ROUTING TRHOUGH A RECTANGLE

K. Mehlhorn and F.P. Preparata

A 83/12

Fachbereich 10
Universität des Saarlandes
D-6600 Saarbrücken
West Germany

ROUTING THROUGH A RECTANGLE

K. Mehlhorn* and F. P. Preparata**

Abstract: In this paper we present an O(NlogN) algorithm for routing through a rectangle. Consider an n by m rectangular grid and a set of N two-terminal nets. A net is a pair of points on the boundary of the rectangle. A layout is a set of edge-disjoint paths, one for each net. Our algorithm constructs a layout, if there is one, in time O(NlogN) which contrasts favorably to the area of the layout which might be as large as N^2. The layout constructed can be wired using 4 layers of interconnect with only O(N) contact-cuts. A partial extension to multi-terminal nets is also discussed.

Keywords: VLSI, routing, layout

*K. Mehlhorn, F3 10, Universität des Saarlandes, 66 Saarbrücken, West Germany

**F. P. Preparata, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, 1101 W. Springfield Avenue, Urbana, IL 61801 USA

This work was supported by the National Science Foundation under Grants MCS-81-05552 and ECS-81-06939 and by the Deutsche Forschungsgemeinschaft SFB 124, VLSI - Entwurf und Parallelität.

I. INTRODUCTION

A rectangle R is a rectangular subset of a rectilinear grid. It consists of n columns numbered 1 to n from left to right and m rows numbered 1 to m from bottom to top. A rectangle routing problem (RRP) consists of N nets $N_1,\ldots,N_N$. Each net is a pair of points on the boundary of rectangle R. The two points are called the terminals of the net. A boundary point (except for the corners) of R can be terminal of at most one net, a corner can be a terminal of at most two nets. A solution (a layout) for an RRP consists of a set of N edge-disjoint paths in rectangle R, one for each net. The path corresponding to (realizing) net N_i connects the two terminals of net N_i, $1 \leq i \leq N$.

Example: The first example shows an RRP with N=5, n=5, m=5. The nets are indicated by labels on the boundary nodes. Nodes with the same label define a net.

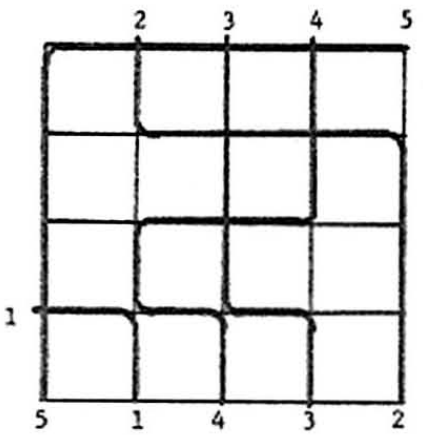


Figure 1. Example of a rectangle routing problem.

Our second example (Figure 2) has n=2m, N=n and consists of nets $N_i = ((i,1),(i+m,m))$, $1 \leq i \leq m$. The following figures show two colutions for this RRP, in the case N = 4:

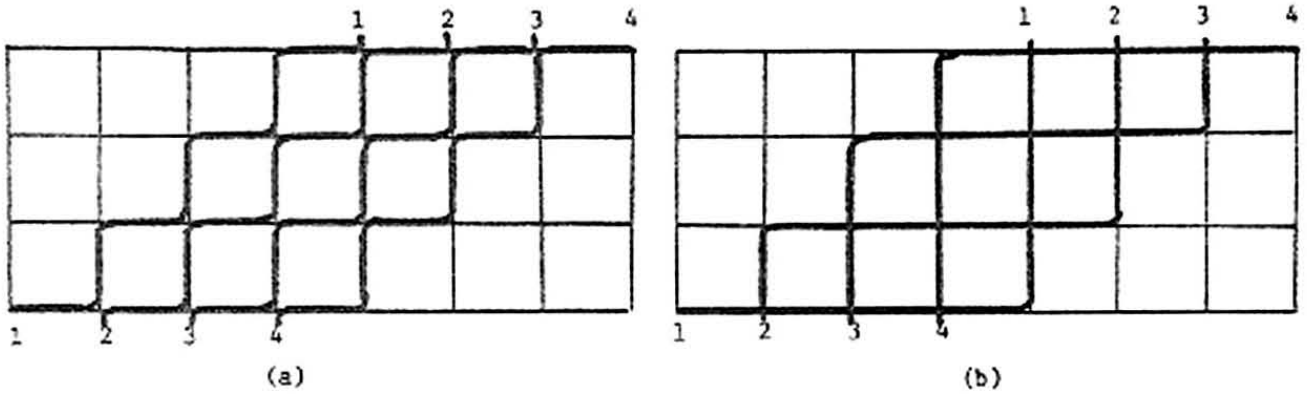


Figure 2. Two solutions of an RRP: (a) A solution with $\Omega(N^2)$ knock-knees; (b) a solution with $O(N)$ knock-knees.

The fundamental characteristic property of RRP's was recently established by A. Frank.

Theorem [Frank]. An RRP is solvable iff the revised row and column criteria hold. Moreover, a layout can be constructed in time $O(nm)$, if one exists.

The revised row and column criteria are defined as follows. A vertical cut (v-cut) is given by a pair of adjacent columns $(a,a+1)$. The capacity of a v-cut is the number of edges between columns a and a+1, i.e. m. The density of a v-cut $(a,a+1)$ is the number of nets which go across the cut, i.e. have exactly one terminal in columns $1,\ldots,a$. A cut is saturated if its density is equal to its capacity. Similar notions are defined for horizontal cuts (h-cuts).

Consider an RRP. Suppose there are $k \geq 0$ saturated horizontal cuts; if $k > 0$, they are $(r_1,r_1+1),\ldots,(r_k,r_k+1)$ with $r_1 < \ldots < r_k$, and let $(c,c+1)$ be any v-cut. The k saturated h-cuts dissect the area to the left of the v-cut into $k+1$ regions $T_1,\ldots,T_{k+1}$.

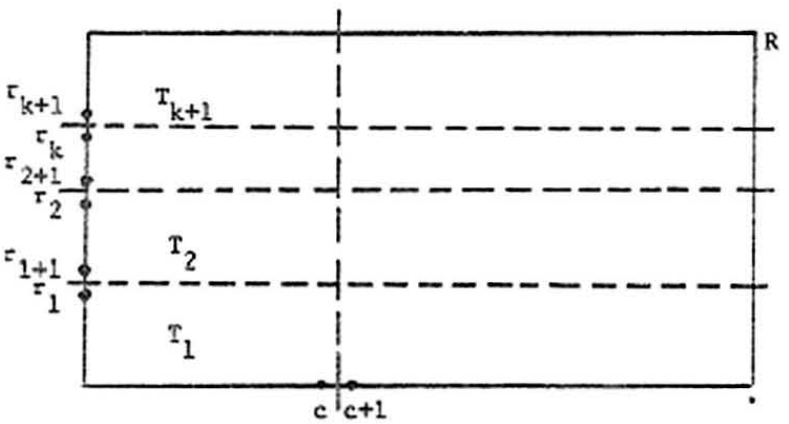


Figure 3. Rectangle dissection caused by saturated h-cuts and an arbitrary v-cut.

The extended degree of a node v of rectangle R is the degree of node v[1] plus the number of nets having v as a terminal. The parity of a set T of nodes is the parity of the sum of the extended degrees of the nodes in T. Finally, the parity-density of v-cut (c,c+1) is the density of the v-cut plus the sum of the parities of sets T_i, $1 \leq i \leq k+1$. The revised column criterion states that the parity density of any v-cut must not exceed its capacity. The revised row criterion is defined analogously.

Frank's algorithm solves an RRP in time $\theta(nm)$. For "squarish" rectangles this quantity is quadratic in the number of nets. Moreover, running time $\Omega(nm)$ is intrinsic to Frank's algorithm because it can generate layouts with $O(nm)$ knock-knees. In Figure 2a the path corresponding to any net bends $\Omega(m)$ times and hence $\Omega(Nm) = \Omega(N^2)$ amount of information is required to describe the layout.

---

[1] Note that the degrees of a corner, lateral, and internal node are 2, 3, and 4 respectively.

In Figure 2b, a layout for the same problem is shown where every path bends at most two times. Thus O(1) amount of information suffices to describe each path. The solution shown in Figure 2b will be constructed by the algorithm described in this paper. (For two-shore problems where all terminals are on the top or bottom of the rectangle it is known that this behavior is achievable [Preparata-Lipski].) The main result of this paper is the following theorem.

Theorem: A layout for a solvable RRP can be constructed in time O(NlogN). Moreover, the layout constructed contains only O(N) knock-knees.

In this paper, we do not adress the question of layer assignment to layouts. However, we want to mention that Brady-Brown have shown recently that every layout can be wired using only 4 layers of interconnect. Moreover, it can be shown that the number of contact cuts required for a 4 layer wiring is at most proportional to the number of knock-knees. Thus we have the following corollary.

Corollary: Every solvable RRP can be wired in four layers using only O(N) contact cuts.

The structure of this paper is as follows. The algorithm, the proof of correctness,and an analysis of the number of knock-knees generated are given in Section III. Section IV describes an implementation and contains the proof of the O(NlogN) running time. Section II develops some technical preliminaries.

II. PRELIMINARIES

In this section we present some simple observations which will be useful in the sequel.

Consider any RRP P. A column (row) of P is empty if it contains no terminals. Suppose that P has more than N+2 empty columns. Then no h-cut of P is saturated because its density cannot exceed the number of nets. Let $(c_1,c_1+1),\ldots,(c_k,c_k+1)$ be the saturated v-cuts. Obtain RRP P' from P by deleting empty columns such that:

1) no row becomes saturated;

2) the number of deleted empty columns between any two saturated h-cuts is even.

It is therefore easy to see that the extended row and column criteria hold for P' iff they hold for P. Hence P' has a solution if P has. Moreover there are at most N+2 empty columns in P' and a solution of P' trivially extends to a solution of P. A similar reduction works if there are more than N+2 empty rows. Thus we can assume w.l.o.g that $n = O(N)$, $m = O(N)$.

An RRP P is <u>standard</u> if every node has even extended degree. Our algorithm only applies to standard RRPs, although this restriction can be overcome by making the algorithm more complex. For every RRP P, an equivalent standard RRP P' is easily constructed in time O(N) as shown by Frank. Indeed, suppose that there are h saturated h-cuts and k saturated v-cuts. They divide the rectangle into (h+1)(h+1) regions $T_1,T_2,\ldots$. Each region has even parity (a proof can be found in [Frank]) and hence contains an even number of nodes of odd extended degree. Let $v_1,v_2,\ldots,v_{2r}$ be the nodes of odd degree in set T_i as they appear on the boundary of rectangle R in clockwise order. To obtain P'

we introduce the additional nets $(v_1,v_2),\ldots,(v_{2r-1},v_{2r})$ for every T_i. In [Frank] it is shown that the extended row and column criteria hold for P' iff they hold for P. Moreover, the parity density of any cut of P' is just its density, and we obtain the following version of the revised row and column criteria

"A standard RRP is solvable iff for any v-cut and h-cut the density never exceeds the capacity".

