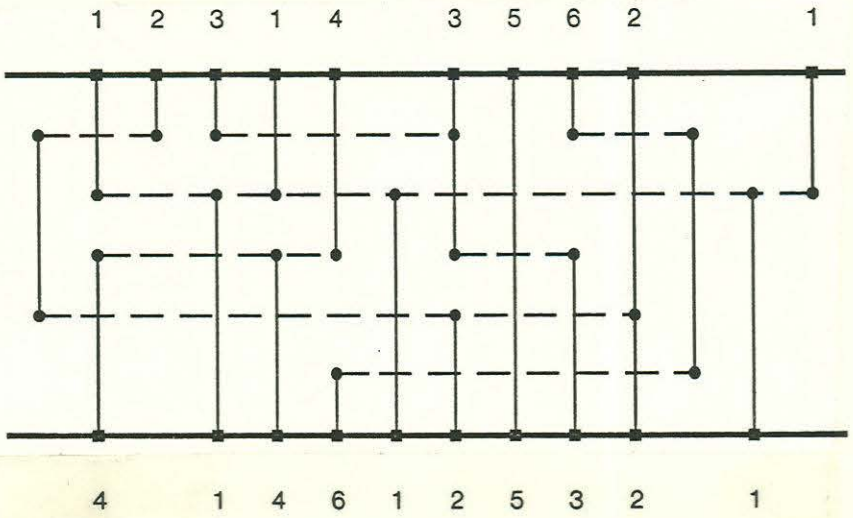
MIT/LCS/TM-238

AN APPROXIMATION ALGORITHM FOR MANHATTAN ROUTING



B. S. Baker, S. N. Bhatt, and F. T. Leighton

February 1983

An Approximation Algorithm for Manhattan Routing

Brenda S. Baker
Bell Laboratories
Murray Hill, NJ 07974

Sandeep N. Bhatt
Lab for Computer Science
M. I. T.
Cambridge, MA 02139

Frank Thomson Leighton
Mathematics Department and
Lab for Computer Science
M. I. T.
Cambridge, MA 02139

Abstract

Density has long been known to be an important measure of difficulty for Manhattan routing. In this paper, we identify a second important measure of difficulty, which we call *flux*. We show that flux, like density, is a lower bound on channel width. In addition, we present a linear-time algorithm which routes any multipoint net Manhattan routing problem with density d and flux f in a channel of width $2d + O(f)$. (For 2-point nets, the bound is $d + O(f)$.) Thus we show that Manhattan routing is one of the NP-complete problems for which there is a provably good approximation algorithm.

Since $f \leq \sqrt{n}$ for any n-net problem, the preceding bound indicates that every n-net problem can be routed in a channel of width $O(d + \sqrt{n})$, thus proving a conjecture of Brown and Rivest. For practical problems, however, the flux appears to be bounded by a small constant. In this case, the algorithm uses $2d + O(1)$ tracks. (For 2-point nets, the bound is $d + O(1)$.) These bounds are (asymptotically) nearly twice as good as those for the best known knock-knee algorithm and nearly as good as those for the best known 3-layer algorithm, yet do not require the use of either knock-knees or 3-layers of interconnect.

The results also have applications to a model of channel routing, which we call the *3-parameter model*, that is closer to the design rules of current fabrication technologies. The 3-parameter model is similar to the Manhattan model except that wires are assumed to be narrower than contact cuts in the 3-parameter model (as is the case in most fabrication technologies). By modifying the Manhattan routing algorithm, we show that *every* 3-parameter problem can be routed in a channel of width $2d + O(1)$. (For 2-point nets, the bound is $d + O(1)$.) Thus the 3-parameter model (like the knock-knees model) is a simple variation of Manhattan routing for which every problem can be routed in $O(d)$ tracks.

Key Words: approximation algorithm, channel routing, density, flux, knock-knees, Manhattan routing, multipoint net, 3-parameter model.

The research of Sandeep Bhatt and Tom Leighton was supported in part by DARPA contract N00014-80-C-0622. In addition, Tom Leighton was supported by Air Force contract OSR-82-0326 and a Bantrell Fellowship. A preliminary version of this paper will be presented at the 1983 ACM Symposium on Theory of Computing.

1. Introduction

Channel routing plays a central role in the development of automated layout systems for integrated circuits. One of the most common models for channel routing is known as *Manhattan routing*. In Manhattan routing, the channel consists of a *2-layer rectangular grid* of *columns* and *tracks* (rows). *Terminals* are located in the top layer of the *top* and *bottom tracks* at points where the tracks intersect a column. A set of terminals to be connected is called a *net*. Nets containing r terminals (r must always be larger than 1) are called *r-point nets*.

The object of the channel routing problem is to connect the terminals in each net with wires in a way which minimizes the width of the channel. The *width* of the channel is the number of tracks used to route the wires. Wires may be routed in either layer of tracks between the top and bottom track and in either layer of any column. (There is usually no restriction on the number of columns at either end of the channel.) Wires may *cross* one another in different layers but may not share corners (knock knees) or otherwise overlap. *Contact cuts* are used to connect wire segments which are in different layers. Figure 1 illustrates these constraints. Notice that the horizontal wire segments are routed in the bottom layer while the vertical wire segments are routed in the top layer. This kind of layer assignment is possible for all Manhattan routings and explains why Manhattan routing is often referred to as *layer per direction* and *reserved layer* routing.

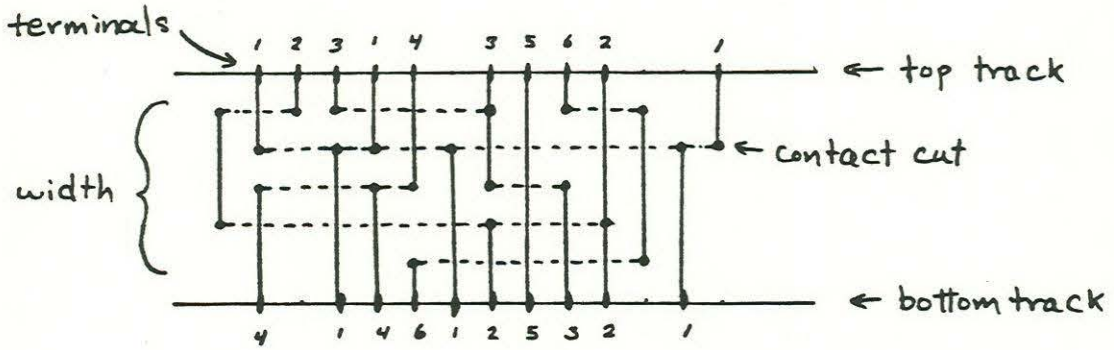


Figure 1: *Example of Manhattan routing. (Dashed lines denote wires in the bottom layer while solid lines denote wires in the top layer.)*

An important measure of a channel routing problem's difficulty is its density. The *density* of a problem is the maximum number of nets that are split by any vertical cut of the channel. A net is *split* by a cut of the channel if at least one terminal of the net lies on each side of the cut. (This definition could differ by one from another common definition of density in which a net is also said to be split if a terminal of the net lies *on* the cut.) For example, the cut shown in Figure 2 splits 10 nets. Inspection reveals that this problem has density 10.

