Layering Algorithms For Single-Row Routing

SANGYONG HAN AND SARTAJ SAHNI, MEMBER, IEEE

Abstract—**We develop two fast algorithms for the layering problem that arises when the single-row routing approach to wire layout is used. Both of these algorithms are for the case when the upper and lower street capacities are two. While neither of these algorithms guarantees the production of an optimal layering, it has been empirically determined that both will produce better layerings than an earlier proposed algorithm [13] for this problem. In addition, our algorithms run much faster than the earlier algorithm.**

Keywords and Phrases: **Single-row routing, layering.**

I. Introduction

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 has been proposed. 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 [11], [8]:

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 third and fourth phases: layering and single-row routing. 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 m nets. Each net N_i represents a set of nodes that are to be made electrically equivalent. The nets satisfy the following conditions:

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

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

The nodes may be regarded as vias/pins that penetrate all layers of the multilayer board.

Node j, $j \in N_i$ is a *touch point* of net i. The nets are to be realized in a minimum number of layers by the use of nonoverlapping wires that are composed solely of horizontal and vertical segments. Fig. 1(a)–(c) shows some of the legal ways to realize the net $\{1, 3, 6, 9\}$ on a single layer. The wire layout must satisfy the additional requirement that a vertical cut made at any point along the axis formed by the nodes intersect, at most, one horizontal segment from each net. Thus, the wiring of Fig. 1(d) is illegal. Another constraint is that no two conductor paths for two distinct nets have a common electrical junction.

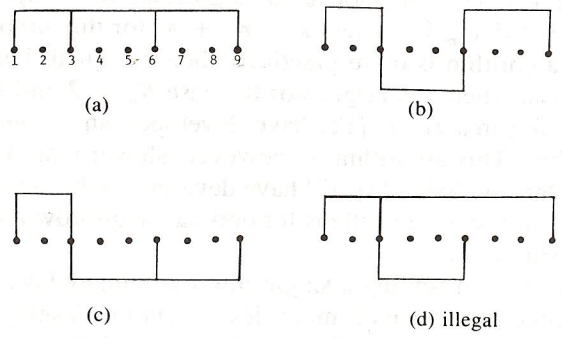

Fig. 1.

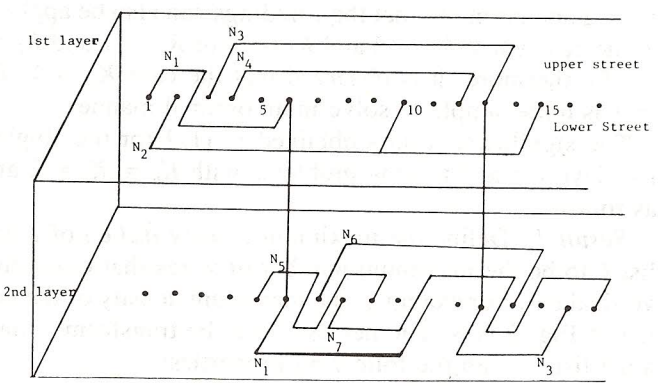

Fig. 2.

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 layered in tracks and no track may hold more than one wire segment at any point (of course, several nonoverlapping wire segments may be packed into the same track). Let K_u and K_l denote the number of tracks in the upper and lower streets, respectively. Fig. 2 shows one way to realize a net list $L = \{(1, 6, 11), (2, 10, 15), (3, 13, 16), (4, 12), (5, 9), (7, 14), (8, 17)\}$ on two layers, both of which have $K_u = K_l = 2$.

Much work has been done on the development of fast algorithms to route a single row on a single layer. Kuh *et al.* [5] and Tsukiyama *et al.* [12] have developed neces-

Manuscript received June 26, 1984; revised February 18, 1986. This Research was supported in part by the National Science Foundation under Grant MCS-8305567.

S. Han is with IBM Kingston, 32JD/723, Kingston, NY 12401.
S. Sahni is with the Computer Science Department, University of Minnesota, Minneapolis, MN 55455.

IEEE Log Number 8611576.

sary and sufficient conditions for a net set to be realizable. In [5], 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 $|L| = m$ to determine if the routing is possible with a given K_u and K_l. Raghavan and Sahni [7] 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 $K_u \leq 2$ and $K_l \leq 2$, Tsukiyama *et al.* [12] have developed an $O(mn)$ algorithm. This algorithm is, however, slower than that of [7]. Han and Sahni [2], [3] have developed what are presently the fastest algorithms for optimal single-row routing on a single layer.

The case of routing a single row when many layers are available has received much less attention. Tsukiyama, Kuh, and Shirakawa [13] have investigated this problem in some detail. They restrict themselves to the case $K_u = K_l = 2$ and point out that their findings can also be applied to the case when $K_u = 2$ and $K_l = 1$ (or $K_u = 1$ and $K_l = 2$). Furthermore, it is observed that the case $K_u = 1$, $K_l = 1$ is quite simple to solve in an optimal manner.

The significant results obtained in [13] for the single-row layering and routing problems with $K_u = K_l = 2$ are as follows.

Result 1: Define the maximum density $d_m(L)$ of a net list L to be the maximum number of wires that cross any vertical cut. For example, the maximum density of the net list of Fig. 2 is 6. The net list L can be transformed into a net list L' with the following properties:

1) Every net in L' contains exactly two nodes.
2) $d_m(L) \leq d_m(L') \leq d_m(L) + 1$.
3) If L' can be routed in k layers when $K_u = K_l = 2$, then so also can L. Furthermore, the routing for L may be obtained in $O(n)$ time (recall that n is the number of nodes) from that for L'.

