SIAM J. COMPUT.
Vol. 16, No. 4, August 1987

006

A 1.6 APPROXIMATION ALGORITHM FOR ROUTING MULTITERMINAL NETS*

TEOFILO F. GONZALEZ† AND SING-LING LEE‡

Abstract. The problem of connecting a set of n terminals belonging to m (signal) nets that lie on the sides of a rectangle to minimize the total area is discussed. We present an $O(n(m+\log n))$ approximation algorithm to solve this problem. Our algorithm generates a solution with area $\leqq 1.6 * \text{OPT}$, where OPT is the area of an optimal solution. The nets are routed according to the following greedy strategy: the wire connecting all points from a net is one whose path crosses the least number of corners of the rectangle. For some nets there are several routes that cross the least number of corners. A subset of these nets is connected by wires whose paths blend with the paths for other nets. The remaining nets are routed using several strategies and 2^6 layouts are obtained. The best of these layouts is the solution generated by our algorithm.

Key words. algorithms, wire routing, VLSI, minimize area, greedy methods

AMS(MOS) subject classification. 68

1. Introduction. Let G be a rectangle and S be a set of n points (terminals) that lie on the sides of G. Let $N_1, N_2, \cdots, N_m$ be any partition of set S. Each subset N_i is called a net. All the points in each net have to be made electrically common by interconnecting them with wires. The wires follow a path consisting of a finite number of horizontal and vertical line segments. These line segments are assigned to two different layers. All the horizontal line segments are assigned to one layer and all the vertical ones are assigned to the other layer. Line segments on different layers can be connected directly at any given point z by a wire perpendicular to the layers if both line segments cross point z in their respective layers. Every pair of distinct and parallel line segments must be at least $\lambda > 0$ units apart and every line segment must be at least λ units from each side of G, except in the region where the path joins a point in S it connects. Also, no path is allowed inside of G on any of the layers.

Problem R1M (routing around one module) consists of specifying the paths for all the wires in such a way that the total area is minimized, that is, placing G together with all the wires (that must satisfy the restrictions imposed above) inside a rectangle (with the same orientation as G) of least possible area. This problem has applications in the layout of integrated circuits ([L] and [R]) and conforms to a set of design rules for VLSI systems [MC].

The 2-R1M is defined similarly, except that all nets are restricted to be of size two. Hashimoto and Stevens present an $O(n \log n)$ algorithm to solve the R1M problem for the case when all the points in S lie on one side of G. An $\Omega(n \log n)$ lower bound on the worst case time complexity for this problem was established in [GLL]. Algorithms to solve the 2-R1M problem appear in [La] and [GL2]. The one in [GL2] is optimal with respect to the time complexity bound. If more than two layers are allowed and wire overlap is permitted, the problem becomes an NP-hard problem [SBR]. Other generalizations of the 2-R1M problem have been shown to be NP-hard [La]. Gonzalez

* Received by the editors July 1, 1986; accepted for publication December 18, 1986. A preliminary version of this paper was presented at the 22nd Allerton Conference on Communication, Control and Computing, October 1984.

† Department of Computer Science, University of California, Santa Barbara, California 93106. The work of this author was supported in part by the National Science Foundation under grant DCR-8503163.

‡ Department of Computer Science, Pennsylvania State University, University Park, Pennsylvania 16802.

and Lee [GL3] present an $O(n(m+\log n))$ approximation algorithm for the R1M problem that generates a solution with area $\leqq 1.69 * \text{OPT}$. In this paper we present an approximation algorithm with a worst case approximation bound of 1.6. This new algorithm is more complex than the previous one and the proof for the 1.6 bound is much more elaborate than the one for the 1.69 bound that appears in [GL3]. The introduction and definitions in this paper are similar to the ones in [GL3].

The first few steps of our procedure follow the initial steps of the algorithms in [La] and [GL2]. These steps reduce our problem to the one of selecting the starting point for the path connecting each global net (a net is called a global net if at least two of its terminals are located on opposite sides of G). The algorithm routes all nets by a path that crosses the least number of corners of G. For some nets there are several paths that cross the least number of corners. A subset of these nets is connected by paths that blend with the paths connecting other nets. The paths connecting the remaining nets are selected using several strategies and 2^6 layouts are obtained. The best of these layouts is the solution generated by our algorithm.

For $i=2, 3, 4$, let R1M-i denote an R1M problem in which all global nets contain terminals on exactly i sides of G. In § 2 we introduce our notation and present some basic results. In order to simplify the exposition of our results, we begin by presenting approximation algorithms for restricted versions of the R1M problem. In § 3 we present an approximation algorithm for the R1M-3 problem. An approximation algorithm for the R1M-4 problem is presented in § 4 and in § 5 we present an approximation algorithm for the R1M-2 problem. In § 6 we combine the results obtained in §§ 3–5 to obtain our 1.6 approximation algorithm for the R1M problem.

The algorithm for the R1M-3 problem is identical to the one in [GL3]. The analysis of this algorithm is a little bit more complex in this paper because one needs to prove the 1.6 bound. The analysis in [GL3] only proves the bound of 1.66. The algorithm for the R1M-2 is a generalization of the one in [GL3]. The main difference is that instead of constructing 2^3 layouts, one has to consider 2^6 layouts. In the former case we can only claim solutions within 69% of optimal, whereas in the latter case the approximation bound is 1.60. In both cases the analysis of the approximation bound is tight, i.e., it cannot be improved. The analysis in this paper for the 2^6 layouts algorithm is much more elaborate than the one in [GL3] for the 2^3 layouts algorithm. The algorithm for the R1M-4 problem is much more complex than the one in [GL3]. For brevity we did not include the postprocessing procedure for the R1M-2 problem in [GL3]. This procedure as well as its formal justification are included in this paper. A significant improvement over the 1.6 approximation bound seems impossible if one is restricted to our techniques. Perhaps a better approximation bound can be obtained by routing all nets concurrently; however, we doubt that the analysis could be simplified.

2. Basic results. We begin by defining the R1M problem and introducing notation similar to the one in [GL1]. Let G be a rectangular component of size h by w (height by width). There are n terminals $(T_1, T_2, \cdots, T_n)$ on the sides of G. These terminals are partitioned into m subsets denoted by $N_1, N_2, \cdots, N_m$. Each subset N_i is called a net and it is assumed that $|N_i|>1$ for all i. The problem depicted in Fig. 2.1 consists of five nets: $N_1=\{T_2, T_4, T_7\}$, $N_2=\{T_1, T_3, T_{10}, T_{14}\}$, $N_3=\{T_5, T_{12}\}$, $N_4=\{T_6, T_8, T_9\}$ and $N_5=\{T_{11}, T_{13}, T_{15}\}$. It is assumed that every pair of terminals is at least $\lambda>0$ units apart and every terminal is located at least λ units from each of the corners of G. All the terminals in each net must be made electrically common by connecting them with wires. The path followed by these wires can be partitioned into a finite number of straight line segments. Each of these line segments must lie on the same plane as

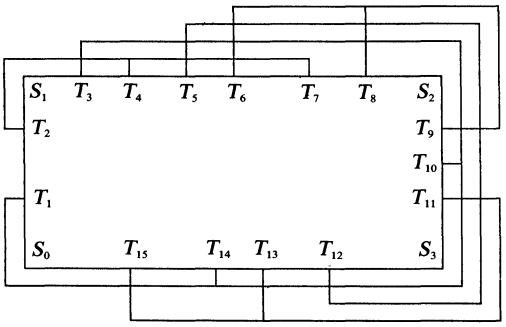

FIG. 2.1

G, be on the outside of G and be parallel to a side of G. Perpendicular line segments can intersect at any point, but parallel line segments must be at least λ units apart. Also, all line segments must be at least λ units away from every side of rectangle G except in the vicinity where a line segment connects a terminal. The *R1M problem* consists of specifying paths for all the interconnections subject to the rules mentioned above in such a way that the total area is minimized, i.e., place the component together with all the wires inside a rectangle (with the same orientation as G) of least possible area.

Label the sides of the component (in the obvious way) left, top, right and bottom. Starting in the bottom left-hand corner of G, traverse the sides of the rectangle clockwise. The ith corner to be visited is labeled S_{i-1}. Assume that the ith terminal visited is terminal T_i. The *close interval* $[x, y]$, where x and y are the corners of G or the terminals T_i, consists of all the points on the sides of G that are visited while traversing the sides of G in the clockwise direction starting at point x and ending at point y. Note that the interval $[x, x]$ consists of a single point. Parentheses are used instead of square brackets for open intervals. We use $[S_0, S_1]$, $[S_1, S_2]$, $[S_2, S_3]$ and $[S_3, S_0]$; 0, 1, 2 and 3; L, T, R and B; and L', T', R' and B' to represent the left, top, right and bottom sides of G, respectively. Terminal T_i is said to belong to side l, $S(i) = l$, if T_i is located in $[S_l, S_{(l+1) \bmod (4)}]$.

The function $I(j)$ indicates the index of the net to which terminal T_j belongs. The function $L(j)$ is defined in such a way that the interval $[T_j, T_{L(j)}]$ is the smallest interval that includes all the terminals from net $N_{I(j)}$. Set $D = \{d_1, d_2, \cdots, d_m\}$ is said to be an *assignment* if D contains exactly one index of a terminal from each net. Any subset of an assignment is said to be a *partial assignment.* An assignment D indicates the starting point for the path connecting all the points in each net. If $i \in D$ then the path connecting all the terninals in net $N_{I(i)}$ starts at terminal T_i moving perpendicular to side $S(i)$ and then it continues in the clockwise direction with respect to G until it reaches $T_{L(i)}$. Each terminal, T_k, in net $N_{I(i)}$ is joined to this path by a line segment perpendicular to side $S(T_k)$. The starting point for the paths connecting some nets might not be defined in a partial assignment. The assignment for the layout given by Fig. 2.1 is $\{2, 3, 5, 6, 11\}$. For any $l \in D$, we say that the path connecting net $N_{I(l)}$ given by D *crosses* point z if $z \in [T_l, T_{L(l)}]$. If $i \in D$, we say that the path connecting net $N_{I(i)}$ *begins* at point T_i and *ends* at point $T_{L(i)}$.

For any assignment (or partial assignment) D we define the *height function* H_D for $x, y \in \{T_1, T_2, \cdots, T_n\} \cup \{S_0, S_1, S_2, S_3\}$ as follows:

$$H_D(x, y) = \max \{\text{number of paths given by } D \text{ that cross point } z \mid z \in [x, y]\}.$$

For example, $H_D(S_0, S_1)$ is 1, $H_D(T_5, T_5)$ is 3 and $H_D(S_2, S_3)$ is 3 for the assignment,

D, whose layout appears in Fig. 2.1. We shall refer to $H_D(x, y)$ as the *height of assignment D for the interval $[x, y]$ of G.* The *height of assignment D for side X* ($X \in$ {*top, right, bottom. left*}) refers to the value of $H_D(x, y)$, where $[x, y]$ is the interval that represents side X. The *vertical* (*horizontal*) *height of assignment D* is the height on the top (left) plus the height on the bottom (right) side of G. The *total height of assignment D* is equal to the vertical plus the horizontal height of assignment D.

The next two lemmas establish that the R1M problem reduces to the problem of finding an assignment D with least

$$(h+(H_D(S_1, S_2)+H_D(S_3, S_0)) * \lambda) * (w+(H_D(S_0, S_1)+H_D(S_2, S_3)) * \lambda)$$

and then in $O(n \log n)$ time one can construct a layout of area h_Q by w_Q for it.

LEMMA 2.1. *For every assignment D, there is a rectangle Q of size h_Q by w_Q, where*

$$h_Q = h+(H_D(S_1, S_2)+H_D(S_3, S_0)) * \lambda,$$

$$w_Q = w+(H_D(S_0, S_1)+H_D(S_2, S_3)) * \lambda$$

with the property that rectangle G together with the interconnecting paths defined by D can be made to fit inside Q.

Proof. The proof is a direct generalization of the proof for the 2-R1M problem that appears in [La]. □

LEMMA 2.2. *For any assignment D a layout with the area given by Lemma* 2.1 *can be obtained in $O(n \log n)$ time.*

Proof. The proof of this lemma is a straightforward generalization of the proof for the 2-R1M problem that appears in [La]. The algorithm that constructs the final layout uses as a subalgorithm the procedure given in [GLL] and [HS]. □

Net N_i is said to be a *global net* if at least two of its terminals are located on opposite sides of G. Net N_i is said to be *local* otherwise, i.e., if all its terminals are located on the same side of G or on two adjacent sides of G. Nets $N_2 = \{T_1, T_3, T_{10}, T_{14}\}$ and $N_3 = \{T_5, T_{12}\}$ are the only global nets in the problem depicted in Fig. 2.1. For assignment D we define the *function $A(D)$* as

$$(h+(H_D(S_1, S_2)+H_D(S_3, S_0)) * \lambda) * (w+(H_D(S_0, S_1)+H_D(S_2, S_3)) * \lambda),$$

i.e., the total area required for a layout of G and all the interconnections given by D.

DEFINITION 2.1. Let D' be the partial assignment in which all the local nets are connected by paths crossing the least number of corners of G.

LEMMA 2.3. *Every assignment D can be transformed to an assignment M such that $D' \subseteq M$ and $A(M) \leqq A(D)$.*

Proof. The proof follows the same line as the one for the 2-R1M problem that appears in [La]. □

Lemma 2.3 shows that for any instance of the R1M problem there exists an optimal solution in which all local nets are connected by paths crossing at most one corner of G. The R1M problem has been reduced to the problem of finding the starting point (by default the direction is clockwise) for the paths connecting all the global nets in the presence of the partial assignment D'. At this point our algorithm abandons the procedures given in [La] and [GL2]. The main difficulty that we encounter in extending the results for the 2-R1M problem to this problem is that the divide-and-conquer step seems not applicable. The reason for this is that there seems to be no rule to separate the nets with terminals located in three or four sides of G into groups that can be optimally routed, independently of each other.

It should be clear that it is only required to specify the paths connecting all the global nets since we know that local nets can be routed optimally by routing them as

indicated in assignment D'. Also, once we have an assignment the proof of Lemma 2.2 (a constructive proof) can be used to find a layout of optimal area for it. In the next three sections we present approximation algorithms for the R1M-i problem, $2 \leqq i \leqq 4$. In § 6 we indicate how to combine these results to obtain our 1.6 approximation algorithm.

3. Approximation algorithm for the R1M-3 problem. In this section we present an approximation algorithm for the R1M-3 problem. Let M_3 be the set of global nets with terminals located on exactly three sides of G. Clearly, all global nets belong to set M_3. For the set of nets M_3 we construct assignment D_3 as follows:

$D_3 = \{$all nets in M_3 are connected by paths that cross the least number of corners of $G\}$.

