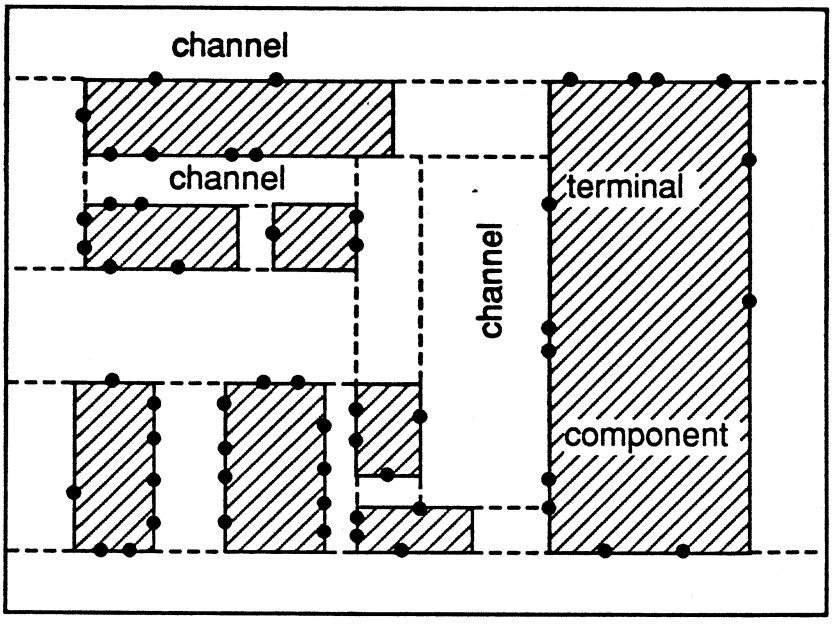

Figure 1: The channel structure of a layout.

is taken to be of a fixed size; the goal here is to route all the wires in the space provided by the channel. In the second type, channels are viewed as having unspecified width, *i.e.* the placement is somewhat flexible, and the goal is to *minimize* the width required to fit all the wires.

One design methodology, in which channel routing has been particularly successful, is that of standard cell (or polycell) layout (*e.g.* [53, 93]). Here, predesigned components, called standard cells, are placed in rows on an integrated circuit. The components all have the same height, yielding rectangular routing regions between the rows of components, *i.e.* natural routing channels.

Channel routing has also been used extensively in less structured layout styles. In full custom layout, the layout is not restricted to follow a particular pattern. However, when channel routing is used, the layout is decomposed into components (often — but not necessarily — rectangular) and routing channels. In this layout style, there is flexibility in how the routing area can be decomposed into routing channels; finding a set of routing channels which yields the best routing becomes an issue. This is called the *channel definition* problem [83]. Custom layout is often done using a hierarchy of components. At each level of the hierarchy, the decomposition into components and channels occurs.

Regardless of the design methodology, when routing channels are used, the channels must be considered by all the phases of the layout process (sometimes even within the generation of individual cells). While placing components and while finding global routes, some measure

of the resulting routing requirements for each channel must be made. If the channels are of fixed size, one must determine whether a given placement can be routed; if the channels have flexible width, one must calculate the width of the channel. Since one is rarely able to compute the channel width without actually calculating the entire routing, a time consuming process, estimates of width which are inexpensive to compute are used.

In this paper we are concerned with algorithms which actually route the wires in a channel. (A more general review of routing problems can be found in [2, 63].) We distinguish calculating routability from calculating the routing. In some cases, such as river routing (treated in Section 5), determining whether a channel is routable can be done (accurately) more quickly than actually calculating the path of every wire. This is due to the fact that generating an actual routing must determine every twist and turn of every wire, and there may be many, while routability can be captured as a set of constraints which guard against the channel getting too congested for successful routing.

Many of the algorithms discussed here are approximation algorithms or heuristic algorithms: they do not find optimal channel routings but will find some legal routing, hopefully a good legal routing. We will evaluate these algorithms both for the quality of the solution they produce and for efficiency. We will judge the quality of solutions under two criteria: the theoretical ability of the algorithm to find or approximate the optimal solution, and the practical performance of the algorithm. We apply the theoretical approximation measure to algorithms which route flexible channels; we wish to bound the *performance ratio* of these algorithms. The performance ratio is the ratio of the channel width found by the algorithm to the minimum channel width. In judging the practical performance of an algorithm, we ask how well it does on specific instances of the problem in comparison with other algorithms and in comparison with lower bounds on the channel width.

Note that an algorithm may produce excellent results in practice but have a terrible theoretical performance ratio because of pathological instances of the problem which rarely or never occur in practice. On the other hand, an algorithm with a good performance ratio may not be a very good practical algorithm. For example, the algorithm of Baker, Bhatt, and Leighton [4] (presented in Section 6), has a performance ratio of two[2], but it always produces channels which are twice as wide as the lower bound suggested by wire congestion. In contrast, a good practical algorithm comes (additively) within one or two of the lower bound fairly often.

In other words, to achieve a provably good performance ratio, an algorithm may have to perform uniformly on all instances, whereas a practical algorithm may do horribly on some instances but very well on those that count. We are still interested in algorithms with provably good performance ratios, since they give us a lot of insight into the structure of the channel routing problem. However, we treat the two classes of algorithms separately.

Although we do survey practical algorithms, our emphasis is on exact analysis of performance, both in terms of the optimality of the results and the complexity of the algorithms

[2]In fact, the width obtained is the maximum between twice the *density* and a larger constant times the *flux*, which are two disjoint measures, but since density is usually much larger it is reasonable to say that the performance ratio is 2.

used[3]. The methods we discuss are discrete. To put this focus into perspective, a brief review is due.

Over the past two decades of research on layout, three major schools have emerged. One school, which comprises Akers [2], Donath [24], Heller *et. al.* [41], Soukup and Royle [94], and El-Gamal [25], and subsequently the simulated annealing community [60, 101], has engaged in continuous analysis of routing phenomena and their effect on placement considerations. They first model the terminals' distribution along the channel's sides with real-valued random variables. Then, using probability theory, they try to obtain good estimates on routing requirements and measure the effect of different distributions on the interaction between routing situations. The heuristics guiding detailed routing (inside the channel) are influenced by this modeling and do not handle local phenomena all that accurately.

The second approach was pioneered by the work on the LTX system at Bell Laboratories conducted in the 1970s [20, 53, 73], and was then followed by Rivest and Fiduccia [85], Kuh *et. al.* [67, 103], Burstein and Pelavin [14, 15], and Sangiovanni-Vincentelli *et. al.* [82, 88]. Here discrete (combinatorial) methods are used to try and capture the fine points that are due to local perturbations in the terminals' set-up. These methods have been rather successful in practice, but no attempt has been made to derive any formal results on the quality of the resulting layouts (in terms of optimality).

The third approach has developed within the theoretical computer science community, where there is an interest in the intrinsic complexity of routing problems and their combinatorial structure. The results of some accurate (and formal) analyses, such as in [4, 11, 59, 81, 96] and other papers, and the advances in the design and analysis of algorithms over the past years, show that indeed a careful examination of the layout problem using discrete tools is both feasible and useful. Some of the insights that were gained by this research have contributed to the maturing of the theory and practice of channel routing, and this is why we have chosen to focus on these developments in this review.

A nice example that brings this point home, by showing how trivial changes to the routing configurations may cause disastrous effects in the quality of the layout, is the *one-shift* channel (Figure 2). Imagine two identical rows of n terminals (equally spaced by the minimum allowed distance — which is the grid size — on each side) that must be connected in order. This goal can be achieved without any cost in area or wire length by simply putting the two rows right on top of each other. If, however, you place them on modules across a channel such that one row is shifted against the other by just one grid unit, you are in for an unpleasant surprise. Depending on how many layers you are allowed to use and on the fashion in which you are allowed to route them, you may have to place the two modules n grid units apart (for a one layer realization [23, 59]), $\Theta(\sqrt{n})$ units apart[4] (for Manhattan routing [11]), or just one unit apart (for "real" two-layer design rules [5, 75]). Thus small perturbations in placement create vastly different routing situations, incurring area penalties that are orders of magnitude apart.

The rest of this paper is organized by issues (rather than specific "results", technological

[3] For a more practically oriented survey, see [13].

[4] In which case you also need to add a similar number of routing columns outside the channel's original boundaries.

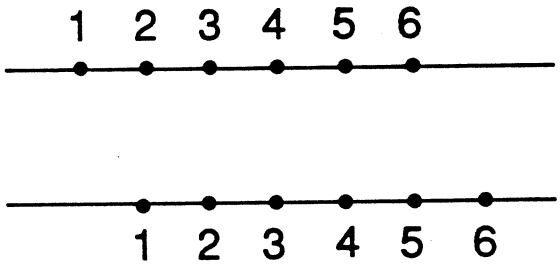

Figure 2: The *one-shift* channel routing problem: large width may be required due to the slight misalignment of terminals.

characteristics, or historic developments) as follows. We start off in Section 2 by defining the terms that are used throughout the paper to discuss the problem (some additional local terminology is introduced on the fly) and the wiring models for which it is being solved. Section 3 discusses lower bounds on channel width that can be established, followed (in Section 4) by a survey of some intractability results that are typical for the most common cases of the problem. Section 5 provides a brief description of single-layer routing, which can indeed be solved optimally, and then some provably good approximation algorithms and algorithms for other special cases are presented in Section 6. We classify and outline the major trends in practically good algorithms in Section 7, followed (in Section 8) by a brief survey of related problems and their solutions (using techniques developed for channel routing). We conclude (in Section 9) with a discussion of how the state of the art as it has been presented in this survey should guide us in trying to solve the channel routing problem, and finally list the most important remaining open problems.

2 Framework

In this section we formally define the basic terminology necessary to discuss the channel routing problem and algorithms for its solution. We first list some of the conventional names for parts and properties of channels, and then present a number of wiring models that are commonly used to solve the routing problem.

2.1 Terminology

A *channel* is the area between two (possibly infinite) parallel lines, called *sides*, with labeled points along them, called *terminals* (or ports). All the terminals bearing the same label constitute a *net*; for convenience, we use the integers $1, \ldots, n$ as labels (hence n is the number of nets), as in Figure 3. Channel routing is the problem of connecting all terminals of each net (by wires or integrated circuitry) within the given space in such a way that no two nets are shorted (electrically); alternatively, the lines are initially separated by an

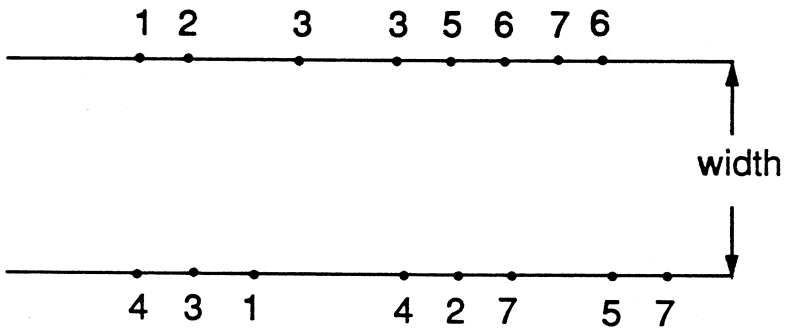

Figure 3: A channel routing problem.

unspecified amount, and then the objective is to minimize the distance between them, called *width.*

A net may contain any number of terminals[5]. We denote a net with k terminals as a k-point net; the most common distinction is between 2-point nets as opposed to nets with more than 2 terminals, that are often referred to as *multi-point* nets. A net may have representatives (terminals) on both sides of the channel, in which case it is said to be going *across* the channel; if all its points are on one side, then it is a *single-sided* net. Terminals on the upper side are called *top* terminals, and those on the other side — *bottom* terminals.

We now turn to the geometry of a channel. Throughout, we assume the existence of an underlying integer grid. Both channel's sides lie on grid lines, and all terminals reside at grid points. This grid will also be used when we discuss wiring models (in the next subsection).

Channels are customarily drawn horizontally, and we shall use this convention. Hence the following names are commonly used: Horizontal grid lines, which are parallel to the channel's sides and lie between them, are called *tracks.* Vertical lines, connecting the two sides, are *columns.* The distance between the sides is the width (as we have noted above), and it is one more than the number of tracks.

Often, the sides of the channel are finite segments (ending at the same columns on the left and the right of the channel). Then, the length of these lines is also the channel's *length.* A distinct notion is the channel's *span*, which is the **closed** interval between the leftmost and rightmost terminals that appear in it (and is always finite). A discrete variant of the span is the set of columns contained in the interval (including the extremes). Similarly, the span of a net is the interval between its leftmost and rightmost terminals. Finally, the channel's *extent* is the horizontal distance between the leftmost and rightmost columns that are used for routing. Again, the extent of an individual net is defined similarly for the wire realizing it. The length, the size of the span, and the extent of a channel may all differ from each other, depending on both the given layout of terminals on the channels' sides and on the actual routing that is being produced by a particular algorithm.

A few notions combine the topology of the channel with its geometry. A *trivial net* is a

[5]Even though nets with one terminal are not interesting, since they do not require any routing.

2-point net going across the channel with both its terminals at the same column. More importantly, the channel's *density* is a metric encapsulating both the geometric and topological structure of the nets in the channel. The density at a point is defined as the number of nets that have terminals both to the left and to the right of the point. If the point happens to be on a column, the nets whose terminals are on the same column do contribute to the density unless they belong to a trivial net (which occupies this very column). The density of the whole channel is the maximal value of densities over all points within the channel.

At times, in addition to having terminals on the channel's sides, nets may have to connect outside the channel through its left and right endings. Such demands are grouped together at both sides into the *left set* and the *right set*, which must be satisfied by putting a wire carrying each member of the set on some track at the designated end.

Finally, there are two primary ways to represent a channel. The first, called *column list*, simply enumerates for each side the net numbers for each column in order from (say) left to right; a vacant terminal position is marked by a 0. The second, called *net list*[6], comprises an enumeration of the nets, each represented as a set of its constituent terminals. A net list is simply the representation of the hypergraph realized by the channel along with the geometric data specifying the terminals' locations.

2.2 Wiring Models

In general, each net can be realized as a tree residing inside the channel and connecting to the terminals on the sides (and at the ends, if necessary). To construct these trees, we have a number of conducting layers at our disposal, as well as ways to interconnect between them. A variety of methods may be used to assign layers to nets; such a method is called a *wiring model* and it significantly affects both the routing algorithm to be employed as well as lower bound arguments.

Wiring models are the rules of the road: they tell us what is allowed when routing a channel, but also guide us as to how to do it. More formally, wiring rules are both an abstraction of the design rules, which in turn reflect technological constraints [68, 100], and at the same time they serve to structure (and often limit) the domain of solutions (or the "search space") for the routing algorithm. For example, Figure 4 shows a possible solution to the problem of Figure 3 using the Manhattan wiring model, that will be discussed in what follows.

All models presented in this paper assume the underlying integer grid, and all wires reside on grid lines. In some cases, extensions can be made that allow other styles of wiring (such as non-integral coordinates, 45° angles, and circular curves), but these will be mentioned only when concrete results pertaining to them are known. Such results are few and far between.

Thus, we can assume (for purposes of the definitions) the existence of a *grid graph*, whose nodes are the grid points, and edges connect points that are adjacent vertically or horizontally (or, in other words, edges are pairs of points at unit distance from each other). Models

[6] Net lists are commonly used to describe the topology of entire designs.

