Routing through a Rectangle

K. MEHLHORN

Universität des Saarlandes, Saarbrücken, West Germany

AND

F. P. PREPARATA

University of Illinois at Urbana-Champaign, Urbana, Illinois

Abstract. In this paper an $O(N \log N)$ algorithm for routing through a rectangle is presented. 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 $O(N \log N)$ time; this contrasts favorably with the area of the layout that might be as large as N^2. The layout constructed can be wired using four layers of interconnect with only $O(N)$ contact cuts. A partial extension to multiterminal nets is also discussed.

Categories and Subject Descriptors: B.7.1 **[Integrated Circuits]**: Types and Design Styles— *VLSI* (*very large scale integration*); B.7.2 **[Integrated Circuits]**: Design Aids—*placement and routing*; E.1 **[Data]**: Data Structures—*arrays*; *trees*; F.2.2 **[Analysis of Algorithms and Problem Complexity]**: Nonnumerical Algorithms and Problems—*routing and layout*

General Terms: Design, Theory

1. *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 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, $1 \le i \le N$.

Example. The first example (Figure 1) 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.

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.

Authors' addresses: K. Mehlhorn, FB 10, Universität des Saarlandes, 66 Saarbrücken, West Germany; F. P. Preparata, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, 1101 West Springfield Avenue, Urbana, IL 61801.

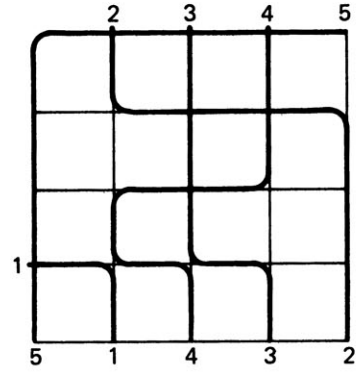


FIG. 1. Example of a rectangle routing problem.

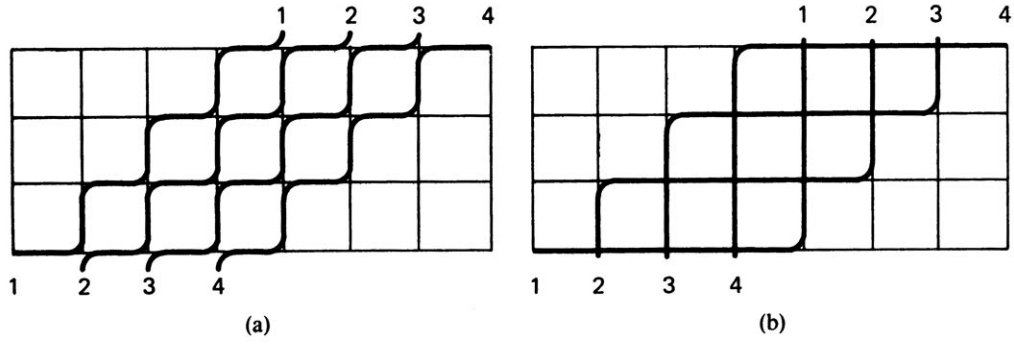


FIG. 2. Two solutions of an RRP. (a) A solution with $\Omega(N^2)$ knock-knees. (b) A solution with $O(N)$ knock-knees.

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

The fundamental characteristic property of RRPs was recently established by Frank [3].

THEOREM 1 [3]. *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 horizontal edges between columns a and $a + 1$, that is, m. The *density* of a v-cut $(a, a + 1)$ is the number of nets that go across the cut, that is, have exactly one terminal in columns $1, \ldots, a$. A v-cut is *saturated* if its density is equal to its capacity. A v-cut is *oversaturated* if its density exceeds its capacity. Similar notions are defined for horizontal cuts (h-cuts).

Consider an RRP (Figure 3). 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 $r_1 < \cdots < r_k$, and we 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, $T_1, \ldots, T_{k+1}$.

The *extended degree* of a node v of rectangle R is the degree[1] of node v plus the number of nets having v as a terminal. The *parity* of a set T of nodes is the parity

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

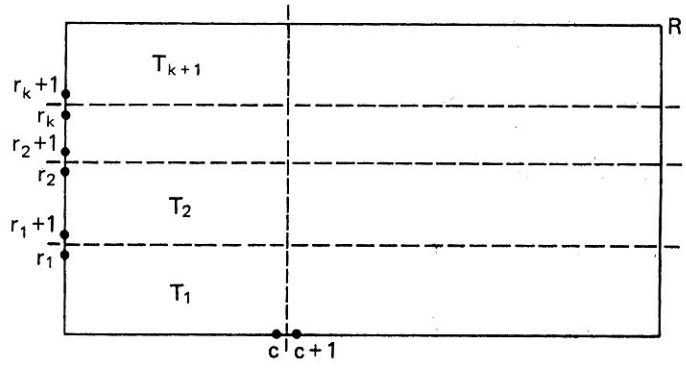


FIG. 3 Rectangle dissection caused by saturated h-cuts and an arbitrary v-cut.

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.

In Figure 2b, we show a layout for the same problem in which every path bends at most twice. Thus $O(1)$ amount of information suffices to describe each path. The solution shown in Figure 2b is constructed by the algorithm described in this paper. (For two-shore problems in which all terminals are on the top or bottom of the rectangle, Preparata and Lipski [6] have shown that this behavior is achievable.) The main result of this paper is the following theorem.

THEOREM 2. *A layout for a solvable RRP can be constructed in time $O(n + m + N \log N)$. Moreover, the layout constructed contains only $O(N)$ knock-knees.*

In this paper, we do not address the question of layer assignment to layouts. However, we want to mention that Brady and Brown [2] have shown recently that every layout can be wired using only four layers of interconnect. Moreover, it can be shown that the number of contact cuts required for a four-layer wiring is, at most, proportional to the number of knock-knees. Thus we have the following corollary.

COROLLARY 1. *Every solvable RRP can be wired in four layers using only $O(N)$ contact cuts.*

This paper is organized as follows. Section 2 develops some technical preliminaries. The algorithm, the proof of correctness, and an analysis of the number of knock-knees generated are given in Section 3. Section 4 describes an implementation and contains the proof of the $O(N \log N)$ running time. Section 5 describes a partial extension to multiterminal nets, and Section 6 addresses the problem of obtaining a multilayer wiring of the layouts. Finally, Section 7 reports about computational experience.

2. *Preliminaries*

In this section we present some simple observations that 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 h-cut becomes saturated,
(2) the number of deleted empty columns between any two saturated v-cuts is even.

Therefore, it is 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 without loss of generality that $n = O(N)$ and $m = O(N)$, and we note that the reductions can be done in time $O(N)$.

An RRP P is *standard* 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 [3]. Indeed, suppose that there are h saturated h-cuts and k saturated v-cuts. They divide the rectangle into $(h + 1)(k + 1)$ regions $T_1, T_2, \ldots$. Each region has even parity (a proof can be found in [3]) 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 a 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 [3], it is shown that the extended row and column criteria hold for P' if and only if 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 if and only if, 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 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 net 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 both the capacity and the density of any h-cut by 1.