Let $M_3^{TB}(M_3^{LR})$ be the set of all nets in M_3 without terminals located on either the left or right (top or bottom) side of G. Let m_3^{TB} and m_3^{LR} represent the number of elements in M_3^{TB} and M_3^{LR}, respectively. In Fig. 3.1(a) we give a layout for the assignment constructed by our algorithm for a net in M_3^{TB}. Suppose that this net is routed as in Fig. 3.1(b) in an optimal assignment D. Let M be D except for the path connecting the net that appears in Fig. 3.1(b) which is connected as indicated in Fig. 3.1(a). It is simple to show that

$$H_M(S_1, S_2) + H_M(S_3, S_0) \leqq H_D(S_1, S_2) + H_D(S_3, S_0) + 1,$$
$$H_M(S_0, S_1) + H_M(S_2, S_3) \leqq H_D(S_0, S_1) + H_D(S_2, S_3).$$

A straightforward generalization of the above observation is given by Lemma 3.1.

LEMMA 3.1. *Let D be an optimal assignment such that $D' \subseteq D$. Let M be D except that all nets in $M_3^{TB} \cup M_3^{LR}$ which are assigned as in our algorithm. Then*

$$H_M(S_1, S_2) + H_M(S_3, S_0) \leqq H_D(S_1, S_2) + H_D(S_3, S_0) + x_3^{TB},$$
$$H_M(S_0, S_1) + H_M(S_2, S_3) \leqq H_D(S_0, S_1) + H_D(S_2, S_3) + x_3^{LR}$$

where x_3^{TB} (x_3^{LR}) is the number of nets in X_3^{TB} (X_3^{LR}), and X_3^{TB} (X_3^{LR}) is a set of all nets in M_3^{TB} (M_3^{LR}) that are connected differently in D and M.

Proof. For brevity the proof is not included. □

Before proving our main result on this section, we establish a lower bound on the area required by any optimal solution. Our lower bound is given in Table 3.1.

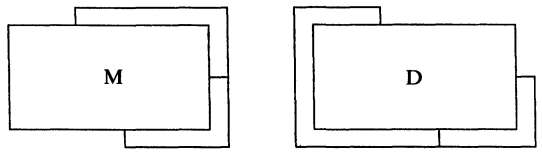

FIG. 3.1

TABLE 3.1
Lower bounds for the R1M-3 *problem.*

Set	Contribution to our lower bound for $h/(\lambda) + H_D(S_1, S_2) + H_D(S_3, S_0)$	Contribution to our lower bound for $w/(\lambda) + H_D(S_0, S_1) + H_D(S_2, S_3)$
M_3^{TB}	$1.5m_3^{TB} + 0.5x_3^{TB}$	$2m_3^{TB} + 0.5x_3^{TB}$
M_3^{LR}	$2m_3^{LR} + 0.5x_3^{LR}$	$1.5m_3^{LR} + 0.5x_3^{LR}$

LEMMA 3.2. *Let D be an optimal assignment such that $D' \subseteq D$. Assignment D and rectangle G satisfy the lower bounds given in Table* 3.1.

Proof. We only prove the lower bounds in the first column of Table 3.1 since the proof for the bounds in the second column is similar. The proof for these bounds can be obtained by adding the lower bounds given in Table 3.2. Therefore, we only need to prove the lower bounds given in Table 3.2.

Each net in M_3^{TB} has at least one terminal located on the left or right side of G and each net in M_3^{LR} has at least one terminal located on the left and on the right sides of G. Therefore, either on the left or the right side of G there are at least $0.5m_3^{TB}+m_3^{LR}$ terminals. Since every pair of these terminals is at least λ unit apart, we know that $h \geqq (0.5m_3^{TB}+m_3^{LR})\lambda$. This lower bound is given by the first column of Table 3.2.

TABLE 3.2

Set	Contribution to our lower bound for $h/(\lambda)$	Contribution to our lower bound for $H_D(S_1, S_2)+H_D(S_3, S_0)$
M_3^{TB}	$0.5m_3^{TB}$	$m_3^{TB}+0.5x_3^{TB}$
M_3^{LR}	m_3^{LR}	$m_3^{LR}+0.5x_3^{LR}$

Let us now establish a lower bound on the number of paths crossing the corners of G. Since in D every path connecting a net in $X_3^{TB} \cup X_3^{LR}$ crosses at least three corners of G and every path connecting a net in $(M_3^{TB}-X_3^{TB}) \cup (M_3^{LR}-X_3^{LR})$ crosses at least two corners of G, we know that

$$\sum_{i=0}^{3} H_D(S_i, S_i) \geqq 2(m_3^{LR}-x_3^{LR}+m_3^{TB}-x_3^{LR})+3(x_3^{LR}+x_3^{TB}).$$

Since the height on the top (bottom) side of G is at least as large as the average height of the two top (bottom) corners of G, we know that

$$H_D(S_1, S_2)+H_D(S_3, S_0) \geqq 0.5 * (H_D(S_1, S_1)+H_D(S_2, S_2)) + 0.5 * (H_D(S_3, S_3)+H_D(S_0, S_0)).$$

The lower bound for the second column of Table 3.2 follows from the above inequalities. The proof for the lower bounds for the first column of Table 3.1 is obtained by adding the bounds given by Table 3.2. This completes the proof of the lemma. □

THEOREM 3.1. *For the* R1M-3 *problem, let D be an optimal assignment such that $D' \subseteq D$ and let M be the assignment generated by our algorithm. Then, $A(M) \leqq 1.6 * A(D)$.*

Proof. From Lemma 3.1 and 3.2 we know that

$$\frac{A(M)}{A(D)} \leqq \left(1+\frac{a}{b}\right) * \left(1+\frac{c}{d}\right)^{1}$$

where

$$a = x_3^{TB},$$
$$b = 1.5m_3^{TB}+0.5x_3^{TB}+2m_3^{LR}+0.5x_3^{LR},$$
$$c = x_3^{LR},$$
$$d = 2m_3^{TB}+0.5x_3^{TB}+1.5m_3^{LR}+0.5x_3^{LR}.$$

[1] Note that a, b, c and d are nonnegative. If b or d is zero, the a, b, c and d will be zero and our assignment is optimal.

Since $m_3^{TB} \geqq x_3^{TB}$ and $m_3^{LR} \geqq x_3^{LR}$, we only need to prove that

$$(1+x/(2x+2.5y)) * (1+y/(2.5x+2y)) \leqq 1.5$$

where x and y are positive integers. A straightforward manipulation of this inequality results in the following inequality:

$$(7.5x^2+15.25xy+7.5y^2)/(5x^2+10.25xy+5y^2) \leqq 1.5.$$

Since the remaining part of the proof is simple, it will be omitted. This completes the proof for Theorem 3.1. □

4. Approximation algorithm for the R1M-4 problem. In this section we present an approximation algorithm for the R1M-4 problem. Let M_4 be the set of all global nets with terminals located on the four sides of G and let m_4 represent the number of nets in it. Clearly, all global nets belong to set M_4. In what follows we partition the set of nets M_4 into sets that will be routed independently of each other. There are several reasons for taking this approach. The main reason is that there are basically two types of nets: the ones with a large number of terminals and the ones with few terminals. For the first set of nets it is difficult to find routings with small area, but for the second type of nets such routing can be easily obtained. In our algorithm we do not route tightly the first type of nets. This routing does not increase our approximation bound significantly since these nets contribute large values to our lower bound function. The other type of nets have to be routed tightly, since their contribution to our lower bound function is small.

Nets N_i and N_j (both in set M_4) are said to be *agreeable* if on at least one side of G no terminal from net N_i is between any two terminals from net N_j and no terminal from net N_j is between any two terminals from net N_i. Procedure PARTITION, defined below, partitions the set M_4 into the sets M_4^A, M_4^N, $M_4^{T'}$, $M_4^{R'}$, $M_4^{B'}$, $M_4^{L'}$, M_4^T, M_4^R and M_4^B. Since some of these sets consist of 1-tuples, 2-tuples and 4-tuples, by partition we mean that every element in set M_4 is in one and only one of the tuples in these sets. After the algorithm terminates, let m_4^X represent the number of tuples in M_4^X, for $X \in \{A, N, T', R', B', L', T, R, B\}$. It is very important to keep in mind that sets M_4^A, $M_4^{T'}$, $M_4^{R'}$, $M_4^{B'}$ and $M_4^{L'}$ consist of 1-tuples, set M_4^N has only 4-tuples and the remaining sets contain only 2-tuples. When algorithm PARTITION terminates, the sets our procedure generates satisfy the following properties. Every net in set M_4^A has exactly four terminals (one on each side of G). Because of this, every pair of such nets is agreeable on each of the four sides of G. Every net in a 4-tuple of set M_4^N has at least two terminals on each side of G and the four nets in each 4-tuple are totally nonagreeable nets, i.e., every pair of these nets is not agreeable (on each of the four sides of G). Each net in set M_4^X, $X \in \{T', R', B', L'\}$, has at least two terminals located on side X and exactly one terminal located on each of the remaining sides of G. Every pair of nets in M_4^X, $X \in \{T', R', B', L'\}$, is agreeable on all sides of G, except possibly on side X. Each 2-tuple in sets M_4^X, for $X \in \{T, R, B\}$, contains a pair of nets that are agreeable on at least one side of G. Every net in set M_4^X, $X \in \{T, R, B\}$ satisfies one of the following conditions:

(i) it contains at least two terminals on at least two sides of G, or

(ii) it contains at least two terminals on side X and either exactly one terminal on the opposite side of X or at least two terminals on the opposite side of X when the two nets in the tuple are agreeable on an adjacent side of X (one net might violate this condition, i.e., it has only one terminal on the opposite side of X).

PROCEDURE PARTITION
//Partition all nets with exactly one terminal located on each side of G.//
$M_4^A \leftarrow \{N_i \mid N_i \in M_4$ and $|N_i| = 4\}$;
$W \leftarrow M_4 - M_4^A$;
//Partition all nets with at least two terminals located on each side of G.//
$Z \leftarrow \{N_i \mid N_i \in W$ has at least two terminals located on each side of $G\}$;
$W \leftarrow W - Z$;
$M_4^T \leftarrow \emptyset$; $M_4^R \leftarrow \emptyset$; $M_4^B \leftarrow \emptyset$; $M_4^N \leftarrow \emptyset$;
while $|Z| \geqq 4$ *do*
  let N_i, N_j, N_k and N_l be any four nets in Z;
  if any two of these four nets are agreeable
    then delete from Z two agreeable nets (two out of the four) and add them as a pair to M_4^T if these two nets are agreeable on the left or right side of G, otherwise add these two nets as a pair to M_4^R
    else delete all four nets from Z
      $M_4^N \leftarrow M_4^N \cup \{(N_i, N_j, N_k, N_l)\}$
  endif;
endwhile;
//Later on we explain what to do when $Z \neq \emptyset$. Note that $|Z| \leqq 3$.//
//Partition all nets with at least two terminals located on some side of G and exactly one terminal located on the remaining sides of G.//

$M_4^{T'} \leftarrow \emptyset$, $M_4^{R'} \leftarrow \emptyset$, $M_4^{B'} \leftarrow \emptyset$, $M_4^{L'} \leftarrow \emptyset$;

while there is a net in W with at least two terminals located on side X and exactly one terminal located on the remaining sides of G *do*.
  Let y be one of such nets and let $X \in \{T', R', B', L'\}$ be the side on which it has more than one terminal;
  $W \leftarrow W - \{y\}$;
  $M_4^X \leftarrow M_4^X \cup \{y\}$;
endwhile;
//Partition the remaining nets, i.e., all nets with at least two terminals located on two sides of G and exactly one terminal located on another side of G.//
$Z \leftarrow \{N_i \mid N_i \in W$ has at least two terminals located on the top side of $G\}$;
$W \leftarrow W - Z$;
while there are two nets in Z with exactly one
    terminal located on the bottom side of G *do*
  delete two of such nets from Z and add them as a pair to M_4^T;
endwhile;
while there are two nets in Z with exactly one
    terminal located on the same side of G *do*
  delete two of such nets from Z and add them as a pair to M_4^T
endwhile;
//Later on we explain what to do when $Z \neq \emptyset$. Note that $|Z| \leqq 3$.//
$Z \leftarrow \{N_i \mid N_i \in W$ has at least two terminals located on the right side of $G\}$;
$W \leftarrow W - Z$;
while there are two nets in Z with exactly one
    terminal located on the left side of G *do*
  delete two of such nets from Z and add them as a pair to M_4^R;
endwhile;

while $|Z| \geqq 2$ *do*
  delete any two nets from Z and add them as a pair to M_4^R
endwhile;
//Later on we explain what to do when $Z \neq \emptyset$. Note that $|Z| \leqq 1$//
$Z \leftarrow \{N_i \mid N_i \in W$ has at least two terminals located on the bottom side of $G\}$;
$W \leftarrow W - Z$;
while $|Z| \geqq 2$ *do*
  remove any two nets from Z and add them to M_4^B as a pair;
endwhile;
//Later on we explain what to do when $Z \neq \emptyset$. Note $|Z| \leqq 1$//
end of procedure

For some problem instances the algorithm fails to assign some nets to any of the sets M_4^X, $X \in \{A, N, T', R', B', L', T, R, B\}$. Let us assume that this is not the case, i.e., Z is always empty whenever we reach each of the "// later on $\cdots$ //" comments in our procedure. In § 6 we explain what to do when this is not the case. From procedure PARTITION it is simple to verify that no more than eight nets will be left out.

(a) *Construction of the assignment* D_A''. In what follows we explain how the assignment D_A'' for all the nets in set M_4^A is constructed. Figure 4.1 gives a layout for the assignment D_A'' constructed by our procedure for a set M_4^A with four nets. For any permutation, π, of the nets in set M_4^A and an integer i $(0 \leqq i \leqq 3)$ we define an assignment (denoted ASG (π, i)) as follows: the first net in π is connected by a path that begins on side i and ends on side $(i+3) \bmod (4)$; and the path connecting the kth net $(1 < k \leqq m_4^A)$ in π begins on side j and ends on side $(j+3) \bmod (4)$, where j is the side on which the path connecting the $(k-1)$st net in π ends. We claim that for every integer i $(0 \leqq i \leqq 3)$ there is a permutation, π (that depends on i) of the nets in set M_4^A such that there is a layout for assignment ASG(π, i) with the property that for any k $(1 < k \leqq m_4^A)$ the path connecting the $(k-1)$st net in π and the path connecting the kth net in π can share the same track on the side where the path connecting the $(k-1)$st net ends. In this case we say that π *is a valid permutation with respect to* i.

