HIERARCHICAL CHANNEL ROUTER

Michael BURSTEIN

IBM T. J. Watson Research Center
Yorktown Heights, New York

Richard PELAVIN

University of Rochester
Rochester, New York

ABSTRACT

The channel routing problem is a special case of the wire routing problem when interconnections have to be performed within a rectangular strip having no obstructions, between terminals located on opposite sides of the rectangle. We present here a new channel routing algorithm, based on reduction of the problem to the case of a $(2 \times n)$ grid and on consistent utilization of a "divide and conquer" approach. For the current implementation of the algorithm, the running time is proportional to $N \times n \times \log(m)$, where N is the number of nets, n is the length of the channel (number of columns) and m is the width of the channel (number of tracks). Traditional technological restrictions are assumed, i.e. net terminals are located on vertical grid lines, two wiring layers are available for interconnections - one layer is used exclusively for vertical segments, another for horizontal and vias are introduced for each layer change. This algorithm consistently outperforms several known routers in quality of wiring. We tested the algorithm on several benchmark problems. One of them - Deutsch's "difficult example" - was routed with only 19 horizontal wiring tracks (the absolute minimum for this case), whereas all other known routers required 20 or more tracks.

INTRODUCTION

The problem of channel routing is a special case of the wire routing problem when interconnections have to be performed within a rectangular strip having no obstructions inside, between terminals located on opposite sides of the rectangle. Channel routing is an essential part of quite a number of layout design systems [1, 2, 3]. Although the first channel routers were introduced more then a decade ago [4], the problem still attracts the attention of researchers and designers and significant results have been obtained quite recently [5, 6, 7]).

We present here a new heuristic channel router. Traditional technological restrictions are assumed, i.e. net terminals are located on vertical grid lines, two wiring layers are available for interconnections - one layer is used exclusively for vertical segments, another for horizontal and vias are introduced for each layer change. Multi-terminal nets are allowed. To address the problem more precisely we need the following definitions. A ***channel*** is a pair of vectors of nonnegative integers - TOP and BOT - of the same dimension

$$TOP = t(1), t(2), \dots, t(n)$$

$$BOT = b(1), b(2), \dots, b(n)$$

with the condition that any positive integer having an entry in one of them $-TOP$ or $BOT-$ has at least one other entry, i.e. every positive integer represented in these vectors is represented at least twice. We assume that these numbers are the labels of grid points located along the top and bottom edge of a rectangle in a rectilinear grid (see Figure 1). Points having the same positive label have to be interconnected, i.e. they define nets. Zeros in TOP or BOT mean that no connection has be made to the corresponding point. The thickness of this rectangle must be determined by the router, i.e. we are allowed to add horizontal tracks to the rectangle, but vertical columns must remain intact.

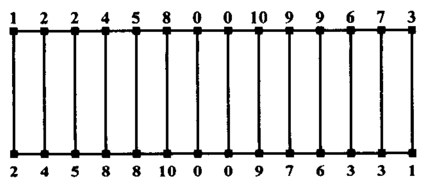


Figure 1.

The objective is of course to minimize the number of tracks, in other words to route within a channel of minimal thickness. Without loss of generality we assume that the set of labels presented in TOP and BOT is $[0, N]$, i.e. all the integers from 0 to N, where N is the number of nets. All vertical columns are numbered from 1 to n, where n hence denotes the length of the channel.

Two graphs are associated with the channel and are important for routing. Both have the set of nets $-\{1, 2, \dots, N\}-$ for the set of vertices. Every column $i (i = 1, 2, \dots, n)$ such that $t(i)$ and $b(i)$ are not zeros introduces a directed edge from the node $t(i)$ to the node $b(i)$ in the Vertical constraints Graph VG. So VG is the directed graph associated with the channel. We assume that edges of this graph are labeled by the corresponding column numbers (see Figure 2).

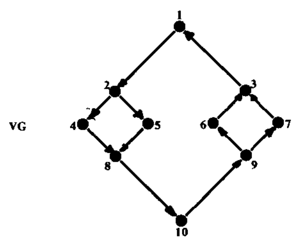


Figure 2.

The horizontal constraints graph HG is constructed as follows. With every net i we associate an interval $I(i)$, where left point of $I(i)$ is the minimal column number k such that $t(k)$ or $b(k)$ equals i, and the right point of $I(i)$ is the maximal such column number. HG is simply an intersection graph of this system of intervals. So it is by construction an interval graph. The density of HG, or clique number, which is the maximal number of intervals crossing the same vertical line, presents a lower bound for channel thickness, i.e. the lower bound for the number of horizontal tracks. This number will be the starting number of tracks of the router. Figure 3 illustrates HG for the example presented in Figure 1.