Result 2: When $K_u = K_l = 2$, the minimum number of layers r needed to route the net list l satisfies the inequality

$$\lceil d_m(L)/4 \rceil \leq r \leq \lceil d_m(L)/3 \rceil .$$

Using results 1 and 2, an $O(n^5)$ layering and routing algorithm is developed. This algorithm is heuristic in nature and does not guarantee to use a minimum number of layers. In fact, at this time, it is not known whether the minimum layering problem for the case $K_u = K_l = 2$ is NP-hard.

In this paper, we develop two new heuristic algorithms for the layering and routing problem when $K_u = K_l = 2$. These are based on the single-layer algorithm described in [3] for $K_u = K_l = 2$. Both algorithms have a time complexity $O(kn)$, where k is the number of layers in the solution. The overhead associated with the second algorithm is slightly higher than that associated with the first. Both algorithms are significantly faster than the one proposed in [13]. Furthermore, experimentation with random instances indicates that our algorithms often use fewer layers.

II. Problem Formulation and Terminology

As a consequence of Result 1 of [13], we can restrict ourselves to net lists L, in which each net consists of exactly two pins. Since our algorithms are based on the algorithms described in [3], we can actually work directly with multipin nets. It is, however, easier to describe the algorithms for the case of two pin nets, and we shall make this restriction in the remainder of this paper.

Our problem is to partition L into the subsets $L_1, L_2, \cdots, L_r$ such that the nets in each partition can be realized in a single layer (with $K_u = K_l = 2$) and r is minimum.

Since we make use of the $K_u = K_l = 2$ algorithm of [3], we briefly restate that algorithm as it relates to two pin nets. This algorithm constructs a wire layout by making a single left to right scan of the nodes. Each node is classified by its type. For the case of two pin nets, only two classifications are needed. Let i, j, $i < j$, be the two pins in some net. i is of type B (beginning) and j of type E (end). The beginning nodes in Fig. 3 are 1, 2, 3, and 5. Nodes 4, 6, 7, and 8 are the end nodes.

As in [7], each possible ordering of nets that can be encountered by a node is maintained simply as an ordered list of net indices. 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. 4(a) and (b), are not treated separately.

The *cut number* of node i is the number of nets that cross node i. This count does not include the net that begins or ends at i. The cut numbers for the nodes of Fig. 3 are: $\text{cut}(1, \cdots, 8) = (0, 1, 2, 2, 2, 2, 1, 0)$.

Let P_i denote the net ordering encountered when node i is reached during a left to right scan of nodes. For Fig. 3, we have $P_1 = \phi$, $P_2 = (1)$, $P_3 = (1, 2,)$, $P_4 = (1, 3, 2)$, $P_5 = (3, 2)$, $P_6 = (4, 3, 2)$, $P_7 = (4, 2)$, and $P_8 = (2)$.

The wire layout algorithm of [2] begins with $P_1 = \phi$ and attempts to obtain a valid sequence $P_1, P_2, \cdots, P_n$. If node i is an E-type node, then we need merely check if the net corresponding to this touch point can, in fact, be wired to this node when $K_u = K_l = 2$. Note that the only time we have difficulty is when $|P_{i-1}| = 4$. The wiring is not possible if the net in question is in position 1 or 4 of P_{i-1} (See Fig. 5).

Let us consider the case when node i is of type B. Let $(N_1, N_2, \cdots, N_b)$ be the permutation of b nets 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 possibilities for the permutation arriving at $i + 1$: $(N_1,$

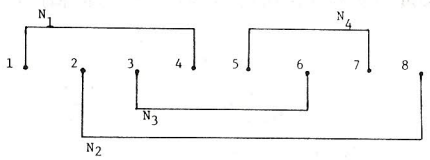

Fig. 3. Two pin nets.

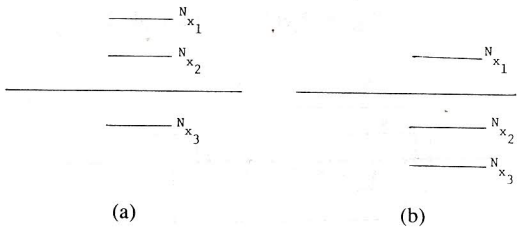

Fig. 4.

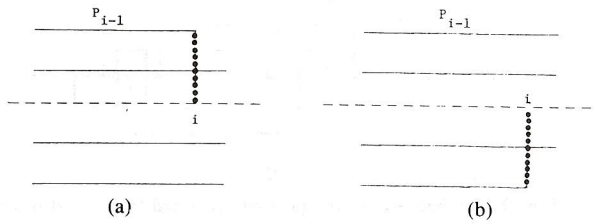

Fig. 5. Outer nets cannot be wired to i.

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 also be 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 $\{v_1, v_2, v_x\}$ 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. 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_c, N_d, N_e, N_f) 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 if $N_a = N_d$ or $N_a = N_e$.

Among the three possibilities for the permutation arriving at $i + 1$, there is exactly one that guarantees $N_a = N_d$ or $N_a = N_e$. 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 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 be the end points of nets N_2 and N_x, respectively. 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 node. Regardless of which permutation is chosen, both N_2 and N_x can be wired to their end points 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.