A column c is said to be *density increasing, preserving, decreasing* depending on whether density$(c - 1, c) <$ density$(c, c + 1)$, density$(c - 1, c) =$ density$(c, c + 1)$, density$(c - 1, c) >$ density$(c, c + 1)$, respectively.

3. *The Algorithm*

In this section we describe an algorithm that solves any standard RRP in time $O(N \log N)$, where N is the number of nets.

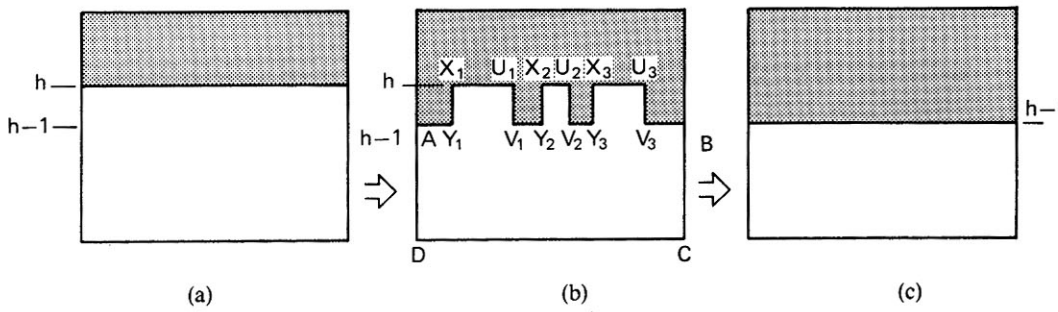


FIG. 4. Processing of a row (row h). (a) Before processing. (b) After the first stage. (c) After the second stage.

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.

The coordinates of the points of the boundary of the near-rectangle in Figure 4b are $A = (1, h-1)$, $B = (n, h-1)$, $C = (n, 1)$, $D = (1, 1)$, $X_i = (c_i, h)$, $Y_i = (c_i, h-1)$, $U_i = (d_i, h)$, and $V_i = (d_i, h-1)$, where $1 \le c_1 < d_1 < c_2 < d_2 < \cdots \le d_s \le n$. The set of points $\{(x, h): c_i \le x \le d_i \text{ for some } i\}$ is called a *segment*. Points X_i and U_i are *convex corners*, whereas points Y_i and V_i are *concave corners*, $1 \le i \le s$.

A *standard near-rectangle routing problem* (SNRRP) consists of a number of nontrivial 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 that 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 that have a terminal on both sides of the cut. We consider v-cuts and h-cuts. The v-cuts are $(a, a+1)$, where $1 \le a < n$, and the h-cuts are $(b, b+1)$, $1 \le b \le h-2$ and $(h, h-1)$, restricted to the segments X_iU_i, $1 \le i \le s$. The h-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_i U_i is not oversaturated.

(b) Let $c_i \le a < d_i$ for some i. If dens$(a, a+1) < h$, then dens$(a, a+1) < h-2$. (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 interior of the near-rectangle. Hence, $h + \text{dens}(a, a+1)$ is even, since h is the number of edges and $\text{dens}(a, a+1)$ is the number of nets that go across v-cut $(a, a+1)$. Thus $\text{dens}(a, a+1) < h$ implies $\text{dens}(a, a+1) \leq h - 2$. □

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

We partition the set of boundary points of a near-rectangle into two 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 that have one terminal in TOP and one terminal in NONTOP. A net $N_i \in$ TOP–NONTOP is also called a TOP–NONTOP net. Sets TOP–TOP, NONTOP–NONTOP are defined similarly. A horizontal net is a TOP–TOP net that has neither point A nor point 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 previously mentioned, 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 $x_1, x_2, \ldots, x_{2l}$ (ordered from left to right) of points in TOP such that the fictitious nets are the l nets (x_{2i-1}, x_{2i}), $1 \leq i \leq l$. We represent the fictitious nets by the ordered sequence $x_1, x_2, \ldots, x_{2l}$ 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 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 with it. We shall 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. Tokens are deposited as "endowments" on nets and segments, and we use the symbols a_{NN}, a_{TN}, a_{TT}, and a_{S} to denote the respective minimum endowments of each NONTOP–NONTOP net, TOP–NONTOP net, TOP–TOP net, and segment. Each action of the algorithm can be viewed as the transformation of a collection of nets and segments (input) into a collection of laid-out wires and a collection of new nets and segments (output). The revenues for such action are the endowments of the input nets and segments, and the expenditures are for the wires (one token per horizontal laid-out wire) and for the endowments of the output nets and segments. Note that only the tokens used for wires are lost; the tokens used for the endowments of new sets and segments are to be reused at a later stage. If we prove that the initially available tokens are sufficient to pay for all horizontal wires, then the number of the latter does not exceed the initial global endowment. We choose the following values:

$$a_{\text{NN}} = 12, \qquad a_{\text{TN}} = 7, \qquad a_{\text{TT}} = 1, \qquad a_{\text{S}} = 4. \tag{1}$$

Initially each net is given its appropriate endowment; thus, the initial global endowment is $O(N)$.

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*, that is, we show that the actions transform a valid problem P into a valid problem P'. Then we do the *accounting*, that is, 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 $N_1 \cdots N_k$ $(k \geq 1)$ such that $N_1, \ldots, N_{k-1}$ are right-going TOP–NONTOP nets and N_k is either a right-going TOP–NONTOP net or a TOP–TOP net (Figure 5). In addition:

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

By *layout of horizontal net*, $N_i = ((f, h), (g, h))$ $(f < g)$ with bound 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. We may view this layout action as a transformation of nets; specifically, 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 transformed to net $N_i' = ((x_{i+1}, h-1), (g, h))$ (see Figure 6a). Analogously, the *layout of a right-going net* $N_i = ((f, h), (g, h))$ with bound 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 $(\min(g, x_{i+1}), h-1)$ 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 is transformed to a trivial vertical net if $g \leq x_{i+1}$. The layout of a left-going net is similarly defined.

Alternately, we may view the transformation of N_i to a (nonempty) N_i' as the *movement of a terminal* from (f, h) to $(x_{i+1}, h-1)$; from this viewpoint, each vertical edge laid out corresponds to moving a terminal from (x, h) to $(x, h-1)$ for $f \leq x < \min(g, x_{i+1})$.