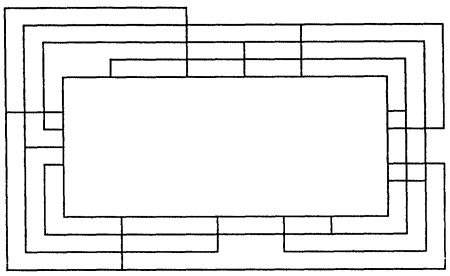

FIG. 4.1. *Assignment* D_A'' *for the case when* $m_4^A = 4$.

CLAIM. For every integer i, $0 \leqq i \leqq 3$, there is a valid permutation for the set of nets in M_4^A.

Proof. The proof of this claim is by induction on the number of nets in M_4^A. Clearly, the claim is true when set M_4^A has only one net. Assume that the claim holds when there are $k \geqq 1$ nets. We now show the claim holds when there are $k+1$ nets in M_4^A. Let $N_j \in M_4^A$ be the net with the topmost terminal on the left side of G. By the induction hypothesis we know that there is a valid permutation π' with respect to some i for the remaining k elements in M_4^A such that the path connecting the last net in the permutation ends on the left side of G. Since N_j was selected to be the net with

the topmost terminal located on the left side of G, we know permutation $\pi = (\pi', N_j)$ is a valid permutation for some value of i. To show the existence of the three other permutations (for the other values of i) we repeat the above procedure three times. The first time we let N_j be the net with the rightmost terminal located on the top side of G, the second time N_j is the net with the bottommost terminal located on the right side of G, and the last time N_j is the net with the leftmost terminal located on the bottom side of G. This completes the proof of the claim.

A valid permutation, π, for $i = 1$ can be constructed by a simple recursive procedure. Once π is obtained, the assignment $D''_A = \text{ASG}(\pi, i)$ can be easily constructed. Figure 4.1 gives a layout for the assignment D''_A constructed by our procedure for set M_4^A with four nets. Note that the contribution to the total height of the assignment is at most $3 * m_4^A + 1$.

(b) *Construction of the assignment* D''_N. The assignment D''_N is constructed by applying the following rule to each 4-tuple in M_4^N. Let (N_i, N_j, N_k, N_l) be any tuple in set M_4^N. Let us assume that these nets have been ordered in such a way that N_i is the net (amongst these four nets) whose bottommost terminal located on the left side of G is closest to the top side of G; net N_j is the net (amongst N_j, N_k and N_l) whose rightmost terminal located on the bottom side of G is closest to the left side of G; and net N_l is the net (amongst N_k and N_l) whose leftmost terminal located on the top side of G is closest to the right side of G. A layout for the assignment constructed for these nets is given in Fig. 4.2.

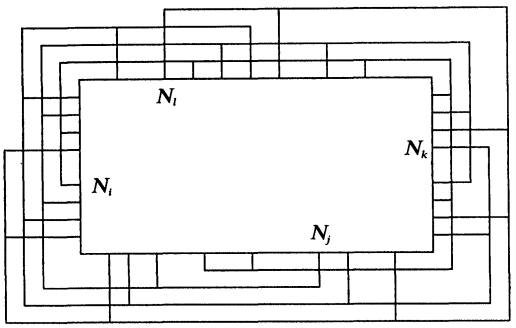

FIG. 4.2. *Assignment for a tuple in* M_4^N.

(c) *Construction of assignment* D''_X *for* $X \in \{T', R', B', L'\}$. Since all the assignments D''_X, $X \in \{T', R', B', L'\}$, are constructed by similar procedures, we only explain the one for assignment $D''_{T'}$. Assignment $D''_{T'}$ for $M_4^{T'}$ is constructed by applying a rule similar to the one used in part (a). The main difference is that we are only interested in valid permutations with respect to $i = 1$ and in a valid permutation it is not required for the $(k-1)$st net in π and the kth net in π to share the same track on the side where the path connecting the $(k-1)$st net ends (for $k = 5, 9, 13, \cdots$). Once a valid permutation is obtained the assignment $D''_{T'} = \text{ASG}(\pi, 1)$ can be easily constructed. Note that the total height of the assignment is at most $3m_4^{T'} + [m_4^{T'}/4]$.

(d) *Construction of assignment* D''_X *for* $X \in \{T, R, B\}$. Assignment D''_X, $X \in \{T, R, B\}$, for the set M_4^X is constructed by applying the following rule to each 2-tuple in M_4^X. Let (N_i, N_j) be any 2-tuple in set M_4^X. From algorithm PARTITION we know that both of these nets are agreeable on at least one side of G because either both of these nets have exactly one pin located on some side of G, or both of the nets have at least two terminals on each side of G and are agreeable on a side adjacent to X.

The pairs that have exactly one terminal on some side of G are assigned in such a way that there is a layout in which these nets share the same track on side Z of G and both nets have exactly one terminal located on side Z. Side Z is selected using the following priorities: the opposite side of X has the highest priority; and the adjacent sides of G have the lowest priorities. Note that side Z cannot be the same as side X. Figure 4.3 shows a layout for the assignment for a pair of nets in M_4^B with Z being the top side of G. The remaining pairs are assigned in such a way that they share the same track on a side adjacent to X.

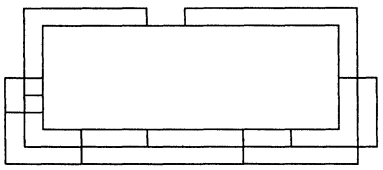

FIG. 4.3. *Layout for a 2-tuple in M_4^B with Z being the top side of G.*

(e) *Final assignment and proof of approximation bound.* Let $D_4 = D''_A \cup D''_N \cup D''_{T'} \cup D''_{R'} \cup D''_{B'} \cup D''_{L'} \cup D''_T \cup D''_R \cup D''_B$. The output of our approximation algorithm is $D_4 \cup D'$. Before showing that our algorithm generates a solution within 60% of the optimal solution value, we need to establish upper bounds for the area of the layouts obtained from assignment D_4 and prove lower bounds for the area of an optimal solution.

In what follows we assume that m_4^A is a multiple of four. In § 6 we indicate what to do when this is not the case.

LEMMA 4.1. *Let D be an optimal assignment such that $D' \subseteq D$. Let M be D except that all the nets in M_4^A are routed as in our approximation algorithm. Then*

$$H_M(S_1, S_2) + H_M(S_3, S_0) \leqq H_D(S_1, S_2) + H_D(S_3, S_0) + 0.5m_4^A + 1,$$

$$H_M(S_0, S_1) + H_M(S_2, S_3) \leqq H_D(S_0, S_1) + H_D(S_2, S_3) + 0.5m_4^A.$$

Proof. From the construction rule for assignment D''_A (a), we know that the contribution of the nets in M_4^A to the total height of any assignment that includes D''_A is at most $3m_4^A + 1$. Also, for any assignment that includes D''_A the maximum contribution to the vertical height by the paths connecting the nets in M_4^A is at most $1.5m_4^A + 1$ and the maximum contribution by these nets to the horizontal height is at most $1.5m_4^A$. In an optimal assignment each path connecting a net in M_4^A must contribute to the vertical and horizontal height at least one. Therefore, when transforming D to M the total increase in vertical and horizontal height is given by the inequalities in the statement of the lemma. This completes the proof of the lemma. □

In what follows we assume that m_4^X, $X \in \{T', R', B', L'\}$, is a multiple of four. In § 6 we indicate what to do when this is not the case.

LEMMA 4.2. *Let D be an optimal assignment such that $D' \subseteq D$. Let M be D except for all the nets in M_4^X, $X \in \{T', R', B', L'\}$, are routed as in our approximation algorithm. Then*

$$H_M(S_1, S_2) + H_M(S_3, S_0) \leqq H_D(S_1, S_2) + H_D(S_3, S_0) + x * m_4^X,$$

$$H_M(S_0, S_1) + H_M(S_2, S_3) \leqq H_D(S_0, S_1) + H_D(S_2, S_3) + y * m_4^X$$

where

$$x = \begin{cases} 0.75 & \text{if } X \in \{T', B'\}, \\ 0.5 & \text{otherwise,} \end{cases} \qquad y = \begin{cases} 0.5 & \text{if } X \in \{T', B'\}, \\ 0.75 & \text{otherwise.} \end{cases}$$

Proof. Since the proof of all cases is similar, we only prove the case for $X = T'$. From the construction rule for assignment $D''_{T'}$ (c), we know that the contribution from the nets in set $M_4^{T'}$ to the total vertical height of an assignment that include $D''_{T'}$ is at most $(1.75) * m_4^{T'}$ and with respect to the total horizontal height is at most $(1.5) * m_4^{T'}$. On any optimal assignment, D, the contribution to the vertical and horizontal height by these nets is at least $m_4^{T'}$. Hence, when transforming D to our assignment, the difference in heights is given by the inequalities in the statement of the lemma. This completes the proof of the lemma. □

For $X \in \{T, R, B\}$ let p_4^X be the faaction of pairs in M_4^X that are connected in D''_X by paths that do not overlap on the top or bottom side of G, and $q_4^X = 1 - p_4^X$. Since all pairs of nets in M_4^B are agreeable on the top side of G, we know from our routing algorithm that $q_4^B = 0$.

LEMMA 4.3. *Let D be an optimal assignment such that $D' \subseteq D$. Let M be D except that all nets in M_4^X, $X \in \{T, R, B\}$, are routed as in our approximation algorithm. Then*

$$H_M(S_1, S_2) + H_M(S_3, S_0) \leqq H_D(S_1, S_2) + H_D(S_3, S_0) + p_4^X * m_4^X + 2q_4^X * m_4^X,$$

$$H_M(S_2, S_3) + H_M(S_0, S_1) \leqq H_D(S_2, S_3) + H_D(S_0, S_1) + 2p_4^X * m_4^X + q_4^X * m_4^X.$$

Proof. Since the proof of all three cases is similar we only prove the case for $X = T$. In assignment D each net with terminals located on the four sides of G contributes to the vertical height and to the horizontal height at least one unit. Therefore, the contribution to the vertical and horizontal heights of D for a pair of nets in M_4^T is at least two. For any assignment a pair of nets routed in such a way that they can share the same track on the top or bottom side of G contributes to the vertical height at most three and their contribution to the horizontal height is at most four. For the nets routed following this sharing rule, we know that transforming D to M increases the vertical and horizontal height as indicated by the inequalities in the statement of the lemma. A similar result can be obtained for pairs of nets that are routed in such a way that they share a track on the left or right side of G. This completes the proof of the lemma. □

LEMMA 4.4. *Let D be an optimal assignment such that $D' \subseteq D$. Let M be D except that all nets in M_4^N are routed as in our approximation algorithm. Then*

$$H_M(S_1, S_2) + H_M(S_3, S_0) \leqq H_D(S_1, S_2) + H_D(S_3, S_0) + 7x * m_4^N,$$

$$H_M(S_0, S_1) + H_M(S_2, S_3) \leqq H_D(S_0, S_1) + H_D(S_2, S_3) + 7y * m_4^N,$$

where x, y are nonnegative reals such that $x + y = 1$.

Proof. One can easily show that we only need to prove that when assigment D is transformed to assignment M the total height increases by at most seven for each tuple in M_4^N. Let us now prove that this statement holds for any tuple, H, in M_4^N. There are two cases.

Case 1. At least one of the paths connecting a net in H begins and ends on the same side of G in D.

For this case we know that the contribution to the total height in D is at least three for the nets connected in D by paths that begin and end on the same side of G. The remaining connecting paths contribute at least two units to the total height.

Therefore the contribution to the total height of D by these nets is at least nine. On assignment, D''_N the contribution to the total height for the same set of nets is at most sixteen. Hence, the total height increases by at most seven after the transformation.

Case 2. In assignment D, no path connecting a net in H begins and ends on the same side of G.

There are four subcases depending on how the nets in H are connected.

Subcase 2.1. In assignment D, two of the paths connecting nets in H begin on adjacent sides of G.

The paths connecting these two nets are depicted in Fig. 4.4. It is simple to see that the contribution to the total height of assignment D by these two nets is at least five. The contribution to the total height of D by the remaining nets is at least four. The proof now follows similar steps to the ones in Case 1.

Subcase 2.2. In assignment D all the paths connecting nets in H begin on the same side of G.

Since the paths connecting all the nets in H begin on the same side of G, we know that only three nets from H are connected differently in D and D''_N. Each time the path connecting one of these nets is changed, the maximum increase of the total height is at most two. Hence, the total height increases by at most six.

Subcase 2.3. In assignment D the paths connecting all the nets in H begin on the left or the right side of G and at least one of these paths begins on each of these two sides.

Assume that no net in H is connected identically in D''_N and D, as otherwise the proof of Subcase 2.2 can be used to complete the proof for this case. The total height increases every time a path is interchanged by at most two, except a path that begins on the left side of G in D''_N. This is because x is always greater than y (see x and y in Fig. 4.5). Remember that such a net was selected to be the one whose bottommost element on the left side of G is closest to the top side. Hence the total height increases by at most seven units.

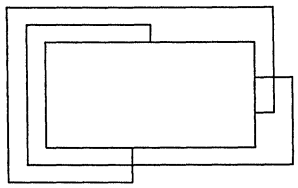

FIG. 4.4

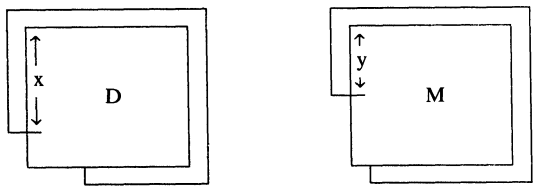

FIG. 4.5

Subcase 2.4. In assignment D the paths connecting all nets in H begin on the top or bottom side of G and at least one of these paths begins on each of these two sides.

Assume that no net in H is connected identically in D''_N and D, as otherwise the proof of Subcase 2.2 can be used to complete the proof for this case. The remaining part of the proof uses similar arguments to the ones used in the proof for Subcase 2.3 because either at least two of these paths begin on the bottom side of G or at least three of these paths begin on the top side of G. Since the connecting path that begins on the bottom side of G in our solution is the second "best" and the one that begins on the top side of G in our solution is the third "best," similar arguments to the ones used in Subcase 2.3 can be used to complete the proof for this subcase.

This completes the proof for Case 2 and the lemma. □

Before proving our main result in this section we establish a lower bound on the area required by any optimal solution. The lower bound is given in Table 4.1.