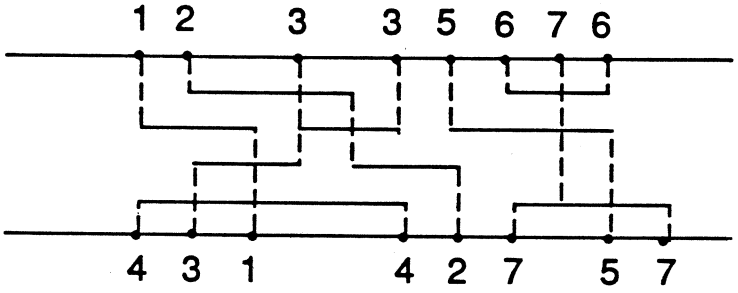

Figure 4: A solution to the channel routing problem of Figure 3.

are then formulated in terms of paths in the grid graph (or derivatives thereof, as explained in due course).

The primary classification for wiring models is the number of conducting layers that are used[7]. A secondary criterion deals with the ways in which these layers are allowed to interconnect and overlap, if at all. We now present a number of rectilinear wiring models, and then conclude this section by stating some criteria for how to relate different models to each other.

- **One-layer Wiring.** Wires in this model are vertex-disjoint paths (or trees) in the grid graph. Obviously, this model can be used only when the interconnection pattern is planar. For example, if all we have are 2-point nets going across the channel, then the terminals' labels must be in the same order (but their positions are not necessarily equally spaced) on both sides, to avoid crossovers; this situation is called *river routing.*

- **The Manhattan Model.** This is by far the most popular and well studied two-layer wiring model. In this model, which can also be called the *directional model*, each direction in the grid is preassigned to one of the two layers. For example, all horizontal lines are in metal, all vertical ones in polysilicon. Whenever a path turns a corner, a connection between the two layers is necessary; in MOS technologies this is facilitated by a contact (or a via). Thus paths may cross over each other, but cannot share segments, or turn at the same grid-point (and, obviously, for any two portions of disjoint nets, no turning point of one path may lie on a segment of the other).

- **The Knock-knee Model.** Introduced by Thompson [97] for other purposes, but later adopted by channel routing theorists as well. Here wires are edge-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 paths (wires), they are on different layers. This model has the advantage of avoiding undesirable electrical properties that are due to overlaps between different layers, such as capacitive coupling (see, for example, [68], Section 1.4), but the complexity of layer

[7] A layer is a medium that can be used independently of other layers, hence — for example — polysilicon and diffusion are considered *one layer*, since their overlap creates a pass transistor.

assignment may become the overriding issue since now we have the freedom to change layers at any "free" grid-point. Problems and solutions pertaining to this subject are discussed in detail in Sections 4.2 and 6.2.

- **General Two-layer Wiring.** This model allows any rectilinear routing in two layers, and wires on different layers can overlap arbitrarily. The only restriction is that wires must lie on the grid, thereby imposing a uniform unit separation design rule. One way to formally define this model is to use the following rule: At each grid point, if the point contains a contact, then only one path goes through it and its layer gets changed there; if no contact is present, up to two paths may go through the point in any way they please. Alternatively, we may duplicate the grid graph, and connect vertices that represent the same grid points (to form a three dimensional grid of height 2). Then wiring in this model amounts to vertex disjoint paths in the new graph.

- **Limited Overlap Two-layer Wiring.** This model is a restriction of the previous, general model, in that it still allows arbitrary assignments of layers to directions, but the amount of overlap between wires is limited. There are two ways to limit this amount: either on the basis of pairs of nets (*i.e.* the total overlap between the trees realizing any two nets does not exceed a certain limit), or on a local basis (namely that any single overlap of two runs is less than a pre-set maximum).

- **One Layer per Net.** In this model (also known as the *multi-wire* model), each net is assigned a layer that is to be used for the whole tree realizing it without any layer changes allowed along the wires. We assume that all terminals are available in all layers, so each net can be assigned a layer by the routing algorithm. The number of layers required in this model is the chromatic number of the corresponding trapezoid graph [19].

- **Other Multi-layer Models.** 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, $i < k < j$. Providing a specific wiring model for three layers may still make sense, but for four layers or more we end up, in fact, routing in a three-dimensional medium [86]. In Sections 6 and 7 we briefly discuss results for three and more layers which generalize two-layer techniques, but three-dimensional routing is beyond the scope of this paper.

How do we measure the relative strength of these models, *i.e.* what does it mean for one model to be stronger than another? There are two primary criteria for doing this. One has to do with whether a model can at all be used to solve a given channel routing problem, and the second deals with the performance of the model in terms of the smallest width that it would allow.

More formally, M_1 is *topologically stronger* than M_2 if every channel that can be routed using M_2 can also be routed using M_1. Obviously, the one-layer model is topologically weaker than all other models. M_1 is *geometrically stronger* than M_2 if it is topologically stronger, and also if for every channel routing problem that can be solved in both models, the smallest width attainable using M_1 (regardless of the algorithm being used) is not larger

than the smallest width attainable under M_2. It is clear that the general two-layer model is geometrically stronger than the Manhattan model.

Clearly, some of the abovementioned models are incomparable. The sometimes intricate interrelation between them will unfold as this survey progresses.

3 Lower Bounds

Evidently, for any given channel there is a minimal width that is required in order to route it correctly, obeying all the rules set by the wiring model. Before we can discuss algorithms for solving the channel routing problem or even characterize its complexity, we should find out what these lower bounds are. All bounds are relative to a wiring model; some are parametric versions of a generic bound or are model specific in other ways, whereas others depend further on certain routing styles and hence may apply only to a class of algorithms that follow this style.

Lower bounds often serve as predictors for area requirements during the placement design phase. Therefore, for each bound that we describe, we state the exact conditions for which it applies, and also state the computational complexity of finding it. We conclude this section with a discussion evaluating the relative merits of the bounds and how they relate to other relevant results.

3.1 Density

Let us reiterate the definition of density. If we impose a Cartesian coordinate system on the plane in which the channel is drawn horizontally, then the density at a point depends only on its x coordinate, and $d(x)$ is then the number of nets whose span contains x. The only exception are trivial nets: they do not contribute to the density at the (points of the) column on which they reside. For a channel, the density d is equal to $\max_{-\infty<x<\infty} d(x)$.

The channel's density is easy to compute: set its initial value to 0, then scan the channel from (say) left to right, and at every column update the value depending on the type of terminals encountered (leftmost in a net, rightmost, or neither). For a channel given in column list format this requires a preprocessing pass for finding the extremal terminals of each net, but the entire process is still linear in l, the length of the channel. If the channel is given in net list form, pre-sorting of the terminals is necessary, taking time $O(n \log n)$. In this case, however, empty columns do not need to be visited during the scan and for sparse channels this may be advantageous.

For the Manhattan wiring model, which does not allow **any** kind of wire overlap, $d+1$ is an obvious lower bound on the channel's width, since at those points along the channel where d equals $d(x)$, at least as many as d tracks are necessary to carry the wires that must cross x in order to realize as many nets (and the width w is one more than the number of tracks).

When overlap between wires is allowed, some modifications to this argument are needed.

For the knock-knee model, which only allows single point overlap between tracks, we can simply redefine spans of nets to be half-closed, half-open intervals (say, closed on the left and open on the right). Then, the definition of density as given above can be used verbatim, since two nets can share a track at a column.

When we move on to wiring models that allow true overlap between wire segments, such as the general two-layer model and the one layer per net model, the channel's width is trivially bounded by $d/k + 1$, where k is the number of layers, since as many as k wires can cross on the same track. Hambrusch [38] has improved this lower bound on the number of tracks to $d/(k-1) + 1$, generalizing the result of Leighton [58] for $k = 2$. Limited overlap does not change this argument, but more complicated analysis can lead to a sharper (*i.e.* larger) bound based on the shape of the function $d(x)$. Specifically, if the number of consecutive columns in which the density is above a certain value is more than the allowed maximum overlap, then obviously $d/k + 1$ cannot be attained.

3.2 Vertical Constraints

The density based lower bounds stem from the usage of tracks. However, the population of the columns in the channel also affects its width, and often in more intricate and complicated ways. We start with the simplest bound of this kind, which applies to a restrictive style of Manhattan routing.

Let us assume that each net is routed using only *one* track, namely every wire is formed out of one straight horizontal strip running along the net's span with straight vertical connections to the terminals. For every column in the channel, the vertical portions occupying it must be disjoint (since overlap is not allowed), hence (for each column) the track realizing the net of the top terminal must lie *above* the track that connects to the bottom terminal. Since there are at most *two* terminals in each column, the relation that this argument induces on nets can be captured [53] by the channel's *vertical constraints graph*, $G = (V, E)$, as follows: V is the set of nets (one node per net), and we draw a directed edge from u to v if there exists a column in the channel in which the lower terminal belongs to u, and the upper to v.[8]

The paths structure in this directed graph reflects two important properties of this restricted form of routing. First, if the graph contains a (directed) cycle, then it is impossible to complete the channel route. This is easy to see, since a cycle means that there is no linear ordering of all tracks that satisfies all the vertical constraints in the cycle. The simplest cycle is a 2-cycle, in which the track of net i must be above that of net j and vice versa. This situation may arise for two 2-point nets whose terminals are in the same columns but in reverse order (the notorious `X` formation), as in Figure 5.

Second, the longest simple path in G is a lower bound on how many tracks are necessary if we route in this restrictive fashion. This is true since any track assignment must obey the partial order specified by G, and in particular the chains — represented by paths — must be followed. A dramatic example of how dominant this bound can be (as compared

[8] Parallel edges — going in the same direction — are merged into one edge.

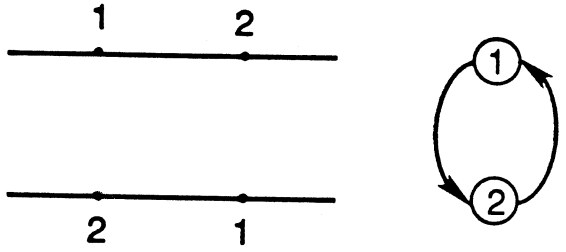

Figure 5: An X formation — the simplest loop in a vertical constraints graph.

to density) is the one-shift, discussed in Section 1, where the vertical constraints graph is simply one long chain with all nets lined up one above the other.

Constructing the vertical constraints graph takes time $O(l)$ or $O(n \log n)$, depending on how the channel is represented. Finding the longest path in it or detecting a cycle may take, in general, as long as $O(|E|)$, which is $O(\min(l, n^2))$. However, if the longest path is of length p (and, thus, obviously there are no cycles), the general algorithm will take only time $O(n \cdot p)$; since often $p \ll n$ this is still reasonable.

This tool can be modified to reason about knock-knee wiring as well. Now we have to put a weight of 0 on every edge that stems from the relation between a leftmost terminal of one net and the rightmost of the other (in either direction), and a 1 on all other edges. A weight of 0 means that the tail's net cannot be assigned to a track above that of the head, but they could use the same track; a weight of 1 indicates, as before, that one track must be strictly above the other. Then, the bound on the number of tracks is the largest sum of weights on any of the paths in the graph. In the case of the one-shift the whole channel now collapses to a single track, and in general the longest weighted paths are shorter.

This style of routing, namely one track per net, is rather restrictive, and it is especially so when many of the nets are multi-point rather than two-point nets. Hence, the following refinement of the vertical constraints graph is sometimes used: Each net is partitioned into *zones* [103], delineated by the successive terminals along it. Now, instead of using one track along the whole net, we may use a different track for each zone. The vertices of the modified vertical constraints graph are now the zones, and an edge is drawn between two zones (in the appropriate direction) if the terminals at one of their endpoints are in the same column. The paths in this new graph are not longer than those in the original graph, resulting in possibly narrower channels. Moreover, cycles may have disappeared: The mere presence of an intermediate terminal allows the change of tracks within the net's span, as in Figure 6.

It is still the case, however, that vertical wiring for a given net occurs only at columns where it has terminals. This restriction is not part of any of the wiring models presented in Section 2, and is indeed unrealistic. Unfortunately, the idea underlying the vertical constraints graph cannot be extended further to deal with these models. If, for example, we break each net into as many zones as the columns it spans, those zones that have no

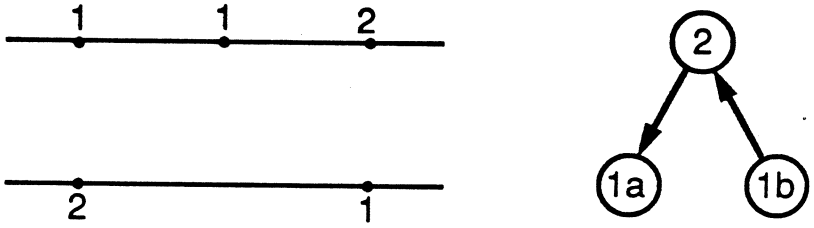

Figure 6: Modifying the vertical constraints graph to be based on zones helps in breaking cycles.

terminals at their ends would "float" about; trying to relate them to other zones (by means of edges) almost amounts to the routing task itself. Furthermore, even this extension does not capture the phenomena occurring when wires extend beyond the net's span.

3.3 Flux and related bounds

The first lower bound on channel width that is driven by vertical misalignment of terminals (of the same net), but is not tied to any specific routing style, was devised in 1981 by Brown and Rivest [11] for the Manhattan model. They provide a simple, albeit powerful counting argument which applies to routing both within a specified channel's length as well as for the case in which the extent is unbounded. We present the former case first; its generalization to the latter is then easier understood.

Let us assume that we have a channel of length l with n two-point nets going across the channel. Starting from the top side, we would like to count how many tracks are necessary so that every wire can carry the corresponding net from the column of the top terminal to that of the bottom terminal. First, we observe that all trivial nets should be routed straight down in their columns; there is nothing to be gained by using up tracks to move out of and then back into such a column. For notational convenience, let us now simply contract (or shrink) the channel by discarding the trivial columns, and adjust the values of l and n accordingly.

The key observation is as follows: in every track, only as many nets as there are *free* columns can move to another column, where a free column is one that is currently not occupied by another net. In the very first track (right under the top side), there are exactly $l - n$ free columns, since all n nets must be routed straight down from their terminals in order to enter the channel. However, regardless of how nets change their column assignment in this track, there will again be exactly $l - n$ free columns for the next track, etc. The total number of column changes that can be effected must be at least as many as the number of nets; formally, if we denote the number of required tracks by t, then

$$(l - n) \cdot t \geq n.$$

Since $w = t + 1$, we obtain

$$w > \frac{n}{l - n}.$$

For example, in the one-shift case (Section 1), this immediately yields the bound $w > n$ (since $l = n + 1$).

Notice, that the above argument did not restrict the track assignment in any way; in particular, the wiring style can be *non-monotonic* in the sense that a wire may use more than one track along a certain range of columns, thereby forming a U- or S-shaped path. This routing strategy seems self defeating at first if one tries to minimize the channel's width[9], but if one is allowed to use columns outside the channel's original length, this strategy may prove somewhat advantageous, as we see next.

On the first track, wires can be extended to two additional columns — one on the left of the channel and one on its right hand side. Even if these new columns are far from the channel's original ends, we can only extend two wires — one in each direction — since at the channel's end points only one wire can occupy the track. Thus, we have added two new columns to the game, allowing $l - n + 2$ nets to change their positions in the first track. For the second track, we have n nets that span $l + 2$ columns, but — applying the same argument again — they can now go to as many as $l - n + 4$ columns. In general, in the ith track we have $l - n + 2i$ columns at our disposal. Summing over the t tracks that are necessary, we obtain

$$\sum_{i=1}^{t}(l - n + 2i) \geq n,$$

or

$$(l - n) \cdot t + 2\sum_{i=1}^{t} i \geq n.$$

Substituting $t(t + 1)/2$ for the series, extracting t, and recalling that $w = t + 1$, we finally have

$$w > (n - l) + \sqrt{(l - n)^2 + 2n}.$$

For the one-shift, again, this establishes a bound that is $\Omega(\sqrt{n})$, which is tight since it can indeed be obtained using non-monotonic routing (as demonstrated in Section 6).