Also the number of nets of P' is O(N) since the length of the boundary of P can be assumed to be O(N) by the preceding remark. From now on we will assume that all RRPs are standard. A net is trivial if its terminals are on opposite sides of the rectangle and lie in the same row or column. Trivial nets can always be routed as straight lines without affecting solvability. This can be seen as follows. Suppose net N_i "runs" vertically in column c. Removal of column c and row N_i changes a standard RRP into a standard RRP. If c=1 then we also have to move all terminals from column 1 to column 2. A similar remark holds for c=n. Moreover it does not change the capacity and density of any v-cut, and decreases the capacity and density of any h-cut by one.

III. THE ALGORITHM

In this section we will describe an algorithm which solves any standard RRP in time O(NlogN) where N is the number of nets.

The algorithm is row-oriented. It processes row after row starting at the top row of the rectangle. Each row is processed in two major stages. The first stage lays out highly structured sets of nets, called "runs" (to be defined later). At the end of this stage, the plane domain still to be processed appears as a "toothy" rectangle, referred to as "near-rectangle". The second stage completes processing by producing the layout in the interior of the teeth. The situation is pictorially illustrated in Figure 4.

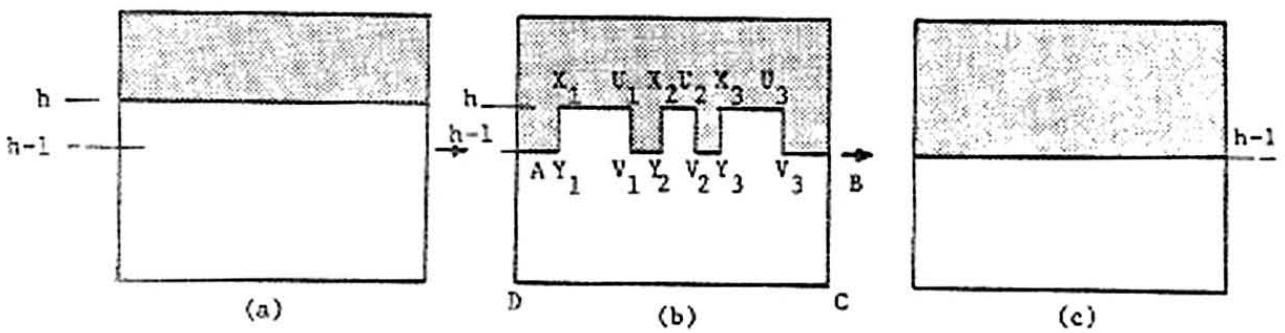


Figure 4. Processing of a row (row h): (a) before processing; (b) after the first stage; (c) after the second stage.

The coordinates of the points on the boundary of the near-rectangle in Figure 4b are A = (1,h-1), B = (n,h-1), C = (n,1), D = (1,1), $X_i = (c_i,h)$, $Y_i = (c_i,h-1)$, $U_i = (d_i,h)$, $V_i = (d_i,h-1)$ where $1 \leq c_1 < d_1 < c_2 < d_2 < \ldots < d_s \leq n$. Points (x,h), $c_i \leq x \leq d_i$ for some i, are called a segment.

A standard near-rectangle routing problem (SNRRP) consists of a number of non-trivial two-terminal nets such that each convex corner of the near-rectangle is the terminal of either zero or two nets, each concave corner is the terminal of no net, and each boundary point which is not a corner is the terminal of exactly one net. In particular, if $c_1 = 1$, and hence $A \equiv Y_1$, then A is the terminal of exactly one net. If a vertex is not a terminal of any net then the vertex is said to be exposed.

As always, the capacity of a cut is the number of edges in the cut, and the density of a cut is the number of nets which have a terminal on both sides of the cut. We will consider vertical (v-cuts) and horizontal cuts (h-cuts). The v-cuts are (a,a+1) where $1 \leq a < n$ and the h-cuts are (b,b+1), $1 \leq b \leq h-2$ and (h,h-1), restricted to the segments X_iU_i, $1 \leq i \leq s$. The cut (h,h-1) in the segment X_iU_i (referred to as facing X_iU_i) consists of the set of vertical edges emanating from X_iU_i. Thus, its capacity is $d_i - c_i + 1$.

The following two simple facts about densities are useful.

Lemma 1: a) If either U_i or X_i is exposed then the h-cut (h-1,h) facing X_iU_i is not oversaturated.

b) Let $c_i \leq a < d_i$ for some i. If dens(a,a+1) < h then dens(a,a+1) $\leq$ h-2. (U_i and X_i are on row h.)

Proof: a) If either U_i or X_i is exposed then the number of terminals on segment X_iU_i is at most $d_i - c_i + 1$ which is also the capacity of the cut.

b) Since we deal with a standard problem every node has even extended degree. Note that this is true for nodes on the boundary as well as for nodes in the inside of the near-rectangle. Hence h+dens(a,a+1) is even, since h is the number of edges and dens(a,a+1) is the number of nets which go across v-cut (a,a+1). Thus dens(a,a+1) < h implies dens(a,a+1) $\leq$ h-2.

A SNRRP is valid if there is no v-cut and no h-cut whose density exceeds its capacity. We will show that every valid SNRRP has a solution.

We divide the set of boundary points of a near-rectangle into 2 disjoint sets. TOP consists of the set of boundary points in rows h-1 and h, and NONTOP consists of the remaining boundary points. Set TOP-NONTOP consists of all nets which have one terminal in TOP and one terminal in NONTOP. A net $N_1 \in$ TOP-NONTOP is also called a TOP-NONTOP net. Sets TOP-TOP, NONTOP-NONTOP are defined similarly. A horizontal net is a TOP-NONTOP net which has neither point A or B as a terminal. A TOP-NONTOP net is right-going (left-going) if its terminal in NONTOP is to the right (left) of its terminal in TOP.

As mentioned above the algorithm only deals with standard problems. In order to keep up the fiction that the problem dealt with is standard, the algorithm will introduce additional nets during its execution. These nets are called fictitious nets. Fictitious nets are always horizontal and they do not overlap. More precisely, there is an even number $x_1, x_2, \ldots, x_{2\ell}$ (ordered from left to right) of points in TOP such that the fictitious nets are the ℓ nets (x_{2i-1}, x_{2i}), $1 \leq i \leq \ell$. We will represent the fictitious nets by the ordered sequence $x_1, x_2, \ldots, x_{2\ell}$ of their terminals. Fictitious nets are an elegant device for dealing with the extended row and column criteria. Fictitious nets are TOP-TOP nets and are usually treated like other top-top nets. In particular, we will often draw wires for fictitious nets. These wires are fictitious and record the fact that certain edges of the grid are not used in the layout.

We can now start to describe the algorithm. For the analysis of the algorithm we count elementary steps. An elementary step consists of drawing a horizontal wire segment of arbitrary length and extending by one unit all the vertical wires intersected by it. We will see that whenever we draw a

horizontal wire segment there will be a knock-knee on both ends. Thus the number of elementary steps also yields a bound on the number of knock-knees. In order to count elementary steps we introduce the concept of token. A token represents the ability to pay for one elementary step. During the execution of the algorithm, tokens will be transferred from nets and segments to (modified) nets and segments. Initially, we have the following token allocation:

a_{NN} = 12 tokens on every NONTOP-NONTOP net

a_{TN} = 7 tokens on every TOP-NONTOP net

a_{TT} = 1 token on every TOP-TOP (or horizontal) net.

The description of the algorithm is quite lengthy and requires many case distinctions. In order to aid the reader, the description is given in the following format. In each case we first give the actions to be taken. We then argue correctness, i.e. we show that the actions transform a valid problem into a valid problem. Then we do the accounting, i.e. we show that all elementary steps can be paid for. Statements in each of these three categories are presented in distinct typographical ways.[2]

Central to our technique is the notion of "run", which we now present.

A right run relative to segment XU, with X = (c,h) and U = (d,h) (c < d) is a maximal sequence $(x_1,\ldots,x_{k+1})$ of columns and an associated string of nets $N_1 \ldots N_k$ ($k \geq 1$) such that $N_1,\ldots,N_k$ are right-going TOP-NONTOP nets and N_k is either a right-going TOP-NONTOP net or a horizontal net. In addition:

1. N_i has its left terminal in column x_i and its right terminal in column x_{i+1} ($i = 1,\ldots,k$);
2. x_{k+1} = min (d, column of right terminal of N_k).

---

[2] Indeed, we suggest that the three sections -- actions, correctness, and accounting -- be approached separately on a first reading of this paper.

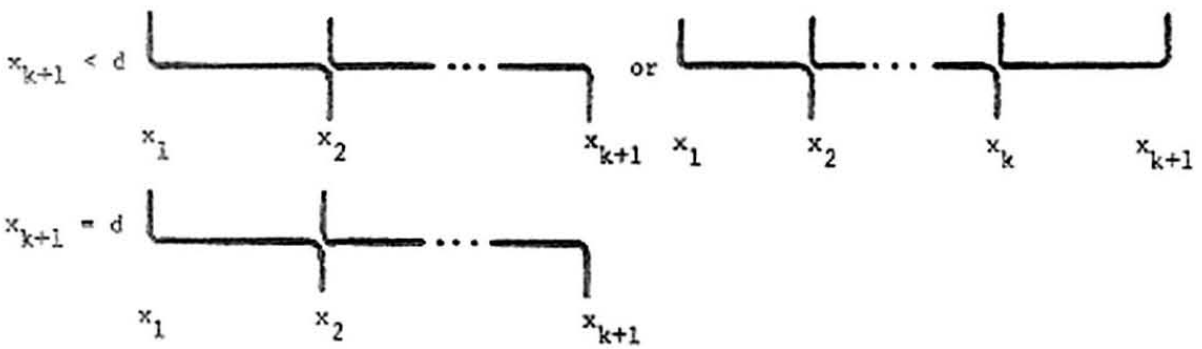


Figure 5. Illustrations of right runs.

