Global Routing Based on Steiner Min-Max Trees

CHARLES CHIANG, STUDENT MEMBER, IEEE, MAJID SARRAFZADEH, MEMBER, IEEE, AND C. K. WONG, FELLOW, IEEE

***Abstract*—We study global routing of multiterminal nets. We propose a new global router; each step consists of finding a tree, called a Steiner min-max tree, that is a Steiner tree with maximum-weight edge minimized (real vertices represent channels containing terminals of a net, Steiner vertices represent intermediate channels, and weights correspond to densities). We propose an $O(\min\{e \log\log e, n^2\})$ time algorithm for obtaining a Steiner min-max tree in a weighted graph with e edges and n vertices (this result should be contrasted with the *NP*-completeness of the traditional minimum-length Steiner tree problem). Experimental results on difficult examples, on randomly generated data, on master slice chips, and on benchmark examples from the Physical Design Workshop are included.**

***Key Words:*—Global routing, greedy approach, Steiner min-max tree, optimal algorithm, gate array, difficult examples, master slice chips.**

I. Introduction

Circuit layout (or simply, *layout*) is the process of placing and interconnecting a set of modules as specified by a collection of multiterminal nets. In the top-down approach to circuit layout, after placing the modules on the plane (*placement*), the routing region (i.e., region of the plane external to the modules) is partitioned into *channels. Global routing*, is to find, for each net, a sequence of channels through which it passes. The final step of circuit layout, *detailed routing*, is to find an exact path (routing) for each net as dictated by the global routing.

Here, we focus on the global routing problem. Researchers have studied global routing for the past two decades. Various approaches have been proposed, for example, hierarchical wiring [2], [18], rerouting techniques [24], [1], [16], [20], simulated annealing [25], [17], multicommodity-based techniques [23], and one-step (theoretical) approaches [10], [22]. A survey of global routing methodologies appears in [11].

In this paper, we propose a greedy approach to global routing. Our strategy is to route the nets one by one. First we decide on an ordering of nets to be processed. Then for each net we obtain a global routing trying to avoid "crowded" channels. (The first step has been traditionally employed in conjunction with min-length Steiner trees [4], [15], [3], [5].) We formulate the second step as a graph problem: given a weighted graph we aim to obtain a Steiner tree, called a Steiner min-max tree, whose maximum-weight edge is minimized over all Steiner trees. We propose an efficient polynomial-time algorithm for obtaining an optimal Steiner min-max tree. This result should be contrasted with the intractability (i.e., *NP*-completeness) of the min-length Steiner tree [7] which is used in traditional global routers (e.g., see [11]). Here, we place emphasis on gate arrays (or any environment where density minimization is the main objective). Experiments on "difficult examples," randomly generated examples, IBM master slice chips, and Primary1 and Primary2 gate arrays from the Physical Design Workshop demonstrate the quality and simplicity of our technique.

This paper is organized in the following manner. In Section II, we outline the proposed algorithm for global routing in two-dimensional arrays (i.e., gate arrays). In Section III, we devise an efficient algorithm for obtaining a Steiner min-max tree in a weighted graph, which is central to the proposed global router. In Section IV, the overall implementation strategy is described. Experimental results showing the effectiveness of the proposed technique are included in Section V. Generalization of the overall approach as well as extension of the algorithm to arbitrary floorplans, along with suggested heuristics, are described in Section VI.

II. Global Routing in Two-Dimensional Arrays

Formally, in two-dimensional array global routing of multiterminal nets there are a set $\eta = \{N_1, \cdots, N_n\}$ of multiterminal nets. The *layout environment* (plane grid) is a two-dimensional $m \times m$ grid, being a square tessellation of the plane. Note that all of our results can be trivially extended to a rectangular $m_1 \times m_2$ grid. We use a square grid just for simplicity. Each k-terminal net N is specified by a k-tuple $[(x_1, y_1), \cdots, (x_k, y_k)]$, where (x_i, y_i), $1 \le i \le k$, are the tiles containing terminals of N. In a global routing, for each net, a sequence of tiles through which it passes, is specified.

The following concepts are demonstrated in Fig. 1. In a global routing (i.e., output of a global router) let $d_h(i, j)$ denote the number of nets crossing the border of tiles (i, j) and $(i, j+1)$, $1 \le i \le m$ and $1 \le j \le m - 1$. Similarly, let $d_v(i, j)$ denote the number of nets crossing the border of tiles (i, j) and $(i+1, j)$, $1 \le i \le m - 1$ and $1 \le j \le m$. $d_h^{\max} = \max_{i,j} d_h(i, j)$ is the *hori-*

Manuscript received August 7, 1989; revised December 20, 1989. The work of C. Chiang and M. Sarrafzadeh was supported in part by the National Science Foundation under Grant MIP-87-09074. This paper was recommended by Associate Editor M. Marek-Sadowska.

C. Chiang and M. Sarrafzadeh are with the Department of Electrical Engineering and Computer Science, The Technological Institute, Northwestern University, Evanston, IL 60208.

C. K. Wong is with the IBM T. J. Watson Research Center, Yorktown Heights, NY 10598.

IEEE Log Number 9037117.