Example 1: Consider the net list $L = \{N_1, N_2, \cdots, N_7\}$, where $N_1 = \{v_1, v_8\}$, $N_2 = \{v_2, v_5\}$, $N_3 = \{v_3, v_{11}\}$, $N_4 = \{v_4, v_7\}$, $N_5 = \{v_6, v_{13}\}$, $N_6 = \{v_9, v_{14}\}$, and $N_7 = \{v_{10}, v_{12}\}$.

The layout shown in Fig. 6 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_{11}). However, N_2 should be in the middle as $v_5 < v_{11}$. At v_9, N_3 should be in the middle as N_3 is the only net having a touch point with cut number 3. □

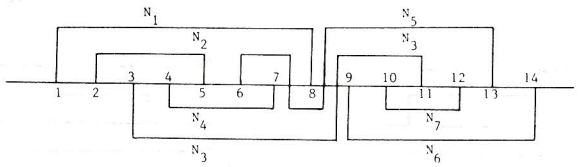

Fig. 6.

III. Greedy Algorithm

The first algorithm that we propose for the layering problem scans the nodes left to right and determines a subset L_1 of L that can be realized in a single layer. This subset is determined following the principles of the strategy outlined in Section II and described in detail below. Once L_1 has been determined, we determine a subset from $L - L_1$ to layout in layer 2. This process is repeated until we have determined $L_1, L_2, \cdots, L_r$ such that $L = L_1 \cup L_2, \cdots \cup L_r$.

Let $P_i = (N_{x_1}, N_{x_2}, \cdots, N_{x_n})$ be the net ordering encountered at node i. To determine L_j, we begin with $P_1 = L_j = \phi$ and $L' = L - L_1 - L_2 - \cdots - L_{j-1}$. Let N_q be the net in L (if any) with touch point i. Note that there may be no net in L' with touch point i. This will happen for several i's when we are dealing with layers 2, 3, 4, $\cdots$, etc. If N_q exists, let v_b and v_e, $v_b < v_e$, be the end points of N_q (clearly, $i \in \{v_b, v_e\}$). We have three cases to consider.

Case 1: *N_q does not exist.*
In this case, we simply set $P_{i+1} = P_i$.

Case 2: *N_q exists and begins at i (i.e., i is of type B).*
In this case, we need to determine whether or not N_q should be added to L_j. This decision is made by considering the four cases below.

a) $|P_i| = 4$.
In this case, the addition of net N_q causes

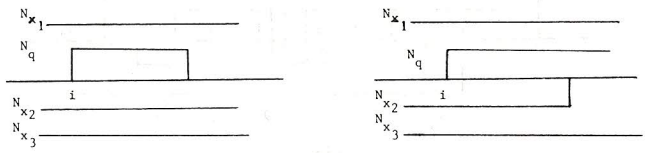

Fig. 7.

infeasibility.

So, set $P_{i+1} = P_i$.

b) $|P_i| = 3$.

Let v_1, v_2, and v_3 be the end points of nets N_{x1}, N_{x2}, and N_{x3}, respectively, (recall that $P_i = (N_{x1}, N_{x2}, N_{x3})$). If $\min\{v_1, v_2, v_3, v_e\} \in \{v_2, v_e\}$, then set $P_{i+1} = (N_{x1}, N_{x2}, N_q, N_{x3})$ (note that $(N_{x1}, N_q, N_{x2}, N_{x3},)$ will do just as well) and add N_q to L_j. Note that the insertion of N_q into the second or third position of P_{i+1} creates no difficulties and also allows all four nets to terminate using the given number of streets (Fig. 7).

When $\min\{v_1, v_2, v_3, v_e\}$ is not an element of $\{v_2, v_e\}$, N_q cannot be added to this layer as the routing of the four nets N_{x1}, N_{x2}, N_{x3}, and N_q cannot be completed with $K_u = K_l = 2$.

c) $|P_i| = 2$.

In this case, N_q can be added to L_j with full assurance that N_{x1}, N_{x2}, and N_q can all be routed to their end points. Of the three orderings (N_{x1}, N_{x2}, N_q), (N_{x1}, N_q, N_{x2}), and (N_q, N_{x1}, N_{x2}) possible for P_{i+1}, choose the one that has the smallest end point in the middle. This will maximize our chances of adding more nets to L_j.

d) $|P_i| < 2$.

Add N_q to L_j. Choose any of the possible orderings for P_{i+1}.

Case 3: *N_q exists and ends at i (i.e. i is of type E).*

Note that because of the way in which nets are selected for N_j and the orderings P_i, P_2, $\cdots$, P_i constructed, we are assured that N_q can terminate at i without violating the street constraints $K_u = K_l = 2$. P_{i+1} is obtained from P_i by deleting N_q. □

Example 2: Consider the following net list with 16 nodes and 8 nets:

$$N_1 = \{1, 5\} \quad N_2 = \{2, 10\} \quad N_3 = \{3, 11\}$$
$$N_4 = \{4, 12\} \quad N_5 = \{6, 15\} \quad N_6 = \{7, 13\}$$
$$N_7 = \{8, 14\} \quad N_8 = \{9, 16\}.$$

This is shown graphically in Fig. 8.