By <u>layout of a horizontal net</u> $N_i = ((f,h),(g,h))$ $(f < h)$ with respect to x_{i+1} we mean the operation of drawing a horizontal wire from (f,h) to $(\min(g,x_{i+1}),h)$ and on to $(x_{i+1},h-1)$ if $x_{i+1} < g$, and of extending all other nets with terminals in set $\{(x,h): f \leq x < \min(g,x_{i+1})\}$ by one vertical edge (so that each terminal is moved from (x,h) to $(x,h-1)$). Also, if $g \leq x_{i+1}$ then net N_i is deleted from the problem, and if $g > x_{i+1}$ then net N_i is replaced by net $N_i' = ((x_{i+1},h-1),(g,h))$. (see Figure 6a). Analogously, the <u>layout of a right-going net</u> $N_i = ((f,h),(g,h))$ with respect to x_{i+1} means the operation of drawing for N_i a horizontal wire from (f,h) to $(\min(g,x_{i+1}),h)$ and a vertical wire from $(\min(g,x_{i+1}),h)$ to $(\min(g,x_{i+1},h-1)$ (thereby moving the terminal from (f,h) to $(\min(g,x_{i+1}),h-1)$) and of extending all other nets with terminals in $\{(x,h): f \leq x < \min(g,x_{i+1})\}$ by one vertical edge (see Figure 6b). Note that net N_i becomes a trivial vertical net if $g \leq x_{i+1}$. The layout of a left-going net is similarly defined. The layout of a run with sequence $(x_1,\ldots,x_{k+1})$ of columns and associated nets $N_1,\ldots,N_k$ is the layout of net N_i with respect to x_{i+1} for $1 \leq i \leq k$. Note that laying out a run turns right-going nets $N_1,\ldots,N_{k-1}$ into trivial nets.

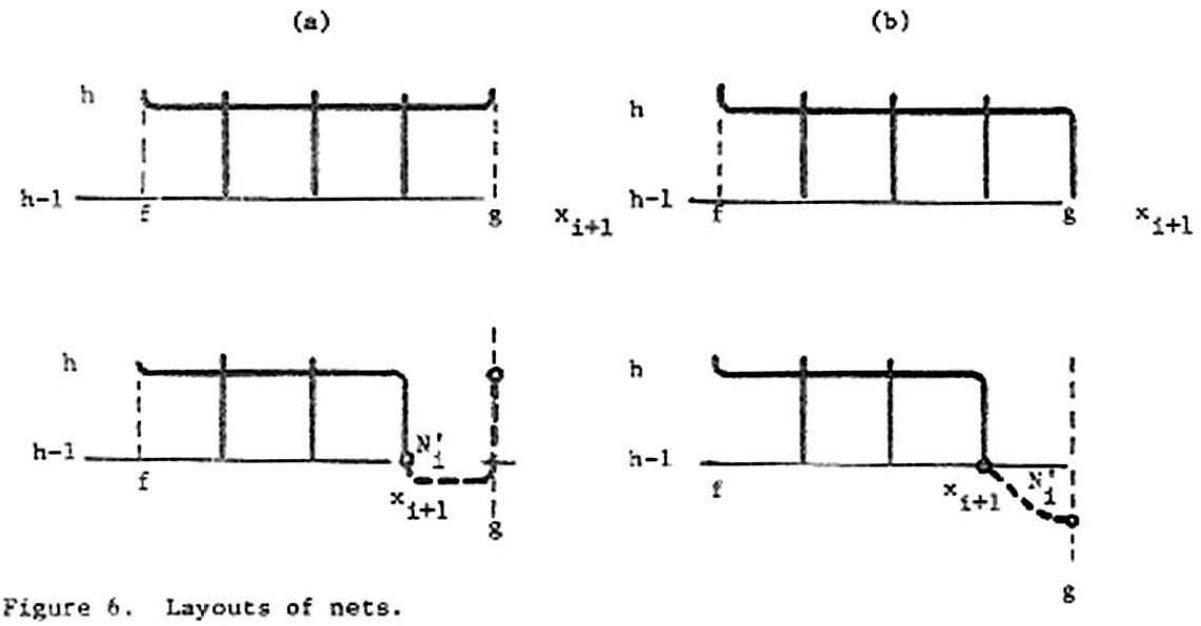


Figure 6. Layouts of nets.

Rectangle routing problems within a rectangle of height one, i.e. h = 1, are trivial. All nets are horizontal and they do not overlap. Thus we can do the layout by drawing one horizontal wire per net. The cost is covered by the $s_{TT}(-1)$ tokens on each TOP-TOP used. So, let us assume that $h \geq 2$. As mentioned earlier, we process row h in two major phases -- run layout and clean-up -- preceded by a simple initialization step, whose objective is to ensure that each segment contains at least one exposed corner. Thus we have

Step 0: Initialization (refer to Figure 7).

```
procedure INITIALIZE (h)
begin move point A ≡ (1,h-1) and B ≡ (n,h-1) from NONTOP to TOP;
      if either X or U is exposed then set c:=1 and d:=n
      else begin N*=((f,h),(g,h)):=a horizontal net with maximal g;
                 lay out N* and delete points (x,h), f < x < g from
                 rectangle, and also (f,h) if f=1 and (g,h) if g=n.
           end
end
```

The results of Step 0 are illustrated in Figure 7. Here, we use the graphics ⊥ or ⊤ to denote exposed nodes. Step 0 is clearly void if either X or U is already exposed (Figure 7a, b, and c); otherwise, two segments X_1U_1 and X_2U_2 are created (Figure 7d). Note that there are no terminals of horizontal nets in X_2U_2.

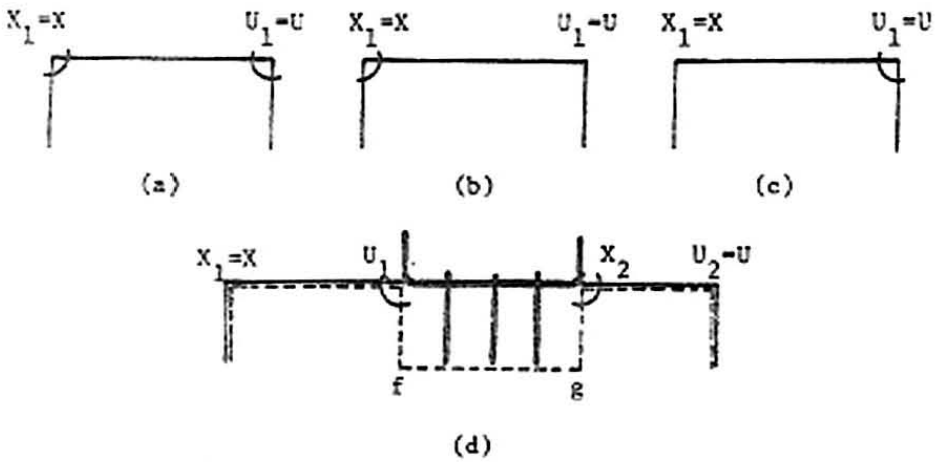


Figure 7. Near-rectangles obtained after the initialization step: in each segment at least one corner is exposed.

Lemma 2: There is at least one net with both terminals in row h or either X or U is exposed.

Proof: If neither X nor U is exposed, each contributes two terminals. Thus there are n+2 terminals in row h and, since h-cut (h-1,h) is not oversaturated, there is at least one net with both terminals in row h. □

Lemma 3: The new problem is valid.

Proof: The lemma is trivially true if only one segment is produced. If two segments are produced, then no h-cut is oversaturated. Also, we reduced the capacity and density of every v-cut (b,b+1),for $f \leq b < g$, by one. Thus no v-cut is oversaturated. □

In the new problem points both $U_1 = (f,h)$ and $X_2 = (g,h)$ are exposed if they exist. Moreover, no node (x,h), $x \geq g$, is a terminal of a horizontal net.

Accounting: Since node A ≡ (1,h-1) moved from NONTOP to TOP, either one net moves from NONTOP-NONTOP to TOP-NONTOP (and hence frees $a_{NN} - a_{TN} = 5$ tokens) or one net moves from TOP-NONTOP to TOP-TOP (and hence frees $a_{TN} - a_{TT} = 6$ tokens). Since node B ≡ (n,h-1) also moves from NONTOP to TOP we conclude that at least $2 \min(a_{NN} - a_{TN}, a_{TN} - a_{TT}) = 10$ tokens become available.

Let $a_8 = 4$. If either X or U is exposed, then we have drawn no horizontal wire. We place $a_8 + 3 = 7$ tokens on the single segment extending from column 1 to n. If neither X nor U is exposed then we have drawn one horizontal wire for the cost of one token. The layout of this net relsases $a_{TT}=1$ token, so 10 + 1 = 11 tokens are available. Also we place $3 + a_8 = 7$ tokens on segment X_1U_1 and 3 tokens on segment X_2U_2. Thus 10 tokens are needed, which contrasts favorably with the 11 tokens available. Thus in either case the cost is more than covered.

Step 1: Run Layout

This step involves two symmetric scans of TOP. We have at this point two segments X_1U_1 and X_2U_2 (the latter possibly void). If X_2U_2 is void we may assume w.l.o.g. that U_1 is exposed (otherwise, we consider the mirror image of the rectangle). The first scan is a left-to-right scan of segment X_1U_1, i.e. a call of procedure RIGHT-RUN-LAYOUT (1, min(d,N)), where $U_1 = (h,d)$ if $U_1 \neq C$.

Execution of this procedure on the segment generates a sequence of segments $X_1'U_1', \ldots, X_k'U_k'$ ($k \geq 0$) such that

a) X_i', U_i', $1 \leq i \leq k$ are all exposed;

b) segments $X_i'U_i'$, $1 \leq i \leq k$, hold $a_s + 1$ tokens each, where $a_s = 4$;

c) no (x,h), $c_i < x < d_i$ and $1 \leq i \leq k$, is the terminal of a right-going TOP-NONTOP net. Here $X_i = (c_i,h)$ and $U_i = (d_i,h)$.

The following program makes use of the "extension of a left-going net", by which we mean the following operation: given a left-going net $N^* = ((f,h),(g,\ell))$ with $g \leq f$, to extend N^* to $e > f$ means to draw for N^* a horizontal wire from (f,h) to (e,h) and a vertical wire from (e,h) to $(e,h-1)$, thereby moving the terminal from (f,h) to $(e,h-1)$. Also all other nets with a terminal in set $\{(x,h): f \leq x < e\}$ are extended by one vertical edge.

```
      procedure RIGHT-RUN-LAYOUT (c,d)
begin u: = c; EXT:=Λ
      if (c,h) is not exposed then
           EXT: = one of the nets starting in (c,h); if possible, a
                  non-vertical net is chosen here: (*this is treated
                  conventionally as an extension*)
      while u < d do
  begin if EXT = Λ then
             p:=f  is minimal such that (f,h) is the terminal of a
             right-going TOP-NONTOP net
        else p:=f is  minimal such that (f,h) is the terminal of a
             right-going TOP-NONTOP net or the left-terminal of a TOP-TOP net

        e:=min (p,d);
        if EXT ≠ Λ then
       begin extend EXT to e; EXT: = Λ end;
        lay out right run starting at e (if it exists);
        g ← column destination of run;
        if (g,h) is the right terminal of a TOP-TOP net N
           or the terminal of a left-going TOP-NONTOP net N then
             EXT: = N ;
        u: = g
  end
end
```

Example: c = 2, d = 7, (c,h) is exposed. In the first iteration of the loop,

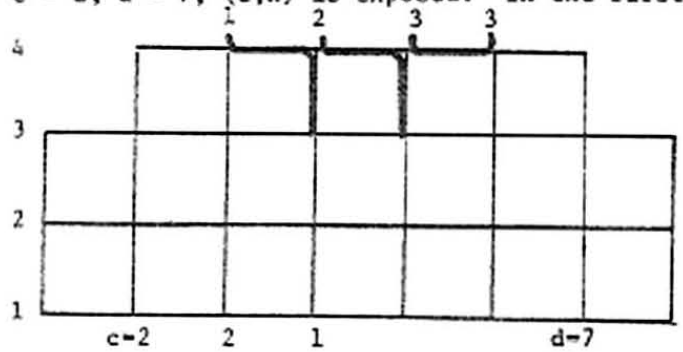


Figure 8. Illustration of the action of RIGHT-RUN-LAYOUT.

we have e = 3. The run starting in column 3 consists of net 1 and extends to column 4. Since (4,h) is the right terminal of a left-going net, we set EXT:=Net 2. In the next iteration we have e = 5,... .