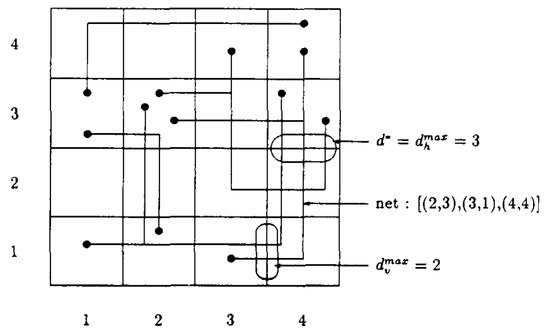


Fig. 1. An instance of global routing in a two-dimensional arrays.

zontal density of the problem and $d_v^{\max} = \max_{i,j} d_v(i, j)$ is the *vertical density* of the problem; $d^* = \max(d_h^{\max}, d_v^{\max})$ is the *global-density* of the problem. Let $c_h(i, j)$ denote the *capacity* of the border of tiles (i, j) and $(i, j + 1)$, $1 \le i \le m$ and $1 \le j \le m - 1$ (i.e., the maximum number of nets that can cross this border). Similarly, let $c_v(i, j)$ denote the maximum number of nets that can cross the border of tiles (i, j) and $(i + 1, j)$, $1 \le i \le m - 1$ and $1 \le j \le m$. $c_h^{\max} = \max_{i,j} c_h(i, j)$ is the *horizontal capacity* of the problem and $c_v^{\max} = \max_{i,j} c_v(i, j)$ is the *vertical capacity* of the problem; $c^* = \max(c_h^{\max}, c_v^{\max})$ is the *global-capacity* of the problem.

An instance of global routing (in a two-dimensional array) is specified by a set η of multiterminal nets and two capacity matrices C_v and C_h, corresponding to vertical and horizontal capacities, respectively, for all i and j. A k-terminal net is said to have *multiplicity* k. Multiplicity of η is the maximum number of terminals per net, over all nets in η. The length of a net, in a global routing, is the number of edges of the grid it crosses. The *bounding length* of a net is **half** the perimeter of the smallest grid rectangle enclosing all terminals of that net. Clearly, the bounding length of a net is a (trivial) lower bound on the length of that net. The global routing problem (GRP) of an arbitrary instance (η, C) is to find a global routing with $d_h(i, j) \le c_h(i, j)$ and $d_v(i, j) \le c_v(i, j)$, for all i and j.

Our strategy is to route the nets one by one. First we decide on an ordering of nets to be processed. Then for each net we obtain a global routing that avoids "crowded" channels.

The first step is to assign a distinct number, called the *order number*, to each net. Here, we give a general description of the order number. In our experiments, we have used a simpled version thereof (see Section IV). Nets that have lower order number will be routed first and intuitively will be shorter. In general, the order number of a net is a function (e.g., the sum) of *priority*, *length*, and *multiplicity* numbers. Each net has a priority number between 1 and α. For example, power nets can be assigned priority number 1 and clock nets can be assigned priority number 2. Length number of a net is proportional to its bounding length (normalized in the range 1 to β). Finally, multiplicity number of a net is its multiplicity (normalized in the range 1 to γ). α, β, and γ dictate importance of each criterion (see Sections IV–VI).

We formulate the second step as a graph problem: given a weighted graph we aim to obtain a Steiner tree, called a Steiner min-max tree, whose maximum-weight edge is minimized over all Steiner trees. We propose (see Section III) a fast algorithm for obtaining an optimal Steiner min-max tree. In an instance (η, C) of global routing, let N_i be the *current* net to be processed. The weighted graph $G_i = (V_i, E_i)$ is dual of the plane grid (i.e., each vertex in the dual graph corresponds to a finite face of the grid graph and vertices corresponding to two adjacent faces are connected by an edge) and hence is itself a grid graph. Thus $|V_i| = (m - 1)^2$ and $|E_i| = 2m(m - 1)$. The weight of an edge is a function (e.g., ratio) of "current" density and capacity and measures "crowdedness" of a border. Each vertex is labeled with *demand* or (potential) *Steiner* depending on whether it is, respectively, a terminal of N_i or not. A Steiner min-max tree of G_i dictates a global routing that minimizes traffic in the densest channel. Each net requires $O(m^2 \text{ loglog } m)$ processing time (see Section III). We concluded the following.

Theorem 1: The proposed global router runs in $O(nm^2 \text{ loglog } m)$ time in an $m \times m$ plane grid, where n is total number of nets.

While the Steiner min-max tree method tends to route nets through less crowded channels, it is also desirable to have nets with short length. Therefore, among all Steiner min-max trees of the given net, we are interested in those with minimum length. As will be discussed in Section III, it is *NP*-hard to find a Steiner min-max tree whose total length is minimized. Thus currently we are employing a number of heuristics that tend to find a Steiner min-max tree with short length. For example, we have noticed that if the terminal closest to the (geometric) center of the bounding rectangle (i.e., the smallest rectangle enclosing all terminals of that net) is chosen to be the starting point of our Steiner min-max tree algorithm (see Section III for more details) then the final tree is shorter than when the starting point is on the boundary of the bounding rectangle.

An example, demonstrating the behavior of an algorithm that minimizes the maximum density, is shown in Fig. 2 (our algorithm is different from this one, for we order the nets in a different manner). In this example, the problem has multiplicity 2 (i.e., all nets are two-terminal nets) and all capacities are equal to 2.