The algorithm begins by constructing L_1. We begin with $L_1 = P_1 = \phi$. At node 1, we get $L_1 = \{N_1\}$ and $P_2 = (N_1)$. At node 2, we get $L_1 = \{N_1, N_2\}$ and $P_3 = (N_2, N_1)$ (note that $P_3 = (N_1, N_2)$ is also acceptable). At node 3, $L_1 = \{N_1, N_2, N_3\}$ and $P_4 = \{N_2, N_1, N_3)$ as N_1 has

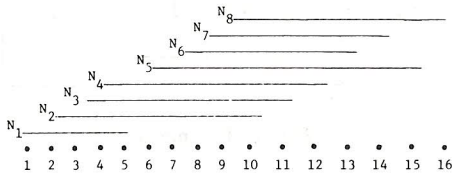

Fig. 8. Net list for Example 2.

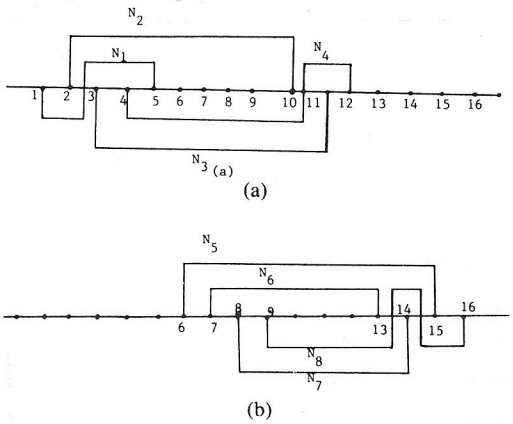

Fig. 9. Realization for the (a) first layer and (b) second layer.

the first end point. At node 4, $L_1 = \{N_1, N_2, N_3, N_4\}$ and $P_5 = (N_2, N_1, N_4, N_3)$ and at node 5, we get $P_6 = (N_2, N_4, N_3)$. No change takes place in L_1. At nodes 6, 7, 8, and 9, no change in L_1 occurs as the nets that begin at these nodes do not satisfy the conditions of case 2b). When the algorithm to determine L_1 terminates, we have $L_1 = \{N_1, N_2, N_3, N_4\}$. The layout for L_1 is easily obtained from the P_i's (see [2]).

To determine L_2, we begin with $L' = L - L_1 = \{N_5, N_6, N_7, N_8\}$. L_2 is determined to be L'. The layout for both layers is shown in Fig. 9. □

Complexity Analysis

Let k be the number of layers used and n the number of nodes. Using appropriate data structures, each L_i can be found in $O(n)$ time. Hence, the total time needed by the greedy algorithm of this section is $O(kn)$. From the P_i's, the wire layout may be obtained easily. □

IV. Blockage Algorithm

The greedy algorithm of Section III selects nets for inclusion into a layer without considering the structure of the nets on the right. This can cause the algorithm to perform poorly on certain net lists. We introduce the concept of *blockage* in order to arrive at a new algorithm that can be expected to give better layerings.

The concept of blockage is defined relative to the set of nets already chosen for a layer. A *blocked area* of nodes begins at a node i, which satisfies the following properties:

1) $|P_i| = 3$.
2) Let $P_i = (N_{x1}, N_{x2}, N_{x3})$ and let v_1, v_2, v_3 be the end points of N_{x1}, N_{x2}, and N_{x3}, respectively, and $v_2 \neq \min\{v_1, v_2, v_3\}$.

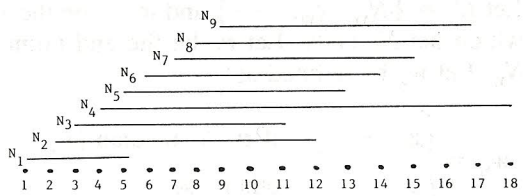

Fig. 10.

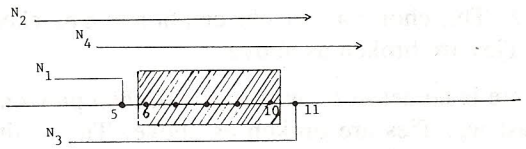

Fig. 11. The blocked area.

The blocked area beginning at node i (above) ends at the node which is min $\{v_1, v_2, v_3\}$. Observe that, within a blocked area, all nodes of type B cause the greedy algorithm to enter case 2b). Only nets that begin and end within the blocked area can be added to L_j.

Example 3: Consider the following net list:

$$N_1 = \{1, 5\} \quad N_2 = \{2, 12\} \quad N_3 = \{3, 11\}$$
$$N_4 = \{4, 18\} \quad N_5 = \{6, 13\} \quad N_6 = \{7, 14\}$$
$$N_7 = \{8, 15\} \quad N_8 = \{9, 16\} \quad N_9 = \{10, 17\}.$$

This net list is shown graphically in Fig. 10.

Our greedy algorithm selects nets N_1, N_2, N_3, and N_4 for L_1. N_1 ends at node 5 and we have $P_6 = (N_2, N_4, N_3)$. The area from node 6 to node 11 gets blocked (Fig. 11) to all but those nets that begin and end in this area. There are no such nets, and we have $L_1 = \{N_1, N_2, N_3, N_4\}$. The maximum density of the net set $L - L_1$ is 5 and two more layers are needed to complete the routing. The routing obtained for the three layers is shown in Fig. 12. The above net list can be routed in two layers using the layout of Fig. 13. □

As can be seen from Example 3, the creation of a blocked area (and its extent) can be determined at the time a fourth net is added to a P_i. When net N_4 is added to $P_4 = (N_2, N_1, N_3)$ to get $P_5 = (N_2, N_1, N_4, N_3)$ we know that a blocked area will begin at node 6 as the end point of net N_4 is not smaller than the end points of nets N_2 and N_3. We also know that this blocked area will extend up to node 11 which is where the first of N_2, N_3, and N_4 end.