It is not difficult to see that the density of a problem is a lower bound on its channel width. In practice, density is also close to an upper bound on channel width, since most practical problems can be routed in channels of width $d + O(1)$. This is not true in general, however. For example, Brown and Rivest [6] found examples of 2-point net problems with density 1 that require channels of width $\sqrt{2n}$ where n is the number of nets in the problem.

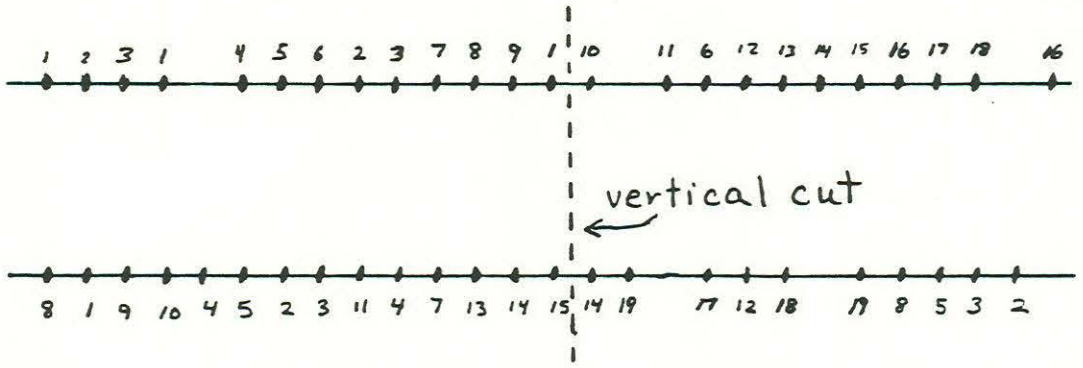


Figure 2: *A vertical cut which splits ten nets. (Vertical cuts measure density.)*

In this paper, we identify a second important measure of difficulty, which we call *flux*. The notion of flux is similar to that of density. Whereas density measures the number of nets split by a vertical cut of the channel, flux measures the number of nets split by a *horizontal* cut of the channel. In particular, we say that a problem has flux f if f is the largest integer such that there is a horizontal cut of the channel which spans $2f^2$ nontrivial columns and splits at least $2f^2 - f$ nontrivial nets. (A *trivial net* consists of two terminals which are in the same column. A *trivial column* contains a trivial net.) For example, both of the cuts shown in Figure 3 split 15 nontrivial nets. The upper cut spans 18 nontrivial columns and 2 trivial columns while the lower cut spans 15 nontrivial columns and 1 trivial column. Inspection reveals that this problem has flux 3.

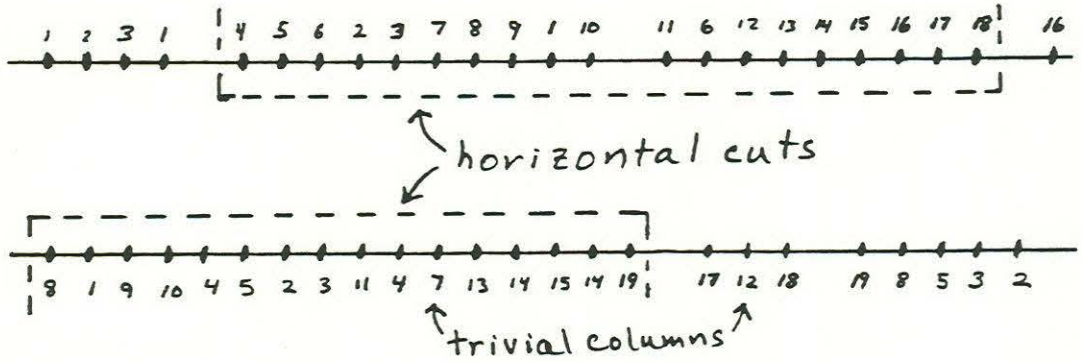


Figure 3: *Two horizontal cuts, each of which splits fifteen nontrivial nets. Note that net* 6 *is not split by the upper cut. (Horizontal cuts measure flux.)*

Like density, flux also serves as a lower bound for channel width. In fact, we will apply the techniques developed by Brown and Rivest in [6] to prove the following.

Theorem 1: *The minimum channel width of any Manhattan routing problem with density* d *and flux* f *is at least* $\max(d, f)$.

In [23], Szymanski showed that Manhattan routing is NP-complete. Recently, this result was strengthened with the aid of Yannakakis to show that Manhattan routing 2-point nets is NP-complete [24]. Thus it is not surprising that the many heuristics that have been proposed for Manhattan routing [1, 2, 7, 9, 10, 11, 12, 17, 20, 22, 26] fail to produce optimal solutions for some problems. In fact, none of these heuristics is known not to produce arbitrarily bad routings for some problems.

In this paper, we combine an analysis of density and flux in order to find a linear-time approximation algorithm for the Manhattan routing problem. Although the algorithm does not perform as well as the heuristics for most practical problems, it *is* guaranteed to produce a routing with width at most a constant times optimal for *all* problems. Hence, Manhattan routing is one of the NP-complete problems for which there is a good approximation algorithm. In particular, we will prove the following.

Theorem 2: *The Manhattan routing algorithm described in this paper routes every problem with density d and flux f in a channel of width $2d + O(f)$. The running time of the algorithm is linear in the area of the routing.*

Theorem 3: *The Manhattan routing algorithm described in this paper routes every 2-point net problem with density d and flux f in a channel of width $d+O(f)$. The running time of this algorithm is linear in the area of the routing.*

In the worst case, an n-net channel routing problem has flux $\Theta(\sqrt{n})$. Thus Theorem 2 implies that every problem can be routed in a channel of width $2d+O(\sqrt{n})$, thereby proving a conjecture made by Brown and Rivest for 2-point nets in [6].

For many problems, however, the flux is substantially smaller than $\Theta(\sqrt{n})$. For example, in practical problems the flux appears to be bounded by a small constant. There are three reasons for this. First, practical problems tend to have columns that contain less than two terminals. Second, practical problems often have nets containing terminals that are close together and on the same side of the channel. (As was illustrated in Figure 3, these are precisely the conditions which lead to a small flux.) Third, flux (unlike density) is a local phenomena and thus is less likely to grow as n increases. As an example, Deutsch's "difficult problem" [7] has 72 nets, 174 columns and density 19, but the flux is just 3.

Fortunately, the Manhattan routing algorithm described in this paper is particularly well suited to problems with constant flux. In particular, the algorithm routes multipoint net problems with constant flux in channels of width $2d + O(1)$. By Theorem 3, this bound can be improved to $d + O(1)$ for 2-point net problems with constant flux. The latter result is within a few (i.e., constant) tracks of optimal while the former result is within a few tracks of twice optimal. In addition, these results are (asymptotically) nearly twice as good as those for the best known algorithm in the 2-layer knock-knees model [21], and are nearly as good as those for the best known algorithm in the 3-layer knock-knees model [5, 19]. (The *knock-knees model* is similar to the Manhattan model except that wire segments are allowed to share corners in the knock-knees model.) Thus the results help to explain why good routings can often be found for practical problems without having to use knock-knees and/or 3 layers of interconnect.