III. Steiner Min-Max Trees

Consider a weighted graph $G = (V, E)$ with each edge e_i having weight $W(e_i)$. The weight of a graph G is sum of weights of its edges; we write, $W(G) = \Sigma_{e_i \in E} W(e_i)$. Each vertex v_i is either a *demand* vertex, denoted by $f(v_i) = d$, or a *Steiner* vertex, denoted by $f(v_i) = s$. A Steiner tree $T = (N, L)$ is an acyclic subgraph of G such that $v_i \in N$ if $f(v_i) = d$, that is, T contains all demand vertices of G and some (or none or all) of the Steiner vertices of

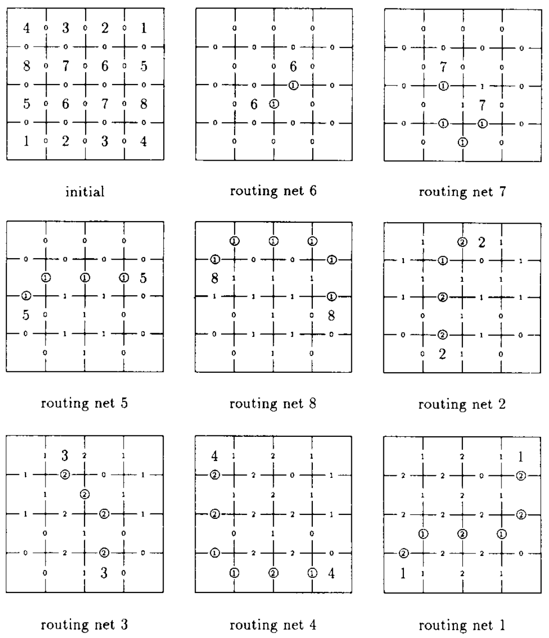


Fig. 2. Example in a 4 × 4 plane grid; local densities are shown on the borders; density increases are circled.

G. The maximum weight edge of L is denoted by $l_{\max}$. A Steiner tree of G with weight $l_{\max}$ minimized is called a *Steiner min-max tree* (SMMT). Here, we propose an efficient algorithm for finding an SMMT of a weighted graph. This result should be contrasted with *NP*-completeness of the Steiner minimum weight-tree problem [7] (that is, a Steiner tree with minimum weight). (Also, a number of closely related problems, for example, the class of center[4] problems, are known to be *NP*-complete [9], [13].)

A minimum spanning tree (MST) $T = (V, L)$ of G is a minimum weight tree containing all (demand and Steiner) vertices of G. Algorithms running in $O(\min\{|E| \log\log |E|, |V|^2\})$ time for finding an MST of $G = (V, E)$ have been proposed [21], [6], [26].

Lemma 1: A MST of $G = (V, E)$ is an SMMT of G if $f(v_i) = d$ for all $v_i \in V$.

Proof: Consider a MST $T = (V, L)$ of G. Let $l_{\max}$ be a maximum weight edge of T. Removing $l_{\max}$ from T leaves two subtrees $T_1 = (V_1, L_1)$ and $T_2 = (V_2, L_2)$ as shown in Fig. 3. Note that $l_{\max}$ is a minimum weight edge connecting V_1 and V_2; otherwise, if there was an edge l^*, $W(l^*) < W(l_{\max})$, connecting V_1 and V_2 then weight of $T^* = (V, \{L - l_{\max}\} \cup l^*)$ would be smaller than weight of T. This is a contradiction, for T is an MST. Thus the maximum weight edge of any spanning tree of G has

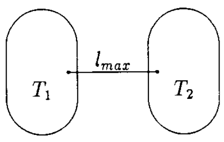


Fig. 3. An MST of G.

weight no less than $l_{\max}$. Equivalently, $T = (V, L)$ is an SMMT of G. □

Given a weighted graph $G = (V, E)$, with a subset D of vertices labeled with d and the rest with s, we obtain an SMMT in the following manner:

```
Procedure ALG-SMMT($G = (V, E), D$);
  begin
    $T :=$ An MST of $G$;
    while there exists a degree 1 Steiner vertex[5] in $T$
    do
      begin
        $v :=$ a degree 1 Steiner vertex of $T$; (* a
          Steiner leaf *)
        Remove $v$ (and the corresponding edge) from
          $T$;
      end
    OUTPUT($T$);
  end.
```

Lemma 2: Procedure ALG-SMMT correctly computes an SMMT of G.

Proof: Let $T^* = (N, B)$ be the output of ALG-SMMT and $T = (V, L)$ be the MST of $G = (V, E)$ computed at the first step. Consider a maximum-weight edge $b_{\max}$ of T^*, that is, $W(b_{\max}) \geq W(b_i)$ for $b_i \in B$. The edge $b_{\max}$ partitions T^* into two subtrees $T_1^* = (N_1, B_1)$ and $T_2^* = (N_2, B_2)$, as shown in Fig. 4. Consider edge $b_{\max} = (n_1, n_2)$ connecting vertices n_1 and n_2 of T_1^* and T_2^*, respectively. Vertex n_1 is either a demand vertex or is eventually connected to a demand vertex $\bar{n}_1 \in T_1^*$, for there are no degree 1 Steiner vertices in T_1^*. Similarly, vertex n_2 is either a demand vertex or is eventually connected to a demand vertex $\bar{n}_2 \in T_2^*$. In any SMMT of G, N_1 and N_2 must be connected by at least one edge because each contains at least one demand vertex. $b_{\max}$ is a minimum weight edge connecting N_1 and N_2. Otherwise, assume there exists an edge $\bar{b}$, with $W(\bar{b}) < W(b_{\max})$, connecting N_1 and N_2. Consider the tree $\bar{T} = (N, \{B - b_{\max}\} \cup \bar{b})$. By adding all degree 1 Steiner vertices (that have been removed from T in ALG-SMMT, that is, $L - B$) we obtain a spanning tree with total weight $(W(b_{\max}) - W(\bar{b}))$ less than the total weight of $T = (V, L)$. This is a contradiction, for T is a minimum spanning tree. Thus the minimum max-weight edge in any tree on G including all demand vertices (and thus any SMMT of G) is $b_{\max}$. □

[4]In center problems, the objective is to locate K centers such that the maximum distance from any demand location to the closest center is minimized. As opposed to SMMT, the resulting network need not be connected.