The above analysis is elegant, but gets weaker as the channel gets sparser, *i.e.* when $l - n$ grows. There are cases where the maximum between the density and a bound derived using the above method is still far less than what the best route can achieve. To remedy this situation, Baker, Bhatt, and Leighton [4] have sharpened the reasoning behind the column assignment argument and have proposed a new measure which is based on it, namely *flux*.

Similarly to density, which is determined by counting how many wires must cross each vertical line cutting the channel, flux is established using *horizontal cuts*. Such a cut is a contiguous interval on one of the channel's sides, and a net is split by the cut if it has a representative both inside the interval and outside it (on either side of the channel). The *length* of the cut is the number of columns it spans, and its *size* is the number of non-trivial

[9] Indeed it is so with respect to vertical non-monotonicity, *i.e.* the usage of columns in the same way [76].

nets split by it. The flux of a channel is the *largest* integer f for which there exists a horizontal cut of length $2f^2$ and of size $2f^2 - f$.

Thus, flux localizes the effect of column congestion — which was previously applied to the entire channel — to the region specified by the cut, and then we maximize over the whole channel. Using analysis that is similar to that of [11], the flux of a channel is a lower bound for the number of tracks necessary to route it. In practice, flux is usually low, but in general it may become the dominant factor for some cases.

Flux is harder to compute than density, *i.e.* it cannot be found by a simple linear scan of the channel. This is largely due to the fact that flux is not a well behaved measure in the following sense: for a given channel we may be able to establish a flux of $f + 1$, but there does not exist a single horizontal cut within it which would substantiate a flux of f. Hence, there are quadratically many cuts to consider, the work required for each cut is linear in its size, and the bookkeeping required to maintain the split nets information is somewhat cumbersome.

To remedy this situation (among other reasons), a new measure called *smooth flux* was recently defined [42]. The smooth flux of a horizontal cut is defined as the *smallest* integer ω for which

$$\omega e + \omega(\omega + 1) + (\omega - 1)(u + r) \geq s,$$

where e is the number of empty columns in the cut, u is the number of unsplit (single-sided) nets in it, s is the number of split nets in it, and r is the number of redundant terminals, namely $r = c - e - s - u$ (c is the number of columns in the cut). The smooth flux of the channel is the largest smooth flux of any of the horizontal cuts in that channel. Smooth flux yields the same results as for flux in terms of channel width lower bounds, but it is easier to compute (although still taking quadratic time in n) and otherwise manipulate.

3.4 Discussion

None of the lower bounds described above can be used by itself, *i.e.* a different measure may dominate the others for any given channel. For the Manhattan wiring model, however, $\max(d, f)$ was found in [4] to be sufficient in the following sense: there exists a routing algorithm that always yields a channel of width $2d + O(f)$ (and when all nets are two-point the width can be guaranteed to be $d + O(f)$). This approximation result, intellectually pleasing as it is, is not useful in practice due to the constant factors involved; we shall see in Section 7 how most heuristics — even though they cannot *guarantee* good worst-case behavior — almost always achieve density.

Of the above bounds, only density is universal. All the others (with the exception of vertical constraints graphs when applied to the knock-knee model) apply only to the Manhattan wiring model. It is important to note that exact mathematical analysis was also performed for two other models. In both cases, which will be described in detail later in this paper, the analysis took the form of supplying necessary and sufficient conditions for routability, rather than rendering a lower bound. They are:

- River routing has been fully analyzed [59], and for any given channel it is possible to find the optimal routing with respect to many criteria, including channel width. These results also extend in part to the one layer per net model, with certain provisions. The full discussion can be found in Section 5.

- Frank [28] provides necessary and sufficient conditions for knock-knee routing of two-point nets in a predefined area, namely when both dimensions (length and width) are given. His analysis makes extensive use of cut sets of the edges in the underlying grid graph (since he is looking at edge disjoint paths), and it allows terminals on all four sides of the routing region. This is directly applicable to switchbox routing, as discussed in Section 8.

Such exact analysis, however, applies only in rare cases. As we shall see in the next section, the problem of determining the channel's width is often intractable. Therefore, the lower bounds that we have described in this section are often used to evaluate the performance of various heuristic algorithms that are employed to solve the channel routing problem in its general setting. As noted above, they also often serve as area estimates during the placement process, when it is too expensive to actually route while making a decision about the relative positioning of circuit components. In both cases, one should be honest and careful when using these bounds, since, as we saw, there are many qualifications and other considerations that affect their validity and sharpness.

4 Intractability of Channel Routing Problems

It is generally accepted that the channel routing problem is a difficult one. Under the two most popular routing models, we have actual mathematical evidence of this, namely NP-completeness results[10]. The main result is that of Szymanski, that channel routing under the Manhattan model is NP-complete. Following the structure of Szymanski's proof, Sarrafzadeh later showed that channel routing under the knock-knee model is also NP-complete. There are also results addressing the complexity of dogleg-free Manhattan routing and layer assignment for knock-knee routing.

4.1 Manhattan Routing

The first NP-completeness result for Manhattan routing was that of LaPaugh [54]. This work addressed the complexity of Manhattan routing when no doglegs are allowed, *i.e.* when each wire must use at most one (horizontal) track. As discussed in Section 3, the length of the longest path in the vertical constraints graph is a lower bound on the width of the channel for this routing model, and the routing cannot be realized if the vertical constraints graph contains a cycle. The NP-completeness result shows that given an instance of a channel routing problem with an acyclic vertical constraints graph, determining whether the

[10] A review of NP-completeness results up to 1982 for various routing problems, including but not limited to channel routing, can be found in David S. Johnson's NP-Completeness column [46].

channel can be routed in width k under the dogleg-free Manhattan model is NP-complete. The reduction is from circular arc graph coloring.

Formally the two problems and the reduction are stated as follows:

Channel routing under the dogleg-free Manhattan model

Input: a routing channel and a positive integer k.

Question: Can the channel be routed in at most k tracks under the dogleg-free Manhattan routing model?

Circular arc graph coloring

Input: a finite set of arcs of a circle and a positive integer k.

Question: Consider the graph defined by the arcs as follows. There is one node for each arc of the circle. Two nodes are connected by an edge if the arcs they represent intersect. This is the circular arc graph defined by the arcs. Can this graph be (node) colored with k colors? (i.e. can the nodes of the graph be colored with k colors so that no two nodes connected by an edge are given the same color.)

The reduction basically cuts the circle at some point, straightens it to make a channel and makes each arc a net. Extra nets are added to preserve the order at the ends of the channel, since at the ends of the channel are nets which were originally arcs cut in two. All nets are 2-point nets; thus NP-completeness is actually proven for dogleg-free Manhattan routing of 2-point nets.

Although dogleg-free routing is not a practical model, this result is of interest because it teaches us something about the complexity of channel routing. It is particularly interesting when compared to the Manhattan routing of a channel with no vertical constraints. When there are no vertical constraints, doglegs are unnecessary; density can be achieved by a dogleg-free Manhattan routing. The routing is done by *interval graph coloring.* An interval graph is similar to a circular arc graph: intervals of a line define the nodes rather than arcs of a circle; overlap again defines the edges. As discussed in Section 6, interval graph coloring can be solved very efficiently, in time $O(n \log n)$. Thus the difference in complexity between circular arc graph coloring and interval graph coloring characterizes the difference between dogleg-free Manhattan routing of channels with and without vertical constraints.

The major complexity result for Manhattan routing is that of Szymanski [96]. He proves that under the Manhattan wiring model with doglegs allowed at any column, channel routing is NP-complete. The reduction is from 3-satisfiability, one of the classic NP-complete problems [32].

Channel routing under the Manhattan model

Input: a routing channel and a positive integer k.

Question: Can the channel be routed in k tracks under the Manhattan routing model?

3-Satisfiability

*Input:*A finite set of Boolean clauses (Boolean sum of literals) with at most three literals in any clause.

Question: Is there a truth assignment to the variables appearing in the clauses that makes all the clauses simultaneously true (or equivalently makes the Boolean product of the clauses true)?

The reduction uses pairs of nets to represent Boolean variables. The order in the channel of the horizontal wires for these nets represents a truth assignment. There is a block of terminals for each clause. The blocks are constructed so that when a literal x appears in a clause, vertical constraints force the nets for x to use an extra track if x is false under the truth assignment — the nets will be in the wrong order for the vertical constraint. Because doglegs are allowed, the actual construction must be very careful to limit the way in which doglegs can occur. *Enforcer blocks* are used between clause blocks to maintain the order of nets. The basic construction produces for each variable a pair of nets that extend across the entire channel and have many terminals — up to four times the number of clauses. A modification of the construction breaks up these nets to produce nets with at most 4 terminals; in fact, Szymanski and Yannakakis have a further modification[11] which will reduce the number of terminals per net to two.

4.2 Knock-knee Routing

The Manhattan routing model implicitly deals with layer assignment by preassigning a layer to each direction. However, in the knock-knee model, the path of each wire and the layer of each wire segment must be determined. The knock-knee model is not a routing model using a specific number of layers. The model requires that edge-disjoint paths be used for distinct nets, but places no restrictions on how layers are used other than the general rules for contacts in multi-layer models. Often the routing problem is broken up into two parts: find a set of edge-disjoint paths which interconnects the given nets and uses a given number of tracks (or minimizes the number of tracks) — the knock-knee wire layout [81]; given a knock-knee wire layout, find a legal layer assignment using a given number of layers or minimizing the number of layers.

Consider first the entire routing problem under the knock-knee model for two layer routing. As discussed in [96], Szymanski's result holds under less constrainted routing models where layers can be assigned to wire segments in any way consistent with the design rules, as long as no horizontal wires are allowed to overlap and only two layers are available. This generalization is achieved by modifying Szymanski's construction slightly to guarantee that vertical wires and horizontal wires must run in different layers. Therefore, determining if a channel can be routed in two layers and k tracks under the knock-knee model is NP-complete.

Sarrafzadeh proves that the knock-knee wire layout problem is NP-complete [89]:

Knock-knee wire layout in a channel

[11] claimed as a footnote in [96]

Input: a routing channel and a positive integer k.

Question: Is there a set of edge-disjoint paths that interconnects the nets of the channel and uses k tracks?

The reduction is a modification of the reduction from 3-satisfiability that Szymanski uses. The outline of the construction is the same; each net constructed has four or five terminals. Thus the knock-knee wire layout problem for multi-point nets with as few as five terminals in the largest net is NP-complete. In contrast, the knock-knee wire layout problem for 2-point nets is solvable in polynomial time [81] (see Section 6). The complexity of the problem when all nets have at most four terminals and some nets have at least three terminals is not known.

Now suppose we have a wire layout and we wish to do the layer assignment? What is the complexity of this problem? Brady and Brown have shown that four layers suffice for any knock-knee wire layout [9]. Thus the question before us is whether we can use two or three layers instead. Pinter has presented a polynomial-time algorithms to determine whether a given wire layout can be realized in two layers [78]. But not all the news is good. Lipski has shown [62] that it is NP-complete to decide if a wire layout can be realized using three layers:

Three-layer wirability

Input: a routing channel with 2-point nets and a set of node disjoint paths interconnecting the nets.

Question: Can each grid segment of each path be assigned one of three layers so that no intersecting paths have the same color and the rules of layer change are followed?

The reduction is again from satisfiability.

The reader should not be confused by the fact that 2-point knock-knee routing under two layers is NP-complete while each of wire layout for 2-point nets and two-layer assignment are polynomial-time solvable. The algorithm of Preparata and Lipski [81] to find a knock-knee wire layout of a set of two point nets in channel width equal to density is guaranteed to produce a wire layout which can be implemented in three layers. However, this layout may not be implementable in two layers, and there may exist some other wire layout that uses width equal to density and can be implemented in two layers.

5 One-layer-per-net — a Tractable Case

Given the NP-completeness results of the previous section, one can only hope to find efficient algorithms which produce optimal solutions for special cases of the channel routing problem. The one layer (*i.e.* planar) wiring model, as well as its natural generalization — the one layer per net model, give rise to some of the few situations for which complete analysis is indeed possible. In most of these cases polynomial-time algorithms exist which indeed find the optimal solution to the routing problem at hand. In this section we give a precise

description of some of these configurations and state the results pertaining to them.

5.1 River-routing (the purely planar case)

A channel can be routed in one layer if and only if the interconnect pattern is planar. If all nets are two-point nets going across the channel, this simply means that the terminals on both sides must appear in the same net ordering. Such a channel can be specified by listing the terminal coordinates for each side; the net structure is then implicit.

The problem of routing such a channel in minimum width can be solved in polynomial-time for several wiring geometries, namely — not just on the rectilinear grid, but also for wires that go off the grid and use real-valued points, for straight line segments at other orientations (*e.g.* 45°), and even for circular arc segments. We present here the simplest case, namely that of rectilinear wiring, following the simple presentation in [59]; other relevant results can be found in [23, 71, 92, 98].

If we denote the terminal positions on the top and bottom sides by $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ respectively, then necessary and sufficient conditions for being able to route the channel using a given number of tracks, t, are

$$a_{i+t} - b_i \geq t$$

and

$$b_{i+t} - a_i \geq t$$

for all $1 \leq i \leq n - t$. The necessity of the conditions can be proven by applying wiring capacity arguments (which are similar to density measuring techniques) to lines that cut the channel both at 45° and 135° angles (these particular angles are derived from the rectilinear nature of the wiring; other angles may be necessary for other orientations and wire forms).

The sufficiency of these conditions can be demonstrated by using the following "greedy" technique[12]: Scan the channel from (say) left to right, and route the nets in the order in which you encounter their leftmost terminal (on the top or the bottom). Every net is routed in its entirety before we process the next net by first routing it across from the exposed terminal straight to the opposite side of the channel until we get within one unit of distance from the other side or previously routed wires, and then hug the contour of the opposite side (including the existing wires) at a distance of one unit until the other (target) terminal is reached. Again, using geometric arguments, it can be shown that this algorithm indeed finds a route within t tracks if the above conditions are fulfilled. The appearance of the wiring as non-intersecting rippling waves gave it — and the entire problem — the name "river routing".

In order to find the minimum possible number of tracks we can perform a binary search on the values of t between 0 and n (since it is obvious that if a channel can be routed with t tracks then it can be routed using $t + 1$ tracks as well). The overall time complexity

[12] Not to be confused with the "greedy" algorithm of Section 7.3.

of this process is $O(n \log n)$, since checking the condition for one value of t takes time proportional to $n - t$. A closer look at the conditions, however, reveals that one pass for finding the minimum t would suffice: start with $t = 0$ and $i = 1$, and iterate on i until a violation occurs; then increment t by 1 and resume with the last valid i. The total number of increments of both i and t is $2n$, resulting in an $O(n)$ time algorithm for finding the channel's width.

It is important to note that the number of conditions that need to be checked in order to decree a channel "routable" is linear in the number of nets, whereas the size of the actual routing, in terms of the number of wire segments, and the time required to create it is quadratic in the worst case (see, for example, [76]). This distinction between the time complexities of testing for the fulfillment of the (necessary and sufficient) routability conditions and that of executing the routing itself is quite singular in this domain (although it is common in other areas of combinatorics).

Moreover, the fact that the algebraic form of the left hand sides of the above conditions is a simple difference between two variables allows their usage for purposes of module placement and compaction with automatic jog introduction, as shown in [59, 65]. Furthermore, the form of the left hand side of the conditions remains similar for other wiring models as well — only the right hand side and the range specifications for the parameter t get more complicated. Consequently, the number of conditions to be checked may grow and become super-linear, but the applications to placement and compaction are still possible.

