Single-Row Routing in Narrow Streets

SANYONG HAN AND SARTAJ SAHNI, MEMBER, IEEE

Abstract—**We develop fast linear time algorithms for single row routing when the upper and lower street capacities are less than or equal to three. A similarly fast algorithm is developed for the case when one of the streets has a capacity 1 and the other has an arbitrary capacity. Experimental results show that our algorithms are many times faster than previously developed algorithms.**

I. Introduction

SO has proposed a systematic approach to the interconnection problem of large multilayer printed circuit boards in which pins and feedthroughs are uniformly spaced on a rectangular grid [6]. This approach consists of a systematic decomposition of the general multilayer wiring problem into a number of independent single layer, single-row routing problems. There are five phases in this decomposition [8], [5]:

1. Via assignment,
2. Linear placement of via columns,
3. Layering,
4. Single row routing,
5. Via elimination.

In this paper, we are concerned only with the fourth phase: Single row routing. In the single-row routing problem, we are given a set $V = \{1, 2, \cdots, n\}$ of n nodes that are evenly spaced along a straight line; and a set $L = \{N_1, N_2, \cdots, N_m\}$ of nets. Each net represents a set of nodes that are to be made electrically equivalent. The nets satisfy the following conditions:

(i) $N_i \cap N_j = \phi, \quad i \neq j$

(ii) $\bigcup_{i=1}^{m} N_i = \{1, 2, \cdots, n\}$

$j \epsilon N_i$ is a *touch point* of net i. The nets are to be realized in a single layer by the use of nonoverlapping wires that are composed solely of horizontal and vertical segments. Fig. 1(a) shows some of the legal ways to realize the net $\{1, 3, 6, 9\}$. The wire layout must satisfy the additional requirement that a vertical cut made at any point along the axis formed by the nodes can intersect at most one horizontal segment from each net. Thus the wiring of Fig. 1(d) is illegal.

The area above the line of nodes is called the *upper street* while that below this line is the *lower street*. Each street has *tracks* that run parallel to the line of nodes (Fig. 2). Horizontal wire segments must be layed in tracks and no track may hold more than one wire segment at any point (of course, several non overlapping wire segments may be packed into the same track). Let K_u and K_l, repectively, denote the number of tracks in the upper and lower streets. Fig. 2 shows one way to realize the nets $(N_1, N_2, \cdots, N_5) = (\{1, 7\}, \{2, 8\}, \{3, 6\}, \{4, 9\}, \{5, 10\})$ when $K_u = 2$ and $K_l = 3$.

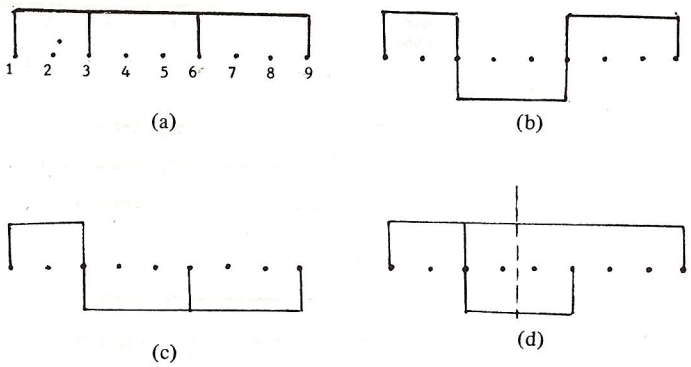

Fig. 1.

In this paper, we are specifically concerned with obtaining net realizations (when they exist) given V (node set), L (net set), K_u (upper street capacity), and K_l (lower street capacity).

This problem has been studied before. Kuh *et al.* [3] and Tsukiyama *et al.* [9] developed necessary and sufficient conditions for a net set to be realizable. In [3], a simple construction is used to show that when K_u and K_l are sufficiently large, L is realizable. This construction, however, results in an algorithm of complexity $O(m!n)$ where $|V| = n$ and $|L| = m$. Raghavan and Sahni [4] have developed an algorithm of complexity $O(k! * k * n * \log k)$ where $k = K_u + K_l$ for this problem. This algorithm is quite practical when k is small but impractical when k is large. For the case $K_u \leqslant 2$ and $K_l \leqslant 2$, Tsukiyama *et al.* [9] have developed an $O(mn)$ algorithm. This algorithm is, however, slower than that of [4].

Here, we shall develop fast $O(n)$ algorithms for the case $K_u \leqslant 3$ and $K_l \leqslant 3$ as well as the case $K_l = 1$ and K_u arbitrary. Experimental results show these algorithms to be several times faster than that of [4] for these street capacities. The case K_u and $K_l \leqslant 3$ is actually solved by developing algorithms for the subcases $K_u = K_l = 2$; $K_u = K_l = 3$; and $K_u = 3$, $K_l = 2$. These algorithms together with that for $K_l = 1$ and K_u arbitrary cover all the possibilities for $K_u \leqslant 3$ and $K_l \leqslant 3$. Note that the case of $K_l = 0$ and K_u arbitrary is trivially solved in linear time.