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 bound x_{i+1} for $1 \leq i \leq k$. Note that layout of a run turns right-going nets $N_1, \ldots, N_{k-1}$ into trivial 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 $a_{TT}(=1)$ tokens on each TOP–TOP net 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." Run-layout consists of two passes, the first from left to right and the second in the opposite direction. The plane domain still to be processed at the end of run-layout is a near-rectangle, whose segments have very special properties. Clean-up effects the processing of exactly these segments. These two major phases are preceded by a simple

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

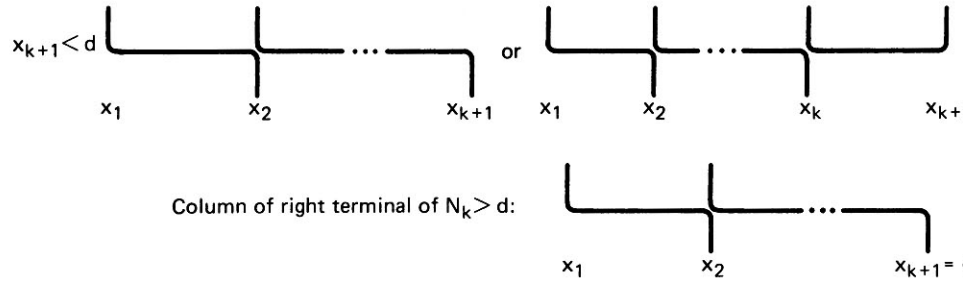


FIG. 5. Illustrations of right runs.

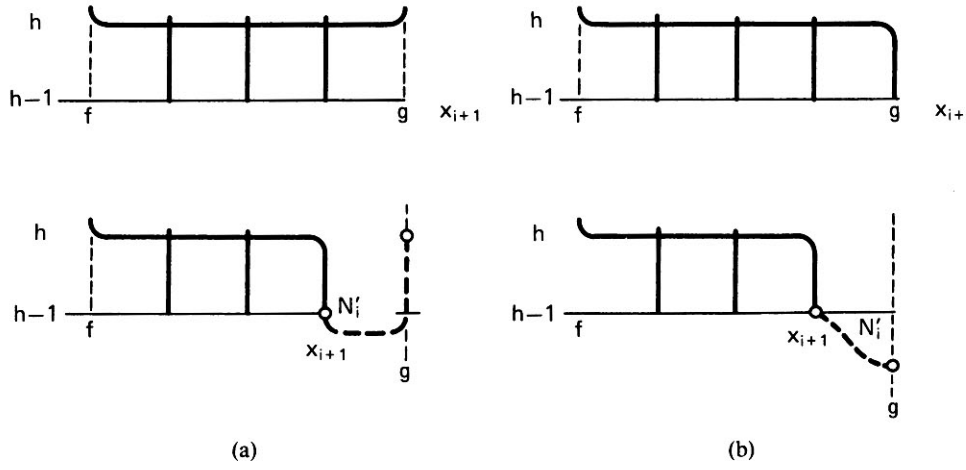


FIG. 6. Layouts of nets.

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 $A \equiv (1, h-1)$ and $B \equiv (n, h-1)$ from set NONTOP to set 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 (Figures 7a–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.

LEMMA 2. *Either there is at least one net with both terminals in row h or either X or U is exposed.*

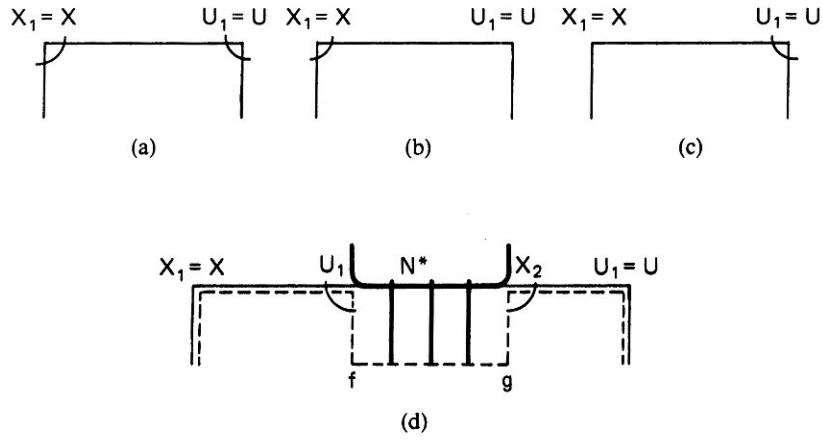


FIG. 7. Near-rectangles obtained after the initialization step; in each segment, at least one corner is exposed. (a, b, c) Step 0 is clearly void if X or U is already "exposed." (d) Two segments, X_1U_1 and X_2U_2, are created.

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 (the problem is solvable), 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 both points $U_1 = (f, h)$ 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* $A \equiv (1, h - 1)$ *moved from NONTOP to TOP, either one net moves from NONTOP–NONTOP to TOP–NONTOP (and hence releases* $a_{NN} - a_{TN} = 5$ *tokens) or one net moves from TOP–NONTOP to TOP–TOP (and hence releases* $a_{TN} - a_{TT} = 6$ *tokens). Since node* $B \equiv (n, h - 1)$ *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 (revenue).*

If either X or U is exposed, then we have drawn no horizontal wire. We place $a_S + 3 = 7$ *(see* (1)*) 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 releases* $a_{TT} = 1$ *token, so* $10 + 1 = 11$ *tokens are available. Also we place* $3 + a_S = 7$ *tokens on segment* X_1U_1 *and* 2 *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 without loss of generality 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, that is, a call of procedure RIGHT-RUN-LAYOUT $(1, \min(d, n))$, where $U_1 = (d, h)$ if $U_1 \neq U$.

Execution of this procedure on the segment generates a sequence of segments $X_1'U_1', \ldots, X_k'U_k'$ ($k \geq 0$), with $X_i = (c_i, h)$ and $U_i = (d_i, h)$,

(i) X_i', U_i', $1 \leq i \leq k$ are all exposed.
(ii) Segments $X_i'U_i'$, $1 \leq i \leq k$, hold $a_S + 1$ tokens each (where $a_S = 4$).
(iii) No (x, h), $c_i < x < d_i$ and $1 \leq i \leq k$, is the terminal of a right-going TOP–NONTOP net.

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 $N^* = ((f, h), (g, l))$ with $g \leq f$, to *extend* N^* *to* $e > f$ means to lay out 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 nonvertical net is chosen here; (/this is treated conventionally as an extension/)
  while $u < d$ **do**
    begin if EXT = Λ **then**
      begin $e := f \geq u$ is minimal such that (f, h) is the terminal of a right-going TOP–NONTOP net;
        if $e = \Lambda$ **then** $e := d$;
        if $e > u$ **then** output (e, p) (/this is a newly created segment/)
      end
    else begin $e := f \geq u$ is minimal such that (f, h) is either the terminal of a right-going TOP–NONTOP net or the left terminal of a TOP–TOP net;
      if $e = \Lambda$ **then** $e := d$
    end;
    if EXT $\neq \Lambda$ **then begin** extend EXT to e; EXT := Λ **end**;
    if $e < d$ **then**
      begin lay out right run starting at e;
        g := destination column of run
      end
    else $g := d$;
    if (g, h) is the terminal of a left-going TOP–NONTOP net N **then**
      EXT := N;
    $u := g$
  end
end.