5.2 The Multi-layer Case

In the one layer per net model, we must first solve the layer assignment problem, *i.e.* we have to decide which wire is routed in what layer so as not to create shorts. If all nets are two-point nets going across the channel, this becomes a permutation graph coloring problem [27] that can be solved in time $O(n \log k)$ where n is the number of nets and k is the number of necessary colors (which can be as large as n in the worst case). On the other hand, if single-sided nets (even with only two points) are allowed, then the layer assignment problem becomes NP-complete (by reduction from circular arc graph coloring, which was described in Section 4.1). There is, however, some hope in the middle ground: if all nets do go across the channel, even if they are multi-point nets, layer assignment is equivalent to coloring a trapezoid graph [19], which can be achieved in time $O(n^2)$.

Layer assignment is only the first step. If all we want is to find the minimum width relative to a given layer assignment, then it suffices to solve — independently — as many river routing problems as there are layers (using the greedy algorithm of Section 5.1) and we are done. Much harder — and still open — is the problem of assigning layers to nets in such a way that the smallest width over all legal layer assignments (with a given number of layers) will be achieved. All the coloring algorithms mentioned above give one particular solution (among many possible ones), but it is not necessarily the one that yields the eventual minimum width (after river routing is applied). A further combinatorial complication may occur if the number of available layers is larger than what is necessary (*i.e.* l is greater than the chromatic number of the permutation or trapezoid graph). It is not known how to

incorporate the routability conditions mentioned above into the color selection mechanism employed by the layer assignment algorithms so as to achieve this result.

One exception to this open problem is again the case in which the routing configuration was planar to start with, namely — the nets appear in the same order on both sides of the channel. Here, if l layers are given (which are $l-1$ more than necessary), and we assign the ith net to layer number[13] $i \bmod l$, then solving the l river routing problems that are thus formed does yield the minimum width channel [5, 59]. We do not know, however, how to generalize this interleaving layer assignment technique for more general situations.

6 Provably Good Algorithms

Even for the cases for which the channel routing problem is NP-complete (indeed, this is the majority of cases, as we saw in Section 4), not all is lost. In the previous section we saw how we can find efficient algorithms which produce optimal solutions for a whole class of problems; in this section we consider some more algorithms in this category and also (similar) algorithms which can be proven to find solutions using a number of tracks within some constant times the optimal. These approximation algorithms should be compared to the algorithms of Section 7, which work well in practice but can, in the worst case, give very bad results.

The results that we review in this section are summarizes in Table 1. Generally the results for 2-point nets are better than those for arbitrary multi-point nets; it should be noted that most authors exclude single-sided nets when discussing 2-point nets. In our review below, whenever we discuss 2-point nets we shall mean 2-point two-sided nets unless we explicitly state otherwise.

The most success in finding solvable special cases and good approximation algorithms (excluding the one-layer case discussed in Section 5) has been for the knock-knee model; there are also results for the Manhattan routing model and for some models that allow wire overlap. Before considering these, we consider a fundamental channel routing problem: channel routing when there are no vertical constraints.

If the vertical constraint graph (as defined in Section 3) has no edges, *i.e.* no two terminals (of different nets) are in the same column, then routing in the Manhattan model amounts to finding the best assignment of nets to tracks without having to pay attention to what happens in the columns. It is easy to show that in this case nets can be realized using one track each and the number of tracks used is equal to the density of the channel. In fact, this solution is optimal for any channel routing model which disallows horizontal overlap of wires, since the number of tracks used is then at least density. The track assignment problem, *i.e.* which track is assigned to every net, can be formulated as an interval graph coloring problem, which can be solved in time $O(n \log n)$, where n is the number of nets [33, 36, 40].

The interval graph coloring problem is obtained from the track assignment problem as

[13] We use the lth layer when the number coming out of the formula is 0.

routing model	number of layers	terminals per net	performance of best known algorithm		best known lower bound	
Manhattan	2	2-point	$d+O(f)$	[4]	$\max(d,f)$	[4]
		arbitrary	$3d/2+O(\sqrt{d\log d})+O(f)$	[30]	$\max(d,f)$	[4]
knock-knee	2	2-point	$2d-1$	[81,84]	$2d-1$	[58]
		arbitrary	$3d+O(\sqrt{d\log d})$	[30,84]	$2d-1$	[58]
	3	2-point	d	[81]	d	
		2 or 3-point	$\left\lfloor \frac{3d}{2} \right\rfloor$	[70]	d	
		arbitrary	$2d-1$	[70]	d	
	4	arbitrary	$3d/2+O(\sqrt{d\log d})$	[30]	d	
unit vertical overlap	2	2-point	$d+O(\sqrt{d})$	[7]	$d+\Omega(\log d)$	[7]
		arbitrary	$3d/2+O(\sqrt{d\log d})$	[30]	$d+\Omega(\log d)$	[7]
overlap every 2^{nd} layer	≥ 5	2-point	$\left\lceil \frac{d}{\lfloor L/2 \rfloor} \right\rceil + 2$	[9]	$\left\lceil \frac{d}{\lfloor L/2 \rfloor} \right\rceil$	
overlap every k^{th} layer, $k \geq 2$	$\geq k+3$	arbitrary	$\left\lceil \frac{d+1}{\lfloor L/k \rfloor} \right\rceil + 2$	[9]	$\left\lceil \frac{d}{\lfloor L/k \rfloor} \right\rceil$	
arbitrary overlap	≥ 2	2-point	$\frac{d}{L-1}+O(\sqrt{d/L}+1)$	[7]	$\frac{d}{L-1}$	[38]
	≥ 3	arbitrary	$\frac{d}{L-2}+O(\sqrt{d/L}+1)$	[7]	$\frac{d}{L-1}$	[38]

Table 1: Performance bounds of provably good algorithms discussed in Section 6. Lower bounds are provided for comparison. Symbol d is the channel density, f is the channel flux, and L is the number of layers available for routing. A reference for each algorithm or unobvious lower bounds is given in brackets.

follows: Each net defines an *interval*, which is its span. The interval graph represents each interval as a node and the overlap of a pair of intervals as an edge between the nodes for those intervals. Coloring the nodes of the graph is equivalent to assigning tracks to the nets, i.e. each track constitutes a color. The chromatic number is the density. Intervals can be assigned colors, or equivalently nets can be assigned to tracks, in a greedy fashion: Scan the channel from (say) left to right. Every time a new net is encountered, it is assigned to an available track, starting with track 1. Tracks are freed when the rightmost terminal of a net is found. The set of all leftmost and rightmost terminals of nets must be sorted for this algorithm to work; this sorting results in the $O(n \log n)$ running time. If the nets are already sorted, the greedy algorithm has $O(n)$ running time. The proof that this algorithm works is quite straightforward: consider the first time the highest numbered track is used. At this point, no lower numbered track is available; there must be a net in each track occupying the starting point of the current net (recall no later-starting net has yet been considered). Therefore, the density must be at least as large as the number of the new track.

6.1 Manhattan Routing

Recall from Section 3 that there are two lower bounds for channel width under the Manhattan routing model: density d and flux f. Baker, Bhatt and Leighton [4] not only define flux, but present an approximation algorithm which will Manhattan route any channel in width $2d + O(f)$. For two-point nets, they can do even better, routing in channel width $d + O(f)$.

To understand the effect of flux, let us first consider a special case of the 2-point channel routing problem for which we know the optimal solution and for which flux is the dominant lower bound: the one-shift. We first introduced the one-shift in Section 1. The n-net right one-shift has top terminals in columns 1 through n and bottom terminals in columns 2 through $n+1$. The flux is $\sqrt{n/2}$ and the density is 2. Brown and Rivest [11] prove a better lower bound of $\sqrt{2n+1} - 1$. A simple routing which uses $2\sqrt{n} - 1$ tracks works as follows: reserve the top $\sqrt{n}/2$ and bottom $\sqrt{n}/2$ tracks for the routing of every $\sqrt{n}^{\text{th}}$ net. These $\sqrt{n}$ nets are routed using two tracks, one in the bottom group and one in the top group, in a U-shape which goes outside the span of the channel and returns — half to the left and half to the right. The remaining nets are in blocks of $\sqrt{n} - 1$ nets. Since each $\sqrt{n}^{\text{th}}$ net has been routed by going outside the channel span, the long staircase has been broken up and each of the $\sqrt{n}$ blocks can be routed in a $\sqrt{n} - 1$ staircase using the same $\sqrt{n} - 1$ tracks. Note that this routing uses columns outside the span of the channel and possibly even beyond the length of the channel. This is typical of the algorithms we will be discussing.

The basic idea of the algorithm by Baker, Bhatt, and Leighton is to break up the routing channel into blocks of size k^2, where k is at most $6(f+1)$. The parameter k is defined so that in each group of k^2 columns, at most $k^2 - 3k$ nets are split by a horizontal cut at the top or the bottom of the channel. (Recall that horizontal cuts are used to define flux.) These blocks are further divided into subblocks of size k.

The routing tracks are also divided into three types of groups, six groups in all:

- **Type 1:** Two groups of $O(k)$ tracks — one group closest to the top of the channel and one group closest to the bottom of the channel — are used to redistribute free columns within the blocks. At the track closest to the center of the channel each subblock of k columns contains at least 3 columns free of wires. These tracks can be viewed as the boundaries of a new channel; we now have a channel routing problem where every subblock of k columns has 3 free columns.

- **Type 2:** One group of $2d + O(k)$ tracks ($d + O(k)$ for 2-point nets) in the center of the channel. These tracks are used for interconnections between the subblocks.

- **Type 3:** Two groups of $O(k)$ tracks, one between the top group of type 1 and the center group and one between the bottom group of type 1 and the center group. These tracks are used for interconnections within one subblock. They connect wires in a type 1 group to the center group.

The routing in the center group of tracks routes the correct number of wires between any two subblocks, but does not worry about the exact location of wires within blocks. It is the most complicated phase of the algorithm and requires a detailed analysis of the problem. The groups of tracks of Type 3 are used to do the cleanup between the routes of the type one groups and the center group; here is where the exact positions of wires is taken care of. Because there are at most $k - 3$ nets in each subblock, routing within each subblock can be done fairly easily in $O(k)$ tracks. This routing relies on an technique of Kawamoto and Kajitani [52]. They observed that for 2-point nets, any channel can be Manhattan routed using at most one column outside the span of the channel. This is done by using one track per net and ordering the tracks according to the order of the vertical constraint graph. To break a cycle in the vertical constraint graph, one net of the cycle uses two tracks: the segments in the two tracks are connected in an empty column (possible the extra column outside the span). Since for 2-point nets all cycles are independent, the same empty column can be used by all cycles if all the tracks for one cycle are consecutive. Since this technique uses at most one extra track per cycle and there are at most $n/2$ cycles, the number of tracks is at most $3/2n$ for n nets.

The running time of the algorithm is analyzed by Baker et. al. to be $O(l(d + f))$, where l is the length of the channel. This is polynomial in the length of the input if the input is presented in column list format. We prefer to use the net list format and not to have running times depend on the number of columns in a channel because the number of columns may be much larger than the number of nets[14]. Upon examination of the phases of the algorithm, we have concluded that there is an implementation which has a running time polynomial in the number of terminals.

The multi-point results of [4] have been improved to $3d/2 + O(\sqrt{d \log d}) + O(f)$ in a paper that addresses the knock-knee and unit vertical overlap models as well as the Manhattan routing model [30]. The technique common to all three models is a decomposition of multi-point nets into *simple nets*; simple nets are 2-point nets and one-sided nets. An arbitrary

[14] Note that the construction of Szymanski produces a dense channel: each column contains a terminal. Therefore, the Manhattan channel routing problem is NP-complete whether the input format is a column list or a net list.

multi-point channel routing problem is transformed into an *extended simple channel routing problem*: a routing problem involving only simple nets but with some added structural properties. Two slightly different transformations are used: one for the Manhattan model which deals with flux as well as density, and one for the other two models. The transformation may increase the density and flux, but in the case of Manhattan routing, the density is increased to no more than $3d/2+O(\sqrt{d\log d})+O(f)$ and the flux in increased to no more than $O(f)$. The extended simple channel routing problems are similar enough to 2-point routing problems that Gao and Kaufmann are able to modify known 2-point routing algorithms to work. In the case of Manhattan routing, they modify the algorithm of Baker et. al. to route an extended simple channel routing problem of density d_e and flux f_e in $d_e + O(f_e)$ tracks. Combined with the transformation of an arbitrary multi-point routing problem to an extended simple routing problem, this gives the desired $3d/2 + O(\sqrt{d\log d}) + O(f)$ track routing for multi-point nets.

6.2 Knock-knee Routing

All the knock-knee routing algorithms we are aware of separate the tasks of finding a wire layout, i.e. a set of edge-disjoint paths for the wires, and finding a layer assignment for the wire segments. In fact, many papers only address the wire layout problem. This approach is justified by the results of Brady and Brown, who show that any knock-knee wire layout has a legal layer assignment using four layers [8]. This result is the best that we can expect: there are wire layouts which cannot be assigned only three layers. (Recall from Section 4 that determining if a wire layout can be assigned three layers is NP-complete.) The result is not limited to routing channels: it holds for knock-knee routing in any closed region of the plane.

Brady's and Brown's proof that four layers suffice is an extension of the theory of layer assignment developed by Preparata and Lipski for three layers [81]. Preparata and Lipski consider the layer assignment of *full layouts*. Full layouts are wire layouts in which every interior grid edge is used. Full layouts can be represented simply by drawing a diagonal line whenever a knock-knee occurs; the diagonal line cuts both wires of the knock-knee (see Figure 7). All other interior grid points contain wire crossings. Any wire layout can be transformed into a full layout by drawing the diagonals at knock-knees and interpreting the resulting diagram as a full layout. (This may give wire paths which are cycles, but this is not important for the layer assignment.) For any $L \geq 2$, a wire layout can be legally assigned L layers if its corresponding full layout can be legally assigned L layers.

The assignment of layers to a full layout is reduced to finding a special two-coloring of the routing area. One requirement of the coloring is that the diagonals of the full layout always be on the boundary of a monochromatic region. The remainder of the boundaries of monochromatic regions must lie on the *partition grid*, which is the unit grid with columns halfway between routing columns and rows halfway between routing rows (so each routing grid point is in the center of a partition grid square — see Figure 7). Preparata and Lipski define a set of forbidden patterns for the boundaries of the monochromatic regions and show that a full layout has a legal 3-layer assignment if and only if such a two-coloring of the routing area exists. Brady and Brown replace the forbidden patterns with a new set of

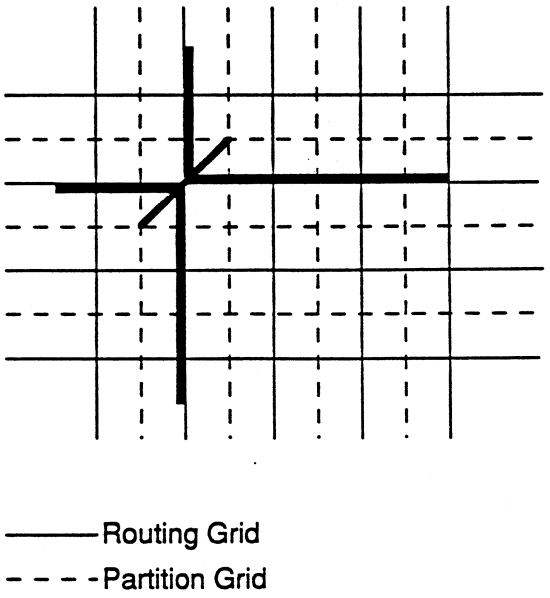

Figure 7: Knock-knee wire layout representation for layer assignment.

forbidden patterns, show that any legal two-coloring under their conditions corresponds to a legal 4-layer assignment of the full layout, and furthermore show that any full wire layout has such a two-coloring of its routing area.