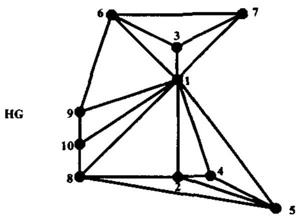


Figure 3.

These graphs $-VG$ and $HG-$ play very significant roles in the case of the ***restricted (no-dogleg)*** channel wiring problem, when the number of horizontal tracks on which any net can be positioned is limited to one. In this case the graph VG must not contain directed cycles and the resulting routing produces a proper coloring of both VG and HG. Note that the algorithm of [5] is unapplicable in the case when VG contains cycles. The "dogleg" router of [2] also may not produce a solution in case when there are directed cycles in VG generated by 2-terminal nets. In [8] it is shown that the restrictive channel routing problem is NP-complete because the circular arc coloring problem can be polynomially reduced to restrictive channel routing ([9] presents an exhaustive search routine for optimal solution of the restrictive case based on branch and bound technique). NP-completeness of the general channel routing problem was shown recently in [11].

It is worth mentioning that certain channels are impossible to wire with even an arbitrarily large number of tracks if there are no free columns. For example, the channel

$$TOP = 1, 2 \text{ and } BOT = 2, 1$$

is not routable. But addition of one vertical column results in channel

$$TOP = 0, 1, 2 \text{ and } BOT = 0, 2, 1$$

which can be routed in three tracks. The router presented in this paper will attempt to add a small number of free columns on either or both sides of the channel if it fails to find a legal routing within the given channel.

GENERALIZED CHANNEL ROUTING PROBLEM

In this section we introduce the generalized channel routing problem. The main reason behind this generalization is that there exists a mechanism for reduction of the generalized problem as well as the original channel routing problem, to a generalized problem of smaller size. An essential part of the generalized problem is that wiring is viewed in terms of grid cells rather then grid points and lines.

Suppose we are given a channel, i.e. a pair of vectors

$$TOP = t(1), t(2), \dots, t(n)$$

$$BOT = b(1), b(2), \dots, b(n)$$

Consider the rectilinear $(m \times n)$ grid $G = G(i, j)$, $i = 1, 2, \dots, m$, $j = 1, 2, \dots, n$. Here $G(i, j)$ denotes a cell of the grid at the intersection of the $i-$ th and $j-$ th strips. Grid lines are comprised of cell boundaries. We distinguish between horizontal and vertical boundaries. The horizontal boundary $H(i, j)$ $(i = 1, 2, \dots, (m-1)$ $, j = 1, 2, \dots, n)$ is merely a segment separating $G(i, j)$ and $G(i+1, j)$. Similarly, the vertical boundary $V(i, j)$ $(i = 1, 2, \dots, m,$ $j = 1, 2, \dots, (n-1))$ is a segment separating $G(i, j)$ and $G(i, j+1)$.

We assume that all boundaries of the grid are weighted by nonnegative integers; the weight of a boundary B will be denoted $W(B)$. We call these weights ***boundary capacities*** and impose on them the following restrictions:

(a) for every horizontal boundary H:

$$0 \leq W(H) \leq 1;$$

(b) capacities of the top and bottom vertical boundaries

$$W(V(k,j)) = 0, \text{ for } k = 1, m \quad (i = 1, 2, \dots, n).$$

The ***graph*** of the grid has cells as its vertices, and two cells are adjacent if they share a common boundary (vertical or horizontal). Any subtree R of this graph is called a ***route.*** We say that the route R crosses a boundary B if a pair of cells attached to this boundary constitutes an edge in R. We say that the route R turns within a cell G if it crosses two non-opposite boundaries of G. Figure 4 illustrates these notions.

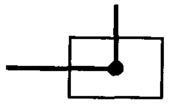


Figure 4.

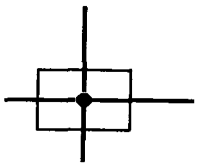


Figure 5.

We assume that the numbers $TOP = t(1), t(2), \dots, t(n)$ are assigned to the top strip of cells of the grid - $G(1,1), G(1,2), \dots, G(1,n)$ and similarly that $BOT = b(1), b(2), \dots, b(n)$ are assigned to the bottom strip $G(m, 1), \dots, G(m, n)$. In other words, these cells are labeled by the integers from TOP and BOT.