Lemma 4: Application of procedure RIGHT-RUN-LAYOUT to segment X_1U_1 generates a valid problem P'.

Proof: Consider a v-cut (a,a+1). If the density of v-cut (a,a+1) has not changed when passing to P', i.e. the cut lies in one of the newly generated segments $X_i'U_i'$, then there is nothing to show. So assume otherwise. Then the capacity of the cut was decreased by one, and also a net was routed across v-cut (a,a+1). If this net was part of a right-going run then the density was also decreased by one and we are done.

The final case to consider is when a net extension was routed across the v-cut and hence the density was increased by one. Suppose that the left-going net extension starts in column x and is routed across the v-cut. Then all nodes (y,h), $x < y \leq a$, are either terminals of left-going TOP-NONTOP nets or the right terminal of a TOP-TOP net. Thus

$$dens(a,a+1) \leq dens(a-1,a) \leq \ldots \leq dens(x,x+1).$$

In view of Lemma 1 it suffices to show dens(x,x+1) < h. This can be seen as follows. If x = 1, and hence (1,h) is the terminal of two vertical nets, we have $dens(x,x+1) \leq h-2$, since there are at most h+2 terminals of column 1, at least 4 of which belong to vertical nets. If x > 1, then $dens(x,x+1) \leq dens(x-1,x) - 2 \leq h-2$, since (x,1) is the right terminal of a top-nontop net and (x,h) is the terminal of a left-going net. □

Accounting: Let $z > 0$ be the number of executions of the body of the *while*-loop, let z_1 be the number of executions which are started with $EXT \neq \Lambda$, and let $z_2 = z - z_1$. Then exactly z_2 segments are created. Let x_{TN} be the number of TOP-NONTOP nets that are laid out completely and let x_{TT} be the number of TOP-TOP nets that are laid out completely. Then the total number of tokens needed is

x_{TN}	for the x_{TN} completed TOP-NONTOP nets
$+ x_{TT}$	for the x_{TT} completed TOP-TOP nets
$+ z_2 \cdot (a_s+1)$	for the z_2 segments created
$+ z_1$	for the z_1 extensions
$+ 1$	for the TOP-NONTOP or TOP-TOP net whose layout is attempted but not completed because it extends beyond column d
$= x_{TN} + x_{TT} + z_2 \cdot a_s + z + 1.$	

The total number of tokens available is:

$x_{TN} \cdot a_{TN}$	from the x_{TN} completed TOP-NONTOP nets
$x_{TT} \cdot a_{TT}$	from the x_{TT} completed TOP-TOP nets
$a_s + 3$	from the segment X_1U_1
$= x_{TN} \cdot a_{TN} + x_{TT} \cdot a_{TT} + a_s + 3$	

Finally observe that $x_{TN} \geq z_1 - 1$, since all but the first extension is preceded by a completed TOP-NONTOP net, and that $x_{TN} \geq z_2 - 1$, since all but the first segment was created by the layout of a run which starts with a TOP-NONTOP net. Thus

$$x_{TN} + x_{TT} + z_2 \cdot a_s + 2 + 1 \leq x_{TN}(1+a_s+2) + x_{TT} + 3 + a_s$$
$$\leq x_{TN} \cdot a_{TN} + x_{TT} \cdot a_{TT} + 3 + a_s$$

since $a_{TN} = 7$ and $a_{TT} = 1$.

This finishes the description of the left-to-right scan. Its effects are illustrated in Figure 9, in the hypothesis that both X and U are initially not exposed.

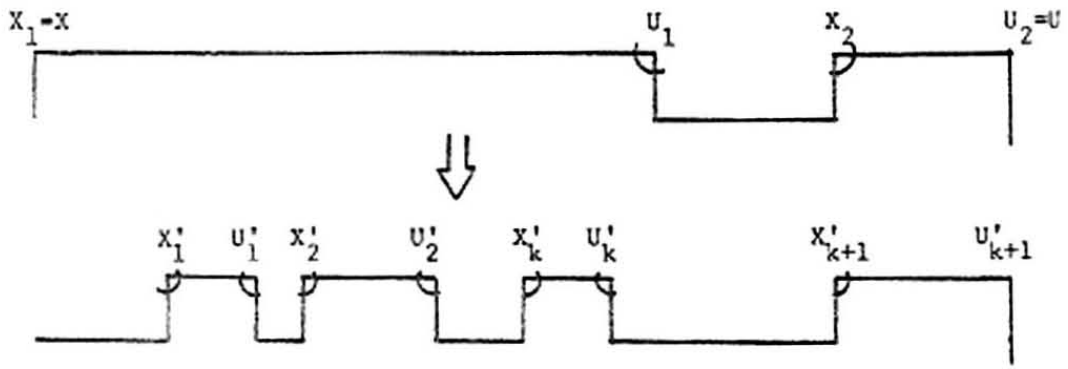


Figure 9. Near-rectangle after the left-to-right scan.

It is immediately verified that the segments $X'_1U'_1,\ldots,X'_kU'_k$ have the properties outlined earlier; in addition there may be a special segment $X'_{k+1}U'_{k+1}$, which was created earlier by the initialization step.

We are now ready to undertake the symmetric right-to-left scan of Step 1, which performs the following action:

for i = k+1 down to 1 do LEFT-RUN-LAYOUT (c_i,d_i)

where:

procedure LEFT-RUN-LAYOUT (c,d) is symmetric of procedure RIGHT-RUN-LAYOUT (c,d) described earlier.

It is clear from the arguments given in connection with RIGHT-RUN-LAYOUT that this scan produces a valid problem. Also, the left-to-right scan shrinks segment $X'_{k+1}U'_{k+1}$ to a void segment and divides each of the segments $X'_iU'_i$,

$1 \leq i \leq \ell$, into a number (≥ 1) of segments. Each newly created segment X"U" has the following properties:

a) X" and U" are both exposed and no node in the segment is the terminal of a TOP-NONTOP net;

b) there are a_s tokens on the segment.

Accounting: *We use the same notation as in the accounting for procedure RIGHT-RUN-LAYOUT.*

Let us concentrate first on segment $X'_{k+1}U'_{k+1}$. This segment exists only if $U'_{k+1} = U$ is not exposed, i.e. if node (h,n) is the terminal of two nets. Since no node (x,h) with $c_{k+1} < x \leq d_{k+1} = n$ is terminal of a horizontal net, we have $s_1 = s$ and hence $s_2 = 0$, i.e. no new segment is created. Hence the total number of tokens needed is at most $x_{TN} + x_{TT} + s_1 + 1$. Also the total number of tokens available is $x_{TN} \cdot a_{TN} + x_{TT} \cdot a_{TT} + 2$ since there are two tokens on segment $X'_{k+1}U'_{k+1}$ (placed by the initialization step). Since $a_{TN} = 2$, $a_{TT} = 1$ and $x_{TN} \geq s_1 - 1$, the cost is amply covered.

It remains to consider segments $X'_iU'_i$, $1 \leq i \leq k$. Since both X'_i and U'_i are exposed, we have $s_2 \geq 1$ and $x_{TN} \geq s_1$. Also, by the accounting for procedure RIGHT-RUN-LAYOUT, the total cost is $x_{TN} + x_{TT} + s_2 \cdot a_s + s_1 + 1$ (note that only a_s tokens are now placed on each newly generated segment) and the total number of tokens available is $x_{TN}a_{TN} + x_{TT}a_{TT} + a_s + 1$ (note that $a_s + 1$ tokens were available on segment $X'_iU'_i$). Thus cost is covered, since $x_{TN} \geq s_1 - 1$, $a_{TN} = 2$, $a_{TT} = 1$, $a_s = 4$. With an endowment of at least a_s tokens per segment we can now begin the clean-up step.

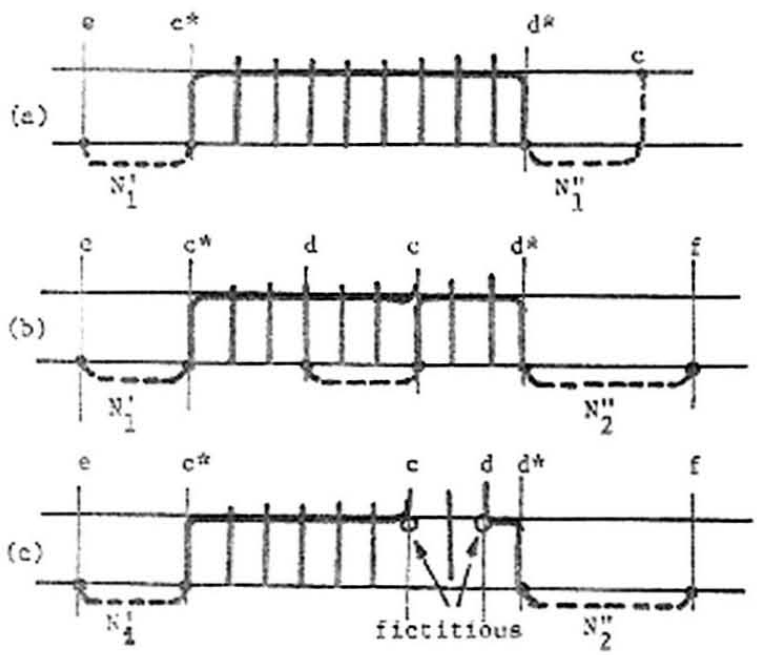
Step 3: Clean-up

As mentioned earlier, at this point we have a sequence of segments $X''_1U''_1, X''_2U''_2, \ldots, X''_sU''_s$ with Properties 1), 2), and 3) described above. To this collection we apply the following algorithm:

```
    for i = 1 to s do CLEAN-UP (c''_i, d''_i)
  where
  procedure CLEAN-UP (c*,d*)
1.  N_1=((e,h-1),(c,h-x)):= TOP-TOP   net with maximal c, e < c* < c, x ∈ {0,1};
2.  N_2=((d,h-y),(f,h-v)):= TOP-TOP   net with minimal d, d < d* < f, v,y ∈{0,1};
3.  if N_1 = Λ then c: = c*;
4.  if N_2 = Λ then d: = d*;
5.  if (c > d) or [(c < d) and for every v-cut (g,g+1), c ≤ g < d, (g,g+1)
      is either nonsaturated or saturated by fictitious net]
6.  then if c > d* then
7.           begin     create nets N'_1=((e,h-1),(c*,h-1)) and N''=((d*,h-1),(d,h-x));
8.  see                draw wire (c*,h-1) → (c*,h) → (d*,h) → (d*,h-1) for net N_1;
9.  Figure 10a         draw vertical wire segments for all other nets with
                                   terminal (z,h), c* < z < d*;
                       end
10. else (/c < d*/) begin create net N'_1=((e,h-1),(c*,h-1));
11.                 draw wire (c*,h-1) → (c*,h) → (c,h) for net N_1;
12.                 if c > d then
13                      begin create nets N'_2=((d,h-y),(c,h-1)) and
                                         N''_2 =((d*,h-1),(f,h-v));
        see
14. Figure 10b          draw wire (c,h-1) → (c,h) → (d*,h) → (d*,h-1) for net N_2;
15.                     draw vertical wire segments for all other nets
                            with terminal (z,h), c* < z < d*;
                        end
16.                 else begin create net N''_2=((d*,h-1)(f,h-v));
                             draw wire (d,h) → (d*,h) → (d*,h-1) for net N_2;
        see
17. Figure 10c               draw vertical wire segments for all other nets
                                 with terminal (z,h), c* < z < d*;
18.                          add c and d to the set of fictitious points;
                             end
    else FULL (c*,d*,c,d)(/to be described later/)
```