The preceding results also have implications for routing in a more realistic model which we call the *3-parameter model*. The 3-parameter model is similar to the Manhattan model except that, as in current fabrication technologies, wires are assumed to be narrower than contact cuts in the 3-parameter model. Hence the design rules specify three (instead of two) parameters: wire width,

minimum size for contact cuts, and minimum separation between wires and/or contact cuts. By modifying the Manhattan routing algorithm, we will prove the following result for 3-parameter routing.

Theorem 4: *For any values of the parameters in the 3-parameter model, there is a constant c such that any 2-point net problem can be routed in a channel of width $d + c$, and any multipoint net problem can be routed in a channel of width $2d + c$. As before, these results can be achieved in linear time.*

The remainder of the paper is divided into three sections. In Section 2, we describe the Manhattan routing algorithm and prove Theorems 1-3. In Section 3, we describe the 3-parameter model in greater detail and prove Theorem 4. We conclude with some remarks and open questions in Section 4. The remarks are primarily concerned with the size of the constant factors and with the practicality of the results.

2. The Algorithm

In this section, we describe an algorithm for Manhattan routing. The algorithm is based on the notions of *flux* and *density*. The importance of these values is demonstrated in Section 2a, where we prove the lower bound of Theorem 1. In Section 2b, we describe a simplified version of the algorithm for routing top-to-bottom nets. (*Top-to-bottom nets* are nets with one terminal in the top track and one terminal in the bottom track.) In Section 2c, we describe the general algorithm and prove Theorems 2 and 3. The running time of the algorithm is discussed in Section 2d.

2a. Lower Bounds

In what follows, we show that flux is a lower bound for channel width. Since density is trivially a lower bound for channel width, we will have thus proved Theorem 1. Our method is a straightforward application of the techniques developed by Brown and Rivest in [6].

Consider a horizontal cut of the channel which spans $2f^2$ nontrivial columns and splits at least $2f^2 - f$ nontrivial nets. For each nontrivial net split by the cut, mark two terminals that are on opposite sides of the cut and that are in different columns. Also mark the terminals in trivial nets which are split by the cut. Let s denote the number of trivial nets split by the cut. For example, see Figure 4.

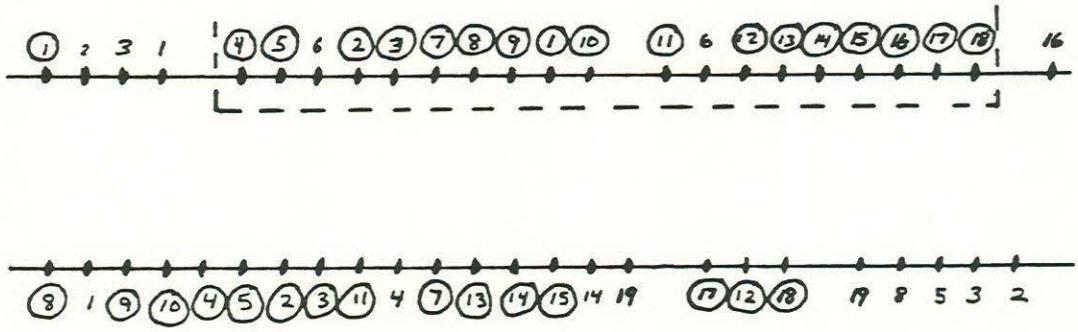


Figure 4: *Marked terminals in nets split by a horizontal cut. (Marked terminals are circled.)*

The marked terminals form a subproblem of the original problem with at least $2f^2 - f + s$ 2-point nets (s of which are trivial). At most f of the $2f^2 + s$ columns spanned by the cut are empty. By the arguments of [6], this means that at most $f + 2$ of the nontrivial nets can be routed into the correct column by using the first track: f into empty columns and one out each side of the cut. After the first track, there are at most $f + 2$ empty columns, the extra 2 having possibly been created by wires exiting through the sides of the cut in the first track. Thus, at most $f + 4$ nontrivial nets can be routed using the second track. In general, at most $f + 2i$ nontrivial nets can be routed using the ith track. Since at least $2f^2 - f$ nontrivial nets must be eventually routed, the minimum channel width w satisfies

$$wf + w(w + 1) \geq 2f^2 - f$$

which means that $w \geq f$. Thus the original problem requires a channel of width at least f.

2b. Manhattan Routing of Top-to-Bottom Nets

The algorithm proceeds in four phases. A brief description of each phase is provided below. This is followed by a more detailed description of Phase 3 (the heart of the algorithm). Figure 5 illustrates the portions of the channel used in each phase.

Phase 1: Route all trivial nets in the trivial fashion. Henceforth disregard these nets and the columns that contain them. (In particular, do not include these columns and nets when computing k below.) Find the least integer k for which there is a partition of the channel into groups of k^2 consecutive columns such that each group contains at least $3k$ columns which do not have terminals in the top track and at least $3k$ columns which do not have terminals in the bottom track. This can be accomplished by trying successive values (starting with 1, 2, 3, ...) until a value for k is found that satisfies the constraint. By the definition of flux, we know that $6(f + 1)$ is an upper bound for k and thus that $k = O(f)$.

Phase 2: Transform the problem into one that can be partitioned into blocks of k consecutive columns such that each block contains at least 3 columns that do not have terminals in the top track and at least 3 columns that do not have terminals in the bottom track. This can be done by routing the first 3 terminals in each block of k columns into columns that do not have terminals in the top track. For every k blocks of k columns, there are $3k$ columns that do not have terminals in the top track (by Phase 1). Thus this process uses at most $3k$ tracks in the upper portion of the channel and (by a symmetric argument) at most $3k$ tracks in the lower portion of the channel. (Henceforth, the term *block* refers to a block of k columns produced in this phase.)

Phase 3: Use the middle d tracks of the channel to route wires *between* blocks. It is not necessary to route wires into the correct column (that will be done in Phase 4); it is necessary only that the wires be routed into the correct block. As the details of this phase are fairly complicated, they will be described later in the section.

Phase 4: Route the subproblems in the upper-middle and lower-middle portion of each block; that is, route each net from the column assigned in Phase 3 to the column assigned in Phase 2. Since each subproblem has at most k nets and at least one "empty" column, these routings can be accomplished in a straightforward way (e.g., using the algorithm of Kawamoto and Kajitani in [11]) using $O(k)$ tracks. Specifically, the nets are routed one per track; the order of routing is

determined by constraints caused by a top terminal for one net lying above a bottom terminal of another net. When a cycle of constraints occurs, one net of the involved cycle is temporarily routed into the empty column to eliminate one constraint, and routed to its other terminal after the other nets in the cycle have been routed. Two tracks are used to route the last net in each such cycle of constraints.