A ***routing*** is a set of routes $R(1), R(2), \dots, R(N)$, such that $R(i)$ contains all cells carrying the label i, $i = 1, 2, \dots, N$. Suppose a certain routing is specified and consider an arbitrary vertical boundary $V(i, j)$. Let LL (left load) denote the number of routes not crossing $V(i, j)$ but turning within the cell $G(i, j)$, and similarly RR (right load) denote the number of nets turning within $G(i, j+1)$ and not crossing $V(i, j)$. The load of the vertical boundary $V(i, j)$ is defined as

$$L(V) = \max \{LL, RL\}.$$

For any boundary B let $C(B)$ denote the number of routes crossing B.

Now we are ready for the final ***definition.***

Routing $R(1), R(2), \dots, R(N)$ is called ***legal*** if and only if

(i) for any boundary B:

$$C(B) \leq W(B);$$

(ii) for any vertical boundary V:

$$C(V) + L(V) \leq W(V).$$

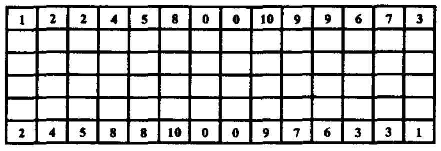
Generating a legal routing for a given grid with appropriate labels from *TOP* and *BOT* and with capacities assigned to the boundaries satisfying (a) and (b), presents a generalized channel routing problem. If all boundary capacities are equal to 1 (except those ruled out in (b)), the legal (generalized) routing corresponds to traditional channel routing, because condition (ii) rules out "overlapping vias" (Figure 5).

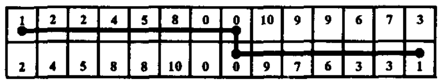
HIERARCHICAL ROUTING

Our approach to channel routing is based on the reduction to a generalized problem for a $(2 \times n)$ grid. The reduction is performed on every level of hierarchy in a way outlined below.

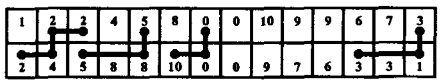
Consider the generalized problem for a $(m \times n)$ grid. Partition the grid into two parts - $([m/2] \times n)$ and $(]m/2[\times n)$ subgrids. Consider vertical strips of these subgrids as single supercells, i.e. factorize the cells of the subgrids making all cells with the same abscissa equivalent (consider them as one).

We end up with two horizontal strips, i.e. a $(2 \times n)$ grid. Vertical boundary capacities of this grid will be the sums of corresponding boundary capacities of the original grid. If the original capacities were all equal to 1, then the new capacities of the $(2 \times n)$ problem will represent the numbers of horizontal tracks crossing the boundaries. The process is illustrated on Figure 6. The $(2 \times n)$ wiring procedure is described in the next section.





First Partition (1-st level of the hierarchy). Net 1.

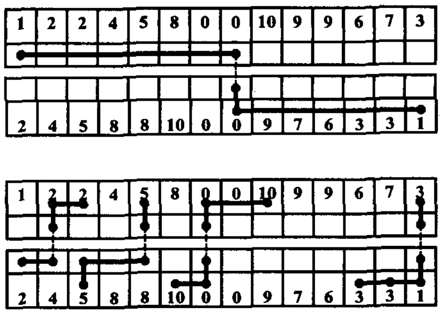


First Partition (1-st level of the hierarchy). Nets 2, 3, 5, 10

Figure 6.

Assuming that the routing within this $(2 \times n)$ grid is obtained, we now partition each of the horizontal strips of the grid into two parts, thus generating two $(2 \times n)$ subproblems (second level of hierarchy).

The global routes from the previous level define terminal positions for wiring of new $(2 \times n)$ subproblems. Figure 7 illustrates this step.



Second Partition (second level of the hierarchy).

Figure 7.

We proceed with the resulting $(2 \times n)$ routing in the same manner until single cell resolution is reached. At this level the routing will be completed.

It is worth mentioning that at every level of hierarchy we obtain routes of all nets, so a net is never positioned exactly before all other nets are routed globally at least on previous levels of hierarchy.

If the $(2 \times n)$ router fails to obtain a legal routing, i.e. the routing satisfying (i) and (ii) of the previous section, then condition (ii) is relaxed. In other words the value $W(V) - C(V) - L(V)$ may become equal to (-1). In this case all vertical capacities of boundaries in the same horizontal strip as V are increased by 1. The increase is equivalent to the addition of a new horizontal track to the original problem.