For example (Figure 8): $c = 2$, $d = 7$, (c, h) is exposed. In the first iteration of the **while** loop, we have $e = 3$. The run starting in column 3 consists of Net 1 and extends to column 4. Since (4, 4) is the right terminal of a left-going net, we set EXT := Net 2. In the next iteration of the **while** loop, we have $e = 5$, etc.

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', that is, 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 1, 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.

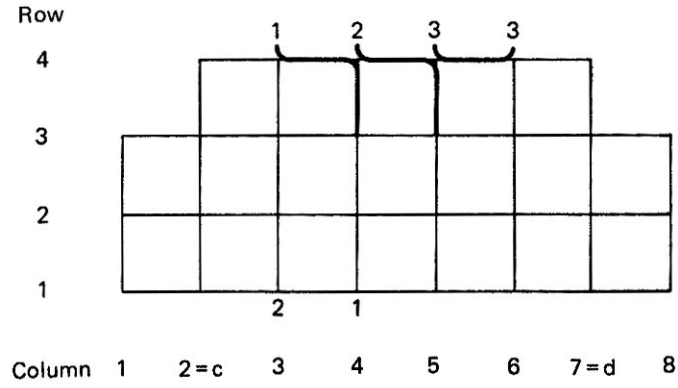


FIG. 8. Illustration of the action of RIGHT-RUN-LAYOUT.

The final case to consider is when a net extension has been routed across the v-cut and hence the density is increased by 1. Suppose that the extension of the left-going net 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 right terminals of a TOP–TOP net. Thus

$$\text{dens}(a, a+1) \leq \text{dens}(a-1, a) \leq \cdots \leq \text{dens}(x, x+1).$$

In view of Lemma 1, it suffices to show $\text{dens}(x, x+1) < h$. This can be seen as follows. If $x = 1$, and hence $(1, h)$ is the terminal of two vertical nets, we have $\text{dens}(x, x+1) \leq h - 2$, since there are at most $h + 2$ terminals in column 1, at least 4 of which belong to vertical nets. If $x > 1$, then $\text{dens}(x, x+1) \leq \text{dens}(x-1, x) - 2 \leq h - 2$, 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 that 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 $X_1 U_1$*

$= x_{TN} \cdot a_{TN} + x_{TT} \cdot a_{TT} + a_S + 3.$

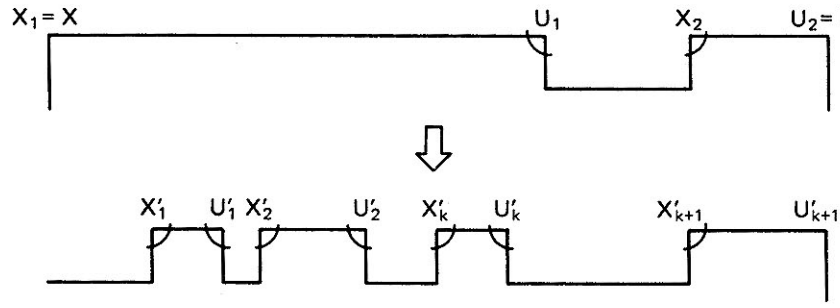


FIG. 9. Near-rectangle after the left-to-right scan.

Finally observe that $x_{TN} \geq z_1 - 1$, *since all but the first extension are preceded by a completed TOP–NONTOP net, and that* $x_{TN} \geq z_2 - 1$, *since all but the last segment are followed by the layout of a run that starts with a TOP–NONTOP net. Thus* $2x_{TN} \geq z - 2$ *and*

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

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, where neither X nor U is initially exposed.

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 right-to-left scan (symmetric to 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 to 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 right-to-left scan shrinks segment $X'_{k+1}U'_{k+1}$ to a void segment and divides each of the segments $X'_iU'_i$, $1 \leq i \leq k$, 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 at least 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, that is, 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 the terminal of a horizontal net, we have* $z_1 = z$ *and hence* $z_2 = 0$, *that is, no new segment is created. Hence the total number of tokens needed to process* $X'_{k+1}U'_{k+1}$ *is at most* $x_{TN} + x_{TT} + z_1 + 1$. *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* $a_{TN} = 7$, $a_{TT} = 1$, *and* $X_{TN} \geq z_1 - 1$, *the cost is amply covered.*

It remains to consider segments $X_i' U_i'$, $1 \le i \le k$. Since both X_i' and U_i' are exposed, we have $z_2 \ge 1$ and $x_{TN} \ge z_1$. Also, as for the accounting for procedure RIGHT-RUN-LAYOUT, the total cost is $x_{TN} + x_{TT} + z_2 \cdot a_S + z_1 + 1$ (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 $X_i' U_i'$). Thus cost is covered, since $x_{TN} \ge z_2 - 1$, $a_{TN} = 7$, $a_{TT} = 1$, $a_S = 4$. With the endowment of at least a_S tokens per segment we can now begin the clean-up step.

Step 2. *Clean-up.* As mentioned earlier, at this point we have a sequence of segments $X_1'' U_1''$, $X_2'' U_2''$, ..., $X_s'' U_s''$ with Properties (i), (ii), and (iii) described above. To this collection, we apply the following algorithm:

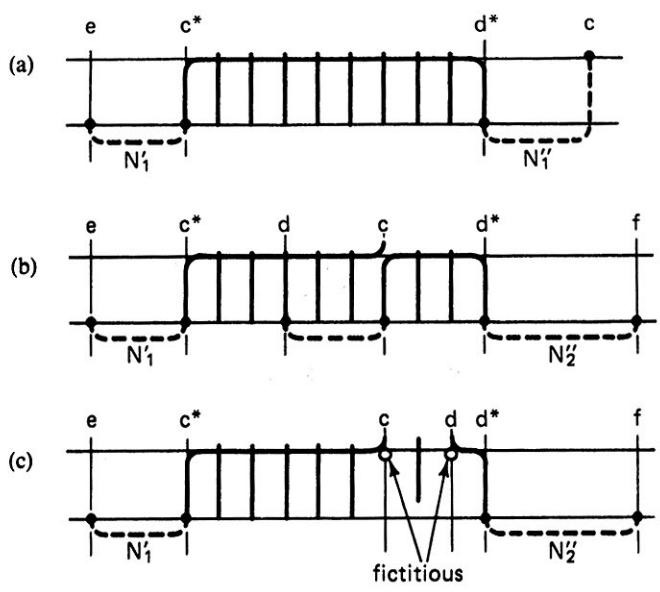
for $i = 1$ **to** s **do** CLEAN-UP (c_i'', d_i'')
where

Procedure CLEAN-UP (c^*, d^*)