Remark: Note that either net N_1 or net N_2 or both might be fictitious. If net N_1 (N_2) is fictitious then the wire drawn for that net is fictitious, i.e. we explicitly declare all the edges on that wire as unused. Also, the newly created nets are fictitious if the original was. For example, if nets N_1 and N_2 are fictitious, i.e. points e, c, d, f are fictitious, and c < d, then points e, c*, c, d, d*, f will be fictitious after termination of CLEAN-UP and we will have drawn three fictitious wires.

The actions of CLEAN-UP described above are illustrated in Figures 10a, b, and c.



	Revenues	Expenditures
(a)	$a_s + a_{TT}$	$1 + 2a_{TT}$
(b)	$a_s + 2a_{TT}$	$2 + 3a_{TT}$
(c)	$a_s + 2a_{TT}$	$3 + 3a_{TT}$

Figure 10. Actions of CLEAN-UP and their accounting.

Lemma 9: The Problem P' obtained in this fashion is valid.

Proof: If c > d then we reduced the capacity and density of every v-cut (g,g+1) with $c^* \leq g < d^*$ by one. Thus validity is preserved.

So let us assume c < d. (Note that c = d is impossible.) For $c^* \leq g < c$ and $d \leq g < d^*$ we reduce capacity and density of v-cut (g,g+1). It remains

to consider g such that $c \leq g < d$. Then, by the condition in line 5 of CLEAN-UP, either v-cut (g,g+1) is not saturated and hence has density at most h-2 by Lemma 1, or v-cut (g,g+1) is saturated and there is a fictitious net going across v-cut (g,g+1). In the first case we are safe since when passing from P to P' we reduce the capacity by one and increase the density by one. Note that there is a fictitious net going across v-cut (g,g+1) in P'. In the second case we are safe since in P' there is no (!!) fictitious net going across v-cut (g,g+1). This follows from the fact that we added c and d to the set of fictitious points and $c \leq g < d$.

Accounting: For the case in Figure 10a, the expenditure is one token for the horizontal wire drawn and one token to place on each of the new TOP-TOP nets N_1' and N_1''. Available are $a_s = 4$ tokens on segment X'U' and one token from net N_1: this is more than adequate to cover expenditures.

For the case in Figure 10b (c > d), we have increased the number of TOP-TOP nets by one and have drawn two horizontal wires. Again, the revenue is $2a_{TT} + a_s$ and the expenditure is $3a_T + 2$, which is covered.

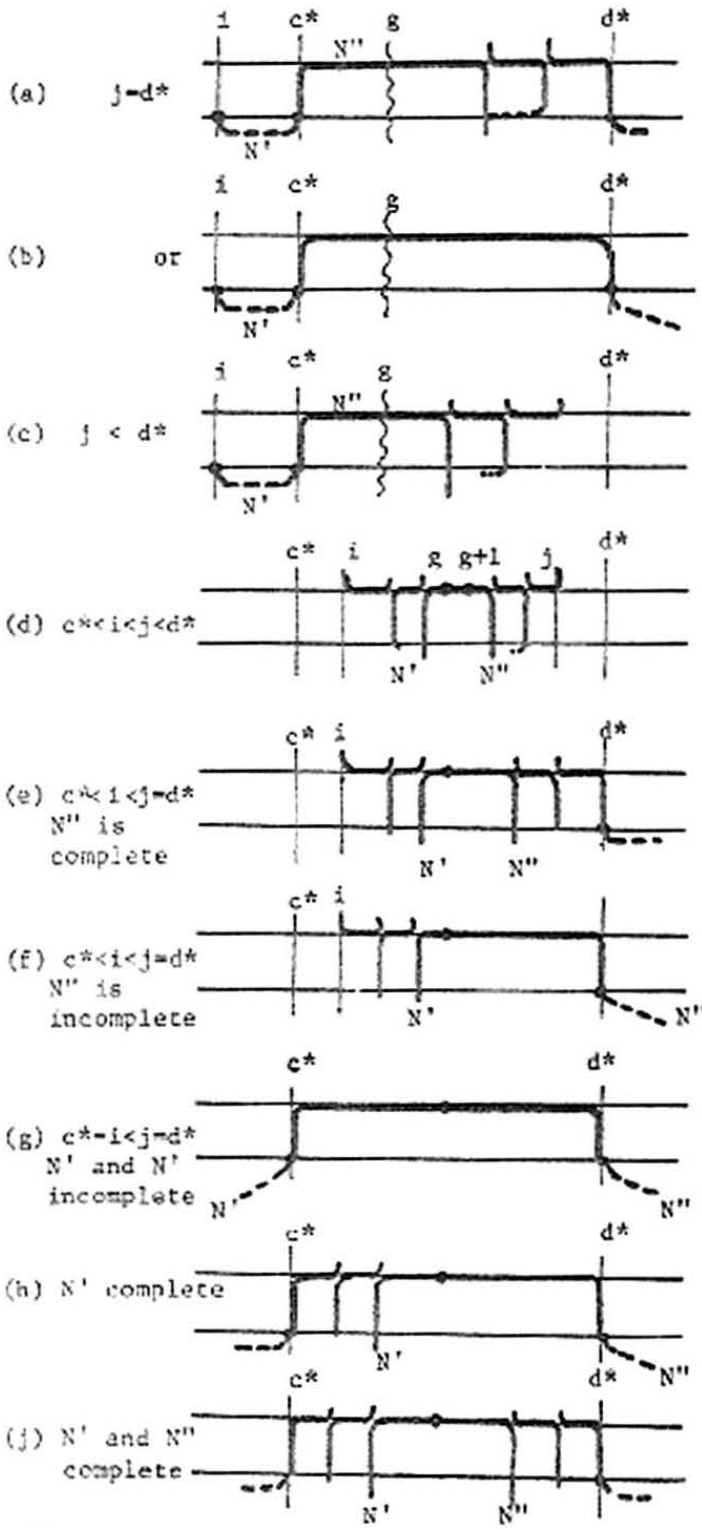
Finally, for the case in Figure 10c (c < d), we have increased the numbers of TOP-TOP nets by one (there is one additional fictitious net) and we have drawn at most three horizontal wires at least one of which is fictitious. Thus, the revenue is $2a_{TT} + a_s$ (from the two processed TOP-TOP nets and the segment), while the expenditure is at most $3a_{TT} + 3$. Since $a_{TT} = 1$ and $a_s = 4$, the cost is always covered.

We are now ready to consider the last case, to which we apply procedure PULL. This case occurs when c < d and there is a saturated v-cut (g,g+1), $c < g < d$, with no fictitious nets across it. In this case, it is mandatory

to use track h in the interval [g,g+1], and the only way to achieve this is by "pulling up" to track h a net not contributing to the vertical density at h. The action is described by the following procedure:

```
        procedure PULL(c*,d*,c,d)
    g:=minimal p, c ≤ p < d, so that v-cut (p,p+1) is saturated with no
       fictitious nets;
    r:=maximal q, so that row q contains the terminal of a TOP-NONTOP or
       NONTOP-NONTOP net across (g,g+1);
    if r=h-1 then
      begin N:=net with terminal in (i,h-1) which goes across cut(g,g+1)
           if i < c* then (/N=((i,h-1),t/)
             begin create nets N'=((i,h-1),(c*,h-1)) and N"=((c*,h),t);
                   draw wire (c*,h) → (c*,h-1) for net N;
                   RIGHT-RUN-LAYOUT (c*,d*)
                   j:=rightmost column reached by layout;
             end
           else (/N-(t,(i,h-1):symmetric of case above/)
      end
 else (r < h-1)
    begin N:=((s,y),(t,z)):=net with r=max(y,z);
          create nets N'=((s,y),(g,h)) and N"=((g+1,h),(t,z));
          draw wire (g,h) → (g+1,h) for net N;
          RIGHT-RUN-LAYOUT (g+1,d*);
          j:=rightmost column reached by layout;
          if j < d* then CLEAN-UP (j,d*);
          LEFT-RUN-LAYOUT (c*,g);
          i:=leftmost column reached by layout;
          if i > c* then CLEAN-UP (c*,i)
    end
```

The actions of PULL are illustrated in Figure 11.



	Revenues	Expenditures
(a) j=d*	$a_s + a_{TN}$	$3 + a_{TT}$
(b) or	a_s	$1 + a_{TT}$
(c) j < d*	$a_s + a_{TN} + a_{TT}$	$3 + a_{TT} + a_s$
(d) c*<i<j<d*	$a_{NN} + 2a_{TT}$	$5 + a_s$
(e) c*<i<j=d* N" is complete	$a_{NN} + a_{TT}$	5
(f) c*<i<j=d* N" is incomplete	$a_{NN} + a_{TT}$	$3 + a_{TN}$
(g) c*=i<j=d* N' and N' incomplete	$a_{NN} + a_s$	$1 + 2a_N$
(h) N' complete	$a_{NN} + a_s$	$3 + a_{TN}$
(j) N' and N" complete	$a_{NN} + a_s$	5

Figure 11. Actions of PULL and their accounting.

Lemma 10: Parameter r (used by CLEAN-UP) is well defined.

Proof: If c = g then column g must be density preserving or increasing; if c < g then column g must be density increasing by the minimality of g. In either case we infer that (g,1) is the terminal of a net which also has a terminal in columns g+1,...,n.

Lemma 11: The new problem P' is valid.