WIRING WITHIN $(2 \times n)$ GRID

Denote $G(i, j)$ the cells of our $(2 \times n)$ grid, $i = 1, 2, \ j = 1, 2, ..., n$. Wiring within the grid of thickness 2 is performed by the algorithm TBN (two-by-n) which allocates routes for all nets, one at a time.

With each interconnection route of a net we associate a cost, which is comprised of the costs of boundary crossings;it is simply the sum of those costs.

The TBN wiring algorithm is a linear algorithm that finds the minimal cost tree interconnecting a set of elements located on a $(2 \times n)$ grid.

The input for the algorithm consists of

1) The terminal locations on the $(2 \times n)$ grid: *EL* is a $(2 \times n)$ Boolean matrix; if $EL(i,j) = TRUE$ then the corresponding cell is a terminal cell.

2) Costs of boundary crossings :
   (h) *HC* is an n element vector which stores the costs of crossing the horizontal boundaries; $HC(j)$ indicates the cost that must be added if a wire crosses the boundary between $G(1, j)$ and $G(2, j)$;
   (v) *VC* is a $(2 \times (n - 1))$ matrix that stores the costs of vertical boundary crossings; $VC(i, j)$ indicates the cost which must be added if a wire crosses the boundary between $G(i, j)$ and $G(i, j + 1)$, $i = 1, 2$.

Note : if the barrier cost is greater than LRG, which is set to a large number, the corresponding boundary is blocked and a wire can not pass through this cell boundary. In other words, cost of crossing of the blocked boundary in infinitely high.

Matrices HC and VC represent our cost-functions. They should reflect the boundary capacities and via conditions. Clearly, $VC(i, j)$ and $HC(j)$ must be a decreasing functions of the number of wiring tracks crossing the corresponding boundaries. Selection of the proper cost functions is extremely important because of its influence on the quality of the final routing. We discuss this point in the next section.

The output of TBN consists of the minimal interconnection tree.

The TBN algorithm is a modification of an algorithm developed in [10], which finds a Steiner tree that interconnects a set of elements located on a $(2\times n)$ grid. But our algorithm solves the problem for an arbitrary cost-function associated with crossing a boundary (matrices HC and VC), whereas the algorithm of [10] assumes the rectilinear Steiner tree, which is the case when all these costs are equal to 1.

We need the following definitions.

1) Let $T^1(k)$ denote the minimal cost tree which interconnects the following set of cells:

$$\{G(i,j) : (j\leq n)\&EL(i,j)\} \cup \{G(1, k)\}.$$

2) Let $T^2(k)$ denote the minimal cost tree which interconnects the following set of cells:

$$\{G(i,j) : (j\leq n)\&EL(i,j)\} \cup \{G(2, k)\}.$$

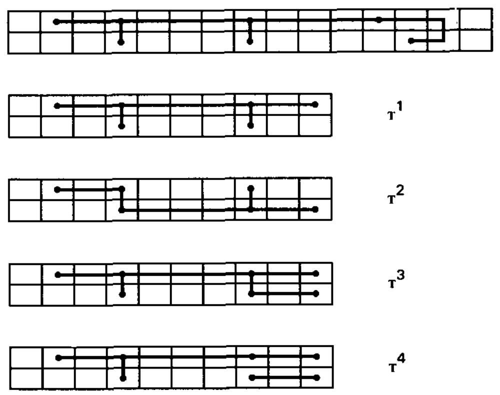
3) Let $T^3(k)$ denote the minimal cost tree which interconnects the following set of cells :

$$\{G(i,j) : (j\leq n)\&EL(i,j)\} \cup \{G(1, k) ; G(2, k)\}.$$

4) Let $T^4(k)$ denote the minimal cost forest, consisting of two different trees T^* and T^{**} : T^* uses cell $G(1, k)$, T^{**} uses cell $G(2,k)$ and the set

$$\{G(i,j) : (j\leq n)\&EL(i,j)\}$$

is interconnected by either one of them (this means that the trees have to be joined later).



Dynamic Programming. Wiring within (2 × N) grid.

Figure 8.

We compute the trees $T^i(k + 1)$, $i = 1, 2, 3, 4$ recursively from $T^i(k)$ (Figure 8.) This procedure is sometimes referred to as dynamic programming. First we need to construct initial trees to start the recursion.