1. $N_1 = ((e, h-1), (c, h-x))$:= TOP-TOP net with maximal c, $e < c^* < c$, $x \in \{0, 1\}$;
2. $N_2 = ((d, h-y), (f, h-v))$:= TOP-TOP net with minimal d, $d < d^* < f$, $v \in \{0, 1\}$, $y \in \{0, 1\}$;
3. **if** $N_1 = \Lambda$ **then** $c := c^*$;
4. **if** $N_2 = \Lambda$ **then** $d := d^*$;
5. **if** $(c > d)$ or [$(c < d)$ and for every v-cut $(g, g+1)$, $c \le g < d$, $(g, g+1)$ is either non-
6. saturated or saturated by fictitious net]
7. **then if** $c > d^*$ **then**
8. see Figure 10a — **begin** create nets $N_1' = ((e, h-1), (c^*, h-1))$ and $N'' = ((d^*, h-1), (c, h-x))$;
9. draw wire $(c^*, h-1) \to (c^*, h) \to (d^*, h) \to (d^*, h-1)$ for net N_1; 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 $(c^*, h-1) \to (c^*, h) \to (c, h)$ for net N_1;
12. **if** $c > d$ **then**
13. **begin** create nets $N_2' = ((d, h-y), (c, h-1))$ and
14. see
15. Figure 10b — $N_2'' = ((d^*, h-1), (f, h-v))$;
    draw wire $(c, h-1) \to (c, h) \to (d^*, h) \to (d^*, h-1)$ for net N_2; draw vertical wire segments for all other nets with terminal (z, h), $c^* < z < d^*$
    end
16. **else begin** create net $N_2'' = ((d^*, h-1)(f, h-v))$;
17. draw wire $(d, h) \to (d^*, h) \to (d^*, h-1)$ for net N_2;
18. see Figure 10c — draw vertical wire segments for all other nets with terminal (z, h), $c^* < z < d^*$; add c and d to the set of fictitious points
    end

else PULL (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, that is, 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, and f are fictitious, and $c < d$), then points e, c^*, c, d, d^*, and f will be fictitious after termination of CLEAN-UP and we shall have drawn three fictitious wires.



Revenues	Expenditures
$a_S + a_{TT}$	$1 + 2a_{TT}$
$a_S + 2a_{TT}$	$2 + 3a_{TT}$
$a_S + 2a_{TT}$	$3 + 3a_{TT}$

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

These actions of CLEAN-UP are illustrated in Figure 10a–c.

LEMMA 5. *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 $c^* \le g < d^*$ by one. Thus validity is preserved.

So let us assume $c < d$. (Note that $c = d$ is impossible.) For $c^* \le g < c$ and $d \le g < d^*$, we reduce capacity and density of v-cut $(g, g+1)$. It remains to consider g such that $c \le g < d$. 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 1 and increase the density by 1. Note that there is a fictitious net going across v-cut $(g, g+1)$ in P'. In the second case, we are also 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 $c \le g < d$. □

ACCOUNTING. *For the case in Figure* 10*a, the expenditure is one token for the horizontal wire drawn and one token to endow 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 10*b ($c > d$), 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 10*c ($c < d$), 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), although 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;
                if j < d* then CLEAN-UP (j, d*)
          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.

LEMMA 6. *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 that has its other terminal in one of columns $g+1, \ldots, n$. □

LEMMA 7. *The new problem P' is valid.*

PROOF. We first consider the case $r = h - 1$, that is, N is a TOP–NONTOP net, whence we have either $i < c^*$ or $d^* < i$ (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 ($r \le h - 2$), the crucial observation is that $\text{dens}(b, b+1) \le n - 2$ for every h-cut $(b, b+1)$ with $r \le b \le h - 2$. We prove $\text{dens}(b, b+1) \le n - 2$ by case distinction on b.

Case 1: $b \ge h - (d^* - c^* + 1)/2$. There are at most $n + 2 - (d^* - c^* + 1)$ terminals in TOP outside of segment $X''U''$ and exactly $2(h - 2 - b)$ terminals in rows $b+1, \ldots, h-2$. Since all terminals in segment $X''U''$ are terminals of

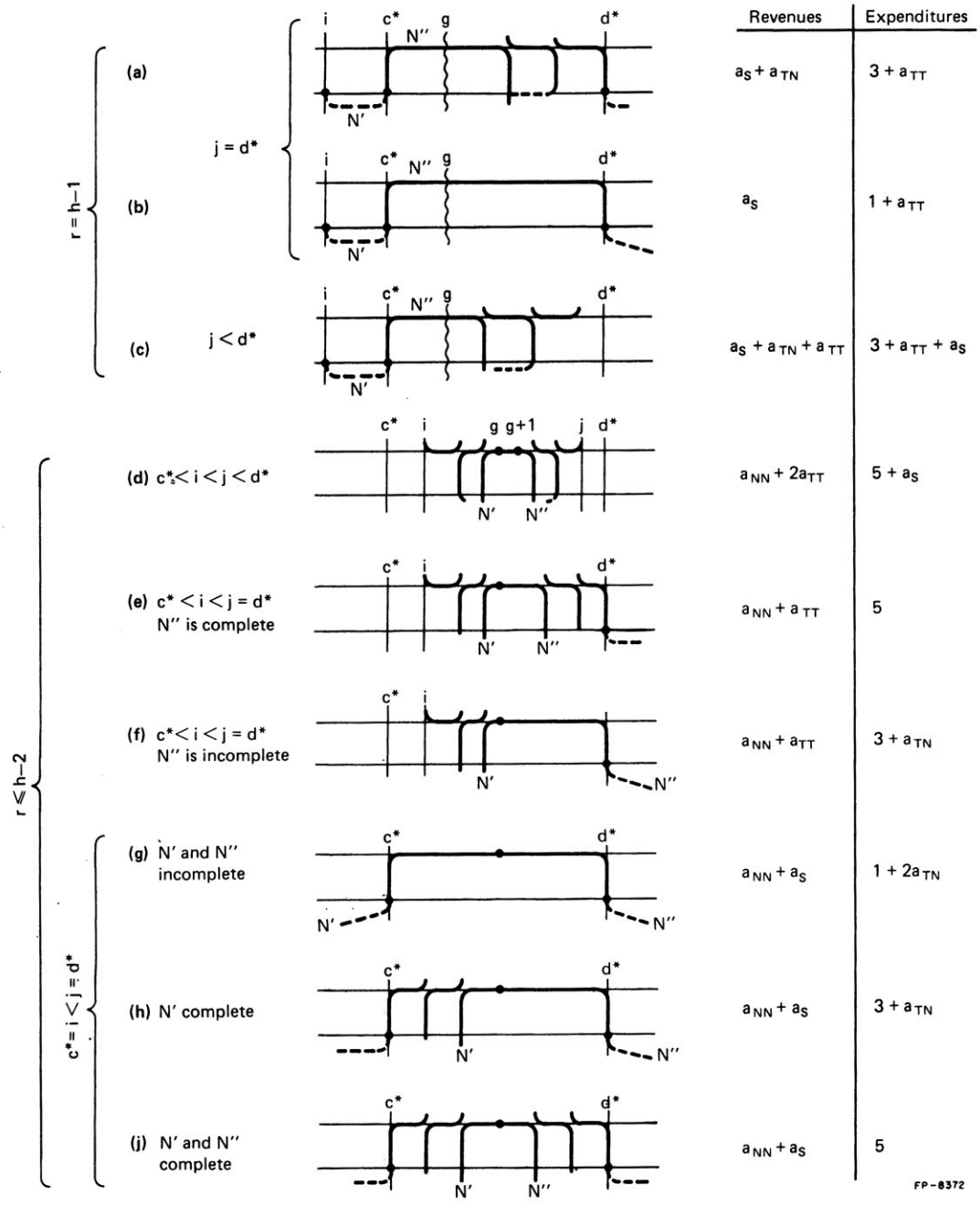


FIG. 11. Actions of PULL and their accounting.

horizontal nets, we conclude

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

because $b \ge h - (d^* - c^* + 1)/2$.