[5]If there exists a Steiner vertex that is connected to at most one other vertex then remove it. Repeat this step while there remains such a Steiner vertex.

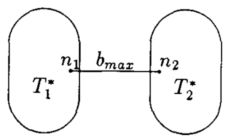


Fig. 4. Output of ALG-SMMT.

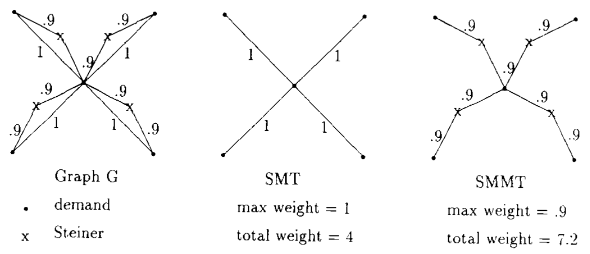


Fig. 5. Total weight of a SMMT.

ALG-SMMT involves computing an MST [21], [6], [26]. Removing degree 1 Steiner vertices requires $O(|V|)$ time. We conclude the following:

Theorem 2: A SMMT of a weighted graph $G = (V, E)$ can be obtained in $O(\min\{|V^2|, |E| \operatorname{loglog} |E|\})$ time.

First, we observe that the weight of a SMMT, in general, is greater than the weight of a Steiner minimum weight tree (SMT) as shown in Fig. 5 (the problem of finding a SMT is known to be NP-complete [7]). The problem of finding an SMMT whose total weight is minimized over all SMMT's (called MSMMT) is not polynomial-time solvable, unless $P = NP$.

Theorem 3: MSMMT is NP-complete.

Proof: We transform SMT, in polynomial time, to MSMMT. SMT is known to be NP-complete [7].

Instance (of SMT): A weighted graph $G = (V, E)$ with vertices classified as Steiner or demand vertices, and an integer K.

Question: Is there an acyclic subgraph $T = (N, L)$ of G with all demand vertices of G in T such that (total) weight of T is less than K.

Consider an arbitrary instance $G = (V, E)$ of SMT. Let $l_{\max}$ be a maximum weight edge of G. Consider a graph $H = (V \cup v, E \cup (v, v_d))$, where v_d is any demand vertex of G, $W(e) = W(l_{\max}) + 1$, where $e = (v, v_d)$. Vertices of H are labeled (demand or Steiner) exactly as vertices of G and vertex v is classified as a demand vertex. Consider an MSMMT T^* of H. Note that edge e must be in T^*, for it is the only edge connecting v to the rest of the vertices. Thus there exists an SMT of G with weight less than K if and only if the weight of T^* is less than $K + W(l_{\max}) + 1$. (By adding (v, v_d) with weight $W(l_{\max}) + 1$ we have essentially removed max-weight edge restriction of MSMMT and this MSMMT will be equivalent to SMT.) □

We have introduced heuristics for obtaining optimal SMMT's with "small" total length. For example, as mentioned in Section II, by choosing, as the starting terminal of a net (in ALG-SMMT), the terminal closest to the geometric center of the smallest rectangle enclosing all terminals of that net, we obtain shorter trees (see Section IV). Other heuristics are outlined in Section VI.

IV. Overall Implementation Strategy

As described earlier, we have incorporated a number of heuristics in our algorithm. In our implementation, we have focused on heuristics that minimize the total length (see Section VI for overflow considerations).

In general, the net ordering should be a function of priority, length, and multiplicity numbers. However, for our test cases, length alone seemed to be a sufficiently good ordering parameter. Specifically, let p_i denote half the length of the perimeter of the smallest rectangle enclosing all terminals of N_i. Let Π be a rank function on the set of p_i's. That is, the smaller p_i the lower its rank $\Pi(i)$. We order the nets as dictated by Π. The algorithm is performed in two phases: the SMMT-phase and the SP-phase (SP-phase is essentially a minimum-spanning tree algorithm, to be elaborated on, below). The SMMT-phase consists of J_1 steps and the SP-phase consists of J_2 steps, where J_1 and J_2 are heuristic design parameters. Intuitively, J_1 and J_2 are, respectively, based on the importance of density and length minimization in a problem (specific values will be given in Section V).

In the SMMT-phase, we route the nets one by one, employing ALG-SMMT (see Section III). At the jth step of the SMMT-phase, if the length of routing of N_1 is within a constant factor, c_j of its "minimum length" (i.e., p_i or half the bounding length) then we accept it. Otherwise, the routing is rejected. If $J_1 = 1$ then we choose c_1 to be a number between 1 and 2, normally, 1.5. Otherwise, we choose, $c_1 = 1$, $c_{J_1} = \infty$, and $c_{i+1} < c_i + \kappa$ for $i > 1$, where κ is a number between 1 and 2. During the SMMT-phase, once a net is routed, it will not be routed again.