Denote by $FIRST$ and $LAST$ the abscissas of the leftmost and rightmost terminal cells, i.e.

$$FIRST = \min \{ k : EL(1,k)\vee EL(2,k)\},$$

$$LAST = \max \{ k : EL(1,k)\vee EL(2,k)\}.$$

Trees $T^i(k)$ for $k\leq FIRST$ are computed trivially and serve as a basis for recursion. In fact, for $k\leq FIRST$ $T^l(k)$ $(l = 1, 2)$ consists of a single vertex $-G(l, k)$, $T^4(k)$ consists of the disjoint pair of vertices $-G(1, k)$ and $G(2, k)$. However, $T^3(k)$ is obviously a path $-G(1, k), \dots , G(1, s)$, $G(2, s), \dots , G(2, k)$ where $1\leq s\leq k$ and

$$H(s) + \sum_{i=s}^{i=k-1} V(1,i) + V(2,i)$$

is ***minimal.***

In rectilinear case, when all costs are the same, $T^3(k)$ is the adjacent pair of vertices $-G(1, k)$ and $G(2, k)$. But in general, the cost $H(k)$ might be too high and detouring might result in cheaper route (see Figure 9.).

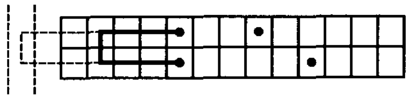


Figure 9.

Suppose that $FIRST\leq k\leq LAST$ and the trees $T^i(k)$, $i = 1, 2, 3, 4$ are constructed. To construct $T^j(k + 1)$ we enumerate all possible extensions from $T^i(k)$, $i = 1, 2, 3, 4$, and select the cheapest one. For example, to construct $T^1(k + 1)$ we select the cheapest way of adding $G(1, k + 1)$ to each of the $T^i(k)$, $i = 1, 2, 3, 4$, and select the cheapest among them. Note, that $T^4(k + 1)$ always becomes either trivial extension of $T^4(k)$ or one of the vertices $G(1, k + 1)$ or $G(2, k + 1)$ is isolated whereas the other component is identical to $T^1(k + 1)$ or $T^2(k + 1)$.

When the value $k = LAST$ is reached our trees are covering all terminal cells. We temporarily select the cheapest among $T^1(LAST)$, $T^2(LAST)$, $T^3(LAST)$ and denote it by T. In rectilinear case it is obviously the interconnection we need. But it is possible that the costs $H(s)$ for $s\leq LAST$ are so high that it is cheaper to take a right side detour. In a way symmetric to the construction of $T^3(FIRST)$ we select a path $-G(1, LAST), \dots , G(1, s)$, $G(2, s), \dots , G(2, LAST)$, where $LAST<s\leq n$ and

$$H(s) + \sum_{i=LAST}^{i=s-1} V(1,i) + V(2,i)$$

is ***minimal.***

Then we join the components of $T^4(LAST)$ by this path, (see Figure 10), compare the cost of resulting interconnection with the cost of T and finally select the cheapest.

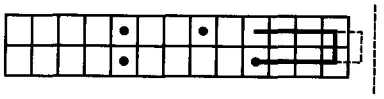


Figure 10.

Note that at each stage of recursion we make a constant number of comparisons, and the total computation time is $O(n)$ because the initiation stage - computing $T^3(FIRST)$ - takes $O(FIRST)$ time, construction of T is performed in time $O(LAST - FIRST)$ and final right side detour is computed in at most $O(n - LAST)$ time.

In case when the interval $[FIRST, LAST]$ is small in comparison with $[1, n]$ we can limit the detours from either side (Figures 9, 10) in order to eliminate unnecessary computations. We usually limit the detour outside of the $[FIRST,LAST]$ segment by 2 or 3 , so in most cases the time spent for $2 \times n$ routing will be $O(LAST - FIRST)$. On the other hand, we may be forced at some point to cross a boundary having an infinite cost LRG. This will result in non resolvable conflict for routing because it means utilization of the same piece of the vertical track for two different nets. In order to avoid this conflict we permit to wire outside of the channel, in other words we add additional vertical tracks at the left or right ends of the channel. This is achieved by introduction of additional boundaries $V(1, i)$, $V(2, i)$, $H(i)$ for $i \leq 0$ and for $i \geq n$ and assigment of high but not infinite cost to their crossing.

COST FUNCTIONS

Costs of crossings of horizontal boundaries - HC - are chosen very simply. Because, by definition, the capacity W of a horizontal boundary $B(j)$ is either 0 or 1, $HC(j)$ is set to LRG if $W(B(j)) = 0$, i.e. if this boundary is already crossed by one of already routed nets, and $HC(j) = C1$ otherwise, where $C1$ is a positive constant.