Given the result by Brown and Brady, we can turn to the description of algorithms which produce provably good knock-knee wire layouts. In some cases the wire layouts produced can actually be legally assigned less than four layers. The most notable of these results is that of Preparata and Lipski [81]: they have shown that any 2-point knock-knee channel routing problem can be routed in three layers optimally. The algorithm runs in linear time. First, the wire layout is produced track by track. When the wire layout is extended by one track, the density of the remaining subproblem is reduced by one; therefore the number of tracks used is density, the minimum number possible in the knock-knee model. After the wire layout is produced, Preparata and Lipski show that the particular wire layout produced can be assigned 3 layers; they use the theory of layer assignment discussed above.

Note that some wire layouts produced by the algorithm of Preparata and Lipski may require only 2 layers, but their layer assignment method is not guaranteed to detect this. Rivest, Baratz, and Miller [84] show[15] that any t-track wire layout can be routed in two layers in $2t-1$ tracks: every second track is used for layer changes. This can require a large number of vias. Leighton [58] has constructed a 2-point channel routing problem with density d which requires $2d-1$ tracks in the two layer knock-knee model; thus the result of [84] is the best possible worst-case result.

[15]The first algorithm to knock-knee route 2-point nets in density was presented in this paper but contained a minor error. The paper did not consider three-layer routing.

For multi-point nets, we no longer have algorithms which can achieve density, even for four layers. However we do have algorithms which can produce wire layouts within a constant factor of optimal: using at most $\lfloor 3d/2 \rfloor$ tracks when all nets have two or three terminals (originally presented in [80]) and using at most $2d - 1$ tracks for arbitrary multi-point nests (originally presented in [90]). In [90], the authors use the layer assignment theory of Preparata and Lipski to show that their wire layout for arbitrary nets can be assigned 3 layers.

In [70], Mehlhorn joins Preparata and Sarrafzadeh to present a unified framework for algorithms producing wire layouts with provably good performance. Within this framework, they present algorithms which reproduce the bound of $\lfloor 3d/2 \rfloor$ tracks for 2 and 3-point nets, the bound of $2d - 1$ tracks for multi-point nets, and reproduce the optimal result of d tracks for 2-point nets. These algorithms proceed column by column in a single pass across the channel; the action in each column depends on the current routing situation. Empty columns and columns containing trivial nets can be ignored; thus the algorithms run in time linear in the number of terminals.

The number of tracks required by a routing algorithm for multi-point nets is reduce by Gao and Kaufmann [30] to $3d/2 + O(\sqrt{d \log d})$. However this reduction is at the cost of a routing layer, since their wire layouts will in general require four routing layers. The technique is as described above for Manhattan routing. However, to route the extended simple channel routing problem they must use the "generalized switchbox routing" of Kaufmann and Mehlhorn [51] rather that the simple 2-point channel routing of [81] or [70]. The generalized switchbox wire layout may require four layers.

6.3 Overlap Models

The major argument for prohibiting overlap of wires is to avoid crosstalk between the wires. When more than two layers are available for routing, this argument is less compelling. Nonadjacent layers in multi-layer technologies are separated by a significant amount of insulating material. Thus it is of practical (as well as theoretical) interest to ask how much better one can route if overlap is allowed. In the discussion below, L will always denote the number of layers available for routing.

Given 3 conducting layers, a very simple extension of the Manhattan routing model immediately gives us a method of routing any multi-point channel in density: Use the top and bottom layers for vertical wires and the middle layer for horizontal wires. Vertical wires connecting to terminals at the top of the channel run on the top layer and vertical wires connecting to the bottom of the channel run on the bottom layer. These layers are allowed to overlap; thus vertical constraints are not an issue and the horizontal wires can be assigned to d tracks using the interval graph coloring algorithm. Note that only vertical wires overlap and no adjacent layers contain overlapping wires. This idea is generalized by Brady and Brown [9] to produce routes using $d/(\lceil L/2 \rceil - 1)$ tracks for $L \geq 3$ layers with no overlap of adjacent layers. Through a more sophisticated use of the idea, they show that if

every k^{th} layer is allowed to overlap, the a multi-point channel can be routed in

$$\left\lceil \frac{d+1}{\lceil L/k \rceil} \right\rceil + 2$$

tracks, where $k \geq 2$ and $L \geq k+3$. For $L = 4$ and $k = 2$, the bound becomes $\lceil (d+1)/2 \rceil + 4$. By using the ideas of Brady and Brown but allowing overlap of adjacent layers (specifically using the top and bottom layers for horizontal wires and the two middle layers for vertical wires), Cong et. al. [18] have achieved an upper bound of $\lceil d/2 \rceil + 2$ tracks for four layers.

Another restriction of overlap meant to limit crosstalk is to restrict each overlap of two wires to a distance of one unit. The routes for two nets are allowed to overlap more than once. By adding the more artificial restriction that overlap only occurs in vertical wires, we obtain the *unit vertical overlap model.* Limiting overlap to the vertical direction maintains density as a lower bound on the channel width. The unit vertical overlap model is viewed as a relaxation of the knock-knee model and has been studied for two-layer routing. The first results appear in [29] : algorithms that routed 2-point nets in $d + O(d^{2/3})$ tracks and multi-point nets in $2d + O(d^{2/3})$ tracks are presented. These results are improved by Berger et. al. [7] to $d + O(\sqrt{d})$ tracks for 2-point nets [16] and $2d + O(\sqrt{d})$ for multi-point nets using algorithms modeled after that of Baker et. al. [4]. The multi-point net result is further improved by Gao and Kaufmann [30] to $3d/2 + O(\sqrt{d \log d})$ tracks, where the extended simple channel routing problem is solved using a variant of the 2-point net algorithm of Berger et. al. [7]. Berger et. al. also prove a lower bound for 2-point nets of $d + \Omega(\log d)$; this lower bound holds even if arbitrary vertical overlap is permitted.

Results also exist for the general routing model which allows unrestricted overlap of wires. Recall from Section 3 that Hambrusch [38] improved the obvious d/L lower bound on the number of tracks to $d/(L-1)$, generalizing the result of Leighton [58] for $L = 2$. Berger et. al. [7] present algorithms that route 2-point nets in $d/(L-1) + O(\sqrt{d/L} + 1)$ tracks ($L \geq 2$) and multi-point nets in $d/(L-2) + O(\sqrt{d/L} + 1)$ tracks ($L \geq 3$). Other interesting algorithms are one which routes multi-point nets in $\lceil (d+2)/\lfloor 2L/3 \rfloor \rceil + 5$ tracks for $L \geq 7$ [9] and one which routes 2-point nets using k-fold overlap in $\lceil d/k \rceil + 1$ tracks when $k \leq \lceil L/2 \rceil - 2$ and $L \geq 3$ [39].

7 Practically Good Algorithms

In this section we review various algorithmic methods that are used to route channels in practice. None of them is provably good, but they all perform very well on real data. Indeed, some have demonstrable poor behavior (both in terms of time complexity and the quality of the results) in the worst case, but these extremes are rarely encountered.

By and large, there are four major paradigms for solving the channel routing problem using two layers: one net at a time, track oriented, column by column, and hierarchical (divide and conquer). Each approach has its pros and cons, as we shall see. Instead of describing

[16] Berger et. al. show how to convert this routing to a routing which contains no overlapping wires but uses 45° wires.

the methods in abstract, generic terms, we have chosen one or two representatives of each approach (and one that combines two of them), and along with some general remarks we present specific algorithms.

In addition to the above methods, which mostly employ the two-layer Manhattan wiring model, we also describe extensions thereof that apply to multi-layer routing. We conclude this section by discussing techniques for post-routing improvements and the usage of jogs.

7.1 Left-edge, with and without doglegs

A natural approach to solving the channel routing problem using the Manhattan model is to attempt to extend the interval graph coloring method that was proven in Section 6 to be optimal for the case in which there are no vertical constraints. When the problem instance contains such constraints, however (and in practice there are usually as many constraints as there are columns, channels being highly populated), respecting the vertical constraints while trying not to widen the channel too much at the same time becomes a challenge. A brute-force method for handling this situation is to exhaustively search through the whole solution space by means of a branch and bound method, as proposed by Kernighan, Schweikert, and Persky [53].

The method still adheres to using only one track for each net, *i.e.* the entire net is routed with only one horizontal wire. As the channel is scanned from left to right, each net is assigned to a track when its leftmost terminal is revealed. The expansion of the search space is bounded by using the vertical constraints graph: before we decide to place a net in a given track, we check whether the number of tracks that we know must be available above it and below it to accommodate other nets will widen the channel. This can be done by computing the longest paths leading from and to the corresponding node in the vertical constraints graph[17], and adding these length to the proposed track assignment. Only assignments that do not require widening are expanded; we backtrack only when a dead end is reached, and then a new channel width is attempted.

The initial channel width is set to the maximum between the channel's density and the length of the longest path in the vertical constraints graph. Exploring the entire search space may take exponential time in the worst case, therefore the track assignment of the first few nets is critical if this method is to be used in a practical setting. Thus, some heuristics for how to select these initial positions are usually applied.

The more serious problem with the above scheme, however, is that the quality of the resulting layout is impaired by the restriction of one track per net. In 1976, David Deutsch [20] was the first to suggest that tracks should be changed along the wire in a systematic manner. His proposal was to limit such track changes, which he calls *doglegs*, to terminals' positions only (thereby also adding a relatively small number of contacts). This way, the size of the search space will still be manageable, but smaller widths would become attainable.

Deutsch's method entails splitting each net to subnets at their own terminals' positions.

[17] These lengths can be computed once as a preprocessing step, and then used during the running of the algorithm.

Thus, a k-point nets becomes $k-1$ adjacent two-point nets. The vertical constraints graph is drawn as before for the entire set of subnets as its vertices, but no edges are introduced due to the relationship between subnets that belong to the same original net. This allows track changes, or doglegs, at a net's own terminals' positions with a controlled amount of extra computation.

In this influential paper [20], Deutsch also introduced the "difficult example", which has since become a common channel routing benchmark. Deutsch has observed that in practice only a few of the seventy nets comprising this channel are indeed difficult (such as clock lines), and it would be enough to allow doglegs only along them, but it is hard to choose these nets algorithmically. Also, the original vertical constraints graph for this example has a longest path of length 28, and the channel's density is 19; the branch-and-bound algorithm of [53] was able to find a 28 tracks wide routing, and Deutsch's version of this algorithm (allowing doglegs) attains a width of 21 tracks.

Yet another variation on the left-edge algorithm was proposed almost a decade later in YACR2 [82]. In their attempt to overcome vertical constraints, the authors suggest to maze route around the conflicts. By deviating from the strict one layer per direction Manhattan wiring model, *jogs* in the layer originally reserved for vertical wires are used to connect to horizontal segments that end one column before (or sometimes after) the span of the net (or the subnet). Another contribution of this work is the scanning in both directions, *i.e.* from left to right and from right to left, based on density considerations.

7.2 Global track assignment

Both left-edge algorithms described in the previous section, as their name implies, scan the channel from left to right and make track assignments as they discover the nets[18]. In 1982, Yoshimura and Kuh [103] suggested to process the nets in some other order using a criterion that is induced by the structure of the vertical constraints graph; this method can be applied either to entire nets or to subnets, thereby producing doglegs. The result is an algorithm that is driven by vertical constraints rather than the horizontal ones (*i.e.* density considerations), which are now being obeyed as we go.

The key idea in [103] is to assign nets to tracks in such a way that the longest path in the vertical constraints graph will not be affected (if at all possible). The basic operation is to assign nets to tracks; whenever a decision is made for a given net to share a track with previously assigned nets (if it does not have a horizontal conflict with them, of course), the corresponding nodes in the vertical constraints graph are *merged*; also, the horizontal constraints are modified so that all the nets that are within the span of the new "super net" are made to interfere with it.

This operation is repeated until all nets are assigned or the channel needs to be widened by adding a track. The order in which nets are picked is crucial; a naive ordering is not good enough, and the authors suggest to use a repeated bipartite matching procedure so as to maximize mergers that do not affect the lengths of paths in the vertical constraints graph.

[18] Even the YACR2 algorithm [82] that scans in both directions follows this paradigm.

Using this added mechanism, Deutsch's difficult example was routed with 20 tracks, 1 more than density.

An important issue that is not addressed by any of the track assignment based algorithms is that of vertical conflict cycle resolution, *i.e.* if the vertical constraints graph contains a cycle then these algorithms cannot successfully route the channel. One way to remedy this is by adding a preprocessing stage which routes only those nets involved in the cycles and then hand over the results to one of the above algorithms. A systematic method for handling cycles while trying to keep the channel's width to a minimum is discussed in [76]. A more effective solution for such cases, however, is to use an approach that is not based on track assignment, as described in what follows.

7.3 Column by column

All the above approaches are based on assigning a net (or a subnet) to a track as an atomic operation. In 1982, Rivest and Fiduccia [85] turned things around, proposing to fill the channel with routing segments by handling one column at a time. Wires thereby grow simultaneously as the scan progresses from left to right, with fragments added to each as we go.

As the channel is being scanned, the terminals are being encountered and they are "brought into" the channel by starting wires from them. These pieces of circuitry are then connected within the current column to existing wires that already realize the same nets when possible, otherwise they are continued to the next column. Other nets improve their positioning with respect to upcoming connections — again, space permitting — by moving to tracks that are closer to the side on which the next terminal will appear.

The different nets compete for the wiring capacity of each column using various resolution methods. As a result, some nets may initially comprise a number of separate wires until they are all merged. This merger may occur only after we leave the span of the net, and the wire is completed in subsequent columns. Moreover, the scan of the whole channel is not completed until all nets are fully connected by wires, so the routing may extend beyond the span — or even the length — of the channel, thus making the extent the dominating measure.

The algorithm proposed in the paper, dubbed by its authors the "greedy" algorithm, is in fact a generic framework with a number of parameters that can be set to tune up the algorithm for different channel populations. These parameters control mostly decisions that have to do with how tracks are being assigned to wiring segments in anticipation of upcoming terminal connections, but also other matters. Three important parameters are:

- how far the algorithm looks ahead when classifying nets as "falling" or "rising", *i.e.* whether their next terminal is on the bottom or top side and therefore they should try and move to a track that is closer to the respective side
- a tolerance on the distance between two consecutive terminals: if this tolerance is not exceeded and if the two terminals are on opposite sides of the channel, then the net

is classified "steady"

- minimum jog length, which is the minimum length of vertical wiring allowed for any given net in a single column for purposes of track changing (*i.e.* jogging)

In addition to these numeric parameters which help to customize the algorithm, there are also a few sub-problems that are well defined as combinatorial problems that need to be solved, but the actual algorithm used to do so can be replaced following a variety of computational and other considerations. For example, the method and criteria for selecting nets that are to be merged in a given column can vary from a greedy procedure using a crude metric (*e.g.* the number of remaining nets which occupy more than one track) to an all out exhaustive search that also takes into account sophisticated measures of subsequent routing.