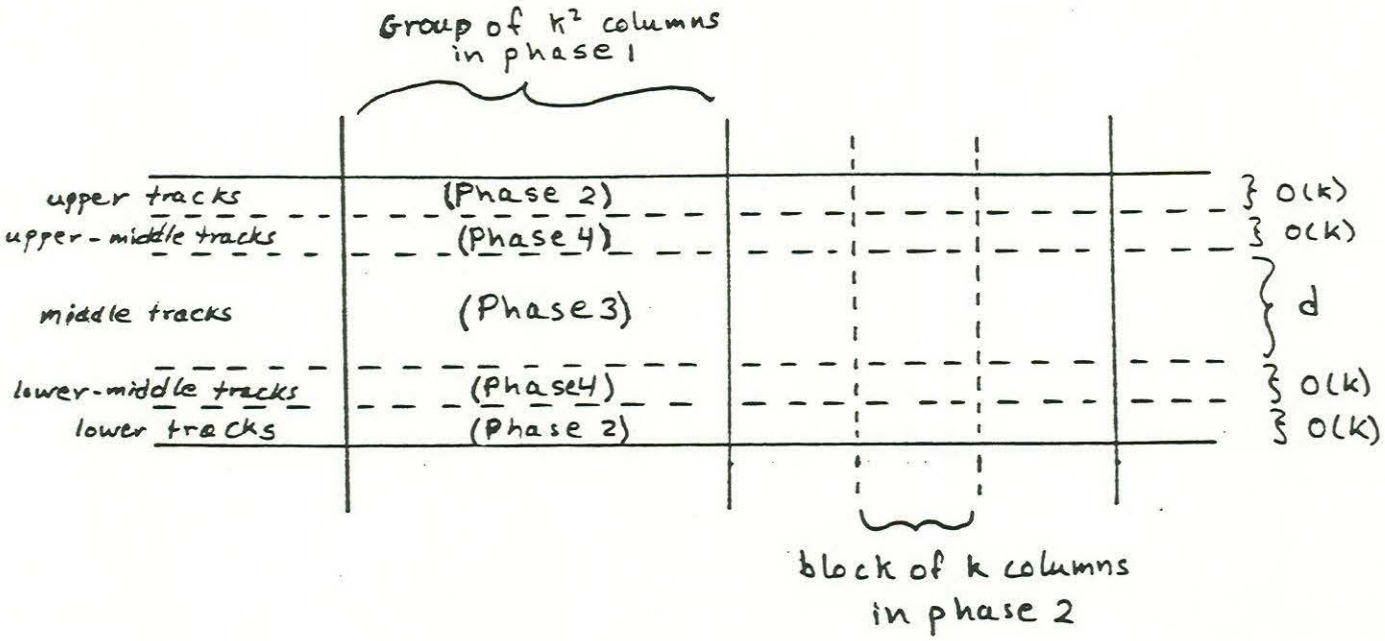


Figure 5: *Illustration of the four phases of the algorithm.*

The interblock routing in Phase 3 is accomplished in a block-by-block fashion, moving from left to right across the middle d tracks of the channel. Before any routing is done, the nets are classified into three categories: *falling nets* (those for which the block containing the top terminal is to the left of the block containing the bottom terminal), *rising nets* (those for which the block containing the top terminal is to the right of the block containing the bottom terminal), and *vertical nets* (those for which both terminals are in the same block). The routing procedure will ensure that before and after each block is routed, the tracks containing the rising nets will be above the empty tracks and the empty tracks will (in turn) be above the tracks containing falling nets.

The routing procedure will also insure that before and after each block is routed, the empty tracks share enough empty columns in the previously routed portion of the channel so that the pyramid structure shown in Figure 6a can be safely inserted into the routed portion of the channel. This fact will be used when the need for *backtracking* (pun intended) arises. As an example, Figure 6b illustrates how the pyramid can be used to make connections in columns that are "blocked" by a vertical segment of wire. Figure 6c illustrates how the pyramid structure is then updated for the remaining empty tracks.

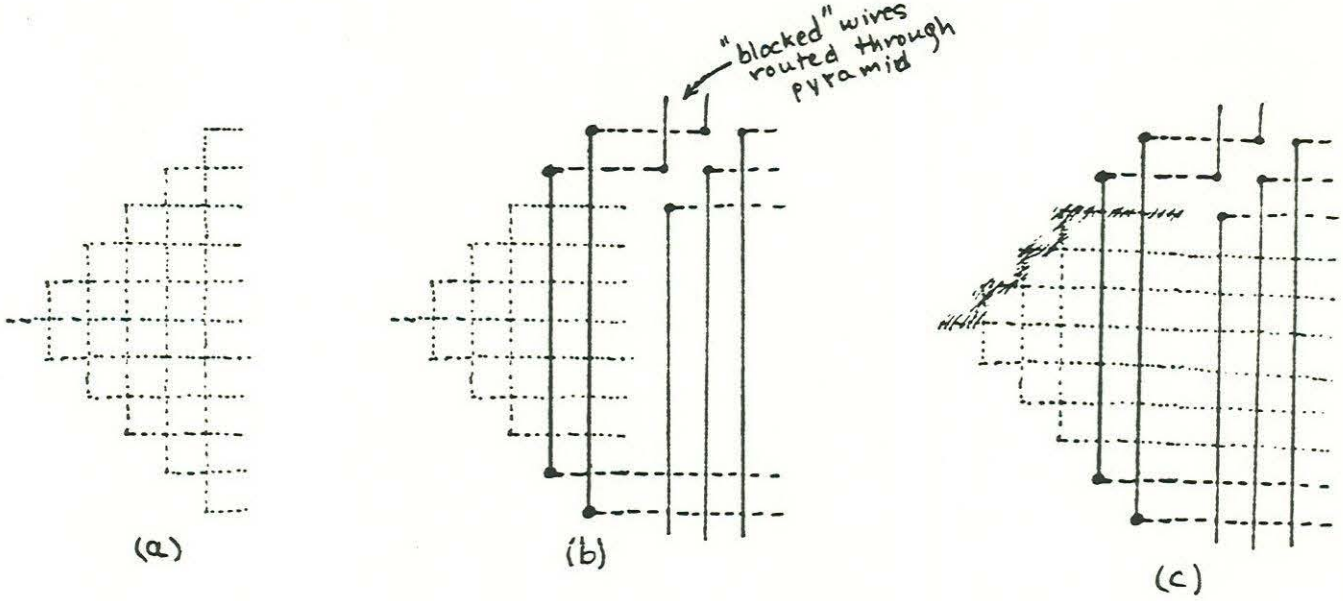


Figure 6: *The pyramid structure: (a) initially, (b) used to route wires, and (c) updated.*

The routing of each block starts as in Figure 7a and proceeds in five steps, as described below.

Step 1: Route the *ending nets* (those that end in the block) so that the wires form the *staircase pattern* shown in Figure 7b. This is done by moving the bottommost ending rising net upward and the topmost ending falling net downward wherever possible.

Step 2: Route the *continuing nets* (those which pass through the block) so that the empty tracks will again be between the tracks containing rising nets and the tracks containing falling nets. As before, the wires should be routed in a staircase pattern. (For example, see Figure 7c.) Note that the staircases generated in this step fit against those generated in Step 1.

Step 3: If more rising nets end than falling nets, make up the difference by routing some *starting* rising nets (those which originate in the block) in the manner shown in Figure 7d. (A symmetric procedure is followed if the opposite condition is true.) This step completes the routing of the initial portion of the block. It is worth noting that the empty tracks are again in the center of the channel and (even though their number may have increased) that the pyramid structure can still be inserted without difficulty. (For example, see Figure 7e.)

Step 4: If there are at least as many remaining starting falling nets than remaining starting rising nets, then route the starting falling nets as in Figure 7f. (a symmetric procedure is followed if the opposite condition holds.) Next route the remaining rising nets (if any) by making use of the pyramid structure for backtracking as shown in Figure 7g. If there are the same number of starting falling nets and starting rising nets, then one starting rising net is placed in the column that is empty in Figures 7f and 7g.