The blockage algorithm to be developed in this section attempts to minimize the extent of blocked areas. This algorithm utilizes a zone representation for the nodes. A *zone* is a region of nodes that consists of some number of B-type nodes followed by some number of E-type nodes (there must be at least one node of each type in the zone). Consider the following net list:

$$N_1 = \{1, 5\} \quad N_2 = \{2, 6\} \quad N_3 = \{3, 10\}$$
$$N_4 = \{4, 13\} \quad N_5 = \{7, 16\} \quad N_6 = \{8, 15\}$$
$$N_7 = \{9, 11\} \quad N_8 = \{12, 14\}.$$

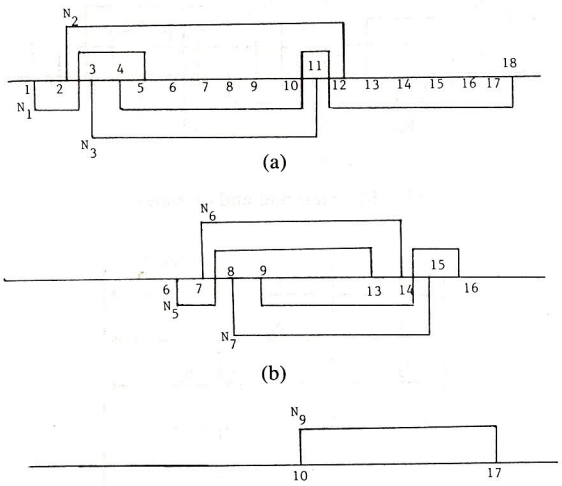

Fig. 12. (a) First, (b) second, and (c) third layers.

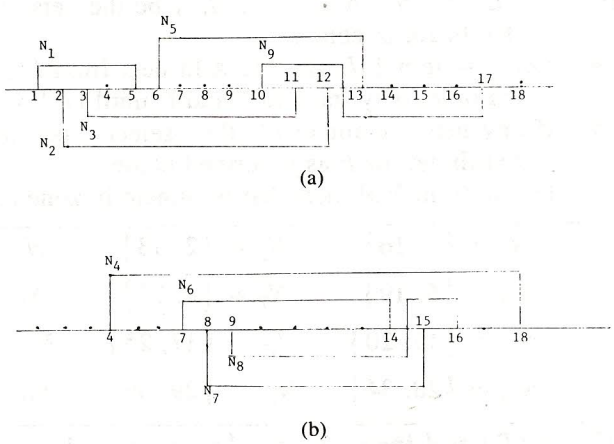

Fig. 13. (a) First and (b) second layers.

The zones created by this list are shown in Fig. 14. Notice that the zone boundaries are displaced slightly from the node positions. The zone representation of a net list is obtained by beginning the net at the left end of the zone in which it begins and ending it at the right end of the zone in which it ends (Fig. 15).

A *critical* zone is defined to be one that contains $4 * \lceil d_m(L)/4 \rceil$ nets. There are no critical zones in Fig. 15.

From the zone representation of a net list, it is easy to determine blockages and their extent. Our blockage algorithm, like our greedy algorithm, constructs L_1, L_2, $\cdots$, in sequence. We describe how each L_j is constructed.

Algorithm Blockage

step1: Let L be the nets not yet assigned to a layer.
Let $P_i = L_j = \phi$.
Construct the zone representation for the nets in L.
number_of_zones ← number of zones
$z \leftarrow 1$

step2: **If** $z >$ *number_of_zones* **then stop**

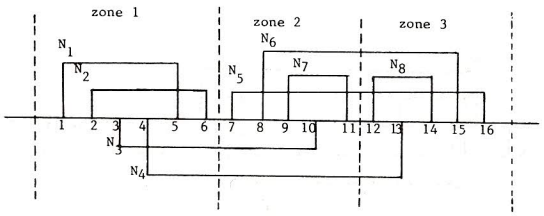

Fig. 14. Start and end of zones.

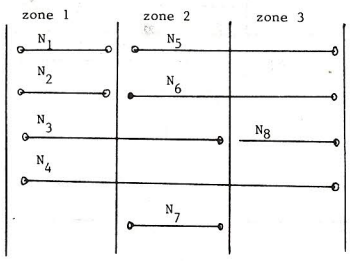

Fig. 15. Zone representation of nets of Fig. 14.

step3: Let $L' = \{N_{x1}, N_{x2}, \cdots, N_{xm}\}$ be the nets in L that begin in zone z.

step4: Let $q = \min\{|L'|, 3\}$. Add nets from L', in the order they start, to P and L_j until $|P| = q$.

step5: If any nets remain in L', then select a possible fourth net for P as described below.

step6: Delete from P all nets that terminate in zone z.

step7: **If** $|P| = 4$ **then** $z \leftarrow \min\{z_1, z_2, z_3, z_4\} + 1$
else $z \leftarrow z + 1$
where z_i is the zone in which the ith net of P terminates;
Delete all nets from P that have terminated before zone z

step8: **go to** *step2*

In step5, the fourth net is selected from L' in such a way as to minimize the extent of the blocked area created. If a net remains in L', then $|P| = 3$. Let $P = (N_{x1}, N_{x2}, N_{x3})$ and let v_1, v_2, v_3, respectively, be the end points of these nets. Let z_1, z_2, z_3 be the zones in which these three nets end. Let $L'' = L' - P$. We consider two cases.

case 1: $v_2 = min\{v_1, v_2, v_3\}$:

(a) $z_2 = z_1$ *or* $z_2 = z_3$.
Following the addition of a fourth net to P, no new nets can be added until zone $z_2 + 1$ or the zone following the one in which the newly added net ends. To allow for maximum net embedding, we choose the fourth net from L'' such that $|z_4 - \max\{z_1, z_3\}|$ is minimized. z_4 is the zone in which the fourth net ends. Ties are broken by choosing the net with rightmost end point.