Before presenting the algorithms, we introduce some notation. Following [4], nodes are classified by type:

(a) Node i is of *type B* if it is the left extreme node of a net,

Manuscript received April 22, 1983. This work was supported in part by the National Science Foundation under Grant MCS 80-005856.

The authors are with the Department of Computer Science, University of Minnesota, Minneapolis, MN 55455.

0028-0070/84/0700-0235$01.00 © 1984 IEEE

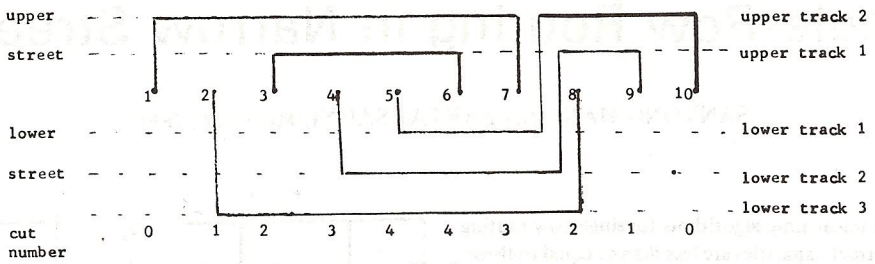

Fig. 2.

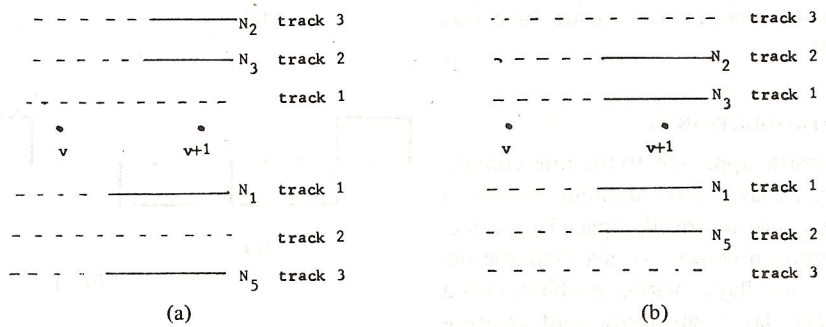

Fig. 3.

i.e., it is the *beginning* node for some net. The type B nodes in Fig. 2 are 1, 2, 3, 4, and 5.

(b) A *type E* node is a node at which a net i ends. I.e., it is a right extreme node for some net. The type E nodes in Fig. 2 are 6, 7, 8, 9, and 10.

(c) A *middle* or "*nonextreme*" node of a net i is a *type M* node. If a and b are, respectively, the B and E nodes of a net i, then $N_i - \{a, b\}$ is the set of middle or type M nodes for this net.

The *cut number* of a node is the number of nets covering that node, i.e., the number of nets between whose extreme nodes the node in question lies. The net (if any) with touch point i is not included in this count. Cut numbers for the nodes of Fig. 2 are given in this figure.

A *k-zone* [9] is a segment of the node axis beginning at a B type node with cut number k and ending at the next E type node with this cut number. Thus all cut numbers within a *k-zone* are greater than or equal to k. The example of Fig. 2 has the following zones:

0-zone: nodes 1 through 10
1-zone: nodes 2 through 9
2-zone: nodes 3 through 8
3-zone: nodes 4 through 7
4-zone: nodes 5 through 6

In general, of course, a net list may have several *k-zones* for any given k.

In all our algorithms, we shall attempt to obtain the net realization by processing the nodes from left to right. As in [4], each possible ordering of nets that can be encountered by a node is maintained simply as an ordered list of net indexes. There is no explicit division between the upper and lower streets. Such an explicit division is neither desirable nor necessary. Only the relative order (top to bottom) is relevant; this same order is reflected in the realization.

An advantage of this approach is that functionally equivalent situations, such as the ones in Fig. 3(a) and (b), are not treated separately.

II. $K_u = k$ AND $K_l = 1$

As noted in Section I, the layout will be constructed by scanning the nodes from left to right. Assume that the layout has been obtained upto node $i - 1$. Let $(N_1, N_2, \cdots, N_b)$ be the permutation of the nets that arrive at node i from the left (Fig. 4(a)). We shall show how to obtain an arrival permutation for node $i + 1$ such that this permutation will lead to a feasible layout if one exists.

If node i is of type E or M, then it must be a touch point for one of the nets N_j, $1 \leqslant j \leqslant b$. If it is a touch point for the net N_p and $p < b - 1$, then the routing cannot be completed with $K_l = 1$ because at least two nets (N_{b-1} and N_b) must pass below node i (Fig. 4(b)). If $p \geqslant b - 1$ and i is of type M, then the arrival permutation for node $i + 1$ is $(N_1, N_2, \cdots, N_b)$. If $p \geqslant b - 1$ and i is of type E, then the arrival permutation for node $i + 1$ is obtained from $(N_1, N_2, \cdots, N_b)$ by simply deleting N_p.