Step 5: Route the vertical nets in the obvious way. The completed routing and updated pyramid structure are shown in Figure 7h. Notice that the conditions illustrated in Figure 7a are now satisfied for the start of the next block.

Since the ending nets are always routed before the starting nets, d total tracks are sufficient to ensure the availability of an empty track whenever a starting net needs to be routed in Phase 3. Hence, the entire algorithm requires just $d + O(f)$ tracks.

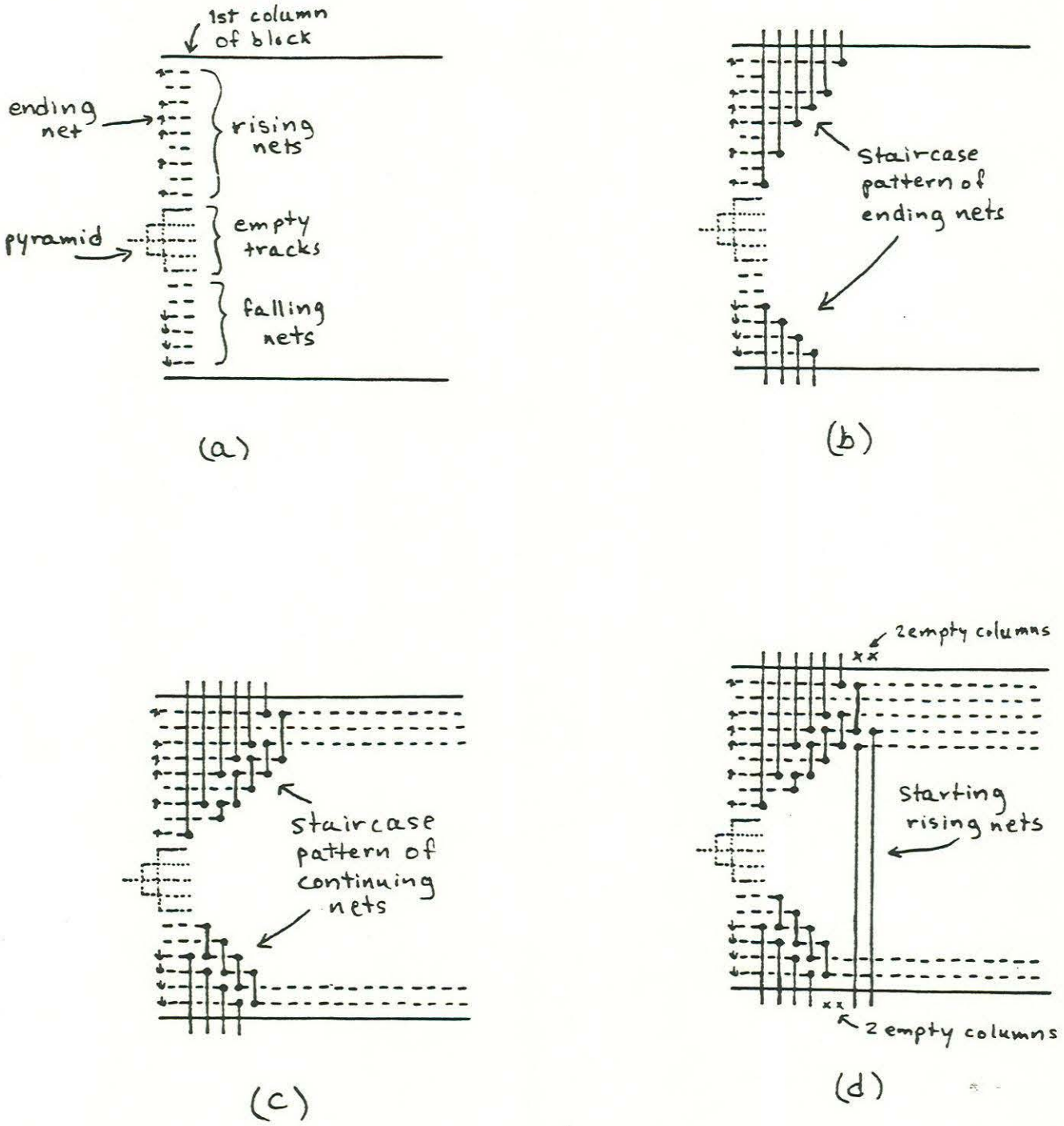


Figure 7: *Detailed illustration of Phase 3: (a) before routing starts (ending nets are marked with arrows), (b) routing of ending nets in Step 1, (c) routing of continuing nets in Step 2, (d) column balancing in Step 3, (e) updating pyramid structure, (f) routing of starting falling nets in Step 4, (g) routing of remaining starting rising nets in Step 4, and (h) completed routing of block and pyramid structure.*