TABLE 4.1
Lower bounds for the R1M-4 *problem.*

Set	Contribution to our lower bound for $h/(\lambda)+H_D(S_1,S_2)+H_D(S_3,S_0)$	Contribution to our lower bound for $w/(\lambda)+H_D(S_0,S_1)+H_D(S_2,S_3)$
M_4^A	$2.5m_4^A$	$2.5m_4^A$
M_4^N	$14m_4^N$	$14m_4^N$
$M_4^{T'} \cup M_4^{B'}$	$2.5m_4^{T'}+2.5m_4^{B'}$	$3m_4^{T'}+3m_4^{B'}$
$M_4^{R'} \cup M_4^{L'}$	$3m_4^{R'}+3m_4^{L'}$	$2.5m_4^{R'}+2.5m_4^{L'}$
$M_4^T \cup M_4^B$	$p_4^T * m_4^T+5m_4^T+6m_4^B$	$q_4^T * m_4^T+6m_4^T+6m_4^B$
M_4^R	$p_4^R * m_4^R+6m_4^R$	$q_4^R * m_4^R+5m_4^R$

LEMMA 4.5. *Let D be an optimal assignment such that $D' \subseteq D$. Assignment D and rectangle G satisfy the lower bounds given in Table* 4.1.

Proof. We only prove the lower bounds in the first column of Table 4.1 since the proof for the bounds in the second column is similar. Before establishing this bound, we prove some intermediate lower bounds. It is simple to prove the following lower bounds:

(a) Each net in M_4^A has exactly one terminal on the left and right sides of G.

(b) The four nets in each tuple in M_4^N have at least eight terminals (two per net) on the left and right sides of G.

(c) Each net in M_4^X, $X \in \{T', B'\}$, has exactly one terminal on the left and right side of G. Each net in M_4^X, $X \in \{R', L'\}$ has at least two terminals on side X and exactly one terminal on the opposite side of X.

(d) Each net in each pair of nets in M_4^T routed in D''_T by sharing a track on the bottom side of T has at least one terminal on the left and right sides of G and at least two terminals on either the left or the right side of G. Each remaining pair in M_4^T has at least two terminals (one per net) on the left and right sides of T. Each pair of nets in M_4^B has at least two terminals on the right side (one per net) and at least four terminals on the left side (two per net). Each pair of nets in M_4^R routed in D''_R by sharing a track on the left side of T has at least two terminals (one per net) on the left side of G and at least four terminals (two per net) on the right side of T. The nets

in the remaining pairs in M_4^R have at least four terminals (two per net) on the left and right sides of G.

Using the above observations we obtain the bounds given by the first column of Table 4.2.

Let us now establish a lower bound on the number of paths crossing the corners of G. Since the path connecting every net in $M_4^A \cup M_4^{T'} \cup M_4^{R'} \cup M_4^{B'} \cup M_4^{L'}$ crosses at least three corners of G; the paths connecting every pair of nets M_4^X, $X \in \{T, B, L\}$ cross at least six corners (each path crosses three corners) of G and the paths connecting each four nets in a tuple of M_4^N cross at least twelve corners (each path crosses at least three corners) of G, we know that

$$\sum_{i=0}^{3} H_D(S_i, S_i) \geqq 3m_4^A + 12m_4^N + 3m_4^{T'} + 3m_4^{R'} + 3m_4^{B'} + 3m_4^{L'} + 6m_4^T + 6m_4^R + 6m_4^B.$$

It is simple to show that

$$H_D(S_1, S_2) + H_D(S_3, S_0) \geqq 0.5 * (H_D(S_1, S_1) + H_D(S_2, S_2)) + 0.5 * (H_D(S_3, S_3) + H_D(S_0, S_0)).$$

The lower bound for the second column of Table 4.2 follows from the above inequalities. The proof for the lower bounds for the first column of Table 4.1 can be obtained by adding the bounds given by Table 4.2. This completes the proof of the lemma. □

THEOREM 4.1. *For the* R1M-4 *problem, let D be an optimal assignment such that $D' \subseteq D$ and let M be the assignment generated by our algorithm. Then, $A(M) \leqq 1.6 * A(D)$.*

Proof. Using the Lemmas 4.1 to 4.5, we know that

$$\frac{A(M)}{A(D)} \leqq \left(1 + \sum_{i=1}^{6} a_i \Big/ \sum_{i=1}^{6} b_i\right) * \left(1 + \sum_{i=1}^{6} c_i \Big/ \sum_{i=1}^{6} d_i\right),^2$$

where a_i, b_i, c_i and d_i are defined in Table 4.3 and x, y are positive integers such that $x + y = 1$.

The proof of the theorem can be obtained by multiplying all terms after replacing 1 by $m_4^A/4$ (remember that $m_4^A \geqq 4$) and obtaining an expression of the form $(ax + by + \cdots)/(a'x + b'y + \cdots)$, where $a, b, \cdots, a', b', \cdots$ are constants and $x, y, \cdots$ are products of variables. The final step consists of showing that $a/a' \leqq 1.6$, $b/b' \leqq 1.6, \cdots$.

This completes the proof of the theorem. □

TABLE 4.2

Set	Contribution to our lower bound for $h/(\lambda)$	Contribution to our lower bound for $H_D(S_1, S_2) + H_D(S_3, S_0)$
M_4^A	m_4^A	$1.5m_4^A$
M_4^N	$8m_4^N$	$6m_4^N$
$M_4^{T'} \cup M_4^{B'}$	$m_4^{T'} + m_4^{B'}$	$1.5m_4^{T'} + 1.5m_4^{B'}$
$M_4^{R'} \cup M_4^{L'}$	$1.5m_4^{R'} + 1.5m_4^{L'}$	$1.5m_4^{R'} + 1.5m_4^{L'}$
$M_4^T \cup M_4^B$	$p_4^T * m_4^T + 2m_4^T + 3m_4^B$	$3m_4^T + 3m_4^B$
M_4^R	$p_4^R * m_4^R + 3m_4^R$	$3m_4^R$

[2] Note that a_i, b_i, c_i and d_i are nonnegative. If b_i or d_i is zero, then a_i, b_i, c_i and d_i and these elements may be deleted from the summation.

TABLE 4.3(a)

i	a_i	b_i
1	$0.5m_4^A+1$	$2.5m_4^A$
2	$7x * m_4^N$	$14m_4^N$
3	$0.75m_4^{T'}+0.75m_4^{B'}$	$2.5m_4^{T'}+2.5m_4^{B'}$
4	$0.5m_4^{R'}+0.5m_4^{L'}$	$3m_4^{R'}+3m_4^{L'}$
5	$q_4^T * m_4^T+m_4^T+m_4^B$	$p_4^T * m_4^T+5m_4^T+6m_4^B$
6	$q_4^R * m_4^R+m_4^R$	$p_4^R * m_4^R+6m_4^R$

TABLE 4.3(b)

i	c_i	d_i
1	$0.5m_4^A$	$2.5m_4^A$
2	$7y * m_4^N$	$14m_4^N$
3	$0.5m_4^{T'}+0.5m_4^{B'}$	$3m_4^{T'}+3m_4^{B'}$
4	$0.75m_4^{R'}+0.75m_4^{L'}$	$2.5m_4^{R'}+2.5m_4^{L'}$
5	$p_4^T * m_4^T+m_4^T+2m_4^B$	$q_4^T * m_4^T+6m_4^T+6m_4^B$
6	$p_4^R * m_4^R+m_4^R$	$q_4^R * m_4^R+5m_4^R$

5. Approximation algorithm for the R1M-2 problem. In this section we present an approximation algorithm for the R1M-2 problem. Clearly, all global nets contain terminals located on only two opposite sides of G. Let $M_2^{TB}(M_2^{LR})$ represent the set of global nets with terminals located only on the top and bottom (left and right) sides of G. Let m_2^{TB} and m_2^{LR} represent the number of nets in M_2^{TB} and M_2^{LR}, respectively. Assume that m_2^{TB} and m_2^{LR} are multiples of 6. In § 6 we indicate the modifications that need to be made to the algorithm when this is not the case. First, let us explain (informally) how the nets in these sets are routed. We will explain only the procedure for M_2^{TB}, since the one for M_2^{LR} is similar. The set M_2^{TB} is partitioned into six equally sized groups. Later on we explain precisely how this partition is obtained. A group is said to be routed by a path type $L(R)$ if all the nets in this group are routed by a path that does not cross the right (left) side of G. We will construct 2^6 assignments. Each of these assignments corresponds to a string of six elements from the alphabet $\{L, R\}$. The ith element in a string specifies the type of path used to connect all nets in the ith group. Let us now define formally this construction process. For each net N_j in M_2^{TB} (M_2^{LR}), let $p(j)$ be the index of the leftmost (bottommost) terminal of N_j located on the top (left) side of G. Let $P^{TB}=\{p(j)\,|\,N_j \in M_2^{TB}\}$, and $P^{LR}=\{p(j)\,|\,N_j \in M_2^{LR}\}$. Each of these sets is partitioned into the sets N_i^{TB} and N_i^{LR} for $1 \leqq i \leqq 6$ as follows

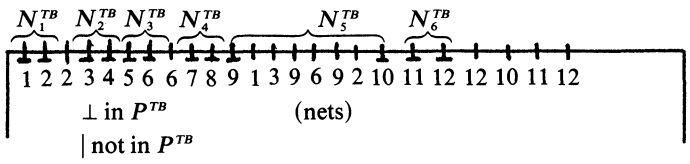

FIG. 5.0

(see Fig. 5.0). (Recall that function $I(j)$ was defined as the index of the net to which terminal T_j belongs.)

$N_i^{TB} = \{I(j) | j$ is the kth smallest value in the set P^{TB} and $((i-1)/6) * |P^{TB}| < k \leqq (i/6) * |P^{TB}|\}$ and

$N_i^{LR} = \{I(j) | j$ is the kth smallest value in the set P^{LR} and $((i-1)/6 * |P^{LR}| < k \leqq (i/6) * |P^{LR}|\}$.

We define the following sets for $0 \leqq i \leqq 63$:

$D_i^{TB} = \{$if in the binary representation of i the lth least significant bit is one then the path connecting each net in set N_l^{TB} does not cross corner S_2, otherwise the path connecting such nets does not cross corner $S_1 | 1 \leqq l \leqq 6\}$, and

$D_i^{LR} = \{$if in the binary representation of i the lth least significant bit is one then the path connecting each net in set N_l^{LR} does not cross corner S_1, otherwise the path connecting such nets does not cross corner $S_0 | 1 \leqq l \leqq 6\}$.

For $0 \leqq i, j \leqq 63$, define assignment $D_{i,j}$ as follows:

$$D_{i,j} = D' \cup D_i^{TB} \cup D_j^{LR},$$

and let P be one of the $D_{i,j}$'s of least area.

For simplicity, let us assume that in all optimal assignments the nets in $M_2^{TB} \cup M_2^{LR}$ are routed by paths that cross exactly two corners of G. In general this is not true; however our results also apply to these problem instances. Just before Lemma 5.4 we will indicate how to handle these problem instances.

Before proving our last upper bound we need to introduce additional notation. First of all let us concentrate on the set of nets in M_2^{TB}. Remember that in our algorithm we partitioned this set of nets into six groups and 2^6 assignments were obtained by routing all nets in each of these sets by paths that either do not cross the left or right side of G. Let us refer to each of these groups by G_i, for $1 \leqq i \leqq 6$ (these sets correspond to the sets N_i^{TB}, for $1 \leqq i \leqq 6$, defined above). Let D be an optimal assignment such that $D' \subseteq D$. Assignment D will be transformed into a nonemtpy set of assignments by an i-string. An i-string consists of a sequence of six elements from the alphabet {"r", "l", "|", "*", "."}. If the number of "*" symbols in the i-string is j, 2^j assignments are generated. The kth element in an i-string indicates how the paths connecting the nets G_k in D are changed. Let us now explain how these changes are performed. When the kth element in an i-string is "l", the paths connecting all the nets in G_k that cross the left side of G in assignment D are changed. These nets will be connected by paths that do not cross the left side of G. If the kth element is "r", the paths connecting all the nets in G_k that cross the right side of G in assignment D are changed. These nets will be connected by paths that do not cross the right side of G. When the kth symbol is "|", it means "l" if the number of paths connecting the nets in G_k that cross the left side of G in D is less than or equal to $(1/2) * |G_k|$; and it means "r" otherwise. When the element is a period "." it means that none of the paths connecting the nets in G_k are changed. The interpretation of "*" is more complex. An i-string with one or more "*" symbols denotes the set of all i-strings *obtainable* from it by replacing each of the "*" symbols with an "l" or an "r." When we apply an i-string with no "*" symbols to an optimal assignment, D, we obtain an assignment identical to D except for the paths connecting some of the nets in the groups G_i. The specific paths

to be changed are indicated by the i-string. Applying an i-string with "*" symbols, I, to an assignment, D, generates a (nonempty) set of assignments. This set of assignments consists of all assignments obtained from D by applying an i-string obtainable from I.

In Lemma 5.1 and the lemmas appearing in the appendices, we assume that when interchanging the paths connecting two nets in M_2^{TB} that cross on the top side of G will increase by at most two the vertical height of the assignment. The assumption is not always true. In Fig. 5.1 we show the only counterexample. We call this interchange a *type I interchange.* The two nets involved in this interchange form a *type I pair.* Interchanging the path connecting a net in M_2^{TB} to one that does not cross the left or right sides of G increases the vertical height of an assignment by at most two. This statement always holds.

Sometimes Lemma 5.1 holds even when type I interchanges occur when transforming an optimal assignment to the ones generated in our algorithm. Since there are cases when this will not be true we need to apply some postprocessing improvement to eliminate the effects of the type I interchanges. We will explain how to do this after Lemma 5.1. We should point out that after this postprocessing procedure the 1.6 approximation bound will hold for all problem instances.

LEMMA 5.1. *Let D be an optimal assignment such that $D' \subseteq D$. There is an assignment $D_{i,j}$ (constructed by our algorithm) such that if M is defined as D except for all the nets in $M_2^{TB} \cup M_2^{LR}$ which are routed as in the assignment $D_{i,j}$, then*

(a) $H_M(S_1, S_2) + H_M(S_3, S_0) \leqq H_D(S_1, S_2) + H_D(S_3, S_0) + (0.6) * m_2^{TB}$,

(b) $H_M(S_0, S_1) + H_M(S_2, S_3) \leqq H_D(S_0, S_1) + H_D(S_2, S_3) + (0.6) * m_2^{LR}$.

(It is assumed that when D is transformed to any of the assignments D_{ij} and two paths that cross on the top side of G are interchanged, such an interchange is not a type I interchange.)