When node i is of type B, we need to be somewhat careful about the construction of the arrival permutation for node $i + 1$. It is easy to see that there are at most two possibilities for this permutation (Fig. 5).

Let N_x be the net that begins at i. If $b > k$, then no feasible layout is possible. So, assume that $b \leqslant k$. When $b = 0$, (N_x) is the only permutation possible for $i + 1$. When $b > 0$, we have two possibilities: $(N_1, N_2, \cdots, N_{b-1}, N_x, N_b)$ and $(N_1, N_2, \cdots, N_{b-1}, N_b, N_x)$. We need to determine which of these two will lead to a layout.

Let v_b and v_z, respectively, be the end points of nets N_b and N_x. Let $v = \min\{v_b, v_x\}$. We observe that if any of the nets $\{N_1, N_2, \cdots, N_{b-1}\}$ have touch points before v, then the layout cannot be completed and this will be detected as soon as

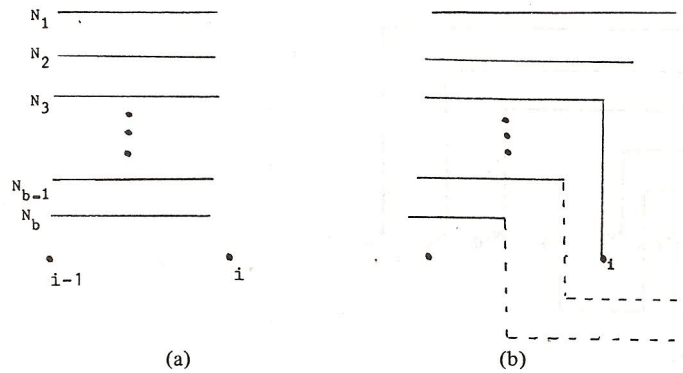

Fig. 4.

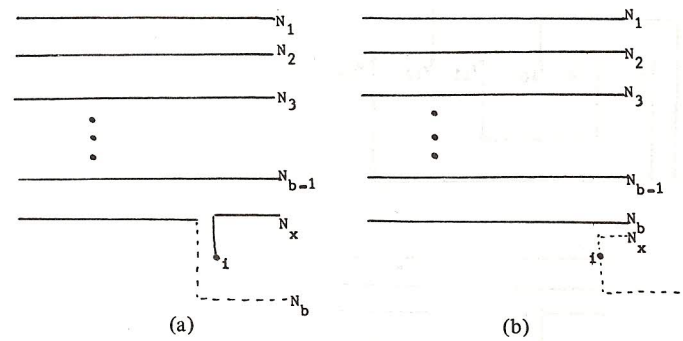

Fig. 5.

one of these points is reached. So, assume that we have the situation of Fig. 6. The cut number at i is b. Let $\{v_1, v_2, \cdots, v_r\}$ be the touch points of N_b and N_x that are no greater than v. It is readily seen that the cut number at each of these points must be at least b.

If every v_j has a cut number of b, then $N_1, N_2, \cdots, N_b$, and N_x are the only nets that arrive at any v_j and regardless of whether the permutation is $(N_1, N_2, \cdots, N_b, N_x)$ or $(N_1, N_2, \cdots, N_{b-1}, N_x, N_b)$, the corresponding net can be wired to it's touch point.

If at least one of the v_j's has a cut number that is larger than b, then there is at least one net that starts after i and arrives at j. This net cannot possibly be above both N_b and N_x. Hence the net with touch point v_j cannot be wired at v_j if it does not occupy the bottom position at i.

Thus if both nets N_b and N_x have touch points in $\{v_1, \cdots, v_r\}$ that have a cut number greater than b, the layout cannot be completed with $K_l = 1$. Therefore, the choice between $(N_1, N_2, \cdots, N_b, N_x)$ and $(N_1, N_2, \cdots, N_{b-1}, N_x, N_b)$ is made by determining if there is a v_j with cut number greater than b. If there is, then pick any one of these and place the net corresponding to this point at the bottom.

By choosing appropriate data structures, the above strategy can be implemented in $O(n)$ time. Also, the resulting algorithm has negligible overhead and can be expected to run many times faster than earlier algorithms.

Example 2.1: In this simple example, we have a net list which has 6 nets and 12 nodes. Net list L is given as follows:

$$N_1 = \{v_1, v_{10}\}, N_2 = \{v_2, v_{12}\}, N_3 = \{v_3, v_{11}\}$$

$$N_4 = \{v_4, v_7\}, N_5 = \{v_5, v_8\}, N_6 = \{v_6, v_9\}.$$

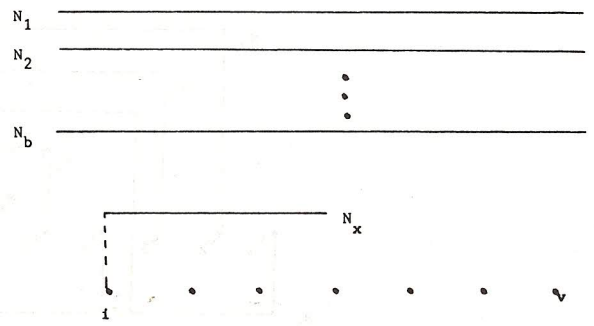

Fig. 6.