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

There are at most $n + 2b$ terminals in rows $1, \ldots, b$. We label these terminals "up" and "cross" as follows. A terminal is labeled up (cross) if it belongs to a net that goes across h-cut $(b, b+1)$ (v-cut $(g, g+1)$). Then no terminal is labeled up and cross since there are no nets in TOP–NONTOP that go across the cut $(g, g+1)$ (recall that $n \le h - 2$) and since every net in NONTOP–NONTOP that goes across the cut has both terminals in rows $1, \ldots, r$ by definition of r. Since v-cut $(g, g+1)$ is saturated, exactly $2(h - l)$ terminals are labeled cross. Thus

$$\begin{aligned} \text{dens}(b, b+1) &\le n + 2b - 2(h - l) \\ &\le n + 2\left(b - h + \frac{d^* - c^* - 1}{2}\right) \\ &\le n - 2 \end{aligned}$$

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

ACCOUNTING. *The accounting is best done by referring to the cases illustrated in Figure* 11.

Consider first $r = h - 1$ *and* $j = d^*$ (*Figures* 11*a and b*). *Here 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 three horizontal wires (note that* N' *could end in a column that also contains the right terminal of a horizontal net (Figure* 11*a*)), *or we have not completed a TOP–NONTOP net drawing one horizontal wire (Figure* 11*b*). *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* 11*c*). *Then the layout of the row starting with net* N'' *uses at most three horizontal wires and also deletes one TOP–TOP net. Therefore, 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 $r \le h - 2$. *In all of these cases, we reduce the number of NONTOP–NONTOP nets by one, which yields* $a_{NN} = 12$ *tokens.*

Case 1 (*Figure* 11*d*): $c^* < i < j < d^*$. *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 for the endowment* $a_S = 4$ *of the newly created segment and is thus easily covered.*

Case 2: $c^* < i < j = d^*$ *or* $c^* = i < j < d^*$. *Here the number of segments does not change. Assume* $c^* < i < j = d^*$ *without loss of generality. 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* 11*e*) (N'' *is laid out completely), we draw at most five 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 11*f*) (N'' *is not laid out completely), we draw at most three 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: $c^* = i < j = d^*$. *Here the* $a_S = 4$ *tokens on segment* $X''U''$ *become available. If neither* N' *nor* N'' *is laid out completely (Figure* 11*g*), *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* $a_{NN} + a_S = 16$ *tokens available.*

If exactly one of N' *or* N'' *is laid out completely (Figure* 11*h), then we draw at most three 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* $a_{NN} + a_S = 16$ *tokens available.*

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

Thus, in all cases the expenses are covered.

We summarize the previous discussion in

THEOREM 3. *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.*

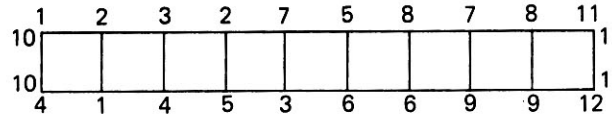
PROOF. Recall that an elementary step corresponds to drawing a horizontal wire of arbitrary length. Knock-knees occur only at the end of horizontal wires. Thus, it suffices to prove the bound on the number of elementary steps.

In Section 2, we described how any (solvable) RRP can be turned into a standard RRP with only $O(N)$ nets.

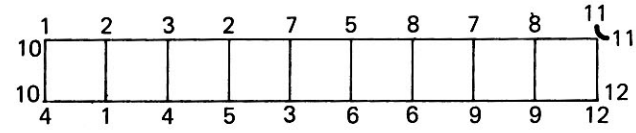
In a standard RRP with $O(N)$ nets, we supply only $O(N)$ tokens initially. Since an elementary step costs one token, the bound follows. □

We conclude this section with an example illustrating the algorithm.

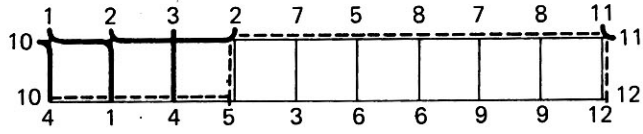
Example 1. The following RRP is given:



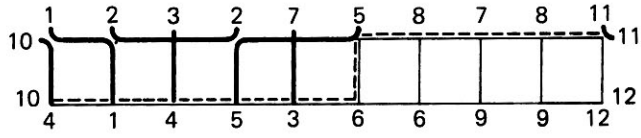
After the execution of INITIALIZE, we have



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

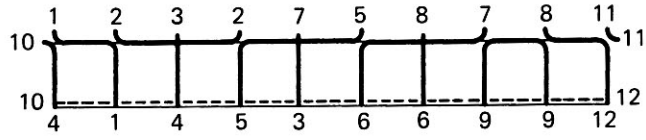


The execution of LEFT-RUN-LAYOUT gives



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

We pull NONTOP-NONTOP net 9 and obtain



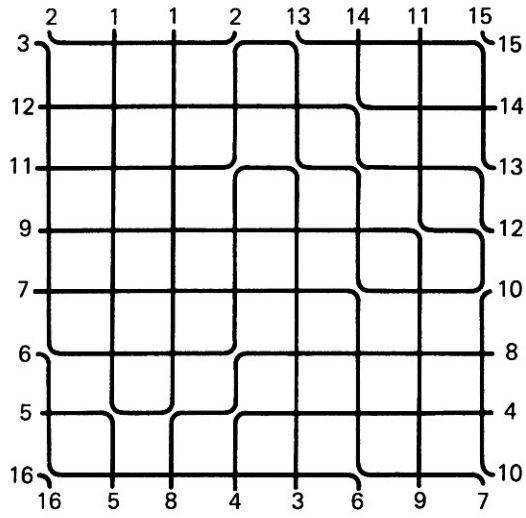
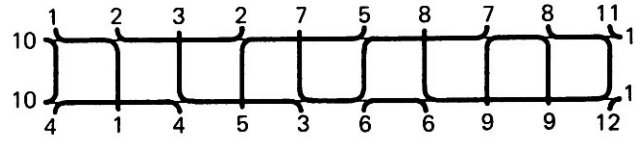
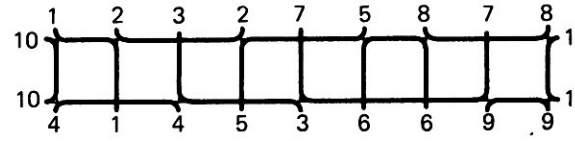


FIG. 12. A more substantial rectangle routing problem.