Proof: We consider first the case r = h-1, i.e. N is a TOP-NONTOP net, whence we have either i < c* or d* < j (if c* < i < d*, the previous scans would have processed it). Thus, in this case, the validity of P' is obvious (since the segment is eliminated).

In the other case ($r \leq h-2$), the crucial observation is that dens(b,b+1) $\leq$ n-2 for every h-cut (b,b+1) with $r \leq b \leq h-2$. We prove dens(b,b+1) $\leq$ n-2 by case distinction on b.

Case 1: $b \geq h - (d^* - c^* + 1)/2$.

There are at most n+2 - (d*-c*+1) terminals in TOP outside of segment X'U' in rows b+1,...,h-2 and exactly 2(h-2-b) terminals in rows b+1,...,h-2. Since all terminals in segment X'U' are terminals of horizontal nets we conclude

$$\begin{aligned} dens(b,b+1) &\leq n + 2 - (d^*-c^*+1) + 2(h-2-b) \\ &\leq n - 2 \end{aligned}$$

Case 2: $b < h - (d^*-c^*+1)/2$.

Let ℓ be the number of horizontal nets which go across cut (g,g+1). All such nets have both terminals on line segment X'U' since $c \leq g < d$. Since X' and U' are exposed we conclude $\ell \leq (d^*-c^*-1)/2$.

There are at most n+2b terminals in rows 1,...,b. We label these terminals "up" and "cross" as follows. A terminal is labelled up (cross) if it belongs to a net which goes across h-cut (b,b+1) (v-cut (g,g+1)). Then no terminal is

labelled up and cross since there are no nets in TOP-NONTOP which go across the cut (g,g+1) (recall that $n \leq h-2$) and since every net in NONTOP-NONTOP which goes across the cut has both terminals in rows 1,...,r by definition of r. Since v-cut (g,g+1) is saturated, exactly $2(h-\ell)$ terminals are labelled cross. thus

$$\begin{aligned} dens(b,b+1) &\leq n + 2b - 2(h-\ell) \\ &\leq n + 2(b-h+(d^*-c^*-1)/2) \\ &\leq n - 2 \end{aligned}$$

since $b < h - (d^*-c^*+1)/2$.

Accounting: The accounting is best done by referring to the case illustrated in Figure 11.

Consider first $r = h-1$ and $j = d^*$ (Figure 11a and b). Then we have increased the number of TOP-TOP nets by one (namely N') and either we have completed a TOP-NONTOP net (namely N") drawing at most 3 horizontal wires (Note that N' could end in a column which also contains the right terminal of a horizontal net (Figure 11a),) or we have not completed a TOP-NONTOP net drawing one horizontal wire (Figure 11b). In either case we can use the $a_s = 4$ tokens available on segment X'U' to cover the expenditures: indeed in the first case we have a revenue of $a_s + a_{TN} = 11$ tokens and need only $3 + a_{TT} = 4$ tokens; in the latter case we have $a_s = 4$ tokens available and need only $1 + a_{TT} = 2$ tokens.

Consider now that $j < d^*$ (Figure 11c). Then the layout of the row starting with net N" uses at most three horizontal wires and also deletes one TOP-TOP net. Thus altogether, the number of TOP-TOP nets is not increased. Thus total expenditure is at most 3 which is easily covered by the $a_{TN} = 7$ tokens on net N.

Consider next all the cases for which $r \leq h-2$. In all of these cases we reduce the number of NONTOP-NONTOP nets by one, which yields $a_{NN} = 12$ tokens.

Case 1. (Figure 11d): $c^* < i < j < d^*$. We lay out nets N' and N" completely and also complete two horizontal nets, which yield $2a_{TT}$ tokens. Expenditure is at most 5 for the horizontal wires drawn and $a_3 = 4$ for the newly created segment and is thus easily covered.

Case 2. $c^* < i < j = d^*$ or $c^* = i < j < d^*$. Then the number of segments does not change. Assume $c^* < i < j = d^*$ w.l.o.g. Then we completely lay out net N' and also one horizontal net. This yields a_{TT} tokens. Net N" is either laid out completely or it is not. In the first case (Figure 11e) (N" is laid out completely) we draw at most 5 horizontal wires and thus have an expenditure of 5. This is readily covered by the $a_{NN} + a_{TT} = 13$ tokens available.

In the latter case (Figure 11f) (N" is not laid out completely) we draw at most 3 horizontal wires and hence total expenditure is at most $3 + a_{TN} = 10$. Recall that we need to put a_{TN} tokens on N" in this case. This is covered by the $a_{NN} + a_{TT} = 13$ tokens available.

Case 3: $c^* = i < j = d^*$. Then the $a_8 = 4$ tokens on segment X'U' become available. If neither N' nor N" is laid out completely (Figure 11g), then we draw one wire and need to place $a_{TN} = 7$ tokens each on N' and N". Thus total expenditure is $2a_{TN} + 1 = 15$ which is covered by the $a_{NN} + a_8 = 16$ tokens available.

If exactly one of N' or N" is laid out completely (Figure 11h), then we draw at most 3 wires and need to place a_{TN} tokens on either N' or N". Thus total expenditure is $3 + a_{TN} = 10$ which is covered by the $a_{NN} + a_8 = 16$ tokens available.

Finally, if both H' and H" are laid out completely (Figure 11j), then we draw at most 5 wires. Thus total expenditure is 5 which is readily covered by the $a_{NH} + a_s = 16$ *tokens available.*

Thus in all cases the expenses are covered.

We summarize in

Theorem 1: Let P be any RRP with N nets. Then a solution (if there is one) with only O(N) knock-knees can be found using only O(N) elementary steps.

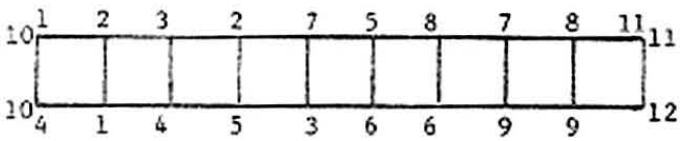
Proof: Recall that an elementary step corresponds to drawing a horizontal wire of arbitrary length. Knock-knees occur only at the end of horizontal wires. Thus it suffices to prove the bound on the number of elementary steps.

In Section II we described how any (solvable) RRP can be turned into a standard RRP with only O(N) nets.

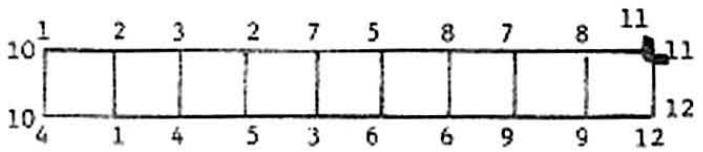
In a standard RRP with O(N) nets we supply only O(N) tokens initially. Since an elementary step costs one token the bound follows.

We conclude this section with an example illustrating the algorithm.

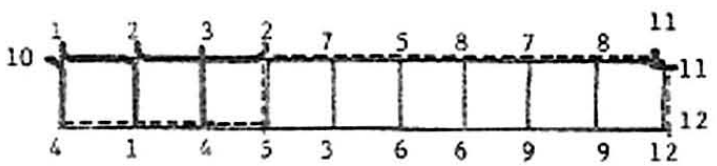
Example 1: The following RRP is given



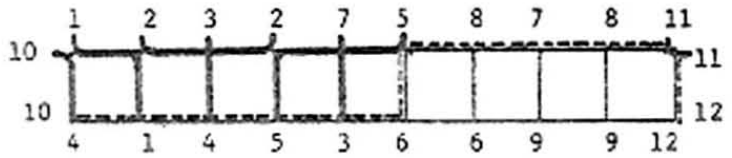
After the execution of INITIALIZE we have:



Next, we have RIGHT-RUN-LAYOUT, which yields

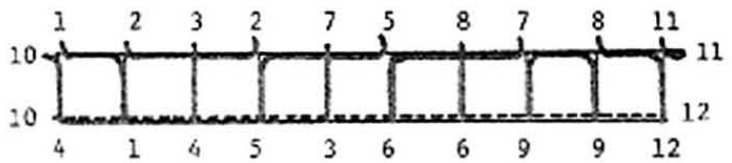


The execution of LEFT-RUN-LAYOUT gives

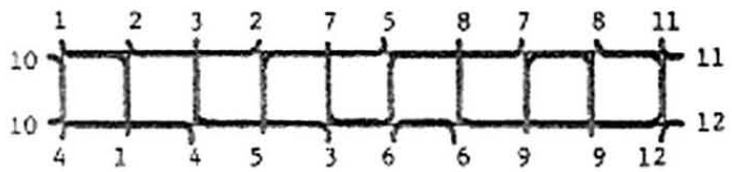


Finally, CLEAN-UP treats a single interval extending from column 6 to column 10.

We pull NONTOP-NONTOP net 9 and obtain

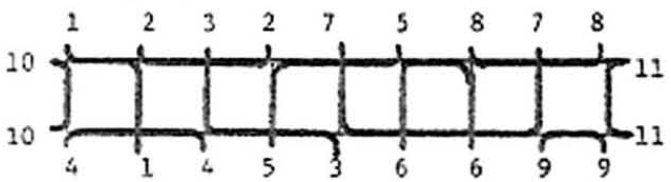


The layout is then completed by the straight-forward processing of row 1



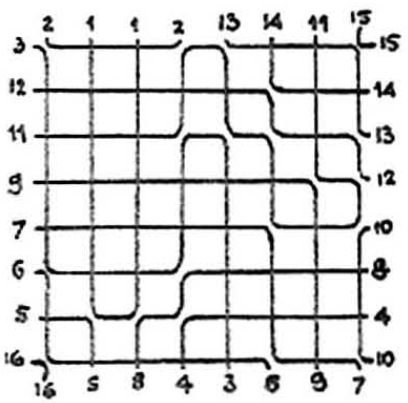
If the last column is suppressed, we leave it as an exercise to show that

the following layout is constructed



Example 2. A more substantial example of routing is illustrated in Figure 13.

Note that all columns but the last one are saturated.



IV. IMPLEMENTATION

The goal of this section is to show that each elementary step can be implemented in time O(logN). We first discuss data structures to be used for nets and then data structures to be used for columns and fictitious points.

The procedure discussed in Section III involves the searches of several collections of items, such as TOP-TOP nets, etc. Each such collection must be organized in a data structure that supports the required operations within the target O(logN) time bound. Before discussing these data structures, we recall, for the reader's convenience, the search operations postulated by the algorithm:

Search 1 Find TOP-TOP net $N^* = ((f,h)(g,h))$ with maximal g (proc. INITIALIZE).

Search 2. Given c, find minimal $f > c$ such that (f,h) is the left-terminal of a TOP-NONTOP net (in proc. RIGHT-RUN-LAYOUT; also, its symmetric operation for LEFT-RUN-LAYOUT).

Search 3. Given c, find minimal $f > c$ such that (f,h) is the left terminal of a TOP-TOP net (in proc. RIGHT-RUN-LAYOUT; also, its symmetric operation for LEFT-RUN-LAYOUT).