When $K_u = 5$ and $K_l = 1$, the above net list can be realized, as in Fig. 7. At v_2, N_2 is placed above N_1 as only N_1 has a touch point, v_{10}, with cut number greater than 1. The same rule is used to place N_4 and N_5 at v_4 and v_5. However, N_3 and N_6 can be placed at either of the two possible locations in the incoming permutation as their touch points do not have a cut number larger than their initial cut numbers. ■

III. $K_u = K_l = 2$

If we know the net permutation that arrives at an E or M type node, then we may easily determine whether or not the net corresponding to this touch point can, in fact, be wired to this point when $K_u = 2$ and $K_l = 2$. Also, the net permutation leaving the node is unique and easily determined.

Therefore, we may restrict our discussion to the case when we are scanning a node i such that node i is of type B. Let $(N_1, N_2, \cdots, N_b)$ be the permutation that arrives at this node and let N_x be the net that begins at this node. Clearly, if $b > 3$, then the net list cannot be realized with $K_u = K_l = 2$. If $b = 0$, then the permutation arriving at $i + 1$ is (N_x). If $b = 1$, then there are two possiblities for the permutation arriving at $i + 1$: (N_1, N_x) and (N_x, N_1). However, these are symmetric and since $K_u = K_l$, if a layout can be achieved using (N_x, N_1), then it can be also achieved using (N_1, N_x). So, we may simply use the permutation (N_1, N_x).

When $b = 2$, there are three possibilities, (N_1, N_2, N_x), (N_1, N_x, N_2) and (N_x, N_1, N_2) for the permutation arriving at node $i + 1$. Let $\{v_1, v_2, \cdots, v_r\}$ be the touch points of N_1, N_2, and N_x that are to the right of node i. If all of these have a cut number that is less than 3, then the net corresponding to any v_i can be wired to v_i independent of the permutation that arrives at v_i. In this case, we may use any one of the three possible permutations for node $i + 1$.

Assume that at least one of the v_i's has a cut number that is 3 (if any node has a cut number > 3, then the net set cannot be realized with $K_u = K_l = 2$). Let v_j be the leftmost such touch point. Let $N_a \in \{N_1, N_2, N_x\}$ be the net with touch point v_j. Let (N_b, N_c, N_d, N_e) be the net permutation that arrives at v_j. Clearly, when $K_u = K_l = 2$, the net connection to v_j can be completed iff $N_a = N_c$ or $N_a = N_d$.

Among the three possibilities for the permutation arriving at $i + 1$, there is exactly one that guarantees $N_a = N_c$ or $N_a = N_d$. This is the permutation that has N_a in the middle. While it may be possible to complete the wiring to v_j using one of the other

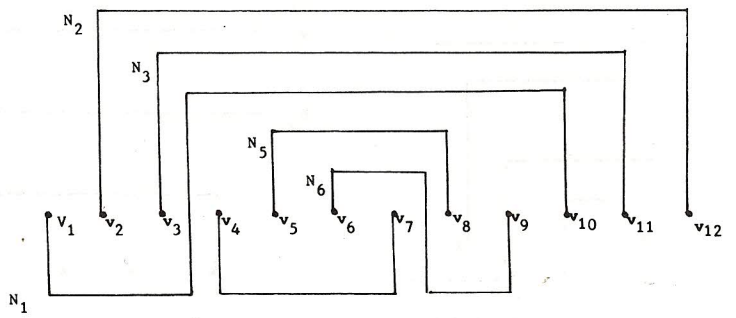

Fig. 7.

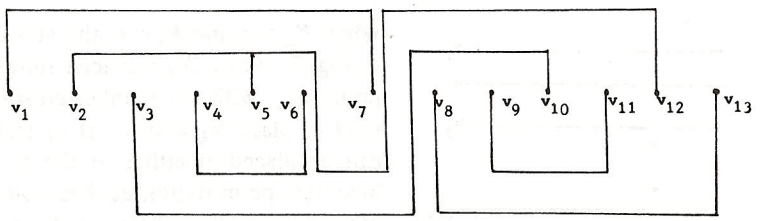

Fig. 8.

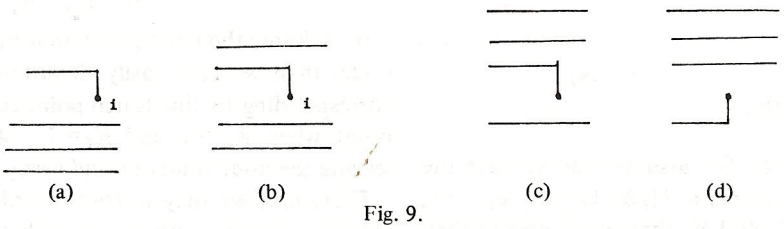

Fig. 9.

two permutations (this happens when one of the other two nets ends before v_j), it is not too difficult to see that if a net set can be realized using one of these other permutations (when $K_u = K_l = 2$), it can also be realized using the permutation with N_a in the middle.