In the SP-phase, we route all the nets one by one (as dictated by Π) employing a shortest path heuristic (to be described later) and utilizing the results from the SMMT-phase. At the jth step, we accept a routing if and only if it is better than the best routing obtained so far.

In instances of global routing with limited capacities, such as the examples of the IBM master slice chips, where our goal is to increase the number of routed nets, we choose a large value for J_1 and a small value for J_2. In problems with enough capacity (i.e., the goal is to minimize the densities) we choose J_1 small and we pick a large value for J_2. (See Section V, for examples of specific numbers. In one case, J_1 is between 8 and 10 and J_2 is between 1 and 2. In the other case, the reverse is true.)

A formal description of the proposed approach follows (for simplicity, we do not pass the grid graph to ALG-SMMT and ALG-SP):

```
for $i = 1$ to $n$ do
$L(N_i) = \infty$; (;* initialize length of all nets *)
```

```
SMMT-phase
Pass $j (j \le j_1)$;
    Assign $c_j$; (* current constant *)
    for $i = 1$ to $n$ do
    begin
        $N_i$: = current net; (* as dictated by $\Pi$ *)
        if $L(N_i) = \infty$ then
        begin
                temp := ALG-SMMT($N_i$); (* length of $N_i$
                  by ALG-SMMT *)
                if temp $\le c_j p_i$ then
                  $L(N_i)$ := temp; (* accepting routing of $N_i$ *)
        end
    end

SP-phase
Pass $j (J_1 < j \le J_1 + J_2)$;
  for $i = 1$ to $n$ do
    begin
      $N_i$ := current net; (* as dictated by $\Pi$ *)
      temp := ALG-SP($i$) (* length of $N_i$ by ALG-SP *)
      if temp $< L(N_i)$ then
        $L(N_i)$ := temp;
              (* changing the routing of previously
              routed nets will make some edges avail-
              able for unrouted nets *)
    end.
```

ALG-SP is any minimum spanning tree algorithm, for example, [21], [26]. Indeed, both ALG-SMMT (see Section III) and ALG-SP can be implemented in a manner similar to Prim's minimum spanning tree algorithm [21]. Consider a net N with terminals $t_1, \cdots, t_k$. We start from a geometric center terminal (i.e., a terminal closest to the center of the smallest rectangle enclosing all terminals of N) and expand from that terminal until all terminals of N are reached. Assume a partial routing R has been obtained. Initially, the partial routing is the center terminal of N. Among all terminals of N which have feasible paths leading to R, i.e., paths not violating the current capacity constraints, we add a terminal t to R if a feasible path from t to R minimizes the maximum-weight edge in ALG-SMMT or minimizes the total-length in ALG-SP. Once all terminals are reached we remove all degree-1 Steiner vertices (see ALG-SMMT, Section III).

V. Experimental Results

The algorithm proposed in Section IV has been implemented in the C language running on a VAX 785 under Berkeley Unix. Performance of our algorithm on "difficult examples," randomly generated data, master slice chips, and benchmark examples from the Physical Design Workshop is very good.

We convert Primary 1, Primary 2, and the IBM chips into their gate-array images (see Fig. 1) using the technique suggested by Nair [20]. That is, we draw vertical lines every k units. Horizontal lines are drawn to cut both the row of cells and the channels.

5.1. Benchmark Examples from Physical Design Workshop

We have tested two gate array benchmark circuits (Primary1 and Primary2) from the 1988 IEEE Workshop on Placement and Routing (Research Triangle Park, NC, May 10-13). The placement of Prim1-GA and Prim2-GA were obtained from TimberWolfSC Version 5.1 [17]. The results of our global router is shown in Table I. To the best of our knowledge, no other global router which uses our particular density metric has been run on these examples with the same value of m (i.e., number of columns).

It is of some (indirect) interest that the exact routing channel (i.e., the region between the rows) densities were reported in [17]. In [17] density is defined as it is normally defined in channel routing problems. Using this (different) metric (and placements different from ours) the densities reported in [17] were 8 tracks for Primary1-GA and 17 tracks for primary2-GA.

It is possible to establish a (trivial) lower bound on the maximum density. Consider the boundary i of two columns i and $i + 1$ (see Fig. 1). Let δ_i denote the number of nets with at least one terminal to the left of boundary i and at least one terminal to the right thereof. Let $\delta_{\max}$ denote the maximum over all $\delta_i s$. Then the maximum density is lower bounded by $\delta_{\max}/m$, where m is the number of rows. In Primary1-GA we obtain $\delta_{\max} = 5$ and in Primary1-GA $\delta_{\max} = 8$. Note that $\delta_{\max}$ is just an indication of how good an upper bound is. Indeed, the (true) lower bound is normally much higher than those indicated by $\delta_{\max}$.