Proof. The proof is given in Appendix A. □

Let L be an optimal solution that includes D'. The existence of at least one of these assignments is guaranteed by Lemma 2.3. Let S be any of the $D_{i,j}$ assignments. The difference between S and L is the way in which some nets in $M_2^{TB} \cup M_2^{LR}$ are routed. Lemma 5.1 shows that at least one of our assignments differs in vertical height from L by at most $0.6 * m_2^{TB}$ and in honrizontal height by at most $0.6 * m_2^{LR}$. As noted in the text appearing immediately before Lemma 5.1, we cannot yet claim that the 0.6 bound holds when type I interchanges occur.

When there is a type I interchange, the height on the top side of G could increase by at most one and the height of the bottom side of G could increase by at most two. If instead of increasing the total vertical height by at most three it would have been by at most two, then the previous lemma would be correct even with interchanges of type I. This is true sometimes, but in general we cannot guarantee that it will always happen. In order for our 0.6 bound to hold we transform each assignment S (each of the $D_{i,j}$'s) to another assignment R such that if the total increase in vertical height when transforming L to S by interchanging some of the nets in M_2^{LR} was $\leqq w+3\alpha$,

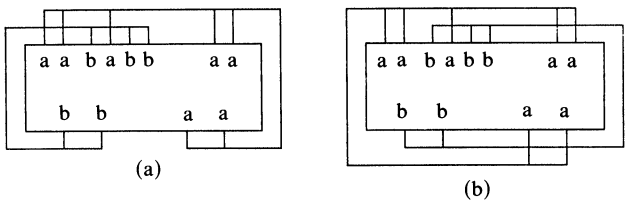

FIG. 5.1. *Type I interchange.* (a) *Before the interchange*; (b) *After the interchange.*

where α is the number of type I interchanges made and w is the contribution from all other interchanges, then the actual difference in vertical height between L and R is at most $w+2\alpha$. Before presenting our transformation algorithm we make a couple of definitions.

DEFINITION 5.1. A pair of nets, (a, b), in M_2^{TB} is said to form a *crossing pair* in an assignment if the following conditions are satisfied (see Fig. 5.2);

(i) Net "a" is connected by a path that crosses the left side of G.

(ii) Net "b" is connected by a path that crosses the right side of G.

(iii) On the bottom side of G all the terminals from net "b" appear to the left of all the terminals from net "a".

(iv) On the top side of G there is at least one terminal from net "a" located to the right of the rightmost terminal from net "b" and at least one terminal from net "a" is located to the left of the leftmost terminal from net "b".

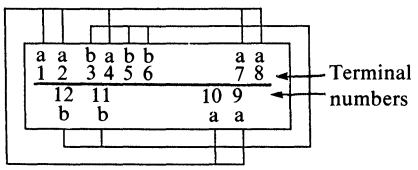

FIG. 5.2. *Crossing pair* (a, b).

One can easily prove that any two nets involved in a type I interchange will, after the interchange, form a crossing pair. However, the converse is not true. Later on we show that such a pair has harming effects only when it is a conflicting pair.

DEFINITION 5.2. A crossing pair (a, b) *includes* point x if either of the two following conditions is satisfied:

(i) If x is located on the top side of G, then all terminals from net "b" located on the top side of G appear to the left of x and the rightmost terminal from net "a" is located to the right of x (interval (6, 8) in Fig. 5.2). Or

(ii) If x is located on the bottom side of G, then all terminals from net "b" located on the bottom side of G appear to the left of x and all the terminals from net "a" are located to the right of x (interval (10, 11) in Fig. 5.2).

DEFINITION 5.3. A crossing pair (a, b) *partially includes* point x if x is located on the bottom side of G and there is a terminal from net b at point x or to the left of x, and there is a terminal from net a at point x or to the right of x (interval [9, 12] in Fig. 5.2).

Note that if point x located on the bottom side of G is included in a crossing pair then it is also partially included in it, but the converse is not always true.

DEFINITION 5.4. A *conflicting pair*, (a, b), is a crossing pair that includes the leftmost point with maximum height located on the top side of G, partially includes all the points with maximum height located on the bottom side of G and includes either the leftmost or the rightmost point with maximum height located on the bottom side of G.

Our postprocessing procedure will find conflicting pairs and interchange their connecting paths. In Fig. 5.3 we show a conflicting pair and in Fig. 5.4 we indicate how these paths are interchanged.

This transformation reverses the effects of type I interchanges.

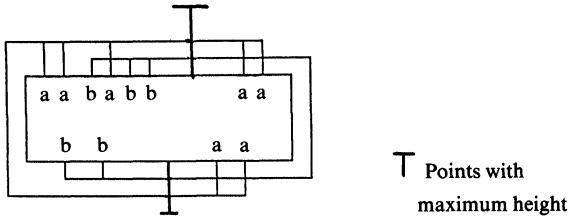

FIG. 5.3. *(a, b) forms a conflicting pair.*

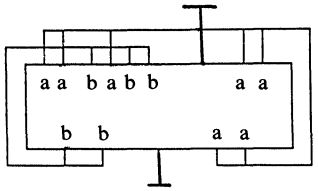

FIG. 5.4 *Interchange of the conflicting pair given in Fig. 5.3.*

ALGORITHM MODIFY
$R \leftarrow S$;
//The statement in this comment is not executed by the algorithm; its presence is to simplify the proof of Lemma 5.2.
Mark all nets involved in type I interchanges when L was transformed into S.//
while there is a conflicting pair in R *do*
interchange a conflicting pair in R
//The statement in this comment is not executed by the algorithm; its presence is to simplify the proof of Lemma 5.2.
Unmark all marked nets involved in the interchange just performed as well as all nets involved in a type I interchange with the nets in the conflicting pair just interchanged.//
endwhile
end of algorithm

During each iteration of the algorithm the maximum height on the top side of G is not increased; however the maximum height on the bottom side of G is decreased by at least one. Note that the maximum height on the top side of G is not increased because a conflicting pair includes the leftmost point with maximum height on the top side of G.

Let α be the number of type I interchanges made when transforming L to S and let $w+3\alpha$ be the total increase in vertical height because of such transformation. Let $R_0 = S$ and $\alpha_0 = \alpha$. For $i \geqq 1$ let R_i be assignment R at the end of the ith iteration of algorithm MODIFY and let α_i be the number of type I pairs that remain marked at the end of the ith iteration of algorithm MODIFY (using the "imaginary" computations described inside the comments).

LEMMA 5.2. *The vertical height of assignment R_i minus the vertical height of assignment L is at most $w+2\alpha+\alpha_i$.*

Proof. We prove this lemma by induction on i. By definition, the statement of the lemma is true for $i=0$. Assume it is true for $i-1 \geqq 0$ and let us now show it is true for i. There are three cases depending on the number of nets unmarked during the ith iteration of the algorithm.

Case 1. No nets are unmarked during the ith iteration.

Since at each iteration of the algorithm we decrease the vertical height by at least one and since no terminal is unmarked at this step,

$$\begin{aligned}(H_{R_i}(S_1, S_2)+H_{R_i}(S_3, S_0))-(H_L(S_1, S_2)+H_L(S_3, S_0))&\\ <(H_{R_{i-1}}(S_1, S_2)+H_{R_{i-1}}(S_3, S_0))-(H_L(S_1, S_2)+H_L(S_3, S_0))&\\ \leqq w+2\alpha+\alpha_{i-1}&\\ =w+2\alpha+\alpha_i&\end{aligned}$$

and the induction hypothesis holds for i.

Case 2. A type I pair is unmarked during the ith iteration.

Since at each iteration of the algorithm we decrease the vertical height by at least one and since $\alpha_i=\alpha_{i-1}-1$,

$$\begin{aligned}(H_{R_i}(S_1, S_2)+H_{R_i}(S_3, S_0))-(H_L(S_1, S_2)+H_L(S_3, S_0))&\\ \leqq(H_{R_{i-1}}(S_1, S_2)+H_{R_{i-1}}(S_3, S_0))-(H_L(S_1, S_2)+H_L(S_3, S_0))-1&\\ =w+2\alpha+\alpha_{i-1}-1&\\ =w+2\alpha+\alpha_i,&\end{aligned}$$

and the induction hypothesis holds for i.

Case 3. Two type I pairs are unmarked during the ith iteration.

Each of the nets in the two pairs contributed two to the 2α term in $(H_{R_{i-1}}(S_1, S_2)+H_{R_{i-1}}(S_3, S_0))-(H_L(S_1, S_2)+H_L(S_3, S_0))$ and one to the α_{i-1} term in the same expression. Now, we know that two of these nets will be routed as in assignment L, so the total increase of the vertical height by each of the two remaining nets is at most two. Hence the total contribution of the interchange is at most four.

$$\begin{aligned}(H_{R_i}(S_1, S_2)+H_{R_i}(S_3, S_0))-(H_L(S_1, S_2)+H_L(S_3, S_0))&\\ \leqq\text{contribution of the nets not involved in this interchange}&\\ \quad\text{(which is the same as in the previous iteration)}+&\\ \quad\text{contribution of the two pairs involved in the interchange}&\\ \leqq(w+2(\alpha-2)+(\alpha_{i-1}-2))+4&\\ =w+2\alpha+\alpha_{i-1}-2.&\end{aligned}$$

Since $\alpha_i=\alpha_{i-1}-2$, the induction hypothesis holds for i and the proof of the lemma follows by induction. □

DEFINITION 5.5. When algorithm MODIFY terminates, let x be the leftmost point located on the top side of G whose height in assignment R is maximum.

LEMMA 5.3. *The vertical height of assignment R minus the vertical height of assignment L is at most* $w+2\alpha$.

Proof. It is simple to see that

$$(H_R(S_1, S_2)+H_R(S_3, S_0))-(H_L(S_1, S_2)+H_L(S_3, S_0))\leqq w+2\alpha+\alpha'$$

where α' is the value of α_i at the end of the last iteration of the algorithm. If α' is zero, there is nothing to prove. So let us assume it is not the case. Let k be the number of type I pairs that do not include point x. Each of these k pairs will only contribute

two to

$$(H_R(x, x)+H_R(S_3, S_0))-(H_L(S_1, S_2)+H_L(S_3, S_0))$$
$$=(H_R(S_1, S_2)+H_R(S_3, S_0))-(H_L(S_1, S_2)+H_L(S_3, S_0)).$$

Hence, if $k=\alpha'$, there is nothing to prove. So, assume that $k<\alpha'$ and let $\alpha''=\alpha'-k$. That is α'' is the number of type I pairs that include point x. Hence,

$$(H_R(S_1, S_2)+H_R(S_3, S_0))-(H_L(S_1, S_2)+H_L(S_3, S_0))\leqq w+2\alpha+\alpha''.$$

Let y_l (y_r) be the leftmost (rightmost) point on the bottom side of G whose height in assignment R is maximum. Since the algorithm terminates when there remains no conflicting pairs, there can be no type I pair that includes x, y_l and partially includes (or includes) y_r; or includes x, y_r and partially includes (or includes) y_l. Let z_l (z_r) be the number of type I pairs that include x and y_l (y_r). There are two cases depending on the values of z_l and z_r.

Case (i): $z_l\leqq z_r$. It is simple to verify that each type I pair is counted (included) in either z_l, z_r or in $\alpha-z_r-z_l$. Every type I pair counted in z_l contributes to

$$(H_R(x, x)+H_R(y_l, y_l))-(H_L(S_1, S_2)+H_L(S_3, S_0))$$

exactly three, each of the type I pairs counted in z_r contributes one, and each of the remaining type I pairs (those not counted in z_r and z_l) that include x contributes at most two. Hence,

$$(H_R(x, x)+H_R(y_l, y_l))-(H_L(S_1, S_2)+H_L(S_3, S_0))\leqq w+2(\alpha-z_l-z_r)+3*z_l+z_r.$$

Since $z_l\leqq z_r$, $z_l+z_r\leqq\alpha''$ and

$$(H_R(x, x)+H_R(y_l, y_l))-(H_L(S_1, S_2)+H_L(S_3, S_0))$$
$$=(H_R(S_1, S_2)+H_R(S_3, S_0))-(H_L(S_1, S_2)+H_L(S_3, S_0)).$$

Hence, $(H_R(S_1, S_2)+H_R(S_3, S_0))-(H_L(S_1, S_2)+H_L(S_3, S_0))\leqq w+2\alpha$. This completes the proof for case (i).

Case (ii): $z_l>z_r$. The proof for this case is similar to case (i), so it will be omitted. This completes the proof of the lemma. □

An algorithm similar to MODIFY has to be applied to the $D_{i,j}$ in order to be able to claim the bound $0.6*m_2^{LR}$. Let us refer to this new procedure MODIFY-LR. Our algorithm works as follows:

Obtain assignments $D_{i,j}$;
Use MODIFY and MODIFY-LR to transform each $D_{i,j}$ into a new assignment, $M_{i,j}$;
Output the assignment $M_{i,j}$ of least area;

We have also been assuming that in all optimal assignments the nets in $M_2^{TB}\cup M_2^{LR}$ are connected by paths that cross exactly two corners of G. In general this is not true. Let D be an optimal assignment and let $m_2^{TB'}$ ($m_2^{LR'}$) be the number of nets in M_2^{TB} (M_2^{LR}) that are connected in D by paths that cross two corners of G. Let $m_2^{TB''}=m_2^{TB}-m_2^{TB'}$ and $m_2^{LR''}=m_2^{LR}-m_2^{LR'}$. When transforming an optimal assignment D to one of the assignments generated by our algorithm, we will first modify the assignment to one in which all nets in $M_2^{TB}\cup M_2^{LR}$ are connected by paths that cross exactly two of the corners of G. This new assignment is obtained by connecting all nets in M_2^{TB} (M_2^{LR}) that are connected by paths that cross the four corners of G, by paths that do not cross the right (top) side of G. The new assignment satisfies the restrictions imposed

before. The cost of this modification must be accounted for (see Lemma 5.4). Note that each time an interchange is performed we will increase by at most one the height of one side of G, but we will decrease the height of another side (adjacent to the previous one) of G by one. From the above discussion and lemmas, we obtain the following lemma which we state without a proof.

LEMMA 5.4. *Let D be an optimal assignment such that $D' \subseteq D$. There is an assignment $M_{i,j}$ (constructed by our algorithm) such that if M is defined as D except for all the nets in $M_2^{TB} \cup M_2^{LR}$ which are routed as in the assignment $M_{i,j}$, then*

$$\text{(a)}\quad H_M(S_1, S_2) + H_M(S_3, S_0) \leqq H_D(S_1, S_2) + H_D(S_3, S_0) + (0.6) * m_2^{TB} + m_2^{TB''} - m_2^{LR''},$$

$$\text{(b)}\quad H_M(S_0, S_1) + H_M(S_2, S_3) \leqq H_D(S_0, S_1) + H_D(S_2, S_3) + (0.6) * m_2^{LR} + m_2^{LR''} - m_2^{TB''}.$$

Before proving our main result in this section we establish a lower bound on the area required by any optimal solution. The lower bound is given in Table 5.1.