(b) $z_2 \neq z_1$ *and* $z_2 \neq z_3$.
Let $L'' = \{N_{y1}, N_{y2}, \cdots\}$ and let z_{y_i} be the zone in which net N_{y_i} ends. Let v_{y_i} be the end point of net N_{y_i}. Let w_{y_i} be defined as:

$$w_{y_i} = \begin{cases} z_{y_1} - z_2 & \text{if } v_{y_i} > \text{median } (v_1, v_2, v_3) \\ 0 & \text{otherwise} \end{cases}$$

If there is a net in L'' with $w_{y_i} = 0$, then add this to P. This choice avoids the creation of a blocked area. Ties are broken as above.

If there is no net in L'' with $w_{yi} = 0$, then pick one with the least w_{y_i}. Ties are broken as above. The addition of this net to P will create a blocked area. If zone z is a critical zone, then add this net to P. If zone z is not a critical zone, then add this net to P only if the blocked area does not include a critical zone.

case 2: $v_2 \neq min\{v_1, v_2, v_3\}$:
We are in a blocked area. Pick a net from L'' with end point less than $\min\{v_1, v_2, v_3\}$. If no such net exists, then add no net to P. If nets with this property exist, add to P the net with rightmost end point that has this property.

Example 4: We consider the example provided in [13, Fig. 4]. The net list is

$$N_1 = \{1, 16\} \quad N_2 = \{2, 13\} \quad N_3 = \{3, 12\} \quad N_4 = \{4, 9\} \quad N_5 = \{5, 10\}$$
$$N_6 = \{6, 19\} \quad N_7 = \{7, 21\} \quad N_8 = \{8, 24\} \quad N_9 = \{11, 33\} \quad N_{10} = \{14, 26\}$$
$$N_{11} = \{15, 20\} \quad N_{12} = \{17, 25\} \quad N_{13} = \{18, 27\} \quad N_{14} = \{22, 38\} \quad N_{15} = \{23, 36\}$$
$$N_{16} = \{28, 34\} \quad N_{17} = \{29, 39\} \quad N_{18} = \{30, 40\} \quad N_{19} = \{31, 37\} \quad N_{20} = \{32, 35\}$$

A graphical representation of this net list is given in Fig. 16 and the zone representation is given in Fig. 17. The nets that begin in each of the six zones of Fig. 17 are

$$L_1' = \{N_1, N_2, N_3, N_4, N_5, N_6, N_7, N_8\}$$
$$L_2' = \{N_9\}$$
$$L_3' = \{N_{10}, N_{11}\}$$
$$L_4' = \{N_{12}, N_{13}\}$$
$$L_5' = \{N_{14}, N_{15}\}$$
$$L_6' = \{N_{16}, N_{17}, N_{18}, N_{19}, N_{20}\}.$$

We begin by constructing L_1. In step4, nets N_1, N_2, and N_3 are added to P. The ordering is $(N_1\ N_3\ N_2)$ as N_3 has the first end point. Since some nets remain in $L_1' - L_1$, we attempt to add a fourth net to P in step5. Since nets N_2 and N_3 end in the same zone (z_2), we are in case 1(a) of the fourth net selection procedure and we choose a net whose end point is in the closest zone to the end point of net N_1. Nets N_6 and N_7 terminate only one zone from net N_1. Of these, N_7 is chosen as it has the rightmost end point. The permutation P becomes $(N_1\ N_3\ N_7\ N_2)$. We

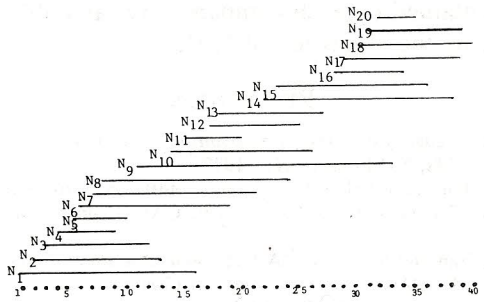

Fig. 16. Net set of [13].

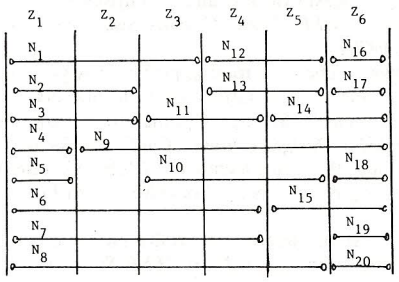

Fig. 17. Zone representation for nets of Fig. 16.