The layout is then completed by the straightforward 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 12. Note that all columns but the last are saturated.

4. *Implementation*

The goal of this section is to show that each elementary step can be implemented in time $O(\log N)$. 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 3 involves the searches of several collections of items, such as TOP–TOP nets. Each such collections must be organized in a data structure that supports the required operations within the target $O(\log N)$ 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 (procedure INITIALIZE).
Search 2. Given c, find minimal $f > c$ such that (f, h) is the left terminal of a TOP–NONTOP net (in procedure 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 procedure 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, $e < c^* < c$ (procedure 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, $d < d^* < f$ (procedure 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 (procedure 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 $(g, g + 1)$ (procedure PULL).

Although 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* [5] and the *segment trees* [1] (see also [4]). Both types of trees deal with integers and integer intervals, for which we use the notation [,].

Priority search trees (PST) 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 layout 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 recall briefly 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 separate data structure), according to the following classification. Referring to Figure 13, TOP is the set of terminals in rows $h - 1$ and h, 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.

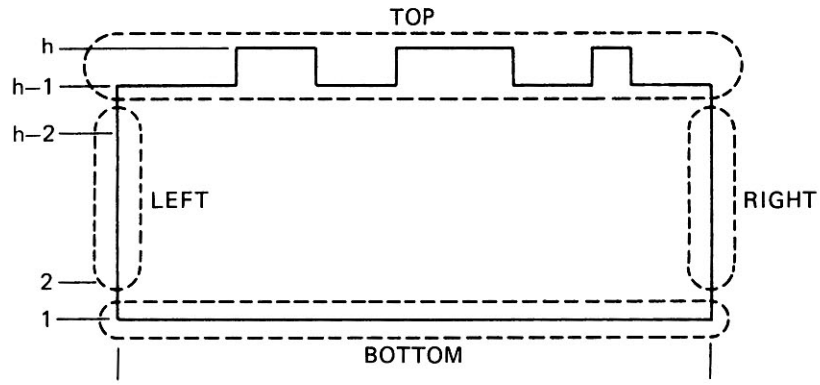


FIG. 13. Classification of the terminals of the near-rectangle.

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 us 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 $O(\log N)$. Furthermore, all data structures can be updated in time $O(\log N)$ after laying out at 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 $1, \ldots, n$. 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—that is, $C[v]$ is the number of nonfictitious nets that start at or before $B[v]$ and end after $E[v]$ and that either start after B[father[v]] or end before or at E[father[v]].

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

Quantity EDENS[v] can be interpreted as follows. For $B[v] \le i < E[v]$, let edens(i, v) be the number of nonfictitious 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] $= \max\{\text{edens}(v, i) : B[v] \le i < E[v]\}$.

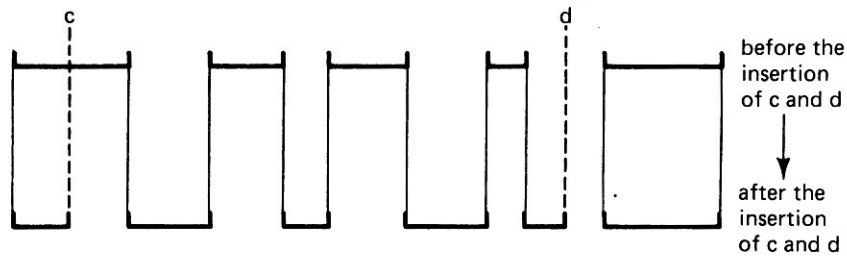


FIG. 14. Change in fictitious nets as a result of inserting points c and d.

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 18, the set of fictitious nets may change dramatically (see Figure 14). 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) + $\sum_{w \in \mathscr{A}(v)} C[w]$, where $\mathscr{A}(w)$ is the set of proper ancestors of v. The latter formula provides 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 $D[v] := \sum_{w \in \mathscr{A}(v)} C[w]$. 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 $(g, g+1)$ that is saturated and has no fictitious net across it. Thus Search 6 takes logarithmic time.

We thus have

THEOREM 4. *Given a rectangle routing problem with N nets, one can construct a layout (if there is one) in time $O(N \log N)$. Moreover, the layout has only $O(N)$ knock-knees.*

PROOF. Immediate from Theorem 1 and the discussion above. □.

5. *Multiterminal Nets*.

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

Let P be any multiterminal 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 retangle is enlarged to a $(2n+1)$ retangle (see Figure 15).

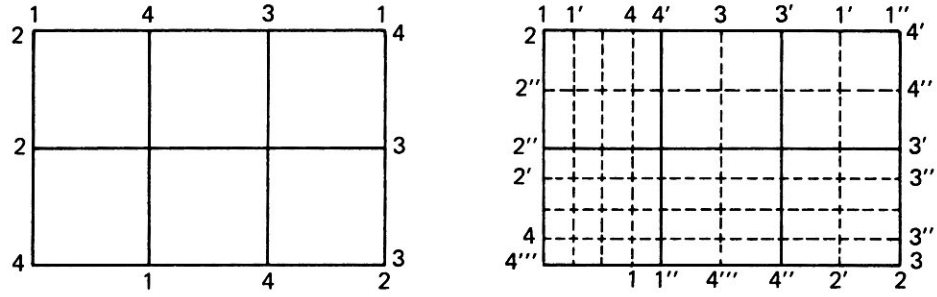


FIG. 15. New grid lines are dotted.

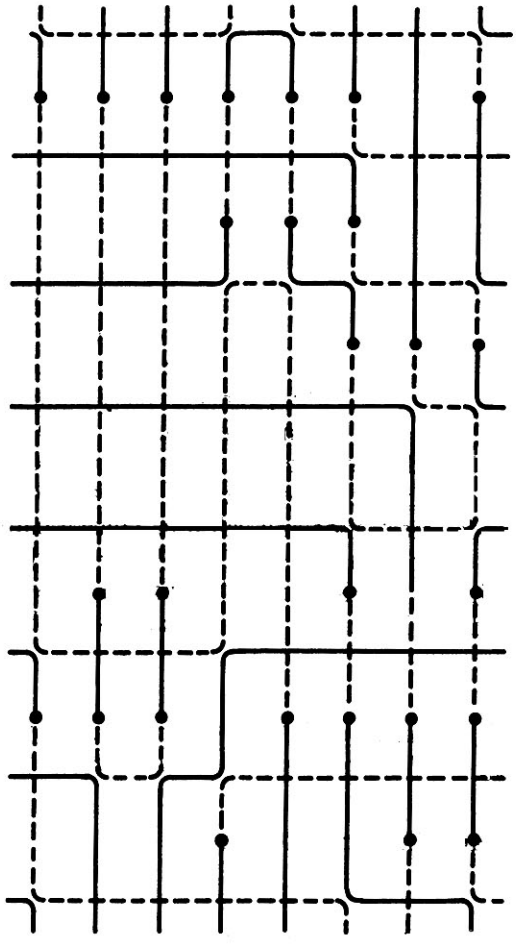


FIG. 16. Two-layer wiring of the layout of Figure 12.