The results reported in the original paper (and then by many follow-up implementations) are quite good. Most channels are routed close to density; Deutsch's difficult example requires 20 tracks, one more than density[19]. The running times are quite reasonable, due to the low computational complexity of the underlying algorithms. These running times are a significant improvement over those of earlier algorithms which yield comparable quality. The favorable running times, in addition to the flexibility and relative simplicity of the algorithm, have been prime contributors to its popularity. It is fair to say that many "industrial strength" routers that have been incorporated into layout systems (such as cell generators, silicon compilers, and other composition systems) since its inception are derivatives of the "greedy" router. The virtue of generating an unbounded number of jogs per net, as it allows, has been substantiated theoretically by a result in [76] showing that indeed $\Omega(\sqrt{n})$ jogs are sometimes necessary in order to obtain minimum width routing.

The greedy algorithm is not free of weaknesses, however: since jogging is permitted to occur quite freely (as opposed to doglegging, which is limited to terminal positions), there is a tendency to generate a large number of contacts; this can be remedied with post-processing (see Section 7.7). A more intrinsic deficiency is that the scan direction creates the asymmetry of adding columns on one end, whereas the other end is typically less populated. One possible solution is to try to start the scanning process in the middle and go to both sides, but there are no reports of successful attempts to implement this idea (which would require overcoming several technical difficulties).

7.4 The hierarchical router

Being a complex combinatorial problem as it is, it is rather surprising that no one had attempted to solve the channel routing problem using a divide and conquer approach until Burstein and Pelavin published their algorithm in 1983 [14]. The easy part was to devise a partitioning of the channel (for the "divide" part); the natural way to do so is to bisect it by a line parallel to the sides. The hard part was to define the respective routing subproblems,

[19] The data used by Rivest and Fiduccia was incomplete, but subsequent runs — of several variations of the algorithm — confirmed this result.

to provide a meaningful interpretation to the inexact intermediate routing that is generated, and finally to figure out how to paste these partial solutions together.

The key idea was to look at every level of the hierarchy as a set of channels with 2 "super tracks", *i.e.* at the top level there is one such channel, at the next level there are two, and so on until the super tracks become simple tracks (of width 1). Once an intermediate problem is solved, each super track needs to be solved separately. This is possible since now the positions at which wires cross the dividing line between the two super tracks can be used as terminal positions for the subproblems.

In order to handle these coarse grain channels that are set up during the top-down refinement (and their solutions), Burstein and Pelavin have defined a generalized wiring model that allows multiple wires to use the same grid line. This extension is in fact necessary only for tracks (which are parallel to the direction of the bisection), since vertical routing remains one wire per column. Each horizontal grid line is assigned a *capacity* which limits the number of wires that can use it; this bound is uniformly set to the current width of the subproblem, as defined by the recursive partitioning procedure.

At each level of the hierarchy the algorithm routes the nets one at a time without exceeding the limits (as established for this level). The net ordering is unspecified, and can be made dependent on a number of factors; the bounds are updated after each net is routed. The routing of a single net is performed by a slight modification to the dynamic programming algorithm of Aho, Garey, and Hwang [1] which generates rectilinear Steiner trees on a grid in time polynomial in the number of grid points. This number is twice the length of the channel for each subproblem; since there are at most $O(w)$ subproblems to be solved, the whole procedure runs in polynomial time.

The results reported in [14, 15] are impressive. They were the first to achieve a width of 19 (which equals density) for Deutsch's difficult example without any manual intervention. These results were obtained after extensive experimentation, which involved tuning of the capacities of super tracks and how they are handled by the modifications to the Steiner tree generation algorithm.

7.5 Net classification

A somewhat unique (and so far unparalleled) approach to channel routing was proposed in 1982 by Marek-Sadowska and Kuh [67]. They suggested to classify two-point nets into five different categories according to their shape: trivial, single-sided with both terminals on the top, ditto for the bottom, rising nets, and falling nets. The key observation was that if each category is assigned layers in a particular manner, then a fixed external ordering between the classes can be established regarding their track assignment, and furthermore an internal ordering in each class can be determined using tractable techniques.

The classes and their layer assignment can be seen in Figure 8. The resulting routing is rectilinear but does not follow the Manhattan rule of one layer per direction. The only wiring model it fits (per Section 2) is the general one, even though no overlap is created.

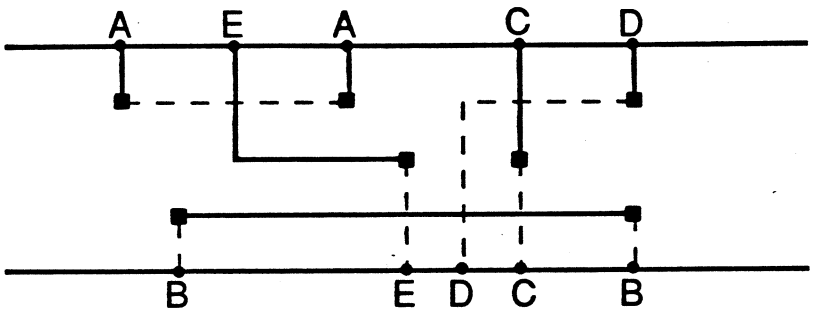

Figure 8: The five net classes.

Multi-point nets can be handled gracefully by simply pasting the constituent 2-point nets together without violating the layer assignment rules. The experimental results (in terms of achieved channel widths) were not particularly good, and this approach was all but abandoned.

7.6 Multi-layer routers

Until two layers of metal became generally available in CMOS technologies, multi-layer routers were an academic exercise when applied to integrated circuits. Now, more and more practitioners have been considering the problem of three-layer routing, using two layers of metal and one layer of polysilicon. Most do not pay much attention to the very different electrical properties and design rules of polysilicon compared to metal. Some research has gone beyond three layers to consider arbitrary multi-layer routing (as we have seen, for example, in Sections 5.2 and 6.2), anticipating the increase in layers in integrated circuit technologies and also to make their work applicable to multi-layer printed circuit boards.

Most of the three-layer routing algorithms which are intended to be practical extend the single direction per layer paradigm of Manhattan routing. We saw in Section 6 that if we assign the top and bottom layers to the vertical direction and the middle layer to the horizontal direction (the VHV model), then we can route any channel in density while creating vertical overlaps. Practitioners, however, prefer to allow overlap in the horizontal direction in hopes of halving the number of tracks needed. Thus, most of the algorithms reported use the top and bottom layers for horizontal wires and the middle layer for vertical wires (the HVH model). With this layer assignment, vertical constraints are again relevant.

Pitchumani and Zhang note [79] that in fact, some mixture of VHV and HVH routing may be best: using VHV routing when the density is low but the vertical constraints chains are long. They partition the channel into two sides: a VHV side and an HVH side; a transition track is used between the sides where vertical wires can change from the top or bottom layer (VHV) to the middle layer (HVH). They present a heuristic algorithm for partitioning a set of nets into a VHV set and an HVH set under dogleg-free routing. The algorithm finds cuts in the vertical constraints graph.

Generally, three-layer algorithms in the HVH model mimic some algorithms in the (two-layer) Manhattan model. Chen and Liu [17] extend the algorithm of Yoshimura and Kuh [103] (discussed in Section 7.2). They assign horizontally conflicting nets to the same track but different layers when there is no vertical conflict between them. The nodes for these nets are then merged. They do not allow doglegs. Bruell and Sun [12] modify the rules of the column by column algorithm of Rivest and Fiduccia [85] (discussed in Section 7.3) to apply it to HVH routing. Heyns [43] uses a left-edge algorithm with doglegs (discussed in Section 1), but departs from the strict direction per layer model. The algorithm allows both horizontal and vertical wires on the top layer, horizontal wires on the middle layer, and vertical wires on the bottom layer. Furthermore, terminals are allowed on both the top and bottom layers at the same position.

Rather than extending a two-layer routing algorithm, Cong, Wong, and Liu [18] start with an actual two-layer routing. They present an algorithm to transform a Manhattan routing (with jogs) into an HVH routing. The algorithm assigns every pair of tracks in the Manhattan route to one two-layer track in the HVH route. This may introduce overlap in vertical wires: two vertical wires that started in separate tracks may now start in the same track. Therefore, the algorithm first modifies the Manhattan routing if necessary so that there will be no such vertical conflicts. The algorithm may insert empty tracks so that it can find a legal solution. Cong, Wong and Liu present the results of applying their algorithm to seven problems from the literature; their algorithm achieves the lower bound of $\lceil d/2 \rceil$ on all the problems. They also present an extension of their algorithm to map a Manhattan route onto four layers using an HVVH model. If the Manhattan routing uses only doglegs (*i.e.* a net only jogs at columns containing its terminals), then each column contains at most two vertical wire segments and a route using t tracks is transformed to an HVVH route using $\lceil t/2 \rceil$ tracks — the best possible.

When considering general multi-layer routing, researchers have also continued the tradition of a single direction per layer. Enbody and Du [26] extend the HVH model to an HVHV...H model (odd number of layers) or HVHV...V model (even number of layers), so that there are $\lceil l/2 \rceil$ horizontal layers. They present a column-by-column algorithm modeled after Rivest and Fiduccia [85] in which they use a five column window and allow backtracking within the window. (They also extend the VHV model to an VHVH...V model which again eliminates vertical constraints and achieves $\lceil d/\lceil l/2 \rceil \rceil$ tracks given l layers and density d.) Braun et. al. [10] take a different approach. They divide the layers into sets of two layers and three layers and distribute the nets among these sets. In this way, they reduce the multi-layer routing problem to several two-layer or three-layer routing problems. The two-layer problems are routed using the YACR2 [82] algorithm (Section 7.1), which uses a modified Manhattan routing model. The three-layer problems are routed using an extension of YACR2 for three layers, which uses a modified HVH model. To maximize the number of layers available for horizontal routing, the partition maximizes the number of three-layer sets, using at most two two-layer sets.

Greenberg et. al. [35] build on the ideas in [10]: they add single-layer routing as a possibility. Thus they partition the routing problem into single-layer, two-layer, and three-layer problems. Since each set of nets assigned to a single layer must be planar (Section 5), it is unclear what is the best partition of layers for a given routing problem. Several partitions

of the layers are attempted — increasing the number of single-layer problems in each partition from zero to all layers — and the area of the routing is estimated for each partition. In the data presented in [35], channel width and net length are often improved and never worsened, but the most impressive improvement is in the number of vias, which is almost always over 15% and in one case is 56%.

7.7 Post-processing

All the wiring models used for solving the channel routing problem that were presented so far are based on a grid abstraction of the actual layout. The design rules, however, are more elaborate. For example, in most technologies contacts take up more area than just the intersection between the wires that are connected; also, metal wires are often wider than polysilicon ones. So, even after the router produces very good results in terms of the grid, improvements can be expected if these results are further processed by layout improvement techniques.

One such common technique is that of *compaction*. In general, layout is compacted by solving a system of linear constraints that reflect the geometric relations between neighboring features. Jog introduction, which leads to sharp increases in complexity, is usually not considered. In 1985, Deutsch [21] observed that channels that were routed using the Manhattan wiring model can be compacted in a special manner, rather than be dealt with by a general purpose algorithm, as follows: The metal wires residing in tracks can be pushed downwards so as to hug the contour of the wires below them. This way, they rise around contacts but can be closer to the bottom side elsewhere. This techniques is reminiscent of the one-layer work mentioned in Section 5 (and follow-on work pertaining to compaction, such as [65]), and the contours around contacts indeed resemble the jogging structures generated by river routing.

If contacts are not concentrated in several areas, but are rather evenly distributed along the channel, the channel can be narrowed considerably. Indeed, the results reported in [21] show a 20-30% reduction in channel widths for a number of examples. Similar results could be obtained by using more general compaction algorithms but at a considerably higher cost.

Another attempt to incorporate actual dimensions into the routing process was made by Rothermal and Mlynski in 1983 [87]. Here vertical constraints were ignored (or, rather, it was assumed they do not occur), but the horizontal wire segments had different widths. The problem that arises is similar to that of interval graph coloring but now the intervals have a height so the coloring rules are more complicated. If many intervals can start at the same coordinate, the problem is in general NP-complete (reduction from bin-packing); otherwise it can be solved greedily by a simple extension to interval graph coloring. Rothermal and Mlynski do not make this distinction and just provide a heuristic which they claim performs well in practice.

A different level at which layout improvement can be carried out is that of symbolic layout, *i.e.* no geometry is involved and the changes are made at the same level of abstraction as the one in which the router operates. A recent contribution in this direction is by Ben-Yehuda and Pinter [6], in which local replacement techniques — based on string matching

algorithms and inspired by compiler optimization methods — are repeatedly applied to the layout. The authors demonstrate their techniques on channel routes resulting from using Rivest's and Fiduccia's "greedy" algorithm (Section 7.3), reporting significant reductions in the number of contacts[20] and sizable increases in the amount of metal being used. This substantiates the claim that channel routing — trying to minimize width — is a hard enough task in itself, and that proper attention to other measures of wiring quality should be deferred to other phases of the layout process.

7.8 Summary

In this section we have reviewed some of the major contributions to the art of channel routing. Both the seminal contributions of the mid 1970's as well as the breakthroughs of the early 1980's were followed by many variations and modifications to the key ideas as they appear in the algorithms surveyed here. But to date, by and large, the workhorses of automatic layout systems — whether commercially available, kept in-house, or used in academia — belong to one of the classes described in this section.

8 Changing the Shape of the Channel

So far we have adhered strictly to the traditional definition of the channel routing problem as defined in Section 2, *i.e.* the sides are straight line segments that can only be moved away from each other. In this section we briefly mention two variations to this geometric constraint. First, we deviate from the one degree of freedom (width) formulation by both adding more freedom (allowing lateral shifting of terminals) as well as taking it away (freezing the sides to form a fixed region). Second, we consider changes to the shapes of the sides from straight lines to polygonal segments. Channel routing techniques and results that were discussed in this survey so far sometimes do pertain (at least in part) to these new situations, but in others they do not quite fit as well, as we shall see next.

8.1 Lateral shifting of terminals

In the *lateral shifting* problem we assume that terminals along the top and bottom of the channel can slide left and right, *i.e.* laterally. We would like to find the optimal position of the terminals, where our optimality criterion may change and will be discussed for each problem. [21] When there was no lateral flexibility, details of the makeup of the channel's sides was unimportant. However, now we must know exactly how terminals are allowed to move laterally. We view terminals as lying on *components*, whose edges make up the channel's sides. The positions of terminals on their components are fixed, and it is the components which can move left and right to lengthen or shorten the channel's length. Thus, all the terminals on one component move in a block. If there is only one component

[20] This end by itself can be attained by running the algorithm from [78] on the layout, but this could be — albeit polynomial — quite costly.

[21] Note that we do not consider more extensive placement modifications such as interchanging terminals.

on a side, then all the terminals on that side move together laterally, their relative positions being fixed; if each terminal is on a separate component, then each terminal may change its distance to other terminals on the side. In between these extremes, we may have several components on each side defining blocks of terminals which move together.

In most cases, we would like to use lateral shifting to minimize the area of the channel after routing. More generally, we would like to know the tradeoffs between the channel's length and width – we may not wish to use the smallest area but may want the best area for a given aspect ratio. How successful we can be at obtaining this information depends on the routing model. Below we discuss when the lateral shifting problem can be solved exactly.

When considering one-layer routing, we are as successful at solving the lateral shifting problem as we are in solving the channel routing problem. Given an arbitrary number of components at each of the top and bottom of the channel, the lateral positions of these components minimizing the channel length for a given channel width can be found in linear time [59]. The algorithm uses the theory discussed in Section 5.1. Since the width is no larger than n, a binary search will find the minimum width in $O(n \log n)$ time. If there is only one component on each side (this is called the *offset problem* [23]), the lateral shifting minimizing the channel width can be found in linear time [71]. As discussed in Section 5.1, other wiring geometries than rectilinear are possible.