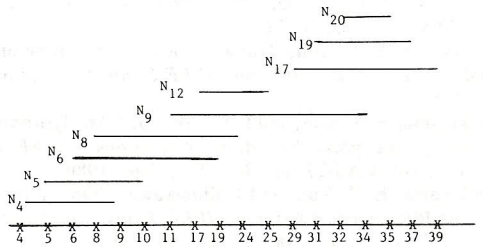

Fig. 18.

next move to zone 3. The nets N_2 and N_3 are deleted from P as these terminate in zone 2. P is now $(N_1\ N_7)$. Net N_{10} is added to P in step4 to get $P = (N_{10}\ N_1\ N_7)$. In step5, net N_{11} is added to P as it does not create any blocked areas. This decision is made in case 1(b). P becomes $(N_{10}\ N_{11}\ N_1\ N_7)$, but N_1 is deleted before moving to zone 4. In zone 4, N_{13} is selected and the ordering becomes $P = (N_{10}\ N_{11}\ N_{13}\ N_7)$. The nets N_7 and N_{11} are deleted as they terminate in this zone. From zone 5, N_{14} is added to P to get $(N_{14}\ N_{10}\ N_{13})$ in step4 and then N_{15} is added in step5 case 1(a) to get $(N_{14}\ N_{15}\ N_{10}\ N_{13})$. Before moving to zone 6, N_{10} and N_{13} are removed from P as these terminate in zone 5. Finally, N_{16} and N_{18} are added to P in zone 6. We get $L_1 = \{N_1, N_2, N_3, N_7, N_{10}, N_{11}, N_{13}, N_{14}, N_{15}, N_{16}, N_{18}\}$.

For the L_2 computations, we have $L = \{N_4, N_5, N_6, N_8, N_9, N_{12}, N_{17}, N_{19}, N_{20}\}$. The zones and zone representations are shown in Figs. 18 and 19, respectively. All the remaining nets get included into L_2. The constructed layouts for the two layers are shown in Fig. 20. □

Complexity Analysis

The beginning and end zone for n nets can be determined in $O(n)$ time. Using appropriate data structures, the work done in each zone can be carried out in time proportional to the number of nets that begin and end in that zone. Hence, the time needed to find an L_j is $O(n)$. If k layers are used, $O(kn)$ time is needed to determine the k L_j's.

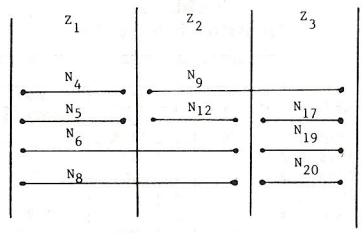

Fig. 19.

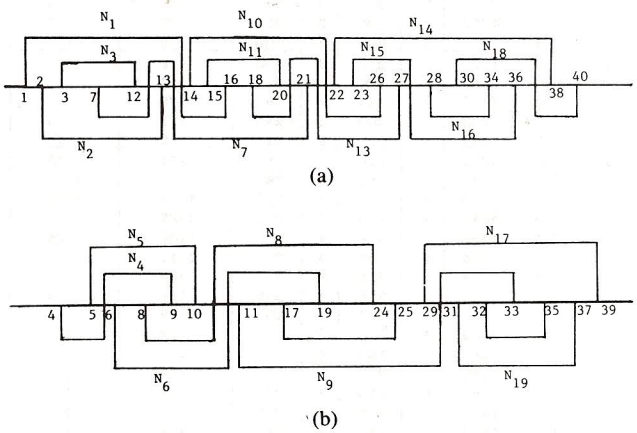

Fig. 20. Realization for the (a) first layer and (b) second layer.

V. Experimental Results

To compare the relative performance of our algorithms and that of [2], we coded all three in Pascal and obtained sample run times and layerings for randomly generated two pin net lists. All experiments were conducted on a VAX 11/780.

Net lists with 100 nodes and 50 nets were generated at random. For each of the densities 7, 8, and 10–20, five instances were generated. Thus, our experiments involved a total of 65 different instances. The results of our experiments are tabulated in Tables I and II.

As can be seen, our greedy algorithm took between $\frac{1}{6}$ to $\frac{1}{35}$ of the time required by the algorithm in [13]. The blockage algorithm took between $\frac{1}{4}$ and $\frac{1}{28}$ of the time taken by [13]. Of the 65 instances tested, the greedy algorithm used fewer layers than that of [13] for seven instances while the algorithm of [13] used fewer layers for three instances. The blockage algorithm used fewer layers than the algorithm in [13] for ten instances and more layers on one.

In addition to the 65 instances reported on in Tables I and II, we laid out the example of [13, Fig. 4]. The greedy algorithm used three layers while the other two used two layers. The time needed to obtain the layout is:

[13]:	533 ms
Greedy:	66 ms
Blockage:	83 ms.