5.2. IBM Master Slice Chips

We have tested examples used in [20] from IBM master slice chips. Table II summarizes our results. The column labeled "Total Nets" is the total number of nets in the circuit. "impossible nets" are obtained by searching all rectangles with size $i \times j$, $1 \le i \le m_1$ and $1 \le j \le m_2$. In each rectangle R, let M denote the set of nets with at least one terminal inside R and at least one terminal outside R. Let c denote the sum of capacities on the boundary of R. Clearly, (at least) $|M| - c$ nets are impossible to route. A formal description of an algorithm for finding "impossible nets" follows:

```
Pass($i, j$) (* rectangle with size $i \times j$ *)
    Set each net's counter to 0; (* initialization *)
    for each $i \times j$ rectangle $R$ do
        begin
            $M$ := the set of nets which have terminals
              both inside and outside rectangle $R$;
            $c$ := the sum of capacities on the boundary
              of $R$;
            If $|M| > c$ then
              $R$ has an overflow so increase the counter
                of each in $M$ by one;
        end
```

TABLE I
BENCHMARK EXAMPLES

Circuit	No. cells	Max Density	Time (sec.)
Primary1-GA	752	6	25.5
Primary2-GA	3000	12	155.05

TABLE II
IBM MASTER-SLICE CHIP

Part	Total Nets	Impossible Nets	Effect. Nets	Unrouted Nets *	Comp(%) **	Time (sec.)
L	312	26	286	6	97.90	1.68
I	1292	5	1287	76	94.10	12.63
R	1726	29	1697	107	94.70	42.08
M	3041	0	3041	124	95.92	147.18
S	3105	0	3105	0	100	53.80

* Number of effective nets which could not be routed
** Percentage of completion of effective nets

```
L := sorted list of net counters in descending
  order;
while first(L) ≠ 0 do (* first element in list L *)
  begin
    N := net of first(L);
    N is designated an impossible net;
    delete first(L) form list L;
    delete all N's terminals from the grid;
    if after deleting N, R (which originally had an
      overflow) no longer has an overflow
then
      decrease the counter of each net in M by 1;
      L := sorted list of net counters in descending
      order;
  end.
```

Effective nets are simply the total nets minus the impossible nets. Note that we searched only rectangles. Instead, if we search all rectilinear regions a better bound on the number of impossible nets is expected. However, the time complexity increases rapidly. Here, we have selected $J_1 = 10$, $J_2 = 0$, $c_1 = 1$, $c_2 = 1.2, \cdots, c_9 = 2.6$, and $c_{10} = \infty$.

5.3. *Difficult and Random Data*

An $m \times m$ grid is partitioned into four equal regions called NW, NE, SE, and SW with obvious meaning. An instance of global routing in an $m \times m$ grid, for even values of m, is called a *difficult-m* instance if there are $m^2/2$ two-terminal nets and each net has one terminal in the NW region and one terminal in the SE region, or if a net has one terminal in the SW region and one terminal in the NE region. Clearly, in a difficult-m instance $d^* \geq m/2$. In fact, as shown in [10] $d^* \geq m/2 + 1$ (for $m \equiv 2, 3 \pmod 4$). We have tried a large number of difficult-m problems and we always achieve $d^* = m/2$ or $d^* = m/2 + 1$. Three instances of difficult-m problems (m = 4, 8, and 16) are shown in Figs. 2, 6, and 7. (As shown in [22], problems involving "short" nets are easier.)

Also, we have tried a number of randomly generated examples involving multiterminal nets. An instance of a randomly generated data in an $m \times m$ grid is denoted by random-m (Table III). In both difficult and random examples we have selected $J_1 = 1$, $J_2 = 5$, and $c_1 = 1.5$.

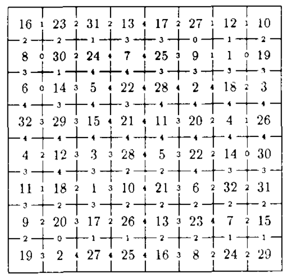


Fig. 6. An instance of difficult-8.

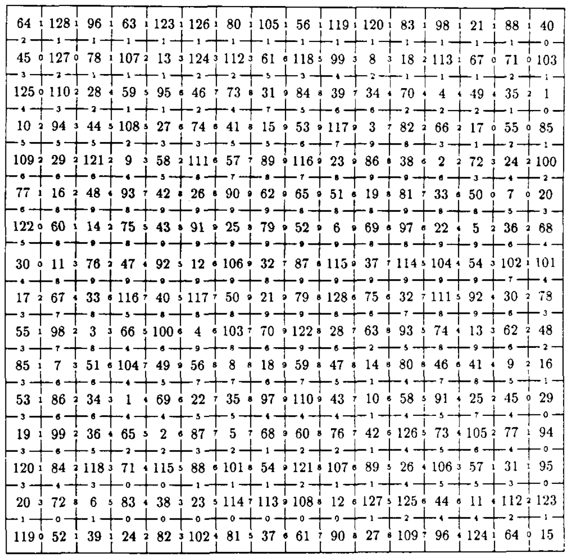


Fig. 7. An instance of difficult-16.

TABLE III
DIFFICULT AND RANDOM DATA RESULTS

	Description				Results	
Name	No. of Nets	Multiplicity	Bounding Length	Max. Cap.	Total Length	Time (sec.)
Difficult-4	8	2	32	2	35	0.07
Difficult-8	32	2	256	4	296	0.15
Difficult-16	128	2	2048	9	2214	2.75
Random-4	5	4	15	2	16	0.10
Random-8	18	4	154	3	160	0.27
Random-16	72	6	1148	6	1229	1.50

VI. EXTENSIONS

In this section, we extend the algorithm proposed in Sections II and IV in two ways; first, we discuss a generalization of the overall approach discussed in Section IV. Second, we consider arbitrary floorplans.