Now, transform the routing problem into a two-terminal net-routing problem as follows (cf. Figure 15). View a multiterminal net as a cycle that 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 multiterminal 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 are saturated (that is 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

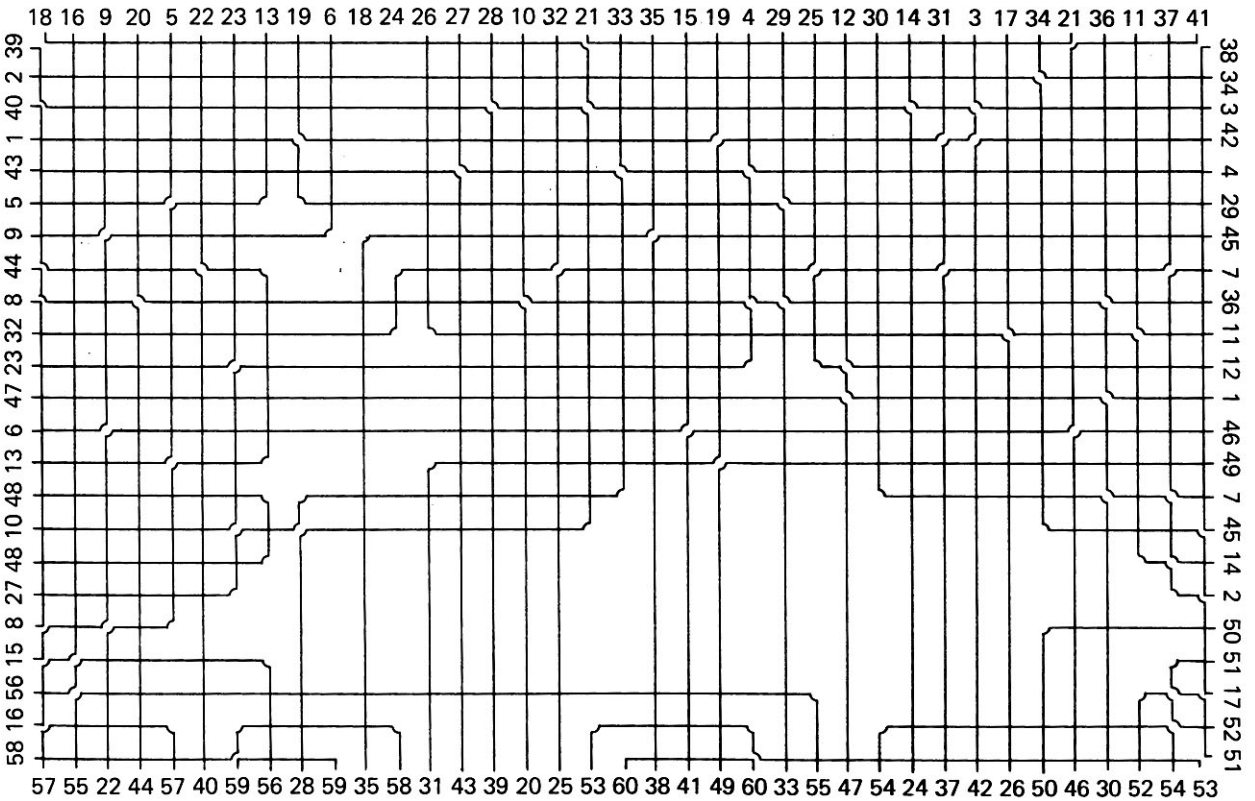


FIG. 17. Layout produced by the algorithm on a moderately complex problem.

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 multiterminal net.

6. *Layer Assignment*

We mentioned already in the introduction that every layout can be wired using four layers of interconnect [2]. 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 grid point, we can place two contacts, one connecting layers i_1 and i_2 and one connecting layers i_3 and i_4, where $1 \le i_1 \le i_2 < i_3 \le i_4 \le 4$. Thus, it is assumed that we can simultaneously connect 1 to 2 and 3 to 4, or, alternatively 1 to 3 and 4 to itself. However, we *cannot* simultaneously connect 1 to 3 and 2 to 4. Let L be any layout with k knock-knees. Then L can be wired in the model described above so that (a) there are only $O(k)$ contact cuts and (b) all but $O(k)$ length of wire runs in preferential layers i and j. The wiring is such that the usage of contact cuts (vias) and of the nonpreferential layers is limited to the vicinity of knock-knees. We refer the reader to [2] for details.

In specific situations one may wish to use fewer than four layers of interconnect. We now show how to wire a knock-knee layout using just *two layers* of interconnect after expanding it by a factor of 2 either in the y or in the x direction. This technique is closely related to the one proposed in [7]. 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 (refer to Figure 16 illustrating the wiring of the layout of Figure 12). There it is reflected upward or downward, and keeps running on layer 1 up to the adjacent new grid line. The other wire sharing the knock-knee runs on layer 2 all the way to its other knock-knee in the row, and so on. We use this strategy 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 segments. We can now use the new grid lines to mend together the solutions for adjacent old grid lines by placing vias wherever necessary.

7. *Computational Experience*

The algorithm presented in this paper was implemented in Pascal by G. Kaninke. The program has about 5000 lines and is available from the first author. Typical running times are 2 seconds for examples with 60 nets on a 30-by-30 grid (see Figure 17). The running times were measured on a SIEMENS 7760 2.8 MOPS machine.

REFERENCES

1. BENTLEY, J. L. Solution to Klee's rectangle problems. Unpublished notes. Carnegie-Mellon University, Pittsburgh, Pa., 1977.
2. BRADY, M., AND BROWN, D. J. VLSI routing: Four layers suffice. *Adv. Comput. Res. 2: VLSI Theory* (1984), 245–257.
3. FRANK, A. Disjoint paths in a rectilinear grid. *Combinatorica 2*, 4 (1982), 361–371.
4. LIPSKI, W., JR., AND PREPARATA, F. P. Finding the contour of a union of iso-oriented rectangles. *J. Alg. 1* (1980), 235–246.

5. McCreight, E. M. Priority search trees. Res. Rep. CSL-81-5. Xerox Corporation, Palo Alto, Calif., January 1982.
6. Preparata, F. P., and Lipski, W., Jr. Optimal three-layer channel routing. *IEEE Trans. Comput. 33*, 5 (May 1984), 427–437.
7. Rivest, R. L., Baratz, A. E., and Miller, G. Provably good channel routing algorithms. In *Proceedings of the CMU Conference on VLSI Systems and Computations* (Pittsburgh, Oct. 19–21). Computer Science Press, Rockville, Md., 1981, pp. 151–159.

RECEIVED NOVEMBER 1983; REVISED NOVEMBER 1984; ACCEPTED JUNE 1985