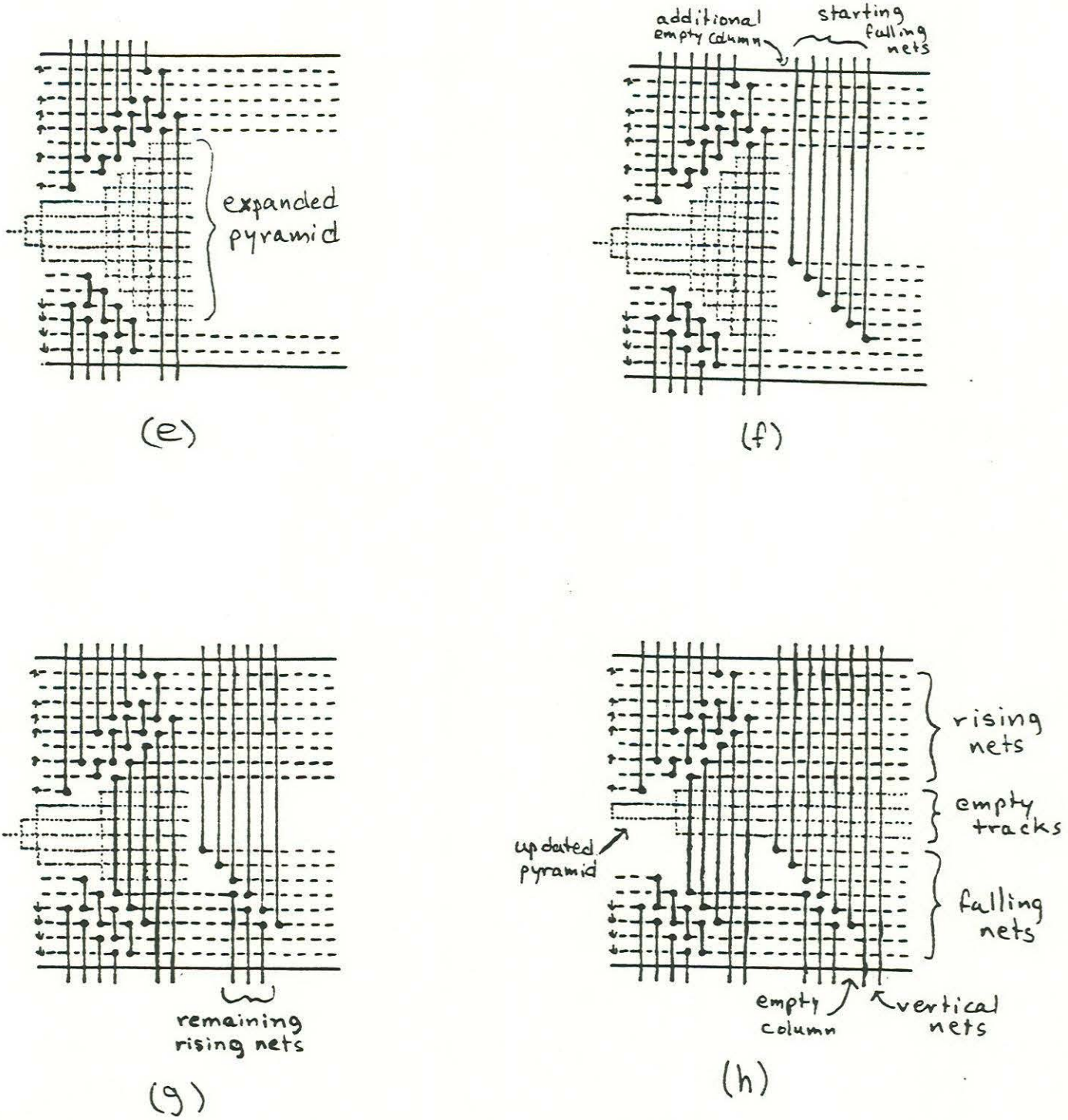


Figure 7: *Detailed illustration of Phase 3: (a) before routing starts (ending nets are marked with arrows), (b) routing of ending nets in Step 1, (c) routing of continuing nets in Step 2, (d) column balancing in Step 3, (e) updating pyramid structure, (f) routing of starting falling nets in Step 4, (g) routing of remaining starting rising nets in Step 4, and (h) completed routing of block and pyramid structure.*

2c. Manhattan Routing of Multipoint Nets

The algorithm for routing multipoint nets is similar to that for routing top-to-bottom nets. As before, the algorithm has four phases. Phase 2 is essentially the same as before. Phase 4 eliminates multiple top terminals and multiple bottom terminals within blocks by connecting them using $O(k)$ tracks, and connects one such terminal for each net to the position determined by Phase 3. Phases 1 and 3 are somewhat different from before. We describe these differences in what follows.

Phase 1: As before, route the trivial nets and henceforth disregard the trivial nets and columns. Find the least integer k for which there is a partition of the channel into groups of k^2 consecutive columns such that at most $k^2 - 3k$ nets are split by the horizontal cut at the top of each group and at most $k^2 - 3k$ nets are split by the horizontal cut at the bottom of each group. As before, $k = O(f)$. Using $O(k)$ tracks at the top of the channel, connect the terminals in the top track that are in the same net and same group for up to $3k$ nets. Do likewise at the bottom of the channel. The resulting problem will have at least $3k$ columns that do not have wires at the top of the channel and at least $3k$ columns that do not have wires at the bottom of the channel within each group of k^2 columns.

Phase 3: The basic strategy is the same as before except that $2d$ tracks may be required for the interblock routing. The details are described in what follows.

The nets are divided into four groups: *falling nets* (those for which the block containing the leftmost top terminal is to the left of the block containing the leftmost bottom terminal), *rising nets* (those for which the block containing the leftmost top terminal is to the right of the block containing the leftmost bottom terminal), *vertical nets* (those for which the leftmost top and bottom terminals are in the same block), and *same-side nets* (those for which all of the terminals are on the same side (top or bottom) of the channel). In addition, each net is divided into a rising portion and a falling portion. The *rising portion* of a net links the block containing the leftmost terminal to the blocks containing terminals in the top track of the channel. The *falling portion* of a net links the block containing the leftmost terminal to the blocks containing terminals in the bottom track of the channel. (Note that some nets don't have both rising and falling portions. For example, see Figure 8.) At most one connection will be made in each block for each portion of a net. Where a portion of a net has more than one terminal in a block, additional connections are made in Phases 1 and/or 4, as described above.

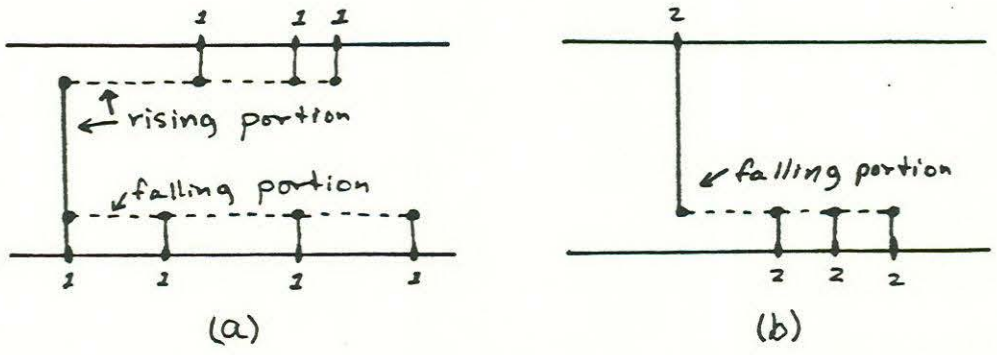


Figure 8: *(a) Division of rising nets into rising and falling portions, and (b) a falling net with no rising portion.*

As before, the algorithm is designed to ensure that (between blocks) tracks containing rising portions of nets are above empty tracks and that empty tracks are above tracks containing falling portions of nets. In addition, the algorithm will ensure that the pyramid structure shown in Figure 6 can be safely inserted in the empty tracks of the previously routed portion of the channel.

The routing proceeds block-by-block from left to right in the middle $2d$ tracks of the channel. Each block is routed in seven steps, as described below. The steps are numbered so as to coincide with the algorithm described in Section 2b. A completed routing is shown in Figure 9.

Step 1: Route the ending portions of nets in staircase patterns at the left end of the block.

Step 2: Route the continuing portions of nets in staircase patterns nestled against those in Step 1. The terminals (if any) for these portions of these nets are placed to the right of the corresponding staircase, and connections are made to them in the straightforward way.

Step 2.5: Route the starting same-side nets in a staircase fashion.

Step 3: If more columns have been used at the top of the channel than at the bottom, make up the difference by routing the rising portions of some starting rising nets. (A symmetric procedure is followed if the opposite is true.)

Step 4: Route the falling portions of starting falling nets and the remaining rising portions of starting rising nets, using the pyramid for backtracking.

Step 4.5: Route the falling portions of starting rising nets and the rising portions of starting falling nets in the straightforward way in empty tracks.

Step 5: Route the falling and rising portions of vertical nets in empty tracks.

Since each net is split into at most two portions, the preceding algorithm routes any problem in a channel of width $2d + O(f)$, as claimed in Theorem 2. For problems with only same-side nets and/or nets with one terminal on the opposite side and to the left of all other terminals, the width can be improved to $d + O(f)$. (This is because such nets have a rising portion or a falling portion, but not both.) Hence, all 2-point net problems can be routed in a channel of width $d + O(f)$, as claimed in Theorem 3.

2d. Running Time Analaysis

It is easily seen that Phases 1,2 and 4 require at most $O(tf)$ steps for t-terminal problems with flux f. Since flux is a lower bound on channel width and $t/2$ is a lower bound on channel length, these phases run in $O(A)$ time where A denotes the area of the routing.