TABLE I
DENSITIES 7, 8, 10–13

Density	[TSUK83]		Greedy		Blockage		Greedy / [TSUK83]	Blockage / [TSUK83]
	l	time	l	time	l	time		
7	3	6916	2	200	2	283	1/35	1/24
	3	6416	2	216	2	266	1/30	1/24
	3	6900	3	233	3	250	1/30	1/28
	3	6550	3	216	2	266	1/30	1/25
	3	6383	2	216	2	300	1/30	1/21
8	3	2216	3	183	3	250	1/12	1/9
	3	1933	3	233	3	250	1/8	1/8
	3	2200	3	216	3	233	1/10	1/9
	3	1750	3	200	3	250	1/9	1/7
	3	1883	3	200	3	300	1/9	1/6
10	3	1226	3	216	3	233	1/6	1/5
	3	1483	3	200	3	250	1/7	1/6
	3	1416	3	183	3	267	1/8	1/5
	3	1333	3	200	3	283	1/7	1/5
	3	1234	3	216	3	217	1/6	1/6
11	4	6616	4	216	4	250	1/31	1/26
	4	6383	4	183	3	266	1/35	1/24
	4	6300	4	216	4	250	1/29	1/25
	4	6683	4	234	4	250	1/29	1/27
	4	6433	4	216	4	233	1/30	1/28
12	4	1966	4	183	4	266	1/11	1/7
	4	1733	4	216	4	233	1/8	1/7
	4	2016	4	216	4	266	1/9	1/8
	4	1633	4	217	4	250	1/8	1/7
	4	2050	4	233	4	233	1/9	1/9
13	4	1266	4	150	4	250	1/8	1/5
	4	1483	4	233	4	250	1/6	1/6
	4	1650	4	216	4	266	1/8	1/6
	4	1516	4	216	4	250	1/7	1/6
	4	1883	4	250	4	266	1/8	1/7

l is the number of layers; time is in milliseconds.

TABLE II
DENSITIES 14–20

Density	[TSUK83]		Greedy		Blockage		Greedy / [TSUK83]	Blockage / [TSUK83]
	l	time	l	time	l	time		
14	4	1566	4	216	4	250	1/7	1/6
	4	1350	4	233	4	300	1/6	1/5
	4	1266	5	216	4	283	1/6	1/4
	5	1416	5	233	4	283	1/6	1/5
	4	1783	4	200	4	258	1/9	1/7
15	5	6066	5	200	5	216	1/30	1/28
	5	4016	5	233	5	250	1/17	1/16
	5	1866	4	233	4	283	1/8	1/7
	5	1783	5	200	5	250	1/9	1/7
	5	3766	4	233	4	266	1/14	1/14
16	5	1766	5	233	5	250	1/8	1/7
	5	2416	5	233	5	216	1/10	1/11
	5	1500	5	183	5	300	1/8	1/5
	5	6216	5	166	5	250	1/37	1/25
	5	1716	5	233	5	233	1/7	1/7
17	5	1400	5	250	5	266	1/6	1/5
	5	1566	5	216	5	250	1/7	1/6
	5	1616	5	216	5	250	1/7	1/6
	5	1366	5	233	5	233	1/6	1/6
	5	1466	5	216	5	233	1/7	1/6
18	5	1850	6	250	5	233	1/7	1/8
	5	1450	5	200	5	250	1/7	1/6
	5	1333	5	166	5	250	1/8	1/5
	5	1716	5	216	5	216	1/8	1/8
	5	1516	5	250	5	316	1/8	1/5
19	6	3433	5	216	6	283	1/16	1/12
	6	3116	6	200	6	266	1/16	1/12
	6	5666	5	166	5	266	1/34	1/21
	6	1483	6	234	6	250	1/6	1/6
	5	1733	6	233	6	316	1/7	1/5
20	6	6266	6	214	6	316	1/29	1/20
	6	3116	6	250	6	284	1/12	1/11
	6	5666	6	250	5	266	1/23	1/21
	6	1483	6	216	6	266	1/7	1/6
	6	5566	6	233	6	266	1/24	1/21

l is the number of layers; time is in milliseconds.

VI. CONCLUSIONS

We have developed two fast heuristics for the layering subproblem of single-row routing. Both are significantly faster than the one proposed in [13]. In addition, the solutions obtained by each compare very favorably to those obtained by the algorithm of [13].

REFERENCES

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

[2] S. Y. Han and S. Sahni, "Single row routing in narrow street," *IEEE Trans. Computer-Aided Design*, vol. CAD-3, no. 3, pp. 235–241, 1984.

[3] S. Y. Han and S. Sahni, "A fast single row routing algorithm," in *Proc. 23rd Annual Allerton Conf. on Communication, Control, and Computing*, 1985, pp. 676–685.

[4] E. Horowitz and S. Sahni, *Fundamentals of Data Structures*. Rockville, MD: Computer Science Press, 1976.

[5] 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.

[6] R. Raghavan, "The wire routing problem: Algorithms and complexity results," Ph.D. dissertation, Univ. Minnesota.

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

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

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

[10] 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.

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

[12] 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.

[13] S. Tsukiyama, E. S. Kuh, and I. Shirakawa, "On the layering problem of multilayer PWB wiring," *IEEE Trans. Computer-Aided Design*, vol. CAD-2, no. 1, pp. 30–38, Jan. 1983.

*

Sangyong Han received the B.S. (engineering) degree from the Seoul National University, Seoul, Korea, and the Ph.D. degree in computer science from the University of Minnesota, Minneapolis, MN.

He was with the Korea Advanced Institute of Science and Technology and is presently with IBM Kingston, NY.

His interests in design automation include placement, routing, and VLSI design.

*

Sartaj Sahni (M'79) is a Professor of Computer Science at the University of Minnesota. He received the B.Tech. (electrical engineering) degree from the Indian Institute of Technology, Kanpur, and the M.S. and Ph.D. degrees in computer science from Cornell University. His publications are on the design and analysis of efficient algorithms, parallel computing, interconnection networks, and design automation. He is also a coauthor of the texts: *Fundamentals of Data Structures* and *Fundamentals of Computer Algorithms*, and author of the texts: *Concepts in Discrete Mathematics* and *Software Development in Pascal*. He is the area editor for Parallel Algorithms and Data Structures for the *Journal of Parallel and Distributed Computing*.