The final case to consider is $b = 3$. Now we have exactly two possible permutations for $i+1$: (N_1, N_2, N_x, N_3) and (N_1, N_x, N_2, N_3). Let v_2 and v_x, respectively, be the end points of nets N_2 and N_x. Let $v = \min\{v_2, v_x\}$. If any one of the nodes $i+1, i+2, \cdots, v-1$ is a touch point of N_1 or N_3 or a node of type B, then the wire layout cannot be completed with $K_u = K_l = 2$; independent of the permutation chosen for $i+1$. So assume that there is no such a node. Regardless of which permutation is chosen, both N_2 and N_x can be wired to their touch points j, $i < j < v$ and the permutation leaving v is (N_1, N_a, N_3) where $a = 2$ if $v_2 > v_x$ and $a = x$ otherwise.

So, when $b = 3$ we may use either permutation. The preceding discussion leads to the following theorem:

Theorem: If a net set L can be realized with $K_u = K_l = 2$, then it can be realized by handling B nodes in the manner described above.

Example 3.1: Consider the net list $L = \{N_1, N_2, \cdots, N_6\}$ where $N_1 = \{v_1, v_7\}, N_2 = \{v_2, v_5, v_{12}\}, N_3 = \{v_3, v_{10}\}, N_4 = \{v_4, v_6\}, N_5 = \{v_8, v_{13}\}$, and $N_6 = \{v_9, v_{11}\}$.

The layout shown in Fig. 8 was obtained by applying the above algorithm. At v_3, it is determined that both N_2 and N_3 have touch points with cut number 3 (v_5 and v_{10}). However, N_2 should be in the middle as $v_5 < v_{10}$. At v_8, N_3 should be in the middle as N_3 is the only net having a touch point with cut number 3. ■

It is an easy matter to arrive at an $O(n)$ algorithm that is based on the above discussion. This algorithm is conceptually much simpler than those of [9] and [4]. It is asymptotically faster than that of [9]. While it's asymptotic complexity is the same as that of the algorithm in [4], it's relative simplicity allows it to run much faster.

IV. $K_u = K_l = 3$

We first observe that if i is a touch point of type M or E in a k-zone for $k \leqslant 3$, then the net corresponding to i can be routed to node i independent of the permutation of the $(k+l)$ nets arriving at i. Fig. 9 illustrates the case $k = 3$.

Futhermore, if node i is of type B and in a k-zone for $k \leqslant 3$, then the net corresponding to node i can be inserted at any point in the permutation arriving at node i. Fig. 10 illustrates the case $k = 3$.

Hence, if there is no 4-zone, then the net set can be realized by making decisions arbitrarily at the B nodes. So, assume that there are some 4-zones (and possibly some 5-zones, too). Let a, b, c, and d be the four nets that enter the first (i.e, leftmost) 4-zone. From the preceding discussion, we note that all 24 permutations of these four nets are possible arrival permutations. However, these 24 may be partitioned into two 12 sets each with the property that the two permutations in each set are symmetric (*abcd* is symmetric to *dcba* and *adcb* is symmetric to *bcda*). So, we need to consider only 12 possible arrival permutations at the start of the first 4-zone.

Once we determine which (if any) of these 12 permutations permits routing beyond the start of the first 4-zone, we can determine the wire layout preceding the first 4-zone.

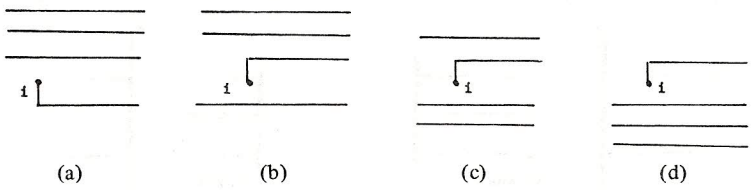

Fig. 10.

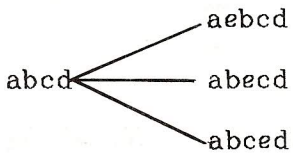

Fig. 11.

TABLE I

NODE	1	2	3	4	5	6	7	8	9	10	11
NET	1	2	3	4	5	4	3	4	5	2	1
TYPE	B	B	B	B	B	M	E	E	E	E	E

Consider one of the 12 nonsymmetric permutations (say *abcd*) that are candidates for the arrival permutation at the start of the first 4-zone. The first node in this zone must be of type *B*. Let *e* be the net that starts at this node. There are only 3 possible entry points for this node (Fig. 11). In all three of the resulting permutations, nets *a* and *d* are the outermost nets. Furthermore, throughout this 4-zone, these two nets will remain the outermost nets of the permutation and the arrival permutation at every node in the 4-zone will contain at least 5 nets. Hence, if either *a* or *d* has a touch point within the 4-zone, the connection to this touch point cannot be made and the permutation *abcd* is infeasible.