Phase 3 is slightly more complicated since nonlinear time might, a priori, be required to update and maintain the pyramid structure for backtracking. The pyramid, however, is used only as an aid in understanding why the algorithm works; the algorithm itself has no need to use it at all. For example, the phrase "by making use of the pyramid structure for backtracking" in Step 4 could be replaced by "by simultaneously backtracking in the uppermost and lowermost empty tracks until both tracks encounter an empty column." Upon some reflection, it can be seen that the two commands result in equivalent routings. The latter command is simpler computationally, however, and results in an easy proof that Phase 3 can be accomplished in $O(dt) = O(A)$ steps.

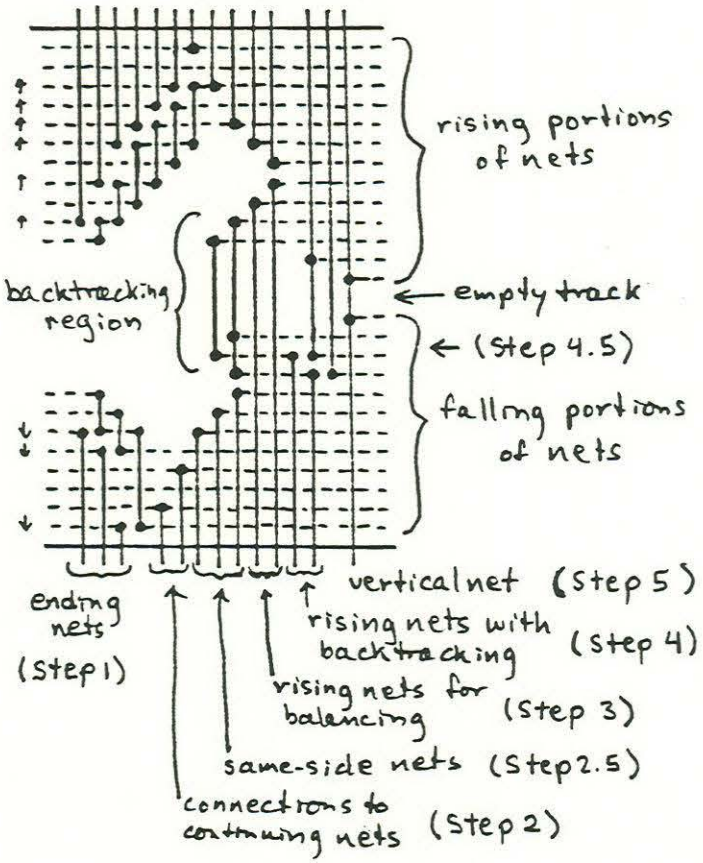


Figure 9: *Phase 3 routing of multipoint nets in a block.*

3. The 3-Parameter Model

The 3-parameter model has parameters p_w, p_c and p_s and the following design rules (see Figure 10).

Rule 1: Wires have width p_w.

Rule 2: Contact cuts are rectangular, and each side has length at least p_c.

Rule 3: Two wires in the same layer, two contact cuts, or a wire and a contact cut must be separated by distance at least p_s, unless they are electrically connected.

Rule 4: In order to be electrically connected, a contact cut and a wire must touch along an interval of length at least p_w. (The contact cut does not need to be centered on the wire).

We assume that $p_c > p_w$ (as in current fabrication technologies), and that terminals are spaced at multiples of $p_c + p_s$ (as is implicitly assumed in the Manhattan model). Wires are either horizontal or vertical, with a layer of interconnect devoted to each. Note that the Manhattan model is equivalent to the 3-parameter model with the restriction $p_c > p_w$ replaced by $p_c = p_w$.

In what follows, we show that for any values of the parameters, there is a constant c such that any 2-point net problem can be routed in a channel of width $d + c$, and any multipoint net problem can be routed in a channel of width $2d + c$, as claimed in Theorem 4. The key idea is to

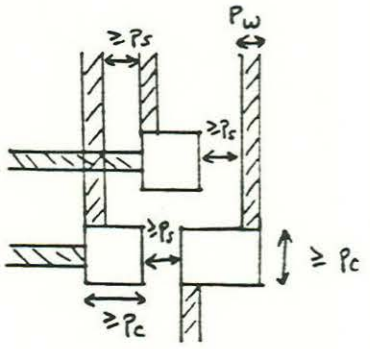


Figure 10: *Design rules of the 3-parameter model.*

place wires closer together than the spacing of the terminals where there is no need for adjacent contact cuts, thus creating empty space that can accomodate extra wires.

Consider a block in the d-track routing produced by Phase 3 of the top-to-bottom net algorithm described in Section 2b. Center all contact cuts on the wires they connect, and space successive horizontal wires at a distance of $p_c + p_s$ (measured from the center of one wire to the center of the next). In a group of wires (e.g., ending wires) where there are no adjacent contact cuts, the vertical wires may be spaced at intervals of $p_s + (p_c + p_w)/2$. Where two adjacent contact cuts occur, an additional separation of $(p_c - p_w)/2$ is needed. The latter situation occurs only between successive staircases. Hence, the wires of successive staircases should be shifted over an extra distance of $(p_c - p_w)/2$, as shown in Figure 11.

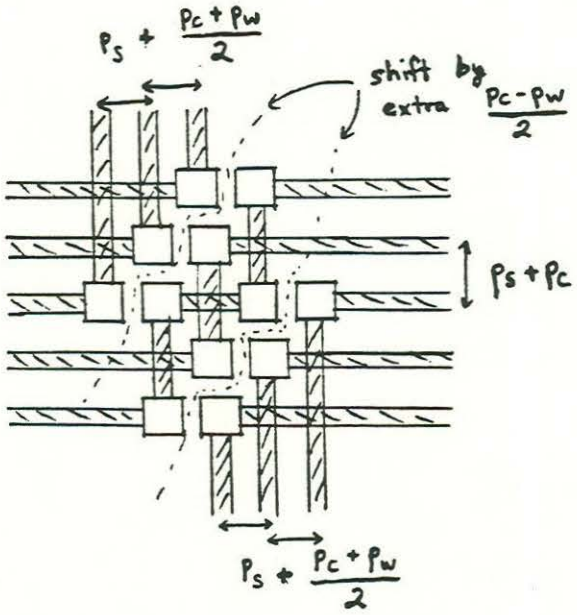


Figure 11: *Spacing between successive staircases.*

If k is the number of columns in the block in the Manhattan routing, then there are at most $k-3$ top terminals and $k-3$ bottom terminals in the block, and the total amount of space required in the 3-parameter routing is

$$k[p_s + (p_c + p_w)/2] + 3(p_c - p_w)/2.$$

If we choose $k \geq \frac{6(p_c+p_s)}{p_c-p_w} + 3$, then

$$(k-3)(p_c + p_s) \geq k[p_s + (p_c + p_w)/2] + 3(p_c - p_w)/2$$

and there is enough space in the region occupied by $k-3$ terminals to accommodate the modified routing of Phase 3. Furthermore, there is enough space to simulate the permutation of Phase 4, requiring $O(k)$ tracks. Finally, it can be shown that $k-3$ additional tracks at the top and bottom are sufficient to connect the (unevenly spaced) wires to the terminals. Thus the entire routing requires $d + c$ tracks where $c = O(k)$.