Costs of vertical crossings - VC - are more complex. If B is a vertical boundary $B(i, j)$ ($i = 1, 2,\ j = 1, 2 ,..., n$), $W(B)$ its initial capacity, $C(B)$ the number of already routed nets crossing B, and $L(B)$ the load, then set

$$X = W(B) - C(B) - L(B)$$

and

$$VC(i,j) = (C2)^{(-X)} + F,$$

where $C2$ is another positive constant and F is an additive corrector, which is determined by the vertical constraints graph VG and is not dependent on previously routed wires. The corrector F reflects relative positions of the nodes of VG corresponding to nets whose intervals in HG cross the j -th vertical line. It makes an influence on wiring only if there are several minimal interconnections of equal cost. In order to avoid random picking of the wire route at the starting stage of TBN the values of F can be set to force nets to be either above or under the cut line. If the currently routed wire belongs to a directed cycle in a subgraph of VG spanned by nets crossing the j-th vertical line, F is set to 0. If it does not belong to a directed cycle then we can estimate how many nets should be above or below the current net in this subgraph, and set F accordingly (F can be both negative and positive). Setting F to 0 all the time also does not hurt, if the net will be rerouted later, as is mentioned in the next section. But proper setting of correctors may result in good wire packing the first time.

The constants $C1$ and $C2$ are defined experimentally during the "fine-tuning" process of algorithm implementation. Moreover, since the range of X is the above formulas is usually small the costs of crossing vertical boundaries with specific X can be precumputed and altered during the "fine-tuning".

In case when the TBN is unable to generate an interconnection of the total cost less then infinite (LRG) we allow detouring outside the channel. Costs of crossing the boundaries $V(1, i)$, $V(2, i)$, $H(i)$ for $i \leq 0$ and for $i \geq n$. are all equal to some big number, but significantly less then LRG; they are chosen in such a way, that a finite cost interconnection within the channel limits is always cheaper then an interconnection crossing these outer boundaries, but crossing of those is still cheaper than LRG (Figures 9, 10).

REFINEMENTS

Because the presented $(2 \times n)$ wiring algorithm performs wiring of one net at a time, it is dependent on the order in which the nets are routed. But the rerouting procedure decreases this order dependency. Once all the nets are imbedded into the $(2 \times n)$ grid we remove one of them, update the boundary capacities, compute new boundary costs (all other nets remain imbedded) and again run the TBN procedure (against the background of all the other wires). We perform this rerouting for every net in random order. Very often the nets choose exactly the same route they had before. But several of them in fact get rerouted, improving the distribution of the remaining boundary capacities. Sometimes the rerouting allows us to get rid of negative boundary capacities, thus eliminating the necessity of introduction of new horizontal track. We experimentally observed that in repeated rerouting (third and subsequent applications of TBN) nets tend not to choose different routes (which is not the case when maze-type rerouting is applied for a general grid). So, more then one cycle of rerouting appears impractical for the case of the $(2 \times n)$ grid. The small thickness of the grid - 2 strips - is probably the reason why the rerouting process converges so fast.

Another trick, which sometimes helps a lot, is rerouting within a sliding $2 \times n$ window; at any level of hierarchy we may select a pair of consecutive rows, usually with largest number of negative boundary capacities, and try to reroute the net segments passing through them.

RESULTS

The complexity of the presented algorithm can be estimated as follows. The time spent for routing of a single net at any level of hierarchy is proportional to the total length of horizontal segments of this net (assuming that a constant time is spent for "detouring"). It is observed that the total length of a net at any level of hierarchy is bounded by $O(n)$, where n the length of the channel (number of columns). Since there are N nets, and $\log_2 m$ levels of hierarchy, where m the width of the channel (number of tracks), the upper bound for algorithm complexity can be expressed as

$$N \times n \times \log(m).$$

This figure should be considered as a worst case behavior, when almost all nets expand through the whole channel. For practical purposes the algorithm is only $O(\log(m))$ times slower then extremely fast router [6]. But we believe that it outperforms the known routers in quality of wiring. It often requires fewer tracks to complete the job. For the channel used as an illustration in this paper (Figure 1), it requires only 5 tracks (which is optimal in this case, although the density is equal to 4). We consider this fact to be a significant achievement because we could not produce the five track routing of this channel manually; neither could several of our colleagues within a reasonable time. The example consists of only 10 nets on 5 x 14 grid. Anyone who will try to find the routing manually in 10 minutes (an intriguing puzzle!) will appreciate the final routing produced by the algorithm - Figure 11.