First we note that the proposed technique can handle multilayer routing. Consider a horizontal edge of a tile (see Section II) with capacity c. If there are k layers allowed to use a horizontal edge then we let up to kc nets to cross this edge. In detailed routing, the wires will be assigned to distinct layers.

Let m_i be the multiplicity of net N_i (i.e., number of terminals of N_i) and p_i denote half the length of the perimeter of the smallest rectangle enclosing all terminals of N_i. For each net N_i we define a *limitation* l_i being a linear combination of p_i and m_i. Similarly, for N_i we define an (*allowable*) *overflow* f_i which is also a linear combination of p_i and m_i. Intuitively, a routing of N_i with total length proportional to l_i and causing overflow proportional to f_i is acceptable; otherwise, it is not acceptable.

Given $\eta = \{N_1, \cdots, N_n\}$ we define an initial order number $\Pi^1(i)$, for each net N_i, as a linear combination of p_i and m_i. A net with $\Pi^1(i) = k$ is the kth net to be routed. Thus Π^1 is a permutation of $(1, \cdots, n)$.

In the first pass, we route the nets, one by one, as dictated by their order number. Then we compare the current routing of a net, if it is routable, with its former routing (initially, the routing length of a net is set to infinity) to decide if this routing is accepted or not. In subsequent iterations, a new order number will be given to each net. A formal description of the proposed heuristic is given below. We run this heuristic J times in the following description, where J is a constant. From step 1 to step J_1 ($J_1 \le J$) we employ ALG-SMMT (see Section III). From step $J_1 + 1$ to step J we use a shortest path heuristic, called ALG-SP (see Section IV), to route each net.

```
Pass $j (j \le J)$;
  for $i = 1$ to $n$ do
    begin
      $N_i$ := current net; (* as dictated by $\Pi^j$ *)
      if $j < J_1$ then
        ALG-SMMT ($N_i$); (* see Sections III and IV *)
      else
        ALG-SP ($N_i$); (* see Section IV *)
      $l_i^j$ := length of routing of $N_i$;
      $f_i^j$ := amount of overflow in routing of $N_i$;
      if $\lambda_1^j l_i^j + \lambda_2^j f_i^j \le \lambda_1^{j-1} l_i + \lambda_2^{j-1} f_i$ then
        accept routing of $N_i$;
      else
        reject routing of $N_i$;
      $\xi_i = [\lambda_1^j l_i^j + \lambda_2^j f_i^j]/[\lambda_1^{j-1} l_i + \lambda_2^{j-1} f_i]$;
      $\Pi^{j+1}(i) = \Phi(\xi_i)$; (* $\Phi$ is a rank function *)
    end
```

For small j, we emphasize producing short nets and for large j we focus on reducing the amount of overflow. Thus for small j, λ_1^j will be large and for large j, λ_1^j will be small. For λ_2^j the reverse is true. Also, the closer a routing to the desired value the smaller is its ξ, so that with new λ's it has a better chance of being routed in the next iteration. Therefore, it will have a smaller order number, as assigned by Φ, in the next iteration.

Next we discuss extension of the algorithm to arbitrary floorplans. Given a floorplan, we represent the input of the global routing by a weighted planar graph $G = (V, E)$, as shown in Fig. 8. In this graph, called an *adjacency graph*, vertices correspond to modules and edges denote adjacency of corresponding modules. The weight of an edge denotes the capacity between the corresponding modules. Also, we are given a set $\eta = \{N_1, \cdots, N_n\}$ of multiterminal nets. Each net specifies a subset of vertices of V to be interconnected. (In the two-dimensional arrays, discussed in Section II, G is a grid graph.)

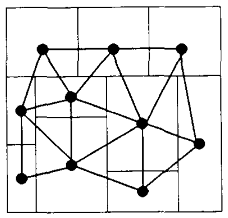


Fig. 8. An arbitrary floorplan.

As before, we propose the following two-step algorithm: first, decide on an ordering of nets (depending on their bounding length, multiplicity, and priority) and then route each net employing the proposed SMMT algorithm (see Section III).

Finally, we observe the following heuristics will enhance the performance of our algorithm.

- *Heuristic 1:* First, consider only very "short" nets (most of which are two-terminal nets). Assign a simple shape (e.g., L shape) to them. This provides an initial routing. Then proceed with the algorithm.
- *Heuristic 2:* Use the proposed algorithm iteratively (i.e., discard the routing of some nets and reroute them), as described in [20].
- *Heuristic 3:* Modify the proposed SMMT algorithm so that a straight path is preferred to a bend (turn), since in most routing models each bend corresponds to a via.

VI. Conclusion

In this paper we proposed a two-step algorithm for global routing: first we order the nets using bounding length, multiplicity, and criticality criteria. Next, for each net, we find a Steiner SMMT, that is, a Steiner tree with maximum-weighted edge minimized. Experimental results on various examples were given.

Acknowledgment

The authors wish to acknowledge many enlightening discussions with Jason Cong, Kai-Win Lee, Ravi Nair, and Carl Sechen. They would also like to thank them for providing the authors with data.

References

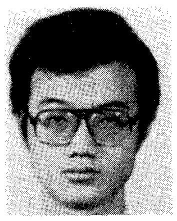
[1] A. Aoshima and E. Kuh, "Multi-channel optimization in gate array LSI layout," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1983.