LEMMA 5.5. *Let D be an optimal assignment such that $D' \subseteq D$. Assignment D and rectangle G satisfy the lower bounds given in Table* 5.1.

Proof. We only prove the lower bounds in the first column of Table 5.1 since the proof for the bounds in the second column is similar. Clearly, each net in M_2^{LR} has at least one terminal on the left and right side of G.

Using the above observation, we obtain the bounds given by the first column of Table 5.2.

Let us now establish a lower bound on the number of paths crossing the corners of G. Since every path for a net in $M_2^{TB'} \cup M_2^{LR'}$ crosses two corners of G, and every path for a net in $M_2^{TB''} \cup M_2^{LR''}$ crosses four corners of G, we know that

$$\sum_{i=0}^{3} H_D(S_i, S_i) \geqq 2(m_2^{TB'} + m_2^{LR'}) + 4(m_2^{TB''} + m_2^{LR''}).$$

It is simple to show that

$$H_D(S_1, S_2) + H_D(S_3, S_0) \geqq 0.5 * (H_D(S_1, S_1) + H_D(S_2, S_2)) + 0.5 * (H_D(S_3, S_3) + H_D(S_0, S_0)).$$

TABLE 5.1
Lower bounds for the R1M-2 *problem.*

Set	Contribution to our lower bound for $h/(\lambda) + H_D(S_1, S_2) + H_D(S_3, S_0)$	Contribution to our lower bound for $w/(\lambda) + H_D(S_0, S_1) + H_D(S_2, S_3)$
M_2^{TB}	$m_2^{TB'} + 2m_2^{TB''}$	$2m_2^{TB'} + 3m_2^{TB''}$
M_2^{LR}	$2m_2^{LR'} + 3m_2^{LR''}$	$m_2^{LR'} + 2m_2^{LR''}$

TABLE 5.2

Set	Contribution to our lower bound for $h/(\lambda)$	Contribution to our lower bound for $H_D(S_1, S_2) + H_D(S_3, S_0)$
M_2^{TB}	—	$m_2^{TB'} + 2m_2^{TB''}$
M_2^{LR}	$m_2^{LR'} + m_2^{LR''}$	$m_2^{LR'} + 2m_2^{LR''}$

The lower bound for the second column of Table 5.2 follows from the above inequalities. The proof for the lower bounds for the first column of Table 5.1 can be obtained by adding the bounds given by Table 5.2. This completes the proof of the lemma. □

THEOREM 5.1. *Let D be an optimal assignment such that $D' \subseteq D$ and let P be the assignment produced by our algorithm. Then, $A(P) \leqq 1.6 * A(D)$.*

Proof. Let M be any of the assignments constructed by the algorithm such that $M = M_{i,j}$ and $M_{i,j}$ is an assignment that satisfies Lemma 5.4. We now prove that $A(M) \leqq 1.6 * A(D)$. Since $A(P) \leqq A(M)$ our bound holds. Using Lemmas 5.4 and 5.5 we know that:

$$A(M)/A(D) \leqq (1 + a_1/b_1) * (1 + c_1/d_1),^3$$

where

$$a_1 = 0.6m_2^{TB'} + 1.6m_2^{TB''} - m_2^{LR''},$$
$$b_1 = m_2^{TB'} + 2m_2^{TB''} + 2m_2^{LR'} + 3m_2^{LR''},$$
$$c_1 = -m_2^{TB''} + 0.6m_2^{LR'} + 1.6m_2^{LR''},$$
$$d_1 = 2m_2^{TB'} + 3m_2^{TB''} + m_2^{LR'} + 2m_2^{LR''}.$$

The proof of the theorem can be obtained by multiplying all terms and obtaining an expression of the form $(ax + by + \cdots)/(a'x + b'y + \cdots)$, where $a, b, \cdots, a', b', \cdots$ are constants and $x, y, \cdots$ are products of variables. The final step consists of showing that $a/a' \leqq 1.6$, $b/b' \leqq 1.6, \cdots$.

This completes the proof of the theorem. □

6. Approximation algorithm for the R1M problem. In this section we show that our algorithm takes $O(n(m + \log n))$ time and generates a solution with area $\leqq 1.6 * \text{OPT}$, where OPT is the area of an optimal solution.

ALGORITHM FOR THE R1M PROBLEM

Construct assignments D', D_3 and D_4;
$D_{i,j} \leftarrow D' \cup D_3 \cup D_4 \cup D_i^{TB} \cup D_j^{LR}$ for $0 \leqq i, j \leqq 63$;
Apply algorithm MODIFY and MODIFY-LR to each $D_{i,j}$ to obtain $M_{i,j}$;
Let P be one of the $M_{i,j}$'s of least area;
Construct and output a layout with area $A(P)$ for P;
end of algorithm

THEOREM 6.1. *The time complexity of our algorithm is $O(n(m + \log n))$.*

Proof. It is simple to verify that all the steps in our algorithm take $O(n)$ time except for the initial sorting of the terminals (if not initially sorted), the construction of assignment D''_A, D''_X ($X \in \{T', R', B', L'\}$), and procedure MODIFY and MODIFY-LR. Clearly, sorting can be performed in $O(n \log n)$ time. Assignment D''_A (as well as assignments D''_X, $X \in \{T', R', B', L'\}$) can be obtained in $O(n \log n)$ time by a simple recursive procedure that manipulates four priority queues and uses a simple marking scheme. The priority queues can be implemented by heap trees [AHU] and the marking scheme can be implemented using a one-dimensional array. A slightly complex marking scheme can be used to reduce the time complexity of this part of the algorithm to $O(n)$. For brevity such a scheme will be omitted. The total number of iterations performed by procedure MODIFY and MODIFY-LR is at most $O(m)$ since at each

[3] See footnote in Theorem 4.1.

iteration the height of assignment R on the bottom side of G is decreased by one and the height can be at most $O(m)$. The statements inside the "while loop" of both procedures can be easily implemented to take $O(n)$ time. Hence, the total time complexity for both these procedures is $O(n(m+\log n))$. □

Before proving our main result we establish a lower bound on the area required by any optimal solution. The lower bound is given in Table 6.1.

TABLE 6.1
Lower bounds for the R1M *problem.*

Set	Contribution to our lower bound for $h/(\lambda)+H_D(S_1,S_2)+H_D(S_3,S_0)$	Contribution to our lower bound for $w/(\lambda)+H_D(S_0,S_1)+H_D(S_2,S_3)$
M_2^{TB}	$m_2^{TB'}+2m_2^{TB''}$	$2m_2^{TB'}+3m_2^{TB''}$
M_2^{LR}	$2m_2^{LR'}+3m_2^{LR''}$	$m_2^{LR'}+2m_2^{LR''}$
M_3^{TB}	$1.5m_3^{TB}+0.5x_3^{TB}$	$2m_3^{TB}+0.5x_3^{TB}$
M_3^{LR}	$2m_3^{LR}+0.5x_3^{LR}$	$1.5m_3^{LR}+0.5x_3^{LR}$
M_4^{A}	$2.5m_4^{A}$	$2.5m_4^{A}$
M_4^{N}	$14m_4^{N}$	$14m_4^{N}$
$M_4^{T'}\cup M_4^{B'}$	$2.5m_4^{T'}+2.5m_4^{B'}$	$3m_4^{T'}+3m_4^{B'}$
$M_4^{R'}\cup M_4^{L'}$	$3m_4^{R'}+3m_4^{L'}$	$2.5m_4^{R'}+2.5m_4^{L'}$
$M_4^{T}\cup M_4^{B}$	$p_4^{T}*m_4^{T}+5m_4^{T}+6m_4^{B}$	$q_4^{T}*m_4^{T}+6m_4^{T}+6m_4^{B}$
M_4^{R}	$p_4^{R}*m_4^{R}+6m_4^{R}$	$q_4^{R}*m_4^{R}+5m_4^{R}$

LEMMA 6.1. *Let D be an optimal assignment such that $D'\subseteq D$. Then assignment D and rectangle G satisfy the lower bounds given in Table* 6.1.

Proof. The proof for this bound follows from Lemmas 3.2, 4.5 and 5.5. □

THEOREM 6.2. *Let D be an optimal assignment such that $D'\subseteq D$ and let P be the assignment generated by our algorithm. Then, $A(P)\leqq 1.6*A(D)$.*

Proof. Let M be any of the assignments constructed by the algorithm such that $M=M_{i,j}$ and $M_{i,j}$ is an assignment that satisfies Lemma 5.4. We now prove that $A(M)\leqq 1.6*A(D)$. Since $A(P)\leqq A(M)$ our bound holds. Using arguments similar to those used in the proofs of Lemmas 3.1, 4.1, 4.2, 4.3, 4.4, and 5.4, and the lower bound given in Lemma 6.1, we know that:

$$\frac{A(M)}{A(D)}\leqq\left(1+\sum_{i=1}^{8}a_i\Big/\sum_{i=1}^{8}b_i\right)*\left(1+\sum_{i=1}^{8}c_i\Big/\sum_{i=1}^{8}d_i\right),^{4}$$

where a_i, b_i, c_i and d_i are defined in Table 6.2 and x, y are positive integers such that $x+y=1$.

Substituting $m_3^{TB}\geqq x_3^{TB}$, $m_3^{LR}\geqq x_3^{LR}$ and $a\geqq m_4^{A}/4$ (remember that $m_4^{A}\geqq 4$) and multiplying all terms, we obtain the expression of the form $(ax+by+\cdots)/(a'x+b'y+\cdots)$, where $a, b, \cdots, a', b', \cdots$ are constants and $x, y, \cdots$ are products of variables. The final step consists of showing that $a/a'\leqq 1.6$, $b/b'\leqq 1.6, \cdots$. This completes the proof of the theorem. □

Our algorithm generates 2^{12} assignments and it outputs one that requires the least layout area. An algorithm that only generates 2^6 assignments can be easily obtained by only taking the best of the modified $D'\cup D_3\cup D_4\cup D_i^{TB}$ together with the best of the modified $D'\cup D_3\cup D_4\cup D_i^{LR}$. Note that a transformation applied to the first assignment does not modify the horizontal height on the second assignment, and a

[4] The footnote in Theorem 4.1 applies for $i=1, 8$.

TABLE 6.2(a)

i	a_i	b_i
1	$0.6m_2^{TB'}+1.6m_2^{TB''}-m_2^{LR''}$	$m_2^{TB'}+2m_2^{TB''}+2m_2^{LR'}+3m_2^{LR''}$
2	x_3^{TB}	$1.5m_3^{TB}+0.5x_3^{TB}+2m_3^{LR}+0.5x_3^{LR}$
3	$0.5m_4^A+1$	$2.5m_4^A$
4	$7x * m_4^N$	$14m_4^N$
5	$0.75m_4^{T'}+0.75m_4^{B'}$	$2.5m_4^{T'}+2.5m_4^{B'}$
6	$0.5m_4^{R'}+0.5m_4^{L'}$	$3m_4^{R'}+3m_4^{L'}$
7	$q_4^T * m_4^T+m_4^T+m_4^B$	$p_4^T * m_4^T+5m_4^T+6m_4^B$
8	$m_4^R+q_4^R * m_4^R$	$p_4^R * m_4^R+6m_4^R$

TABLE 6.2(b)

i	c_i	d_i
1	$-m_2^{TB''}+0.6m_2^{LR'}+1.6m_2^{LR''}$	$2m_2^{TB'}+3m_2^{TB''}+m_2^{LR'}+2m_2^{LR''}$
2	x_3^{LR}	$2m_3^{TB}+0.5x_3^{TB}+1.5m_3^{LR}+0.5x_3^{LR}$
3	$0.5m_4^A$	$2.5m_4^A$
4	$7y * m_4^N$	$14m_4^N$
5	$0.5m_4^{T'}+0.5m_4^{B'}$	$3m_4^{T'}+3m_4^{B'}$
6	$0.75m_4^{R'}+0.75m_4^{L'}$	$2.5m_4^{R'}+2.5m_4^{L'}$
7	$p_4^T * m_4^T+m_4^T+2m_4^B$	$q_4^T * m_4^T+6m_4^T+6m_4^B$
8	$p_4^R * m_4^R+m_4^R$	$q_4^R * m_4^R+5m_4^R$

transformation applied to the second assignment does not modify the vertical height on the first assignment. For brevity we will not prove that this solution also satisfies our approximation bounds. In §§ 4 and 5 we assumed that the number of nets in some sets was a multiple of some fixed constant. More specifically, we might have to delete at most five nets from sets M_2^{TB} and M_2^{LR}; at most three nets from sets M_4^A, $M_4^{T'}$, $M_4^{R'}$, $M_4^{B'}$ and $M_4^{L'}$; and procedure PARTITION deletes at most eight nets (those never added to M_4^N, M_4^T, M_4^R and M_4^B). All of these nets with a "small" number of terminals can be routed optimally by trying all possible routing paths and then selecting the best of the solutions generated. If some of these nets do not have a "small" number of terminals then select any routing paths for them. Note that it will not make too much difference which routing path is selected since their contribution to the lower bound in an optimal solution is large (contribution from the number of terminals). There are better ways of dealing with the remaining 33 nets; however, for brevity these other methods will not be discussed in this paper.

7. Discussion We have shown that there is an efficient approximation algorithm that generates a solution within 60% of optimal for the R1M problem. The algorithm takes $O(n(m+\log n))$ time and the constant associated with this bound is small. Our algorithm generates 2^6 assignments (see comment at the end of § 6) and it outputs one with least layout area. We conjecture that the solutions generated by our algorithm are usually closer to optimal than the bound of 60%. The reason for this is that our bounds are rough and are based on analyzing groups of nets separately. We took this approach in order to simplify the analysis, which in spite of this became very complex. One of the interesting aspects of our algorithms is that the proof of Lemma 5.1 can be obtained by solving a set of Linear Programming problems. Before we proved this lemma we programmed a personal computer (IBM with INTEL 8088 and 8087 processors) to prove the lemma for us. The computer solved several hundred linear

programming problems using the simplex method. In each of these problems there were around eighty equations and about one hundred variables. The simplex method that we programmed took about 30 iterations to generate the optimal solution to each of the LP problems. The total execution time was about 50 CPU hours.