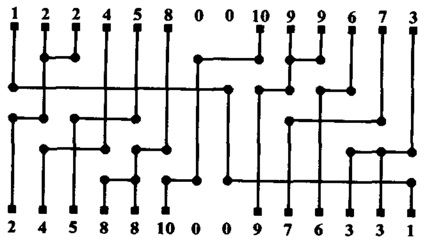


Figure 11.

The algorithm was tested on several artificially created random examples with approximately 500 nets, channel length of 1000. The results were close to optimal, in 7 out of 10 examples the lower bounds for channel thickness were achieved, in 2 of them the achieved thickness was 1 track above the lower bound, and only 1 required two additional tracks.

We also tested the algorithm on the well-known "Difficult Example" of D. Deutsch [2], which has recently become a benchmark test for channel routing algorithms. The channel has density of 19. But no known channel router was able to produce the wiring within 19 tracks. The "dogleg router" of D. Deutsch [2] produced the solution with 21 tracks. T. Yoshimura and E. S. Kuh [5] were able to come up with 20 track wiring. The 20 tracks are also required by "Greedy" router of R. Rivest and C. M. Fiduccia [6]. Note that the ***restrictive*** routing of this example (when a net can occupy at most one track) requires at least 26 tracks.

In our first experiment the interconnection data was copied from [6]. It subsequently came to our attention that the data presented in [6] was not correct: it actually contains only first "three quarters" of the original channel of D. Deutsch. We were not aware of this fact when the Research Report [12] was written and referred to this partial channel as to the complete "difficult example". R. Rivest and C. M. Fiduccia kindly informed us about the "bug". The 19 track wiring of the complete "difficult example" is reproduced on the Figure 12. It uses 336 vias, 31 "doglegs" , the total wire length equals to 5023, only one net - 51 - is "detouring", i.e. using the columns outside of it's interval span, however several other nets are "detouring internally". This solution is slightly different (better!) from our first 19 track routing of the "difficult example" reported in [13] ; it was obtained by varying costs of boundary crossings and rerouting within sliding $2 \times n$ subgrids. This is the best we were able to achieve in terms of numbers of vias and "doglegs" after about 20 different program runs. We were unable to come close to the solution produced manually by D. Deutsch ([14], manual editing of 21 track solution) which uses only 9 doglegs and no "detouring".

Remark. The "difficult example" is not a pure channel routing problem because there is a requirement that a net #1 (connecting all points labeled 1) has to be brought out through the left side of the channel. But we "purified" it by appending 1 to the beginning of the TOP vector and 0 to the BOT of this example, thus forcing the net 1 to go out.

ACKNOWLEDGEMENTS

Discussions of the hierarchical routing idea with S. J. Hong and R. Nair were very useful. A. Sangiovanni-Vincentelly supplied us with the correct data on the "difficult example". We are also grateful to E. Yoffa, D. Deutsch, R. Rivest, C. M. Fiduccia, R. Pinter and T. Szymanski for helpful comments and suggestions.

REFERENCES

[1] Soukup J., "Circuit Layout", Proc. IEEE, Vol. 69, 10, 1981, 1281-1304.

[2] Deutsch D., "A dogleg channel router", Proc. 13th DA Conf., 1976, 425-433.

[3] Persky G., Deutsch D. N., Schweikert D. G., "LTX - A minicomputer-based system for automatic LSI layout", J. DA & Fault-Tolerant Comput., vol. 1, 3, 1977, 217-256.

[4] Hashimoto A., Stevens J., "Wire routing by optimizing channel assignment", Proc. 8th DA Conf., 1971, 214-224.

[5] Yoshimura T., Kuh E.S., "Efficient Algorithms for Channel Routing", IEEE Trans. on CAD of ICs and Systems, Vol. CAD-1, No. 1, 1982, 25-35.

[6] Rivest R.L., Fiduccia C.M.,"A 'Greedy' Channel Router", Proc. 19th DA Conf., 1982, 418-424.

[7] Rivest R.L., Baratz A., Miller G., "Provably good channel routing algorithms", Proc. 1981 Carnegie-Mellon Conf. on VLSI, October 1981, pp. 178-185.

[8] LaPaugh A.S., "Algorithms for Integrated Circuit Layout : An Analytic Approach", PhD thesis, MIT, Laboratory for Computer Science, Cambridge, MS, 1980.

[9] Kernighan B.W., Schweikert D.G., Persky G., "An Optimum Channel-Routing Algorithm for Polycell Layouts of Integrated Circuits", Proc. 11th DA Workshop, 1973, pp. 26-46.