For general two-layer channel routing, we do not have a nice theory of channel routing on which to build. Rather than using the actual channel width, which we do not know how to compute, researchers have used channel density as an estimate of channel width when considering lateral shifting problems. Note that when considering density, for each net only the leftmost and rightmost terminal on each side of the channel are relevant. LaPaugh and Pinter [55], and independently Atallah and Hambrusch [3], have presented $O(n^2 \log n)$ algorithms for computing the optimal offset of two components to minimize the density of a channel, where n is the number of nets. When one or both sides of the channel contains more than one component, Johnson, LaPaugh, and Pinter [47] have an $O(n^3)$ time algorithm to compute lateral positions of components minimizing the channel density. When each terminal can slide individually, Gopal, Coppersmith, and Wong [34] have an algorithm which computes lateral positions of terminals to minimize density in time (n^2).

For Manhattan wiring we can also consider trying to reduce the smooth flux (as defined in Section 3.3) of a routing problem. We do not know how to compute the optimal lateral shifting of components to minimize the smooth flux. However, if one is allowed to add columns anywhere in the channel (so terminals can move independently, and both top and bottom terminals are pushed apart when a column is added), the minimum number of columns needed to reduce the smooth flux of a channel to a given value can be computed in time $O(n^4)$, where n is the number of terminals [42].

For the knock-knee routing model, only partial results are available. If representative paths are given for the nets, the problem of minimizing the channel's length for a given width is a special case of the *one-dimensional homotopic compaction* problem [31, 50]. If, furthermore, the routing instance is *locally even*[22], then the compaction problem can be solved in time

[22] Locally even is a technical term, but, roughly speaking, it requires that there be a terminal at each grid

$O(n^2 \log n)$, where n is the number of terminals [50].

8.2 Switchboxes

A switchbox is a rectangular routing region which contains terminals on all four sides. Thus, it is a channel which has left and right edges which may contain terminals in addition to the top and bottom edges. Switchboxes are usually viewed as being of fixed area, but sometimes they are allowed to stretch horizontally and vertically. Clearly the loss of flexibility makes this an even harder routing problem than channel routing, but the problems are similar enough to share techniques.

For switchbox routing we do still have some models for which the problem can be solved exactly. For single-layer routing, Pinter [77] has developed an algorithm to determine the routability of a switchbox in linear time, and an algorithm to actually generate the routes in $O(n^2)$ time. For the knock-knee model, the routing of a set of two-point nets can be found, if it exists, in time linear in the circumference of the switchbox and $O(n \log n)$ with respect to the number of nets using $O(n)$ knock-knees [69]. This algorithm is based on the work of of Frank [28], addressing the design of an efficient algorithm. The foundation for finding edge-disjoint paths in a switchbox is a result in multicommodity flow [72]. The sufficiency of four layers still holds [8].

Turning to algorithms designed to be practical for two layers, many have appeared in the literature. We only review some representative results, emphasizing how techniques for channel routing were applied to switchboxes.

One class of switchbox routers extends routing algorithms designed for channel routing. Burstein and Pelavin [15] apply their hierarchical router (described in Section 7.4) to switchbox problems. They also provide an example (for which their router fails to route one net) which has been used by many others as the "difficult example" of switchbox routing[23]. Several people have extended the column oriented algorithm of Rivest and Fiduccia [85] (discussed in Section 7.3). Luk [64] allows rows and columns to be added to the switchbox in his extension of the Rivest and Fiduccia algorithm. He focuses on the scan direction and presents a heuristic for choosing it. Hamachi and Ousterhout [37] extend the Rivest and Fiduccia algorithm without allowing extra rows or columns, so the algorithm can fail. Their algorithm can avoid obstacles such as prewired (*e.g.* manually wired) nets. By prewiring one net in the difficult example, they were able to route it in the 15 rows and 23 columns provided. In comparison, Luk's algorithm routes the example without manual intervention, but adds a row. A switchbox router for three-layer routing and based on a column scan is presented in [49]. This router routes the difficult example in two layers in 15 rows and 22 columns. (It does not use one of the columns provided in the original specification of the example.) Marek-Sadowska [66] routes nets based on a set of rules; this technique is reminiscent of the router in [67] but is not an extension of it.

Another class of switchbox routers approaches the problem using methods most often ap-

point on each side of the channel.

[23] In fact, this example was initially conjectured to be unroutable, until Deutch [22] succeeded to find a manual solution.

plied to general (not channel) routing problems. In particular, several routers try to route one net at a time using a general path in the grid, rather than using a column-based or row-based approach. Unlike the routers described in the paragraph above, these routers often do not use the Manhattan wiring model. Among these, Supowit [95], who does use the Manhattan model, presents a heuristic for choosing the order in which to route nets. Shin and Sangiovanni-Vincentelli [91] present an algorithm which makes one connection at a time and can either move or remove existing wires to make room for new connections. Lin, Hsu, and Tsai [61] also present an algorithm which modifies existing routes as it progresses. This algorithm, however, has the interesting property that intermediate routings are allowed to have design-rule violations. Joobbani [48] has developed an expert system to route switchboxes. The system uses a large number of criteria in choosing which net segment is to be routed next and how — *e.g.* wire length, congestion, vertical constraints, special patterns and "common sense".

8.3 Jagged edges

So far, we have kept the sides of the channel smooth, *i.e.* they were parallel (finite or infinite) straight line segments resulting from module boundaries, aligned rows of polycells, or artificial partitions of the routing area. All of these could become rectilinear polygonal paths if we change the geometry of the constituent circuits or the composition methodology. This variation is applicable both to channels with movable sides as well as to fixed-shape routing regions, thus a switchbox would now turn into a simple rectilinear polygon.

We first discuss channels with ragged sides. The sides are both assumed to be monotonic in the horizontal direction. Thus, a vertical line crosses each side exactly once. In other words, there are no "coves" along the sides. The terminals, in general, can lie anywhere along each side.

The above requirement is equivalent to postulating the following: if the channel's sides are far enough from each other (in the vertical direction), then there exists a straight horizontal "line of sight" between a point on the right end of the channel and one on its left end; now we require that every point on the channel's sides can be connected by a straight vertical line lying entirely inside the channel to such a line of sight. If terminals were to lie only on horizontal segments, then this suggests a fairly straightforward generalization of most existing routing methods. Note, however, that the existence of the horizontal line of sight is in itself not a requirement for finding a valid routing — the smallest attainable width could be such that the channel's boundaries block such a straight line while all the routing still fits inside the ragged channel.

Not much work has been reported on this problem, since most design systems adhere to rectangular building blocks and smooth composition rules. An early attempt to dealing with the problem is reported in [44], where the "depressions", *i.e.* those parts of the channel that are above or below the lowest or highest y-coordinate of the upper or lower side (respectively), are filled in as much as possible by using a general purpose two-dimensional router (*e.g.* from [57]). Only then, the rest of the channel is routed as usual; no details are given on the interface between the two stages.

This strategy has a major flaw: all solutions that it produces require the existence of a line of sight between the channel's ends. As we have pointed out previously, this may not be necessary to obtain minimal width. Therefore, Hsu [45] developed an adaptation to conventional two-dimensional routing which tries to maximally utilize track usage. Tracks are added only when necessary, and reducing the number of turns in a wire (which each gets instantiated as a contact in the Manhattan model) is also used to make this algorithm behave more like a channel router. The results that are reported are quite good: the router makes reasonable usage of depressions and also routes the rest of the channel quite well[24].

Lauther [56] reports of his success in modifying a mainstay channel router to a ragged situation. He uses Yoshimura's algorithm [102], which is a refinement of the track assignment algorithm from [103] which was described in Section 7.2. The horizontal wiring segments are fitted into the available spaces, which comprise full-length or partial tracks.

The complexity of the track assignment problem for ragged channels by itself, however, was not studied until recently. Casting their results in terms of the simple three-layer model described in Section 6.3 (the VHV model), in which there are no vertical conflicts and hence all there is to look at are the track assignments of straight horizontal wires, Vijayan, Chen, and Wong [99] prove that this problem is NP-complete. They proceed to providing a (worst-case exponential time) branch and bound algorithm for solving the problem, patterned after [53] (described in Section 7.1), as well as a number of well performing heuristics. These all require that all terminals reside on the horizontal portions of the sides. It seems, however, that the last word on this realistic variant of classic channel routing has not been written yet.

Finally, we turn to ragged edges enclosing a fixed shape polygonal region. One approach is to treat them as switchboxes with ragged edges (rather than being simply rectangular). Exact algorithms still exist for single-layer routing [77] and knock-knee routing [51]. The algorithm for knock-knee routing has running time at worst $O(A^2)$, where A is the number of grid points in the routing region; the running time is much better — $O(A \log^2 A)$ — if the routing problem is locally even (see Section 8.1). Several of the algorithms reviewed in the previous section actually were designed for these more general switchboxes [49, 66, 95]. Also, Hsu's algorithm [45], which was mentioned above, applies to this situation as well.

Another approach tries to analyze specific shapes. Chen [16] presents a mapping of an `L`-shaped channel routing problem to a regular channel routing problem by using 45° wires; he also addresses Manhattan routing by decomposing an `L`-shaped channel into a regular channel and a 3-sided switchbox. In 1981, Pinter [74] studied the interaction between portions of `T`- and `X`-shaped channels. The requirements on the channel's shape are stringent: it must be decomposable into two rectangles (for a `T`) or five rectangles forming an `X`. The study focused on how to arrange the net's crossings between one rectangular portion and the next. In a `T`-shaped channel, for example, if all nets have one representative in the lower rectangle (the "trunk") and one in the upper part, necessary and sufficient conditions (along with a routing algorithm) are given to determine routability. The partial order imposed on net crossings is then used to study X-shaped channels as well.

[24] It would have been interesting to apply this technique to ordinary channels, with smooth sides, but this is not reported.

9 Discussion and Open Problems

We have discussed various computational aspects of the channel routing problem. This is not only an important practical problem, lying at the heart of the automatic layout generation process, but it also poses a rather intricate mathematical challenge. We have tried to present and evaluate the two approaches — practical and theoretical — by giving a comprehensive survey of the state of the art. We conclude by trying to relate these two viewpoints. We list a number of lessons to be learnt from the various theoretical results. We believe it is worthwhile for the practitioner to take these into account when writing or using a channel router. We close with what we believe are the most important open problems pertaining to channel routing whose solution will lead to even better practical solutions.

First here is our list of lessons.

- Density is not the only lower bound on channel width. The use of vertical constraints, for example, to compensate for this phenomenon, is both limited to certain models and does not not reflect an intrinsic difficulty. Flux and flux related bounds should be used more widely, since those cases where they do become dominant are indeed the most problematic.
- Not all channel routing problems are dictated rigidly. In many cases, they are set up by a previous stage of the design process (which may or may not be automatic). Designers and implementors of design automation systems should be aware of the fact that channel routers produce better results for some situations than for others, and setting up favorable configurations — by, for example, reducing density as described in Section 8.1 — can help considerably.
- Picking the right wiring model and fully understanding the implications of this choice are crucial. Each model has its limitations, both in terms of applicability and in its potential to yield results that match certain bounds. When deciding which model to use, one should be fully aware of these issues.
- In some cases, the distinction between layer assignment and path determination can be quite effective. This paradigm, which appears in different forms in many of the results surveyed here, is yet another example of how abstraction and separation of concerns can help in finding solutions to hard problems.

The last word on the channel routing problem has not been written as yet. In addition to the unresolved problems that were mentioned throughout the above text, pertaining to specific issues, here are, in our opinion, the three major open problems in this area:

- What is the complexity of the general two-layer channel routing problem? Even if no overlap is allowed, but wires can run in either layer in both directions, we still do not know whether the problem of finding the smallest channel width is NP-complete or polynomial time solvable.

- Can one find tighter lower bounds and matching guaranteed performance bounds for routing in more than two layers? The results described in Section 6.3 have only partially resolved this question.

- Finally, can non-rectilinear geometries help in reducing channel width? The analysis conducted for the one-layer model (*e.g.* [92]) indicates that the benefits may not be as dramatic as one would hope for, but both the theoretical and practical communities ought to further investigate this issue.

The future of practical channel routing is in multi-layer models. However, it is not clear whether, under the general model, routing channels in two layers is fundamentally different from routing in more layers. Results in two-layer routing may still have impact for years to come.

Acknowledgements. The work of Andrea LaPaugh was supported in part by NSF grant number MIP 86119335 and DARPA/ONR grant N00014-88-K-0459, and the work of Ron Pinter was supported in part by ONR grant number N00014-89-J-1906.

We would like to thank Ron Rivest (of MIT) who got us both started in the integrated circuits layout business about a decade ago, and who has influenced both the research in this field in general and our own approach (as it shows through in this paper) in particular. We would also like to thank Sandeep Bhatt (of Yale) for his useful comments on an earlier draft of this survey. Finally, we would like to thank Ike Burke, the production editor of Annual Reviews, Inc. for his patience and confidence.

References

[1] A. V. Aho, M. R. Garey, and F. K. Hwang. Rectilinear Steiner trees: Efficient special case algorithms. *Networks*, 7:37–58, 1977.

[2] S. B. Akers. Routing. In M. A. Breuer, editor, *Design Automation of Digital Systems: Theory and Techniques*, Chapter 6. Prentice-Hall, 1972.

[3] M. J. Atallah and S. E. Hambrusch. Optimal rotation problems in channel routing. *IEEE Transactions on Computers*, C-35(9):843–847, September 1986.

[4] B. S. Baker, S. N. Bhatt, and F. T. Leighton. An approximation algorithm for Manhattan routing. In F. P. Preparata, editor, *Advances in Computing Research 2 (VLSI Theory)*, pages 205–229. JAI Press, 1984.

[5] A. E. Baratz. *Algorithms for Integrated Circuit Signal Routing.* PhD thesis, Massachusetts Institute of Technology, August 1981.

[6] S. Ben-Yehuda and R. Y. Pinter. Symbolic layout improvement using string matching based local transformations. In *Proceedings of the Decennial Caltech Conference on VLSI*, pages 227–239. MIT Press, March 1989.

[7] B. Berger, M. L. Brady, D. J. Brown, and F. T. Leighton. Nearly optimal algorithms and bounds for multilayer channel routing. unpublished manuscript, 1988.

[8] M. L. Brady and D. J. Brown. VLSI routing: Four layers suffice. In F. P. Preparata, editor, *Advances in Computing Research 2 (VLSI Theory)*, pages 245–257. JAI Press, 1984.

[9] M. L. Brady and D. J. Brown. Optimal multilayer channel routing with overlap. In *Proceedings of the Fourth MIT Conference on Advanced Research in VLSI*, pages 281–296, April 1986. to appear in *Algorithmica.*

[10] D. Braun, J. L. Burns, F. Romeo, A. L. Sangiovanni-Vincentelli, K. Mayaram, S. Devadas, and H.-K. T. Ma. Techniques for multilayer channel routing. *IEEE Transactions on Computer-Aided Design*, CAD-7(6):698–712, June 1988.

[11] D. J. Brown and R. L. Rivest. New lower bounds for channel width. In *Proceedings of the CMU Conference on VLSI Systems and Computations*, pages 178–185. Computer Science Press, October 1981.

[12] P. Bruell and P. Sun. A 'greedy' three layer channel router. In *Proceedings of the International Conference on Computer-Aided Design*, pages 298–300. IEEE Computer Society, November 1985.

[13] M. Burstein. Channel routing. In T. Ohtsuki, editor, *Layout Design and Verification (Advances in CAD for VLSI, Vol. 4)*, pages 133–167. North-Holland, 1986.