A better solution (not a better approximation bound) can be obtained by selecting more groups for the nets in set M_2^{TB} and M_2^{LR}, as well as some postprocessing improvements similar to those performed by procedure MODIFY. For brevity we will not discuss such extensions. If less than six groups are selected for M_2^{TB} or M_2^{LR}, one cannot prove the approximation bound 1.6. When three groups are selected (as in [GL3]), the approximation bound is exactly 1.69.

Appendix A.

LEMMA A.1. *Let D be an optimal assignment such that $D' \subseteq D$. There is an assignment $D_{i,j}$ (constructed by our algorithm) such that if M is defined as D except for all the nets in $M_2^{TB} \cup M_2^{LR}$ which are routed as in the assignment $D_{i,j}$, then*

(a) $H_M(S_1, S_2) + H_M(S_3, S_0) \leqq H_D(S_1, S_2) + H_D(S_3, S_0) + (0.6) * m_2^{TB}$,

(b) $H_M(S_0, S_1) + H_M(S_2, S_3) \leqq H_D(S_0, S_1) + H_D(S_2, S_3) + (0.6) * m_2^{LR}$.

It is assumed that when D is transformed to any of the assignments D_{ij} and two paths that cross on the top side of G are interchanged, such an interchange is not a type I interchange.

Proof. Since the proof for part (b) is similar to the proof of part (a), we only prove part (a). Before proving the lemma we make the following observations (note that we are assuming that there are no type I interchanges):

(i) The interchange of a path connecting a net in M_2^{TB} in assignment D to one that crosses the least number of corners in G will increase by at most two the vertical height of D.

(ii) The interchange of two paths that cross on the top side of G in assignment D will increase by at most two the vertical height of D.

Let G_i, for $1 \leqq i \leqq 6$, be the partition of the nets generated by our algorithm (these sets were called N_i^{TB}, for $1 \leqq i \leqq 6$, in our algorithm). For assignment D, let l_i be the number of nets in G_i that are connected by a path that crosses the left side of G and let r_i be the number of nets in G_i that are connected by a path that crosses the right side of G.

In what follows we apply several i-strings to assignment D. When an i-string contains three "*" symbols and three "." symbols, for example ".*..**", eight assignments are obtained. In Appendix B we prove that at least one of these assignments, M, has the property that

$$H_M(S_1, S_2) + H_M(S_3, S_0) \leqq H_D(S_1, S_2) + H_D(S_3, S_0) + (1/3) * m_2^{TB}.$$

When the i-string contains five "*" symbols and a "." symbol, for example "**.***", 32 assignments are generated. In Appendix B we show that at least one of these assignments, M, has the property that

$$H_M(S_1, S_2) + H_M(S_3, S_0) \leqq H_D(S_1, S_2) + H_D(S_3, S_0) + (8/15) * m_2^{TB}.$$

In what follows we will use i-strings that contain three "*" symbols and three letters, for example "*r*rl**." Using the lemma in Appendix B we know that the i-string ".*..**" when applied to D produces an assignment with some given property. If such assignment is modified by the i-string "r.rl.." we obtain an assignment whose vertical height differs from the one in D by at most $2 \max\{r_1 + r_3, l_4\} + (1/3) * m_2^{TB}$. A similar interpretation is given to i-strings with five "*" symbols and one "|" symbol. In this

case the bound on the increase of the vertical height is given by the lemma in Appendix C.

We prove part (a) of the lemma by contradiction. Suppose that all assignments M obtained from D by applying the i-string "******" do not satisfy the lemma. There are three cases that need to be considered.

Case 1: $\max\{l_i, r_i\} \geqq (2/15) * m_2^{TB}$, for some $1 \leqq i \leqq 6$. Let j be such that $\max\{l_j, r_j\} \geqq (2/15) * m_2^{TB}$. Since $r_j + l_j = m_2^{TB}/6$, we know that $\min\{l_j, r_j\} \leqq m_2^{TB}/30$. Applying the i-string that consists of all "*" symbols except for a "|" symbol in the jth position to D generates an assignment whose vertical height differs from the one in D by at most

$$2 \min\{l_j, r_j\} + (8/15) * m_2^{TB},$$

which we know must be greater than $(0.6) * m_2^{TB}$. Substituting the upper bound for "min" obtained above we know that $(9/15) > 0.6$, a contradiction. Hence, the conditions of Case 1 do not hold.

Case 2.

(A.2.0) $$\max\{l_i, r_i\} < (2/15) * m_2^{TB} \text{ for } 1 \leqq i \leqq 6, \text{ and}$$

(A.2.1) $$r_2 + r_4 \geqq l_3 + l_5.$$

First of all we prove a pair of inequalities (A.2.2 and A.2.3) hold. These inequalities will be useful later on. Applying the i-string "*rrl****" to D generates an assignment whose vertical height differs from the one in D by at most $2 \max\{r_1 + r_2, l_3\} + m_2^{TB}/3$, which we know must be greater than $(0.6) * m_2^{TB}$. This is equivalent to

$$\max\{r_1 + r_2, l_3\} > (2/15) * m_2^{TB}.$$

If $l_3 \geqq r_1 + r_2$ then from the above inequality we know that $l_3 > (2/15) * m_2^{TB}$. This contradicts (A.2.0). So it must be that

(A.2.2) $$r_1 + r_2 > l_3.$$

Applying the i-string "***rll*" to D and using similar arguments, we know that

(A.2.3) $$l_5 + l_6 > r_4.$$

Hence, it must be that inequalities (A.2.2) and (A.2.3) hold as otherwise there is a contradiction.

We now show that for this case there is a contradiction. There are two subcases.

Subcase 2.1.

(A.2.1.1) $$l_4 + l_5 + l_6 \geqq r_1 + r_2 + r_3.$$

CLAIM. *If one of the following inequalities does not hold then there is a contradiction*:

(A.2.1.2) $$l_3 + l_4 + l_6 \geqq r_1 + r_2 + r_5,$$

(A.2.1.3) $$l_2 + l_4 + l_6 \geqq r_1 + r_3 + r_5,$$

(A.2.1.4) $$r_5 < l_6,$$

(A.2.1.5) $$r_4 < l_5,$$

(A.2.1.6) $$r_3 < l_4,$$

(A.2.1.7) $$r_2 > l_3.$$

Proof of Claim. In what follows we show that (A.2.1.2)-(A.2.1.7) hold as otherwise there is a contradiction.

Proof of (A.2.1.2)-(A.2.1.3). If any of these inequalities does not hold then replace in it r_j by $m_2^{TB}/6 - l_j$ and l_k by $m_2^{TB}/6 - r_k$ for appropriate values of k and j, and we obtain a contradiction to inequality (A.2.1.1).

Proof for (A.2.1.4). Since inequality (A.2.1.1) holds we know that applying the i-string "*rrrlll*" to D produces an assignment with a vertical height that differs from the one in D by at most

$$2(l_4 + l_5 + l_6),$$

which we know must be greater than $(0.6) * m_2^{TB}$. If $r_5 \geqq l_6$ then replacing l_6 by r_5 and replacing $r_5 + l_5$ by $m_2^{TB}/6$ in the above inequality we know that $l_4 > (2/15) * m_2^{TB}$. This contradicts inequality (A.2.0). Hence, it must be that $r_5 < l_6$. This completes the proof for (A.2.1.4).

Proof for (A.2.1.5). Since the proof for this part is similar to one for the previous one, it will be omitted.

Proof for (A.2.1.6). Same as the proof for (A.2.1.4) but applying the i-string "*rrllrl*" and using inequalities (A.2.1.4) and (A.2.1.2).

Proof for (A.2.1.7). The proof follows from (A.2.1.5) and (A.2.1).

End of Claim. □

We now show that when inequalities (A.2.1.2)-(A.2.1.7) and (A.2.0) hold then there is a contradiction. Applying the i-string "*rlrlrl*" to D and using inequalities (A.2.1.4), (A.2.1.6) and (A.2.1.3), we know that

$$l_2 + l_4 + l_6 > (0.3) * m_2^{TB}.$$

Similarly, applying the i-string "$|$*rlrlr*" and using inequalities (A.2.1.7) and (A.2.1), we know that

$$\min\{r_1, l_1\} + r_2 + r_4 + r_6 > (0.3) * m_2^{TB}.$$

Adding these two inequalities and replcing $r_j + l_j$ by $(1/6) * m_2^{TB}$, we know that

$$\min\{r_1, l_1\} > (0.1) * m_2^{TB},$$

a contradiction (since $r_1 + l_1 = (1/6) * m_2^{TB}$). This completes the proof for Subcase 2.1.

Subcase 2.2.

(A.2.2.1) $$l_4 + l_5 + l_6 \leqq r_1 + r_2 + r_3.$$

CLAIM. *If one of the following inequalities does not hold then there is a contradiction.*

(A.2.2.2) $$r_1 + r_2 + r_4 \geqq l_3 + l_5 + l_6,$$

(A.2.2.3) $$r_1 + r_3 + r_4 \geqq l_2 + l_5 + l_6,$$

(A.2.2.4) $$r_1 + r_3 + r_5 \geqq l_2 + l_4 + l_6,$$

(A.2.2.5) $$r_1 + r_2 + r_5 \geqq l_3 + l_4 + l_6,$$

(A.2.2.6) $$r_1 > l_2,$$

(A.2.2.7) $$r_2 > l_4,$$

(A.2.2.8) $$r_1 > l_3,$$

(A.2.2.9) $$r_2 > l_3,$$

(A.2.2.10) $$r_3 > l_4,$$

(A.2.2.11) $$l_3 + l_5 + l_6 \geqq r_2 + r_4,$$

(A.2.2.12) $$l_1 + l_5 < (2/15) * m_2^{TB}.$$

Proof of Claim. In what follows we show that (A.2.2.2)-(A.2.2.12) hold as otherwise there is a contradiction.

Proof of (A.2.2.2)-(A.2.2.5). If any of these inequalities does not hold then replace in it r_j by $m_2^{TB}/6 - l_j$ and l_k by $m_2^{TB}/6 - r_k$ for appropriate values of j and k, and we obtain a contradiction to inequality (A.2.2.1).

Proof of (A.2.2.6). The proof for this part uses arguments similar to the ones in the proof of (A.2.1.4) in Subcase 2.1. The i-string applied in this case is "*rrlrll*" and inequalities (A.2.2) and (A.2.2.2) are used to establish the bound on the difference in vertical heights.

Proof of (A.2.2.7). Similar to the proof for (A.2.2.6).

Proof of (A.2.2.8). Same as above but applying the i-string "*rrrlll*" and using inequality (A.2.2.1).

Proof of (A.2.2.9). Similar to the proof for (A.2.2.8).

Proof of (A.2.2.10). Same as above but applying the i-string "*rlrrll*" and using inequalities (A.2.2.6) and (A.2.2.3).

Proof of (A.2.2.11). If $l_3 + l_5 + l_6 < r_2 + r_4$, then since (A.2.2.9) holds we know that applying the i-string "|*rlrll*" to D produces an assignment with a vertical height that differs from the one in D by at most

$$2 \min \{l_1, r_1\} + 2r_2 + 2r_4,$$

which we know must be greater than $(0.6) * m_2^{TB}$. Since $\min \{r_1, l_1\} \leqq m_2^{TB}/12$ and $r_j + l_j = m_2^{TB}/6$ for all j, the above inequality becomes

$$l_2 + l_4 < (7/60) * m_2^{TB}.$$

Also, applying the i-string "*rl*l***" to D generates an assignment whose vertical height differs from the one in D by at most

$$2 \max \{r_1, l_2 + l_4\} + m_2^{TB}/3,$$

which must be greater than $(0.6) * m_2^{TB}$. Substituting $l_2 + l_4 < (7/60) * m_2^{TB}$ and inequality (A.2.0) in the above expression results in the inequality $0.6 > 0.6$, a contradiction. So it must be that

$$l_3 + l_5 + l_6 \geqq r_2 + r_4.$$

Proof of (A.2.2.12). Applying the i-string "*rrllrl*" and using inequalities (A.2.2.8), (A.2.2.7) and (A.2.2.5), we know that

(eq.A.1) $$2(r_1 + r_2 + r_5) > (0.6) * m_2^{TB}.$$

Similarly, using "*rlrlrl*," (A.2.2.6), (A.2.2.10) and (A.2.2.4), we know that

(eq.A.2) $$2(r_1 + r_3 + r_5) > (0.6) * m_2^{TB}.$$

Substituting r_i by $1/6 - l_i$ in the inequality obtained by adding (eq.A.1), (eq.A.2) and four times $(l_1 + l_5 \geqq (2/15) * m_2^{TB})$, we know that

$$l_2 + l_3 < 2/15 * m_2^{TB}.$$

Now, applying the i-string "*rll****" to D generates an assignment whose vertical height differs from the one in D by at most

$$2 \max \{r_1, l_2 + l_3\} + m_2^{TB}/3,$$

which must be greater than $(0.6) * m_2^{TB}$. Since $\max\{r_1, l_2+l_3\} \leqq (2/15) * m_2^{TB}$, we know that $0.6 > 0.6$. A contradiction. So it must be that

$$l_1 + l_5 < (2/15) * m_2^{TB}.$$

End of Claim. □

We will now show that at least one of the above inequalities does not hold. There are two subcases:

Subcase A.2.2.1: $r_1 + r_3 \geqq l_2 + l_4 + l_5$. Applying the *i*-string "*rlrllr*" and using inequalities (A.2.2.6) and $r_1 + r_3 \geqq l_2 + l_4 + l_5$, we know that

$$2(r_1 + r_3 + r_6) > (0.6) * m_2^{TB}.$$

Similarly, using "*lrlrll*", (A.2.3) and (A.2.2.11), we know that

$$2l_1 + 2(l_3 + l_5 + l_6) > (0.6) * m_2^{TB}.$$

Adding these last two inequalities, we know that

$$l_5 > m_2^{TB}/10.$$

Since r_1 is less than $(2/15) * m_2^{TB}$, it must be that

$$l_1 > m_2^{TB}/30.$$

Adding these two inequalities, we know that $l_1 + l_5 > (2/15) * m_2^{TB}$, which contradicts (A.2.2.12).