Search 4. Given c*, find a TOP-TOP net $N_1 = ((e,h-1),(c,h-x))$, $x \in \{0,1\}$, with maximal c, $e < c^* < c$ (proc. CLEAN-UP).

Search 5. Given d*, find a TOP-TOP net $N_2 = ((d,h-y),(f,h-v))$, $v,y \in \{0,1\}$, with minimal d, $d < d^* < f$ (proc. CLEAN-UP).

Search 6. Given interval [c,d], find minimal g, such that $c \leq g < d$, v-cut (g,g+1) is saturated and there is no fictitious net across the v-cut. If there is no such g report that fact (proc. CLEAN-UP and PULL).

Search 7. Given g, find maximal q such that row q contains the terminal of a TOP-NONTOP or NONTOP-NONTOP net N = ((s,y)(t,z)), q = max(y,z), across v-cut (g,g+1) (proc. PULL).

While operations 1, 2, and 3 could be carried out using very conventional data structures (priority queues realized by height-balanced trees), the other operations require recourse to more sophisticated structures: these are the (discrete range) priority search trees [McCreight (82)] and the segment trees [Bentley (77); see also, Lipski-Preparata (80)]. Both types of trees deal with integers and integer pairs, denoted by [,].

Priority search trees support the following operations on a dynamic set S of points, in time logarithmic in the size of their coordinates. We denote points stored in priority search trees by [x,y] in order to distinguish them from points in rectangles:

PST 1. Insert (delete) a point;

PST 2. Given query integers x_0, x_1, and y_1, find $[x,y] \in S$ such that $x_0 \leq x \leq x_1$, $y \geq y_1$ and x is minimal;

PST 3. Given query integers x_0 and x_1, find $[x,y] \in S$ such that $x_0 \leq x \leq x_1$ and y is maximal.

It is now appropriate to briefly recall the structure of segment trees. A segment tree T(a,b) over an interval [a,b] consists of a root v, with two parameters B[v] = a and E[v] = b, and, if b-a > 1, of a left subtree $T(a,\lfloor(a+b)/2\rfloor)$ and a right subtree $T(\lfloor(a+b)/2\rfloor,b)$; v is a leaf if b-a = 1. Besides the essential parameters B[v] and E[v], associated with each node there are additional auxiliary parameters required by the specific applications. Given an interval [c,d], with $a \leq c < d \leq b$, segment tree T(a,b) supports in logarithmic time the operation of partitioning [c,d] into O(log(a-b)) segments, each of which is allocated to a node of T(a,b).

Nets are conveniently subdivided into subsets (each to be maintained in a seperate data structure), according to the following classification. Referring to Figure 14, TOP is the set of terminals in rows h-1 and h,

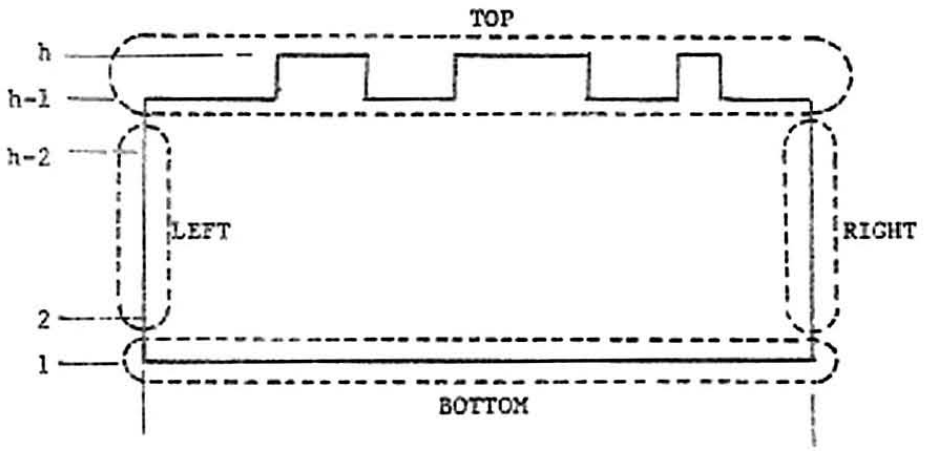


Figure 14. Classification of the terminals of the near-rectangle.

BOTTOM is the set of terminals in row 1, LEFT is the set of terminals in column 1, rows 2 through h-2, and RIGHT is defined analogously. We shall use eight collections of nonfictitious nets: TOP-TOP, TOP-LEFT, TOP-RIGHT, TOP-BOTTOM; LEFT-BOTTOM, RIGHT-BOTTOM; LEFT-RIGHT, BOTTOM-BOTTOM (no sophisticated data structure is needed for LEFT-LEFT and RIGHT-RIGHT). We now discuss the data structure for each of these collections.

TOP-TOP: Priority search tree and priority queue. Nets are of the form ((f, h-a), (g, h-b)) for some a,b $\in$ {0,1}. We store [f,g] as a point in a priority search tree; in addition, points g are stored in a priority queue. Search 1 is solved by choosing $x_0 = 1$ and $x_1 = n$ in PST3; Search 3 is solved by choosing $x_0 = c$, $x_1 = n$, and $y_1 = c$ in PST2; the symmetric version of Search 3 is solved by finding the maximal g less than or equal to some

given d in the priority queue for the right terminals; Search 4 is solved by choosing $x_0 = 1$, and $x_1 = c^*$ in PST3; Search 5 is solved by choosing $x_0 = d^*$, $x_1 = n$, and $y_1 = 1$ in PST2. We leave it as an open problem to find a single data structure for TOP-TOP nets that supports all searches.

TOP-LEFT: Priority queue, ordered according to the column of the right terminal. In Search 7, this enables us to determine if there is a TOP-LEFT net across (g,g+1).

TOP-RIGHT: Same as TOP-LEFT.

TOP-BOTTOM, BOTTOM-BOTTOM: Same as TOP-TOP.

LEFT-BOTTOM: Priority search tree. (Nets are of the form ((1,y),(x,1)).) In the priority search tree we store points [x,y]. In Search 7, we can find a LEFT-BOTTOM net with maximal y across (g,g+1) by setting $x_0 = g+1$ and $x_1 = n$ in PST3.

RIGHT-BOTTOM: Same as LEFT-BOTTOM.

LEFT-RIGHT: Priority queue, ordered according to the row that contains the higher terminal. This enables to find the desired NONTOP-NONTOP net in Search 7.

The above discussion outlines the implementation of searches 1, 2, 3, 4, 5, and 7 with respect to nonfictitious nets; all are executable in time O(logN). Furthermore, all data structures can be updated in time O(logN) after laying out a net.

It remains to describe the data structure used for columns and fictitious nets. This data structure also supports Search 6. We resort to a segment tree T(1,n) with n leaves numbered 1,...,n. Leaf i represents column i and v-cut (i,i+1). In segment tree T(1,n) we store all nonfictitious nonvertical nets as intervals of the type [column containing left terminal, column containing right terminal -1]. The segment tree has the following auxiliary node parameters:

FICT[v]: = number of fictitious points in subtree rooted at v

C[v]: = number of interval segments allocated to node v, i.e., C[v] is the number of nonfictitious nets which start at or before B[v] and end after E[v] and which either start after B[father[v]] or end before or at E[father[v]].

$$EDENS[v] = \begin{cases} C[v] & \text{if } v \text{ is a leaf} \\ C[v] + \max(EDENS[lson[v]], EDENS[rson[v]]) & \text{if } v \text{ is not a leaf.} \end{cases}$$

Quantity EDENS[v] can be interpreted as follows. For $B[v] \leq i < E[v]$, let edens(i,v) be the number of nonficititious nets that go across v-cut (i,i+1) and have at least one terminal in interval [B[father[v]],E[father[v]]]. Then $EDENS[v] = \max\{edens(v,i): B[v] \leq i < E[v]\}$.

Using auxiliary node parameter FICT it is easy to carry out searches 1, 3, 4, and 5 with respect to fictitious nets and also to determine the second terminal of a fictitious net if one is known. Also all parameters of the segment tree can be updated in logarithmic time when inserting or deleting a nonfictitious net or when inserting or deleting fictitious points. There is one subtle point however. When inserting fictitious points c, d in procedure CLEAN-UP, line, the set of fictitious nets may change dramatically (see Figure 15).

Figure 15. Change in fictitious nets as a result of inserting points c and d.

It is therefore essential that only nonfictitious nets be considered in computing parameter EDENS[v].

We finally describe how to do Search 6. Note that Search 6 is tantamount to deciding whether edens(g,root) = h for some g with c $\leq$ g < d and if so to find a minimal such g. Next note that if node v is an ancestor of leaf g then edens(g,root) = edens(g,v) + Σ{C[w]; w is proper ancestor of v}. The latter formula gives rise for an algorithm for Search 6. Conceptually, insert interval [c,d-1] into the segment tree. For every node v, such that [c,d-1] is allocated to v, compute D[v]: = Σ{C[w]: w is proper ancestor of v}. This is easily done in logarithmic time and in fact can be combined with the insertion routine. Finally compute EDENS[v] + D[v] for every such node and determine the leftmost node with EDENS[v] + D[v] = h. If there is no such node the outcome of Search 6 is negative. If there is such a node, take the leftmost node and call it v. Trace a path from v to a leaf by following the points to the left son if EDENS[v] = C[v] + EDENS[lson[v]] and by following the points to the right son otherwise. In this way we find a v-cut (g,g+1) which is saturated and has no fictitious net across it. Thus Search 6 takes logarithmic time.

We thus have

Theorem 2: Given a rectangle routing problem with N nets one can construct a layout (if there is one) in time O(NlogN). Moreover, the layout has only O(N) knock-knees.

Proof: Immediate from Theorem 1 and the discussion above.

V. MULTI-TERMINAL NETS

A multi-terminal net may have any number of terminals, i.e., a multi-terminal net is an artitrary subset of the boundary points of the rectangle. It is not known whether the results of the previous section carry over to multi-terminal nets. However, the results of the previous section can be used to obtain an approximation algorithm for multi-terminal net routing problems.

Let P be any multi-terminal routing problem such that the density of no v-cut and h-cut exceeds its capacity. Enlarge the grid by inserting new empty grid lines between any pair of adjacent vertical or horizontal grid lines, but insert three grid lines between columns 1 and 2 and rows 1 and 2. Thus an n by m rectangle is enlarged to an (2n+1) by (2m+1) rectangle (see Figure 16).