[2] M. Burstein and R. Pelavin, "Hierarchical wire routing," *IEEE Trans. Computer-Aided Design*, vol. CAD-6, pp. 223-234, Oct. 1987.

[3] J. P. Cohoon, D. S. Richards, and J. S. Salowe, "An optimal Steiner tree algorithm for a net whose terminals lie on the perimeter of a rectangle," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 398-407, Apr. 1990.

[4] F. K. Hwang, "On Steiner minimal trees with rectilinear distance," *SIAM J. of Appl. Math.*, vol. 30, pp. 104–114, Jan. 1976.
[5] J. M. Ho, G. Vijayan, and C. K. Wong, "A new approach to the rectilinear Steiner tree problem," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 185–193, Feb. 1990.
[6] J. B. Kruskal, Jr. "On the shortest spanning subtree of a graph and the traveling salesman problem," in *Proc. Amer. Math. Soc.*, vol. 7, no. 1, 1956, pp. 48–50.
[7] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations.* Plenum Press, 1972, pp. 85–163.
[8] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Sci.*, vol. 220, pp. 671–680, 1983.
[9] O. Kariv and S. L. Hakimi, "An algorithmic approach to network location problem—Part I: The p-centers," unpublished manuscript.
[10] R. M. Karp, F. T. Leighton, R. L. Rivest, C. D. Thompson, U. Vaziriani, V. Vaziriani," "Global routing in two-dimensional arrays," *Algorithmica*, vol. 2, no. 1, pp. 113–130, 1987.
[11] E. S. Kuh and M. Marek-Sadowska, "Global routing," in *Layout Design and Verification*, T. Ohtsuki, ed. New York: Elsevier Science, 1986.
[12] C. Y. Lee, "An algorithm for path connection and its application," *IRE Trans. Electron. Comput.*, vol. EC-10, pp. 346–365, 1961.
[13] D. Lichtenstein, "Planar formulae and their uses," *SIAM J. Comput.*, vol. 11, 1982, pp. 329–343.
[14] R. Linsker, "An iterative improvement penalty function drive wire routing system," *IBM J. Res. Develop.*, vol. 28, no. 5, pp. 613–624, Sept. 1984.
[15] J. H. Lee, N. K. Bose, and F. K. Hwang, "Use of Steiner's problem in sub-optimal routing in rectilinear metric," *IEEE Trans. Circuits Syst.*, vol. CAS-23, pp. 470–476, July 1976.
[16] J. T. Li and M. Marek-Sadowska, "Global routing for gate arrays," *IEEE Trans. Computer-Aided Design*, vol. CAD-3, pp. 298–307, Oct. 1984.
[17] K.-W. Lee and C. Sechen, "A new global router for row-based layout," in *Proc. IEEE Int. Conf. on Computer-Aided Design*, Nov. 7–10, 1988, pp. 180–183.
[18] W. K. Luk, P. Sipala, M. Tamminen, D. Tang, L. S. Woo, and C. K. Wong, "A hierarchical global wiring algorithm for custom chip design," *IEEE Trans. Computer-Aided Design*, vol. CAD-6, pp. 518–533, July 1987.
[19] E. F. Moore, "Shortest path through a maze," *Annals of Computation Laboratory.* Cambridge, MA: Harvard University Press, 1959, pp. 285-292.
[20] R. Nair, "A simple yet effective technique for global routing," *IEEE Trans. Computer-Aided Design*, vol. CAD-6, pp. 165-172, 1987.
[21] R. C. Prim, "Shortest connection networks and some generalizations," *Bell Syst. Tech. J.*, no. 36, pp. 1389–1401, 1957.
[22] M. Sarrafzadeh and D. Zhou, "Global routing of short nets in two-dimensional arrays," *The Int. J. Computer Aided VLSI Design*, vol. 2, no. 2, pp. 197–211, 1990.
[23] E. Shargowitz and J. Keel, "A global router based on multicommodity flow model," *INTEGRATION: The VLSI J.*, vol. 5, pp. 3–16, 1987.
[24] B. S. Ting and B. N. Tien, "Routing techniques for gate arrays," *IEEE Trans. Computer-Aided Design*, vol. CAD-2, pp. 301–312, Oct. 1983.
[25] M. P. Vecchi and S. Kirkpatrick, "Global wiring by simulated annealing," *IEEE Trans. Computer-Aided Design*, vol. CAD-2, pp. 215–222, 1983.
[26] A. Yao, "An $O(|E| \text{ loglog } |E|)$ algorithm for finding minimum spanning trees," *Inform. Proc. Lett.*, vol. 4, pp. 21–23, 1975.

*



Charles Chiang received the B.A. degree from Tunghai University, Taichung, Taiwan, R.O.C. in 1980 and the M.S. degree in computer science from Northwestern University, Evanston, IL in 1988. He is currently working towards the Ph.D. degree at Northwestern University.

Science 1987, he has been a Research Assistant with Department of Electrical Engineering and Computer Science, Northwestern University. He was with the VLSI group of Department of Computer Science at IBM T. J. Watson Research Center, Yorktown Heights, NY in 1989. His research interests include computer-aided design and layout of VLSI circuits.

*

Majid Sarrafzadeh (S'82–M'87), for a photograph and biography, please see page 426 of the April 1990 issue of this TRANSACTIONS.

*

C. K. Wong (M'71–SM'78–F'85), for a photograph and biography, please see page 193 of the February 1990 issue of this TRANSACTIONS.