Subcase A.2.2.2: $r_1 + r_3 < l_2 + l_4 + l_5$. Applying the *i*-string "*rlrllr*", and using inequalities (A.2.2.6) and $r_1 + r_3 < l_2 + l_4 + l_5$, we know that

(eq.A.3) $$2(l_2 + l_4 + l_5) + 2r_6 > (0.6) * m_2^{TB}.$$

Similarly, using "*lrlrll*," (A.2.3) and (A.2.2.11), we know that

(eq.A.4) $$2l_1 + 2(l_3 + l_5 + l_6) > (0.6) * m_2^{TB}.$$

Using "*lrlrlr*," (A.2.2.9) and (A.2.1), we know that

(eq.A.5) $$2l_1 + 2(r_2 + r_4 + r_6) > (0.6) * m_2^{TB}.$$

Replacing $l_i + r_i$ by $m_2^{TB}/6$ in $2 * (\text{eq.A.4}) + 2 * (\text{eq.A.2}) + (\text{eq.A.3}) + (\text{eq.A.5})$, we know that

$$l_1 + l_5 > (2/15) * m_2^{TB},$$

which contradicts (A.2.2.12). This completes the proof of this subcase and Case 2.

Case 3: $\max\{l_i, r_i\} < (2/15) * m_2^{TB}$, and $r_2 + r_4 \leqq l_3 + l_5$.

The proof for this case is symmetric to the one for Case 2.

This completes the proof of the lemma. □

Appendix B.

LEMMA B.1. *Let D be an optimal assignment such that $D' \subseteq D$ and let I be an i-string with three "*" symbols and three "." symbols. Applying the i-string I to D generates an assignment M with*

$$H_M(S_1, S_2) + H_M(S_3, S_0) \leqq H_D(S_1, S_2) + H_D(S_3, S_0) + (1/3) * m_2^{TB}.$$

It is assumed that when D is transformed to any of the assignments obtainable from D and I, all the interchanges of pairs of paths that cross on the top side of G are not type I interchanges.

Proof. It is simple to show that the observations (i) and (ii) given in the proof of Lemma 5.1 hold for this case.

Let G_i, for $1 \leqq i \leqq 3$, represent the ith group of nets (from left to right) for which there is a "*" symbol in I. For assignment D, let l_i be the number of nets in G_i that are connected by a path that crosses the left side of G and let r_i be the number of nets in G_i that are connected by a path that crosses the right side of G. In what follows we will use i-strings of size three. Their meaning is obvious.

We prove our lemma by contradiction. Suppose that all assignments M obtained from D by using the i-string "***" do not satisfy the lemma.

Applying the i-string "*rll*" to D generates an assignment whose vertical height differs from the one in D by at most

$$2 \max \{r_1, l_2 + l_3\},$$

which we know must be greater than $(1/3) * m_2^{TB}$. If $r_1 \geqq l_2 + l_3$ then it must be that $r_1 > (1/6) * m_2^{TB}$, a contradiction. Therefore, $l_2 + l_3 > (1/6) * m_2^{TB}$, which is equivalent to

$$l_3 > r_2. \tag{B.1}$$

Applying the i-string "*lrl*" to D generates an assignment whose vertical height differs from the one in D by at most

$$2 * l_1 + 2 * \max \{r_2, l_3\},$$

which we know must be $> (1/3) * m_2^{TB}$. Substituting inequality (B.1) in this inequality, we know that

$$l_1 + l_3 > (1/6) * m_2^{TB}. \tag{B.2}$$

Using similar arguments with the i-strings "*rrl*" and "*rlr*," we know that

$$r_1 + r_3 > (1/6) * m_2^{TB}.$$

Adding this inequality to inequality (B.2) gives that the number of elements in G_1 and G_3 is greater than $(1/3) * m_2^{TB}$. This is a contradiction. Hence, at least one of the assignments obtained from the above i-strings satisfies the lemma. This completes the proof of the lemma. □

Appendix C.

LEMMA C.1. *Let D be an optimal assignment such that $D' \subseteq D$ and let I be an i-string with five "*" symbols and one "." symbol. Applying the i-string I to D generates an assignment such that*

$$H_M(S_1, S_2) + H_M(S_3, S_0) \leqq H_D(S_1, S_2) + H_D(S_3, S_0) + (8/15) * m_2^{TB}.$$

It is assumed that when D is transformed to any of the assignments obtainable from D and I, all the interchanges of pairs of paths that cross on the top side of G are not type I interchanges.

Proof. It is simple to show that the observations (i) and (ii) given in the proof of Lemma 5.1 hold for this case.

Let G_i, for $1 \leqq i \leqq 5$, represent the ith group of nets (from left to right) for which there is a "*" symbol in I. For assignment D, let l_i be the number of nets in G_i that are connected by a path that crosses the left side of G and let r_i be the number of nets in G_i that are connected by a path that crosses the right side of G. In what follows we will use i-strings of size five. Their meaning is obvious.

We prove our lemma by contradiction. Suppose that all assignments M obtained from D by applying the i-string "*****" do not satisfy the lemma. There are two cases:

Case 1.

$$r_1 + r_3 > l_2 + l_4. \tag{C.0}$$

CLAIM.

$$r_1 + r_2 > l_3 + l_4, \tag{C.1}$$

$$r_1 > l_2. \tag{C.2}$$

Proof of Claim. We now show that the above inequalities hold as otherwise there is a contradiction. We prove each part separately.

Proof of (C.1). The proof can be easily obtained by replacing r_j by $(1/6) * m_2^{TB} - l_j$ and l_k by $(1/6) * m_2^{TB} - r_k$, for appropriate values of j and k, in inequality (C.0).

Proof of (C.2). Applying the i-string "*rrll*|" to D and using inequality (C.1) gives that the increase of the vertical height is at most

$$2(r_1 + r_2) + 2 \min \{l_5, r_5\},$$

which we know must be greater than $(8/15) * m_2^{TB}$. Since $\min \{l_5, r_5\} \leqq (1/12) * m_2^{TB}$, we know that

$$r_1 + r_2 > (11/60) * m_2^{TB}.$$

If $r_1 \leqq l_2$ then substituting in the above inequality, we know that

$$l_2 + r_2 > (11/60) * m_2^{TB}.$$

But $l_2 + r_2 = (1/6) * m_2^{TB}$, a contradiction. Hence, r_1 must be greater than l_2.

End of Claim. □

We now show that when (C.1) and (C.2) hold there is a contradiction. There are two subcases.

Subcase 1.1.

$$r_2 + r_4 < l_3 + l_5. \tag{C.1.0}$$

CLAIM.

$$r_2 + r_3 < l_4 + l_5, \tag{C.1.1}$$

$$r_4 < l_5. \tag{C.1.2}$$

Proof of Claim. We now prove if any of the above inequalities does not hold there is a contradiction. We prove each part separately.

Proof of (C.1.1). Same as the proof for (C.1) but uses inequality (C.1.0).

Proof of (C.1.2). Same as the proof for (C.2) but uses the i-string "|*rrll*" and inequality (C.1.1).

End of Claim. □

We now show that when the above inequalities hold there is a contradiction. Applying the i-string "*rlrlr*" to D and using inequalities (C.2) and (C.0) we know that the increase in vertical height is

$$2(r_1 + r_3) + 2r_5$$

which we know must be greater than $(8/15) * m_2^{TB}$.

Using similar arguments with i-string "*lrlrl*", and inequalities (C.1.2) and (C.1.0), we know that

$$2l_1 + 2(l_3 + l_5) > (8/15) * m_2^{TB}.$$

Adding these last two inequalities and replacing $r_j + l_j$ by $1/6$, we know that

$$1 > 16/15,$$

a contradiction. This completes the proof of this subcase.

Subcase 1.2.

(C.2.0) $$r_2 + r_4 \geqq l_3 + l_5.$$

CLAIM.

(C.2.1) $$r_2 + r_3 \geqq l_4 + l_5,$$

(C.2.2) $$r_2 > l_3,$$

(C.2.3) $$r_1 + r_2 < l_3 + l_4 + l_5,$$

(C.2.4) $$r_1 + r_3 < l_2 + l_4 + l_5,$$

(C.2.5) $$r_2 + r_3 > (4/15) * m_2^{TB}.$$

Proof of Claim. We now show that if any of the above inequalities does not hold then there is a contradiction. We prove each part separately.

Proof of (C.2.1). The proof for this part is similar to the proof for (C.1). The main difference is that we use inequality (C.2.0).

Proof of (C.2.2). The proof for this case follows the same arguments as the ones in the proof for (C.2) but uses the i-string "$|rrll$" and inequality (C.2.1).

Proof of (C.2.3). If $r_1 + r_2 \geqq l_3 + l_4 + l_5$ then applying the i-string "*rrlll*" to D generates an assignment whose vertical height differs from the one in D by at most

$$2(r_1 + r_2),$$

which we know must be greater than $(8/15) * m_2^{TB}$. Replacing r_j by $(1/6) * m_2^{TB} - l_j$, we know that

$$l_1 + l_2 < (1/15) * m_2^{TB}.$$

Applying the i-string "*ll****" to D, we know that

$$2(l_1 + l_2) + (1/3) * m_2^{TB} > (8/15) * m_2^{TB}.$$

Substituting the first of these inequalities into the second, we know that

$$7/15 > 8/15,$$

a contradiction.

Proof of (C.2.4). The proof for this case is by replacing r_2 by $(1/6) * m_2^{TB} - l_2$ and l_3 by $(1/6) * m_3^{TB} - r_3$ in (C.2.3).

Proof of (C.2.5). Applying the i-string "*rlrlr*" to D and using inequalities (C.2) and (C.0), we know that

(eq.C1) $$2(r_1 + r_3) + 2r_5 > (8/15) * m_2^{TB}.$$

Applying the i-string "*lrlrl*" to D and using inequalities (C.2.2) and (C.2.0), we know that

(eq.C2) $$2l_1 + 2(r_2 + r_4) > (8/15) * m_2^{TB}.$$

Applying the i-string "*rrlll*" to D and using inequality (C.2.3), we know that

(eq.C3) $$2(l_3 + l_4 + l_5) > (8/15) * m_2^{TB}.$$

Applying the i-string "$rlrll$" to D and using inequalities (C.2) and (C.2.4), we know that

(eq.C4) $$2(l_2+l_4+l_5)>(8/15)*m_2^{TB}.$$

Replacing r_j+l_j by $(1/6)*m_2^{TB}$ in the expression

$$2(\text{eq.C1})+2(\text{eq.C2})+(\text{eq.C3})+(\text{eq.C4}),$$

we know that

$$r_2+r_3>(4/15)*m_2^{TB}.$$

End of Claim. □

We now show that if the above inequalities hold then there is a contradiction.

Applying the i-string "$*ll**$" to D generates an assignment D whose vertical height differs from the one in D by at most

$$2(l_2+l_3)+(1/3)*m_2^{TB},$$

which we know must be greater than $(8/15)*m_2^{TB}$. Replacing l_j by $(1/6)*m_2^{TB}-r_j$, we know that

$$r_2+r_3<(7/30)*m_2^{TB}.$$

But this contradicts (C.2.5). This completes the proof of this subcase and Case 1.

Case 2.

(C.3) $$r_1+r_3\leqq l_2+l_4.$$

Subcase 2.1. $r_2+r_4<l_3+l_5$.

The proof of this case is symmetric to the one for Subcase 1.2 in Case 1.

Subcase 2.2.

(C.4) $$r_2+r_4\geqq l_3+l_5.$$

CLAIM.

(C.4.1) $$r_1+r_2\leqq l_3+l_4,$$

(C.4.2) $$r_2+r_3\geqq l_4+l_5,$$

(C.4.3) $$r_3<l_4,$$

(C.4.4) $$r_2>l_3.$$

Proof of Claim. We now show that if any of the above inequalities does not hold then there is a contradiction. We prove each part separately.

Proof of (C.4.1). The proof for this part is similar to the proof for (C.1). The main difference is that we use inequality (C.3).

Proof of (C.4.2). The proof for this part is similar to the proof for (C.1). The main difference is that we use inequality (C.4).

Proof of (C.4.3). The proof for this case follows the same arguments as the ones in the proof for (C.2) but uses inequality (C.4.1).

Proof of (C.4.4). The proof for this case follows the same arguments as the ones in the proof for (C.2) but uses the i-string "$|rrll$" and inequality (C.4.2).

End of Claim. □

We now show that when the above inequalities hold there is a contradiction. Applying the i-string "$rlrl|$" to D and using inequalities (C.4.3) and (C.3) we know that the increase in vertical height is at most

$$2(l_2+l_4)+2\min\{l_5, r_5\}$$

which we know must be greater than $(8/15) * m_2^{TB}$.

Using similar arguments with i-string "$|rlrl$" and inequalities (C.4.4) and (C.4), we know that

$$2r_2+2r_4+2\min\{l_1, r_1\} > (8/15) * m_2^{TB}.$$

Adding these last two inequalities and replacing r_j+l_j by $1/6$ and $\min\{l_k, r_k\} \leqq (1/12) * m_2^{TB}$ results in the inequality

$$1 > 16/15,$$

a contradiction. This completes the proof of this subcase and Case 2.

This completes the proof of the lemma. □

REFERENCES

[AHU] A. AHO, J. HOPCROFT AND J. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1975.

[GL1] T. GONZALEZ AND S. LEE, *An optimal algorithm for optimal routing around a rectangle*, Proc. 20th Annual Allerton Conference on Comm. Control and Computing, Univ. of Illinois, Urbana, IL, October, 1982, pp. 636-645.

[GL2] ———, *An $O(n \log n)$ algorithm for optimal routing around a rectangle*, Technical Report #116, Programs in Computer Science, The University of Texas at Dallas, Dallas, TX, November 1982. (Revised May 1985.)

[GL3] ———, *Routing multiterminal nets around a rectangle*, IEEE Trans. Comput., C-35 (1986), pp. 543-549.

[GLL] U. I. GUPTA, D. T. LEE AND J. LEUNG, *An optimal solution for the channel-assignment problem*, IEEE Trans. Comput., C-28 (1979), pp. 807-810.

[HS] A. HASHIMOTO AND J. E. STEVENS, *Wire routing by optimizing channel assignment without large apertures*, Proc. 8th IEEE Design Automation Conference, 1971, pp. 155-169.

[L] A. S. LAPAUGH, *A polynomial time algorithm for optimal routing around a rectangle*, Proc. 21st IEEE Foundations of Computer Science, 1980, pp. 282-293.

[La] ———, *Algorithms for integrated circuit layout, an analytic approach*, Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, 1980

[MC] C. MEAD AND L. CONWAY, *Introduction to* VLSI *Systems*, Addison-Wesley, Reading, MA, 1980.

[R] R.L. RIVEST, *The* PI (*Placement and Interconnect*) *System*, Proc. 19th IEEE Design Automation Conference, pp. 475-481.

[SBR] S. SAHNI, A. BHATT and R. RAGHAVAN, *The complexity of design automation problems*, Proc. 17th Design Automation Conference, June 1980, pp. 402-411.