Example 4.1: Consider the net list given in Table I. This net list consists of 11 nodes and 5 nets. There is only one 4-zone (from node 5 to node 7) and no 5-zone. Two nets (N_3 and N_4) have touch points inside this 4-zone.

The 12 possible asymmetric arrival permutations for node 5 are:

$N_1\ N_2\ N_3\ N_4$ $\quad$ $N_1\ N_2\ N_4\ N_3$
$N_1\ N_4\ N_2\ N_3$ $\quad$ $N_4\ N_1\ N_2\ N_3$
$N_1\ N_3\ N_2\ N_4$ $\quad$ $N_1\ N_3\ N_4\ N_2$
$N_1\ N_4\ N_3\ N_2$ $\quad$ $N_4\ N_1\ N_3\ N_2$
$N_3\ N_1\ N_2\ N_4$ $\quad$ $N_3\ N_1\ N_4\ N_2$
$N_3\ N_4\ N_1\ N_2$ $\quad$ $N_4\ N_3\ N_1\ N_2$.

Since N_3 and N_4 have touch points within the 4-zone, permutations with these nets at position 1 or 4 are infeasible. Eliminating these infeasible arrival permutations leaves us with just two candidates ($N_1\ N_3\ N_4\ N_2$ and $N_1\ N_4\ N_3\ N_2$) for the arrival permutation at node 5. Both of these yield the same set of emerging permutations ($\{N_1\ N_5\ N_4\ N_2, N_1\ N_4\ N_5\ N_2\}$) from the 4-zone. Since, there are no other 4-zones, we may arbitrarily pick one of the remaining feasible candidates (say $N_1\ N_3\ N_4\ N_2$) and obtain the routing. In order to get this permutation at node 5, we need to place N_2 below N_1 at 2, N_3 between N_1 and N_2 at 3 and N_4 between N_3 and N_2 at 4.

Regardless of whether N_5 is placed above or below N_4 at 5, the routing can be completed. Suppose we elect to place N_5 between N_4 and N_2. The resulting layout is shown in Fig. 12. ■

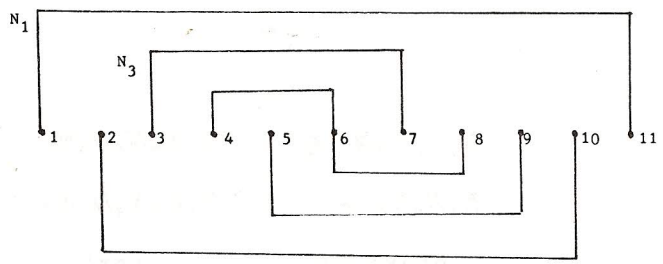

Fig. 12.

Suppose that *abcd* is not infeasible in the sense of the preceding paragraph. Then, we may lay out nets *a* and *d*, as in Fig. 13. Pick any of the three inner track permutations (*ebc*, *bec*, *bce*), say *ebc*. If there is no 5-zone within the 4-zone, then each of these 3 nets (*e*, *b*, *c*) may be connected to it's touch points within the 4-zone. If there is a 5-zone, then let *f* be the net that starts at the first 5-zone. There are exactly two possibilities for the permutation following the first node in this 5-zone: *efbc* and *ebfc*.

Within the 5-zone, however, no new nets may start. Also, only nets *b* and *f* can connect to their touch points. Hence, if either *e* or *c* has touch points in this 5-zone, then the permutation *aebcd* is infeasible. If this is not the case, then at the end of this 5-zone the emerging permutation is *aebcd* or *aefcd* depending on which of *b* or *f* terminates. So, the emerging permutation and feasibility are not affected by the placement of net *f*, that is, it does not matter whether the permutation *aefbcd* or *aebfcd* is used within the 5-zone.

We may continue with the permutation that emerges from the first 5-zone and proceed to route the remainder of the first 4-zone. If all three of the permutations of Fig. 11 are determined to be infeasible, the *abcd* should be discarded from the list of candidate entry permutations for the first 4-zone.

The procedure described above may be applied to each of the (at most 12) candidate permutations. If all are determined to be infeasible, then the net set cannot be realized. If some candidates remain, then let S be the corresponding set of permutations that can emerge from the 4-zone. From S, symmetric permutations may be eliminated. Hence, $|S| \leq 12$.

Example 4.2: In this example, the net list consists of 16 nodes and 7 nets as below.

$$N_1 = \{1, 16\}, N_2 = \{2, 14\}, N_3 = \{3, 6, 12\}, N_4 = \{4, 8\}$$

$$N_5 = \{5, 13\}, N_6 = \{7, 10, 15\}, \text{and } N_7 = \{9, 11\}.$$

This net list has one 4-zone (from node 5 to node 12) and two 5-zones within this 4-zone. The first of these 5-zones starts at node 7 and ends at node 8 while the second 5-zone begins at node 9 and ends at node 11.