A similar argument shows that multipoint nets can be routed in $2d+c$ tracks, thus completing the proof of Theorem 4. In addition, the results show that the 3-parameter model gets around the flux lower bound inherent in the Manhattan model, as well as the $2d-1$ lower bound for 2-point nets in the knock-knees model [14].

The preceding 3-parameter layout spaces adjacent horizontal wires at distances of $p_c + p_s$ as in the Manhattan model. An obvious question is whether the horizontal tracks can be spaced closer together to reduce the total channel width. In fact, for any $\epsilon > 0$ it is possible to modify the preceding algorithm to require channel width at most

$$d[p_s + (p_c + p_w)/2 + \epsilon] + c$$

for some constant c. This is an asymptotic improvement over the density lower bound of $d(p_s + p_c) + p_s$ required by the Manhattan model. Similarly for multipoint nets and $\epsilon > 0$, it is possible to modify the multipoint net algorithm to require channel width at most

$$2d[p_s + (p_c + p_w)/2 + \epsilon] + c,$$

for some constant c.

It appears unlikely that these results can be improved to the point where the average horizontal wire spacing is below $p_s + (p_c + p_w)/2$, since that separation does not allow a contact cut to appear between two wires. For the same reason, however, it is to be expected that most optimal routings will require channel width close to $d[p_s + (p_c + p_w)/2]$, although instances occur that require only the minimum separation of $d(p_s + p_w) + p_s$.

4. Remarks

The Manhattan routing channel width bounds discussed in this paper have the form $\alpha d + \beta f$ where α and β are constants. For 2-point nets, $\alpha = 1$ in both the upper and lower bounds. For multipoint nets, however, $\alpha = 1$ in the lower bound and $\alpha = 2$ in the upper bound. It is our feeling that (for both the upper and lower bounds) the αd term might be expressible as $d + e$ where e measures (in some sense) the number or concentration of *conflicting* nets (those which share terminals in several columns or blocks). Since most practical multipoint net problems can

be routed in channels of width $d + O(1)$, e should be defined so that (like f) it is usually constant in practice. Should this turn out to be the case, it would improve our algorithm by a factor of 2 (making it very close to the optimal solution for practical multipoint net problems). It would also be interesting to know the value for e in the worst case. In particular, are there n-net problems which require $2d$ tracks for every d and n?

Most of our efforts in this paper have been devoted to optimizing the value of α since it is more important than β for most problems. For problems with large flux and small density, however, β is of primary importance. Although the upper and lower bounds calculated for β in this paper are very crude, there are more sophisticated arguments which show (under a slightly different definition of f and for $\alpha = 3$) that $\sqrt{2} \leq \beta \leq \sqrt{3}$ for top-to-bottom nets. For multipoint nets, the bounds are somewhat worse and merit further study.

Although these results are not as good as those achieved by heuristics for most practical problems, they might be useful as a worst-case backup to good heuristics (particularly in the future as problems with larger and larger densities are encountered). In any case, they do provide us with good worst-case theoretical bounds and with insight into why practical problems often have good solutions.

Acknowledgements

We would like to thank the following people for helpful discussions: Donna Brown, Dave Deutsch, Gary Miller, Ron Pinter, Franco Preparata, and Ron Rivest.

References

[1] S. Alford, *DYCHAR: A Channel Router which uses dynamic channel assignment,* S.B. thesis, Dept. of Electrical Engineering and Computer Science, M. I. T., (1980).

[2] T. Asano, T. Kitahashi, and K. Tanaka, "On a method of realizing minimum width wiring," *Electronics and Communications in Japan* Vol. **J59-A**, 2 (1976).

[3] T. Bolognesi, *A Channel Routing Algorithm Bounding Channel Width and Maximum Wire Length,* M.S. thesis, (1982).

[4] T. Bolognesi and D. Brown, "A channel routing algorithm with bounded wire length," unpublished manuscript, (1982).

[5] D. Brown and F. Preparata, personal communication, (1982).

[6] D. J. Brown and R. L. Rivest, "New lower bounds on channel width," *Proceedings CMU Conference on VLSI Systems and Computations* (1981).

[7] D. N. Deutsch, "A 'Dogleg' channel router," *Proceedings 13th. IEEE Design Automation Conference* (1976).

[8] D. Dolev, K. Karplus, A. Siegel, A. Strong, and J. Ullman, "Optimal wiring between rectangles," *Proceedings 13th. ACM Symposium on Theory of Computing* (1981).

[9] A. Hashimoto and J. Stevens, "Wire routing by optimizing channel assignment within large apertures," *Proceedings 8th. IEEE Design Automation Workshop* (1971).

[10] D. Hightower, "The interconnection problem - a tutorial," *Computer* Vol. 7, 4 (1974).

[11] T. Kawamoto and Y. Kajitani, "The minimum width routing of a 2-row 2-layer polycell-layout," *Proceedings 16th. IEEE Design Automation Conference* (1979).

[12] B. Kernighan, D. Schweikert, and G. Persky, "An optimal channel-routing algorithm for polycell layouts of integrated circuits," *Proceedings 10th. IEEE Design Automation Workshop* (1973).

[13] A. S. LaPaugh, *Algorithms for Integrated Circuit Layout: an Analytic Approach,* Ph.D. thesis, Dept. of Electrical Engineering and Computer Science, M. I. T., (1980).

[14] F. T. Leighton, "New lower bounds for channel routing," M. I. T. VLSI Memo 82-71, (1981).

[15] C. E. Leiserson and R. Y. Pinter, "Optimal placement for river routing," *Proceedings CMU Conference on VLSI Systems and Computations* (1981).

[16] C. Mead and L. Conway, *Introduction to VLSI Systems,* Addison-Wesley, (1980).

[17] G. Persky, D. Deutsch, and D. Schweikert, "A minicomputer-based system for automated LSI layout," *Journal of Design Automation and Fault-Tolerant Computing* Vol. 1, 3 (1977).

[18] R. Y. Pinter, "On routing 2-point nets across a channel," *Proceedings of the 19th IEEE Design Automation Conference* (1982).

[19] F. Preparata and W. Lipski, "Three layers are enough," *Proceedings Twenty third Annual IEEE Symposium on Foundations of Computer Science* (1982).

[20] R. L. Rivest, "The 'PI' (Placement and Interconnect) System," *Proceedings 19th. IEEE Design Automation Conference* (1982).

[21] R. L. Rivest, A. Baratz, and G. L. Miller, "Provably good channel routing algorithms," *Proceedings CMU Conference on VLSI Systems and Computations* (1981).

[22] R. L. Rivest and C. M. Fiduccia, "A greedy channel router," *Proceedings 19th. IEEE Design Automation Conference* (1982).

[23] T. Szymanski, "Dogleg channel routing is NP-Complete," unpublished manuscript, (1981).

[24] T. Szymanski and M. Yannakakis, personal communication, (1982).

[25] M. Tompa, "An optimal solution to a wire-routing problem," *Proceedings 12th. ACM Symposium on Theory of Computing* (1980).

[26] T. Yoshimura and E. Kuh, "Efficient algorithms for channel routing," U. C. Berkeley Electronics Research Laboratory Memo. M80/43, (1980).