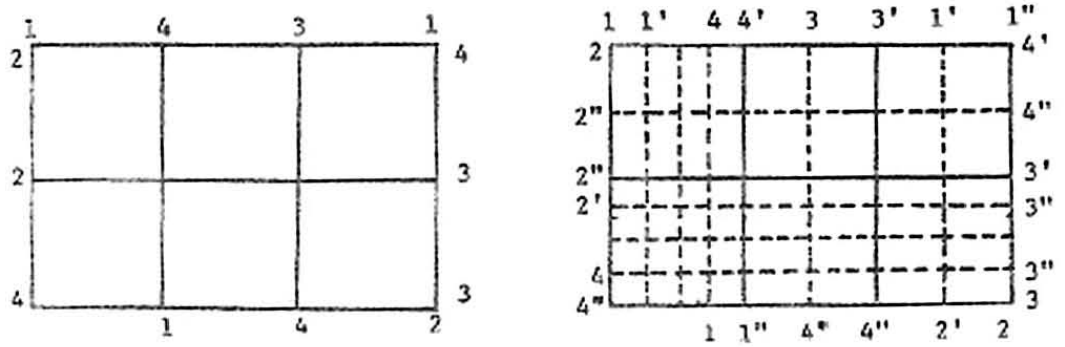


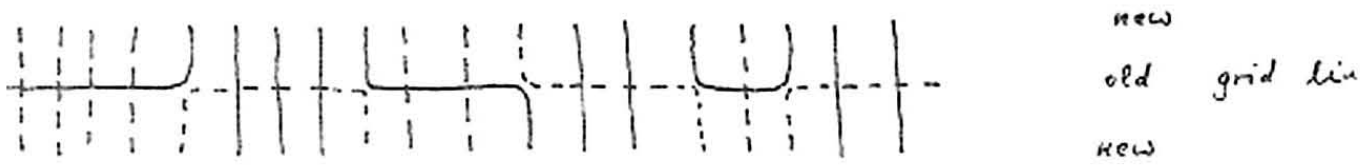
Figure 16. New grid lines are dotted.

Now, transform the routing problem into a two-terminal net routing problem as follows (cf. Figure 16). View a multi-terminal net as a cycle which is attached to the boundary at some number of points (its terminals). Create a copy of each terminal and move it to an adjacent empty grid line. In this way a multi-terminal net with d terminals gives rise to d two-terminal nets. Also this transformation at most doubles the density. Thus in the new problem no column and row is saturated (that's the reason why we inserted three grid lines and not only two between columns 1 and 2 and rows 1 and 2) and hence the revised row and column criteria hold. Thus the problem is valid and has a solution. Moreover, it is easy to see that edges between the two copies of a terminal need not be used in the layout and can therefore be used to connect the different pieces of a multi-terminal net.

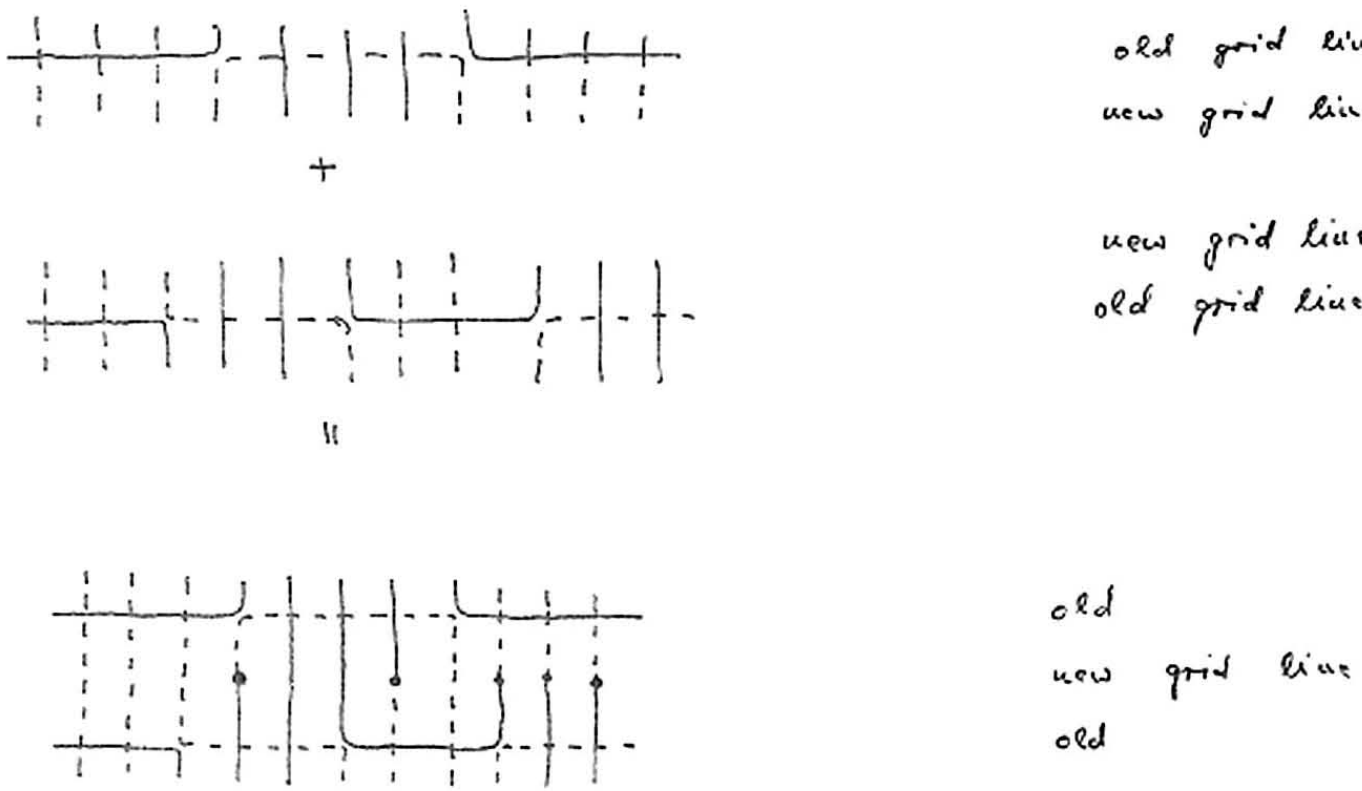
VI. Layer Assignment

We mentioned already in the introduction that every layout can be wired using four layers of interconnect (Brady/Brown). The precise model is as follows. We have four layers, say 1,2,3,4, of interconnect, two of which, say i and j (i < j), are electrically preferential. In any gridpoint we can place two contacts, one connecting layers i_1 and i_2 and one connecting layers i_3 and i_4 where $1 \leq i_1 \leq i_2 < i_3 \leq i_4 \leq 4$. Thus it is assumed that we can simultaneously contact from layers 1 to 2 and from 3 to 4, or from 1 to 3 and from 4 to 4. However, we cannot simultaneously contact from 1 to 3 and from 2 to 4. Let L be any layout with k knock-knees. Then L can be wired in the model described above such that (a) there are only O(k) contact cuts and (b) all but O(k) length of wire runs in special layers i and j. The wiring is such that the usage of contact cuts and the non-special layers is limited to the vicinity of knock-knees. We refer the reader to Brady/Brown for details.

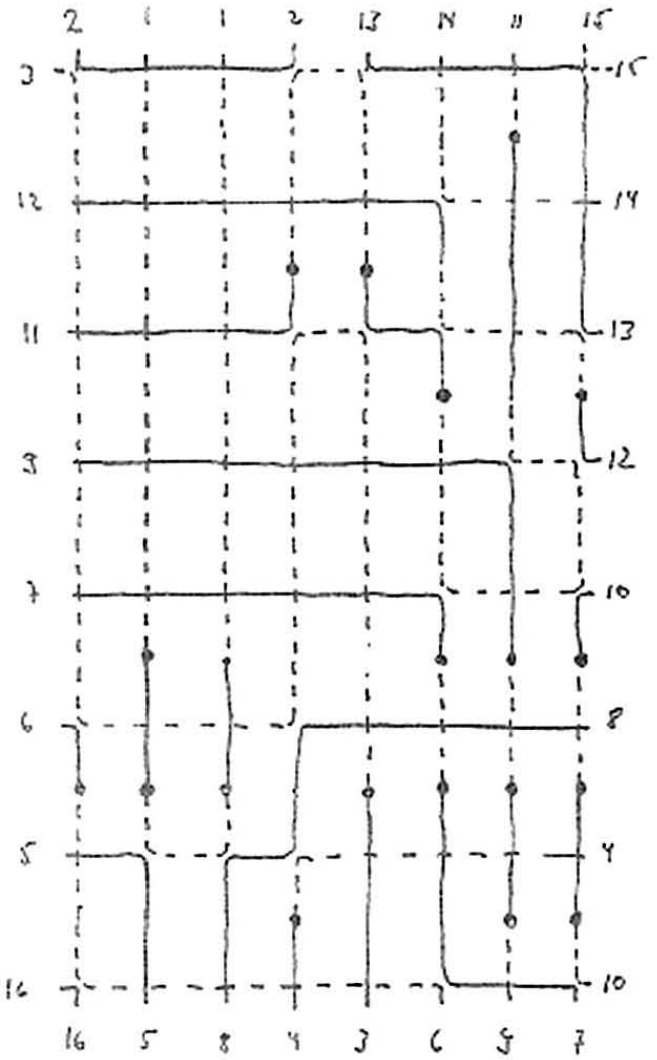
Four layers of interconnect may not be technically feasible at present. We will now show how to wire a layout using two layers of interconnect after expanding it by a factor of two in the y-direction (or x-direction). Let L be a layout. Expand the grid by inserting a new empty grid line between any pair of adjacent horizontal grid lines. Thus an n by m rectangle is enlarged to an n by 2m - 1 rectangle. Contacts are only placed on new grid lines. The layer assignment to wires on old grid lines is as follows. The wire entering a row from the left runs on layer 1 all the way to its knock knee. There it is reflected upwards or downwards. We continue to run it on layer 1 to the adjacent new grid line. The other wire taking part in the knock knee runs on layer 2 all the way to its other knock knee in the row, ... The following diagram illustrates the strategy.



We use this stragegy independently on every old grid line. Note that the layer of every vertical wire segment crossing an old grid line is fixed by the layer assignment of the horizontal segment. We can now use the new grid lines to mend together the solutions for adjacent old grid lines by placing contacts at appropriate places. The diagram below illustrates the mending process.



A complete writing for the final example of section III is shown below.



REFERENCES

[Bentley]
J. L. Bentley, "Solution to Klee's rectangle problems." Unpublished notes, Carnegie-Mellon University, 1977.

[Brady-Brown]
M. L. Brady and D. J. Brown, "Arbitrary planar routing with four layers." Manuscript, Coord. Sci. Lab., University of Illinois; September 1983.

[Frank]
A. Frank, "Disjoint paths in a rectilinear grid." (March 1982); to appear in Combinatorica.

[Lipski-Preparata]
W. Lipski, Jr. and F. P. Preparata, "Finding the contour of a union of iso-oriented rectangles," Journal of Algorithms, vol. 1, pp. 235-246; 1980.

[McCreight]
E. M. McCreight, "Priority search trees," Xerox Corp. Research Rep. CSL-81-5; January 1982.

[Preparata-Lipski]
F. P. Preparata and W. Lipski, Jr., "Three layers are enough," Proc. 23rd IEEE Symposium on Foundation of Computer Science, Chicago, Illinois, pp. 350-357; November 1982.