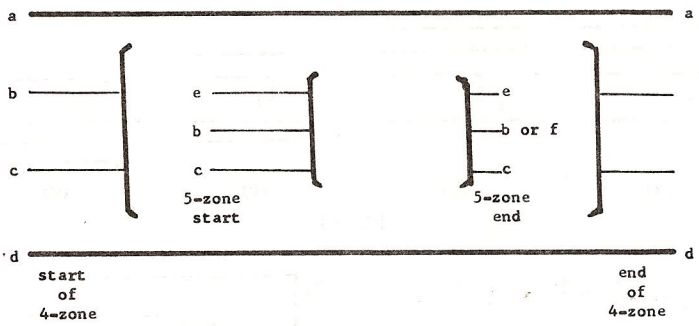

Fig. 13.

$N_1N_3N_4N_2$ <---- $N_1N_3N_4N_5N_2$ <---- $N_1N_3N_6N_5N_2$ <---- $N_1N_6N_5N_2$

$N_1N_4N_3N_2$ <---- $N_1N_5N_4N_3N_2$ <---- $N_1N_5N_6N_3N_2$ <---- $N_1N_5N_6N_2$

start of 4-zone | start of 1st 5-zone | start of 2nd 5-zone | end of 4-zone

Fig. 14.

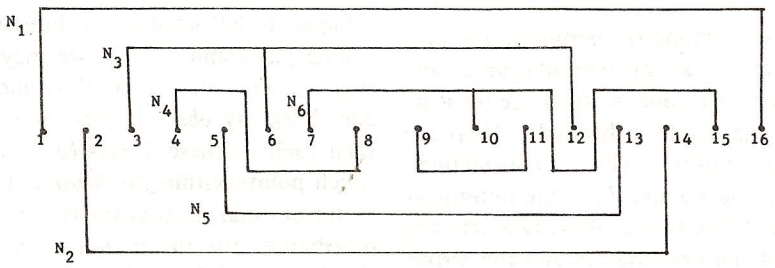

Fig. 15.

All feasible permutations at the start and end of the 4-zone are shown in Fig. 14. Feasible permutations at the start of the two 5-zones are also shown. Since nets starting in a 5-zone may be arbitrarily placed in one of the two possible locations (3rd or 4th), the figure shows just one placement. Fig. 15 shows a layout corresponding to the top row of Fig. 14. ■

If there is no next 4-zone, then we may pick any of the remaining candidates for the first 4-zone and complete the routing.

Assume that there is a next 4-zone. Let A be the set of nets entering this 4-zone. Let A be the set of nets emerging from the previous 4-zone. Clearly, $|A| = |B| = 4$. Let $j = |A \cap B|$.

When $j = 4$, S becomes the set of candidate permutations for the start of the new 4-zone. In between the previous and the new 4-zone, no new nets may begin and connections to all touch points can be made.

Consider the case $j = 3$. Let $B = \{u, v, w, x\}$ and let $x = B - A$ and $y = A - B$. To get the candidate permutations for the start of the new 4-zone, delete x from each of the permutations in S and insert y at each of the four possible insertion points (Fig. 16). From the resulting set T eliminate symmetric permutations. Following this, $|T| <= 12$.

Where $j < 3$, we may arbitrarily pick one permutation, from S and use this to complete the layout in and preceding the previous 4-zone. All 12 asymmetric permutations of A become candidate permutations for the new 4-zone.

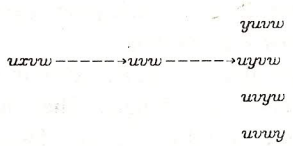

Fig. 16.

Using the process described above, we may proceed from one 4-zone to the next until we are either done with the last 4-zone or we have determined that the net set cannot be realized. If the last 4-zone is completed, then the layout following this is easily obtained. The strategy presented above can be implemented to run in $O(n)$ time.

V. $K_u = 3$ and $K_l = 2$

This case can be solved efficiently in a manner similiar to the case $K_u = K_l = 3$. Now, of course, we need to be concerned with the 3-zone and 4-zones rather than the 4-zones and 5-zones (there can be no 5-zones in the $K_u = 3$, $K_l = 2$ case if the net set is to be realizable).

VI. Experimental Results

Our algorithm for the cases $K_u = K_l = 3$ and $K_u = 3$, $K_l = 2$ were coded in Pascal and run on a VAX 11/780. For com-

TABLE II

Number of nodes	Number of successes	RAGH's Algorithm min time	RAGH's Algorithm max time	RAGH's Algorithm avg time	OUR Algorithm min time	OUR Algorithm max time	OUR Algorithm avg time
50	8	1600	6583	4304	183	466	306
100	8	4650	9800	6960	383	716	525
200	6	14433	17583	16003	1000	1466	1173
300	4	23250	28333	24721	1550	1816	1625

times are in milliseconds

TABLE III

Number of nodes	Number of successes	RAGH's Algorithm min time	RAGH's Algorithm max time	RAGH's Algorithm avg time	OUR Algorithm min time	OUR Algorithm max time	OUR Algorithm avg time
50	8	966	1866	1354	200	316	256
100	7	2533	3216	2807	516	616	543
200	3	5483	6450	5861	1033	1083	1061
300	2	9450	9466	9458	1566	1633	1599

times are in milliseconds

parative purposes, we also ran the Pascal code for the algorithm of [4]. This code was made available to us by Dr. Raghavan. The observed run times are shown in Tables II and III.