[14] M. Burstein and R. Pelavin. Hierarchical channel router. *Integration — the VLSI Journal*, 1(1):21–38, 1983.

[15] M. Burstein and R. Pelavin. Hierarchical wire routing. *IEEE Transactions on Computer-Aided Design*, CAD-2(4):223–234, October 1983.

[16] H. H. Chen. Routing L-shaped channels in nonslicing-structure placement. In *Proceedings of the Twenty-fourth Design Automation Conference*, pages 152–158. IEEE, 1987.

[17] Y. K. Chen and M. L. Liu. Three-layer channel routing. *IEEE Transactions on Computer-Aided Design*, CAD-3(2):156–163, April 1984.

[18] J. Cong, D. F. Wong, and C. L. Liu. A new approach to three- or four-layer routing. *IEEE Transactions on Computer-Aided Design*, CAD-7(10):1094–1104, October 1988.

[19] I. Dagan, M. C. Golumbic, and R. Y. Pinter. Trapezoid graphs and their coloring. *Discrete Applied Mathematics*, 21:35–46, 1988.

[20] D. N. Deutsch. A dogleg channel router. In *Proceedings of the Thirteenth Design Automation Conference*, pages 425–433. IEEE Computer Society, June 1976.

[21] D. N. Deutsch. Compacted channel routing. In *Proceedings of the International Conference on Computer-Aided Design*, pages 223–225. IEEE Computer Society, November 1985.

[22] D. N. Deutsch. Solutions to a switchbox routing problem. *IEEE Transactions on Computer-Aided Design*, CAD-4(2):163, April 1985.

[23] D. Dolev, K. Karplus, A. Siegel, A. Strong, and J. D. Ullman. Optimal wiring between rectangles. In *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*, pages 312–317, May 1981.

[24] W. E. Donath. Placement and average interconnection lengths of computer logic. *IEEE Transactions on Circuits and Systems*, CAS-26(4):272–277, April 1979.

[25] A. El-Gamal. Two-dimensional stochastic model for interconnections in master slice integrated circuits. *IEEE Transactions on Circuits and Systems*, CAS-28(2):127–138, February 1981.

[26] R. J. Enbody and H. C. Du. Near-optimal n-layer channel routing. In *Proceedings of the Twenty-third Design Automation Conference*, pages 709–714. IEEE, 1986.

[27] S. Even, A. Pnueli, and A. Lempel. Permutation graphs and transitive graphs. *Journal of the Association for Computing Machinery*, 19(3):400–419, July 1972.

[28] A. Frank. Disjoint paths in a rectilinear grid. *Combinatorica*, 2(4):361–371, 1982.

[29] S. Gao and S. E. Hambrusch. Two-layer channel routing with vertical unit-length overlap. *Algorithmica*, 1(2):223–232, 1986.

[30] S. Gao and M. Kaufmann. Channel routing of multiterminal nets. In *Proceedings of the Twenty-eighth Symposium on Foundations of Computer Science*, pages 316–325. IEEE, October 1987.

[31] S. Gao, M. Kaufmann, and F. M. Maley. Advances in homotopic layout compaction. In *Proceedings of the ACM Symposium on Parallel Algorithms and Architectures*. ACM, June 1989, pages 273-282.

[32] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., San Francisco, Ca, 1979.

[33] F. Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM Journal of Computing*, 1(2):180–187, June 1972.

[34] I. S. Gopal, D. Coppersmith, and C. K. Wong. Optimal wiring of movable terminals. *IEEE Transactions on Computers*, C-32(9):845–858, September 1983.

[35] R. I. Greenberg, A. T. Ishii, and A. L. Sangiovanni-Vincentelli. Mulch: A multi-layer channel router using one, two, and three layer partitions. In *Proceedings of the International Conference on Computer-Aided Design*, pages 88–91. IEEE Computer Society, November 1988.

[36] U. I. Gupta, D. T. Lee, and J. Y.-T. Leung. An optimal solution for the channel-assignment problem. *IEEE Transactions on Computers*, C-28(11):807–810, November 1979.

[37] G. T. Hamachi and J. K. Ousterhout. A switchbox router with obstacle avoidance. In *Proceedings of the Twenty-first Design Automation Conference*, pages 173–179. IEEE, 1984.

[38] S. E. Hambrusch. Using overlap and minimizing contact points in channel routing. In *Proceedings of the Twenty-first Annual Allerton Conference on Communication, Control, and Computing*, pages 256–257, October 1983.

[39] S. E. Hambrusch. Channel routing algorithms for overlap models. *IEEE Transactions on Computer-Aided Design*, CAD-4(1):23–30, January 1985.

[40] A. Hashimoto and J. Stevens. Wiring routing by optimizing channel assingment within large apertures. In *Proceedings of the Eighth Design Automation Workshop*, pages 155–169. IEEE, 1971.

[41] W. R. Heller, W. Mikhail, and W. Donath. Prediction of wiring space requirements for LSI. *Journal of Design Automation and Fault Tolerant Computing*, 2:117–144, 1978.

[42] F.-L. Heng, W. W. Lin, A. S. LaPaugh, and R. Y. Pinter. Decreasing channel width lower bounds by channel widening. Technical Report 218-89, Department of Computer Science, Princeton University, 1989.

[43] W. Heyns. The 1-2-3 routing algorithm or the single channel 2-step router on 3 interconnection layers. In *Proceedings of the Nineteenth Design Automation Conference*, pages 113–120. IEEE, 1982.

[44] D. W. Hightower and R. L. Boyd. A generalized channel router. In *Proceedings of the Seventeenth Design Automation Conference*, pages 12–21. IEEE Computer Society, June 1980.

[45] C.-P. Hsu. A new two-dimensional routing algorithm. In *Proceedings of the Nineteenth Design Automation Conference*, pages 46–50. IEEE Computer Society, June 1982.

[46] D. S. Johnson. The NP-completeness column: An ongoing guide. *Journal of Algorithms*, 3(3):381–395, September 1982.

[47] D. S. Johnson, A. S. LaPaugh, and R. Y. Pinter. Minimizing channel density by lateral shifting. unpublished manuscript, 1988.

[48] R. Joobbani. *An Artificial Intelligence Approach to VLSI Routing.* Kluwer Academic Publishers, Boston, MA, 1986.

[49] J. M. Jou, J. Y. Lee, and J. F. Wang. A new three-layer detailed router for VLSI layout. In *Proceedings of the International Conference on Computer-Aided Design*, pages 382–385. IEEE Computer Society, November 1987.

[50] M. Kaufmann and F. M. Maley. Parity conditions in homotopic knock-knee routing. unpublished manuscript, 1989.

[51] M. Kaufmann and K. Mehlhorn. Routing through a generalized switchbox. *Journal of Algorithms*, 7:510–531, 1986.

[52] T. Kawamoto and Y. Kajitani. The minimum width routing of a 2-row 2-layer polycell-layout. In *Proceedings of the Sixteenth Design Automation Conference*, pages 290–296. IEEE Computer Society, June 1979.

[53] B. W. Kernighan, D. G. Schweikert, and G. Persky. An optimum channel routing algorithm for polycell layouts of integrated circuits. In *Proceedings of the Tenth Design Automation Workshop*, pages 50–59. IEEE Computer Society, 1973.

[54] A. S. LaPaugh. *Algorithms for Integrated Circuit Layout: An Analytic Approach.* PhD thesis, Massachusetts Institute of Technology, August 1980. Available as Technical Report MIT/LCS/TR-248.

[55] A. S. LaPaugh and R. Y. Pinter. On minimizing channel density by lateral shifting. In *Proceedings of the International Conference on Computer-Aided Design*, pages 123–124. IEEE Computer Society, September 1983.

[56] U. Lauther. Channel routing in a general cell environment. In *Proceedings of the International Conference on Very Large Scale Integration: VLSI '85*, pages 393–403. IFIP, August 1985.

[57] C. Y. Lee. An algorithm for path connections and its applicatins. *IRE Transactions on Electronic Computers*, VEC-10(9):346–365, September 1961.

[58] F. T. Leighton. New lower bounds for channel routing. unpublished manuscript, 1982.

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

[60] H. W. Leong, D. F. Wong, and C. L. Liu. A simulated-annealing channel router. In *Proceedings of the International Conference on Computer-Aided Design*, pages 226–228. IEEE Computer Society, November 1985.

[61] Y. L. Lin, Y. C. Hsu, and F. S. Tsai. A detailed router based on simulated evolution. In *Proceedings of the International Conference on Computer-Aided Design*, pages 38–41. IEEE Computer Society, November 1988.

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

[63] M. J. Lorenzetti and D. S. Baeder. Routing. In B. T. Preas and M. J. Lorenzetti, editors, *Physical Design Automation of VLSI Systems*, chapter 5, pages 157–210. Benjamin/Cummings Publishing Co., CA, 1988.

[64] W. K. Luk. A greedy switch-box router. *Integration — the VLSI Journal*, 3:129–149, 1985.

[65] F. M. Maley. Compaction with automatic jog introduction. In *Proceedings of the Chapel-Hill Conference on VLSI*, pages 261–283. Computer Science Press, May 1985.

[66] M. Marek-Sadowska. Two-dimensional router for double layer layout. In *Proceedings of the Twenty-second Design Automation Conference*, pages 117–123. IEEE, 1985.

[67] M. Marek-Sadowska and E. S. Kuh. A new approach to channel routing. In *Proceedings of ISCAS-82*, pages 764–767, 1982.

[68] C. Mead and L. Conway. *Introduction to VLSI Systems.* Addison-Wesley, Reading, MA, 1980.

[69] K. Mehlhorn and F. P. Preparata. Routing through a rectangle. *Journal of the Association for Computing Machinery*, 33(1):60–85, January 1986.

[70] K. Mehlhorn, F. P. Preparata, and M. Sarrafzadeh. Channel routing in knock-knee mode: Simplified algorithms and proof. *Algorithmica*, 1(2):213–221, 1986.

[71] A. Mirzaian. Channel routing in VLSI. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, pages 101–107. ACM, May 1984.

[72] H. Okamura and P. D. Seymour. Multicommodity flows in planar graphs. *Journal of Combinatorial Theory, Series B*, 31:75–81, 1981.

[73] G. Persky, D. N. Deutsch, and D. G. Schweikert. LTX — a minicomputer-based system for automatic LSI layout. *Automation and Fault-Tolerant Computing*, 1(3):217–256, 1977.

[74] R. Y. Pinter. Optimal routing in rectilinear channels. In *Proceedings of the CMU Conference on VLSI Systems and Computations*, pages 160–177. Computer Science Press, October 1981.

[75] R. Y. Pinter. *The Impact of Layer Assignment Methods on Layout Algorithms for Integrated Circuits.* PhD thesis, Massachusetts Institute of Technology, August 1982. Available as Technical Report MIT/LCS/TR-291.

[76] R. Y. Pinter. On routing two-point nets across a channel. In *Proceedings of the Nineteenth Design Automation Conference*, pages 894–902. IEEE Computer Society, June 1982.

[77] R. Y. Pinter. River routing: Methodology and analysis. In *Proceedings of the Third Caltech Conference on VLSI*, pages 141–163. Computer Science Press, March 1983.

[78] R. Y. Pinter. Optimal layer assignment for interconnect. *Journal of VLSI and Computer Systems*, 1(2):123–137, 1984.

[79] V. Pitchumani and Q. Zhang. A mixed HVH-VHV algorithm for three-layer channel routing. *IEEE Transactions on Computer-Aided Design*, CAD-6(4):497–502, July 1987.

[80] F. P Preparata and M. Sarrafzadeh. Channel routing of nets of bounded degree. In P. Bertolazzi and F. Luccio, editors, *VLSI: Algorithms and Architecture*, pages 189–203. Elsevier (North-Holland), May 1984.

[81] F. P. Preparata and W. Lipski, Jr. Optimal three layer channel routing. *IEEE Transactions on Computers*, C-33(5):427–437, May 1984.

[82] J. Reed, A. Sangiovanni-Vincentelli, and M. Santomauro. A new symbolic channel router: YACR2. *IEEE Transactions on Computer-Aided Design*, CAD-4(3):208–219, July 1985.

[83] R. L. Rivest. The PI (Placement and Interconnect) system. In *Proceedings of the Nineteenth Design Automation Conference*, pages 475–481. IEEE Computer Society, June 1982.

[84] R. L. Rivest, A. E. Baratz, and G. L. Miller. Provably good channel routing algorithms. In *Proceedings of the CMU Conference on VLSI Systems and Computations*, pages 153–159. Computer Science Press, October 1981.

[85] R. L. Rivest and C. M. Fiduccia. A 'greedy' channel router. In *Proceedings of the Nineteenth Design Automation Conference*, pages 418–424. IEEE Computer Society, June 1982.

[86] A. L. Rosenberg. Three-dimensional integrated circuitry. In *Proceedings of the CMU Conference on VLSI Systems and Computations*, pages 69–80. Computer Science Press, October 1981.

[87] H.-J. Rothermal and D. A. Mlynski. Automatic variable-width routing for VLSI. *IEEE Transactions on Computer-Aided Design*, CAD-2(4):271–284, October 1983.

[88] A. Sangiovanni-Vincentelli and M. Santomauro. YACR: Yet another channel router. In *Proceedings of the Custom Integrated Circuits Conference*, pages 460–466, 1982.

[89] M. Sarrafzadeh. Channel-routing problem in the knock-knee mode is NP-complete. *IEEE Transactions on Computer-Aided Design*, CAD-6(4):503–506, July 1987.

[90] M. Sarrafzadeh and F. P. Preparata. Compact channel routing of multiterminal nets. In G. Ausiello and M. Lucertini, editors, *Annals of Discrete Mathematics*, pages 255–279. North-Holland, 1985.

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

[92] A. Siegel and D. Dolev. The separation for general single layer wiring barriers. In *Proceedings of the CMU Conference on VLSI Systems and Computations*, pages 143–151. Computer Science Press, October 1981.

[93] J. Soukup. Circuit layout. *Proceedings of the IEEE*, 69(10):1281–1304, October 1981.

[94] J. Soukup and J. Royle. On hierarchical routing. *Journal of Digital Systems*, 5(3):265–289, Fall 1981.

[95] K. J. Supowit. A minimum-impact routing algorithm. In *Proceedings of the Nineteenth Design Automation Conference*, pages 104–112. IEEE, 1982.

[96] T. G. Szymanski. Doglog channel routing is NP-complete. *IEEE Transactions on Computer-Aided Design*, CAD-4(1):31–41, January 1985.

[97] C. D. Thompson. *A Complexity Theory for VLSI.* PhD thesis, Carnegie-Mellon University, August 1980. Available as Technical Report CMU-CS-80-140.

[98] M. Tompa. An optimal solution to a wire-routing problem. *Journal on Computer and System Sciences*, 23(2):127–150, October 1981.

[99] G. Vijayan, H. H. Chen, and C. K. Wong. On VHV-routing in channels with irregular boundaries. *IEEE Transactions on Computer-Aided Design*, CAD-8(2):146–152, February 1989.

[100] N. H. E. Weste and K. Eshraghian. *Principles of CMOS VLSI Design — a Systems Perspective.* Addison-Wesley, Reading, MA, 1985.

[101] D. F. Wong, H. W. Leong, and C. L. Liu. *Simulated Annealing for VLSI Design.* Kluwer Academic Publishers, Boston, MA, 1988.

[102] T. Yoshimura. An efficient channel router. In *Proceedings of the Twenty-first Design Automation Conference*, pages 38–44. IEEE Computer Society, June 1984.

[103] T. Yoshimura and E. S. Kuh. Efficient algorithms for channel routing. *IEEE Transactions on Computer-Aided Design*, CAD-1(1):25–35, January 1982.