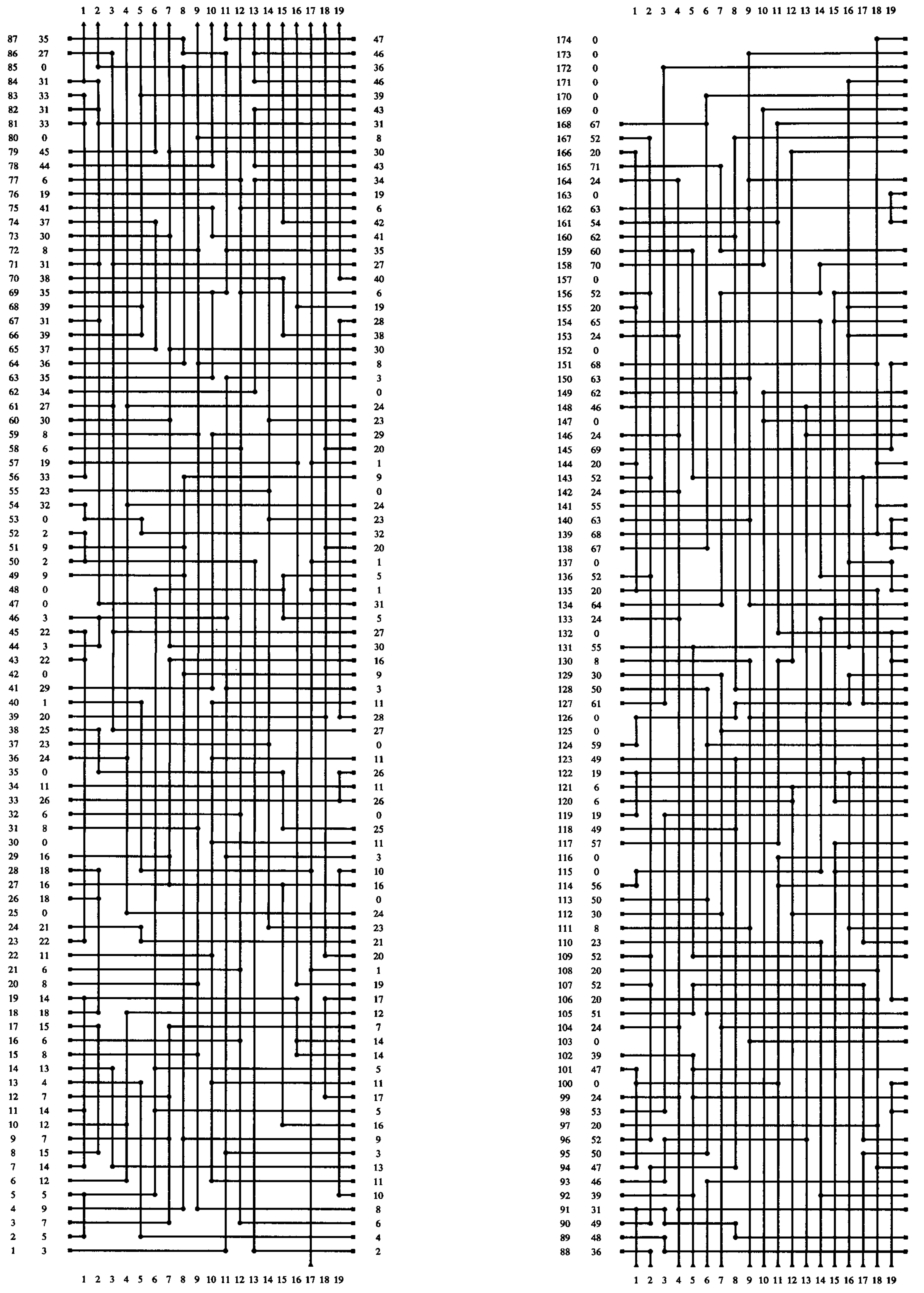
[10] Aho A., Garey M. R., Hwang F. K., "Rectilinear Steiner Trees: Efficient Special Case Algorithms", Networks, 7, 1977, pp. 37-58

[11] Szymanski T. G., "Dogleg Channel Routing is NP-complete", unpublished manuscript, Bell Laboratories, Murray Hill, 1982

[12] Burstein M., Pelavin R., "Hierarchical Channel Router", IBM T. J. Watson Research Center, RC 9715, 1982.

[13] Burstein M., Pelavin R., "Routing of the 'Difficult Example' ", IBM T. J. Watson Research Center, RC 9830, 1983.

[14] Deutsch D., Private Communication, 1/26/1983.



19 track wiring of the "difficult example"

Figure 12.