We considered single-row routing instances with 50, 100, 200, and 300 nodes. For each node size 10 random instances were generated in such a way that the maximum cut number was less than the number of available tracks. Tables II and III give the number of instances (out of 10) that could in fact be routed with the given street capacities.

As can be seen from the tables, our algorithm for the case $K_u = K_l = 3$ took about one fourteenth of the time taken by the algorithm of [4] on the instances tested. For the instances generated for the case $K_u = 3$ and $K_l = 2$, our algorithm took about one fifth the time taken by the algorithm of [4].

We also generated a data set with 87 nodes and 33 nets that had many nodes with cut number 5. On this data set our algorithm took 516 ms while that of [4] took 21 s: a factor of 40 difference! For this run, K_u and K_l were both 3.

VII. Conclusion

The single-row routing problem is a very important problem for the design automation of digital systems. The single row routing problem was previously shown to be *NP*-hard for a general case. Therefore, it is unlikely that there are efficient algorithms that always generate an optimal routing. We have developed, however, several single row routing algorithms for a narrow street congestion which is particularly of interest in the design of practical case. For the case when $K_u = k$, $K_l = 1$ and $K_u = K_l = 2$, a linear $O(n)$ algorithm was shown. And for the case when $K_u = 3$ and $K_l = 2$ or 3, a very efficient routing algorithm was presented and it's performance was analyzed.

References

[1] *Design Automation of Digital Systems.* M. A. Breur (Ed.), Englewood Cliffs, NJ: Prentice-Hall, 1972.

[2] E. Horowitz and S. Sahni, *Fundamentals of Data Structures.* Computer Science Press, 1976.

[3] E. S. Kuh, T. Kashiwabara, and T. Fujisawa, "On optimum single row routing," *IEEE Trans. Circuits Syst.*, vol. CAS-26, pp. 361–368, June 1979.

[4] R. Raghavan and S. Sahni, "Optimal single row router," in *19th ACM IEEE Design Automation Conf.*, pp. 38–45.

[5] K. Stevens and W. Van Cleemput, "Global via elimination in a generalized routing environment," in *IEEE ISCAS Proc.*, pp. 689–692, 1979.

[6] H. C. So, "Some theoretical results on the routing of multilayer printed-wiring boards," in *IEEE Symp. Circuits Syst.*, pp. 296–303, 1974.

[7] B. S. Ting, E. S. Kuh, and I. Shirakawa, "The multilayer routing problem: Algorithms and necessary and sufficient conditions for the single row single layer case," *IEEE Trans. Circuits Syst.*, vol. CAS-23 pp. 768–778, Dec. 1976.

[8] B. S. Ting and E. S. Kuh, "An approach to the routing of multilayer printed circuit boards," in *IEEE Symp. Circuits Syst.*, pp. 902–911, 1978.

[9] S. Tsukiyama, E. S. Kuh, and I. Shirakawa, "An algorithm for single row routing with prescribed street congestions," *IEEE Trans. Circuits Syst.*, vol. CAS-27, pp. 765–771, Sept. 1980.

*

Sangyong Han was born in Seoul, Korea, on October 2, 1952. He received the B.S. degree in engineering from the Seoul National University in 1975.

Since graduation, he had been with the Korea Advanced Institute of Science and Technology. He is currently a Ph.D. student at the Department of Computer Science, University of Minnesota, Minneapolis.

*

Sartaj K. Sahni (M'79) received the B.Tech (electrical engineering) degree from the Indian Institute of Technology, Kanpur, India, and the M.S. and Ph.D. degrees in computer science from Cornell University, Ithaca, NY.

He is at present Professor of Computer Science at the University of Minnesota, Minneapolis. He has published in *JACM*, *JCSS*, *SIAM Journal on Computing*, IEEE Transactions on Computers, IEEE Transactions on Circuits and Systems, *ACM Transactions on Mathematical Software*, *Operations Research*, *International Journal on Theoretical Computer Science*, *Mathematics of Operations Research*, and the *Journal of Statistical Computation and Simulation*. His publications are on design and analysis of efficient algorithms, parallel computing, interconnection networks, and design automation. He is also a coauthor of the texts: *Fundamentals and of Data Structures and Fundamentals of Computer Algorithms*, and author of *Concepts of Discrete Mathematics*. He is the area editor for Parallel Algorithms and Data Structures for the *Journal of Parallel and Distributed Computing*.