An Approximation Algorithm for the Via Placement Problem

TEOFILO F. GONZALEZ AND SHASHISHEKHAR KURKI-GOWDARA

Abstract—**In this paper, we consider the via placement problem that arises in multilayer printed circuit board (MPCB) layout systems. It is shown that this problem can be formulated as an integer linear max-flow problem. Since the integer linear max-flow problem is an *NP*-complete problem, it is unlikely that one can find an efficient algorithm for its solution. However, a solution to our via placement problem can be obtained by relaxing the integer constraints in the integer linear max-flow problem. A solution to the relaxed linear max-flow problem can be obtained by solving a linear programming problem. Our procedure generates in time bounded by a low order polynomial a placement with no more than 2 * d_{OPT} density, where d_{OPT} is the density in an optimal placement.**

Keywords—**Multilayer printed circuit boards, routing, via placement, polynomial time approximation algorithms, NP-complete problems, linear max-flow problem.**

I. Introduction

LARGE SCALE computer systems are built by interconnecting multilayer printed circuit boards (MPCB). Each MPCB consists of a grid with r rows and n columns. The computer chips (components) are mounted on top of the MPCB and their terminals are inserted in drilled-through holes called *pins*. The pins are located at grid points (intersection of a row with a column). Terminals from different chips are connected by printed wires located on any of the layers. Each set of points (pins) that need to be made electrically common is called a *net*. In this paper we use the routing model proposed by So [17]. In this model only points (pins) on the same line (row or column) can be connected directly. The physical routes for these wires must be confined within the channels on both sides of the line. All the row connections are performed on one layer, and all the column connections are carried out on another layer. The main advantage of this model is that the MPCB routing problem is reduced to solving no more than $r + n$ single-line single-layer routing problems. The single-line single-layer routing problem can be solved by the algorithms given in [10], [13], [14], and [19]. A sample problem is shown in Fig. 1(a) and a wire layout is given in Fig. 1(b).

Manuscript received March 3, 1988; revised August 23, 1988 and October 21, 1988. This work was supported in part by the National Science Foundation under Grant DCR-8503163. A preliminary version of this paper entitled "A suboptimal algorithm for the via placement problem" appeared in *Proc. 25th Ann. Allerton Conf. on Communications, Control and Computing*, Oct. 1987. The review of this paper was arranged by Associate Editor R. J. M. Otten.

The authors are with the Department of Computer Science, University of California, Santa Barbara, CA 93106.

IEEE Log Number 8824375.

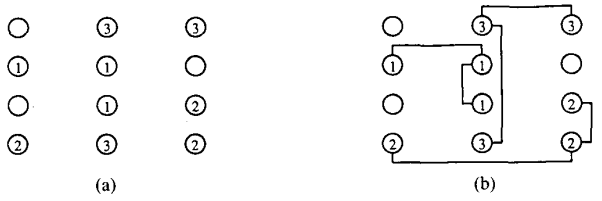


Fig. 1. Simple MPCB routing problem.

It is simple to find problem instances that cannot be solved under the above model (Fig. 2(a)). To solve these problem instances, one introduces a set of via columns. These via columns are initially placed to the right of all the pin columns and the vias are located at grid points. Hereafter, when we refer to a point, we mean a pin or a via; and when we refer to So's model, we use the new definition for point. We assume that n also includes the via columns. The vias are assigned to the nets in such a way that a routing under the above model exists (see Fig. 2(b)).

The *via assignment problem* (VAS) consists of adding the least number of via columns and specifying the points (pins and/or vias) that need to be connected directly in such a way that no two pins from the same net are connected to vias from different via columns and a routing under the above model exists. The *multi-via assignment problem* (MVAS) is defined similarly, except that we allow the pins from a net to be connected to vias from more than one via column. Fig. 3 illustrates via assignments and a final layout for an instance of the VAS and MVAS problems. As noted in [18], an optimal solution to the MVAS problem requires no more via columns than an optimal solution to the VAS problem; however, the number of vias used in an optimal solution to the MVAS problem is usually larger than those in an optimal solution to the VAS problem (e.g., see Fig. 3).

As noted in [7], finding an efficient approximation solution for the VAS problem is as hard as finding an efficient approximation algorithm for the (node) graph coloration problem. The problem of finding a good approximation solution to the (node) graph coloration problem is a computationally difficult problem [3]. On the other hand, finding a good approximation solution for the MVAS problem is a computationally tractable problem [7]. Approximation algorithms and heuristics for these problems appear in [1], [7], [18], and [20]. For the MVAS problem the algorithm in [7] guarantees a solution with a

0278-0070/89/0300-0219$01.00 © 1989 IEEE

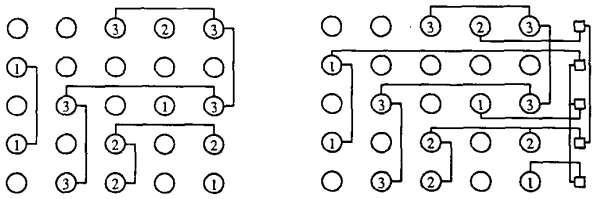


Fig. 2. MPCB routing problem. (a) MPCB routing problem that cannot be solved without introducing via columns. (b) A solution to (a) with one via column.

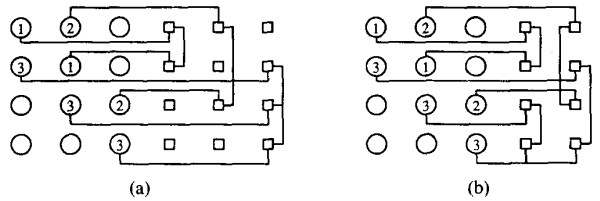


Fig. 3. Solutions to the VAS and MVAS problem.

number of via columns which is not more than 3 * OPT, where OPT is the number of via columns in an optimal solution.

The above formulation suffices to solve all MPCB routing problems as long as there are enough tracks between each pair of adjacent lines. This assumption is not realistic. In practice, there are only a small number (usually two) of tracks on each side of the line in each of the single-line single-layer routing problems. In order to solve the MPCB routing problems when the number of tracks is small, it is required to increase the number of layers. That is, if there are h layers for routing the row wires and v layers for routing the column wires, then instead of solving $r + n$ single-line single-layer routing problems, we solve r single-row h-layer routing problems and n single-column v-layer routing problems. The *layer assignment (layering) problem* consists of finding the least number of layers $(h + v)$ required to solvé an MPCB routing problem. There are several heuristics and approximation algorithms to solve this problem [8], [9], [21]. Gonzalez and Kurki-Gowdara [8] developed a polynomial time dynamic programming algorithm that generates an optimal solution to the layering problem when there are a small number of tracks and a fixed number of layers. The main drawback of this algorithm is that the constant associated with the worst case time complexity bound is large; however, the experimentation reported in [8] indicates that quite often the algorithm generates near-optimal solutions quickly.

There are MPCB routing problems whose optimal solution under the above model requires an excessive number of layers. A solution with a smaller number of layers can be obtained by moving the via columns and placing them between adjacent chips. This operation is realistic since it can be carried out by moving the chips apart while maintaining their relative columnwise ordering. The placement of the via columns is such that the maximum density of any of the single-row routing problems is least possible, where by *density* we mean the maximum number of nets that cross (with a pin to the right and to the left) any channel (the region between adjacent points) on the row. This problem is called the *via placement optimization (VPO) problem*. Note that different placements have exactly the same column density, but their row densities may be different. Another important observation is that minimizing the density does not necessarily imply that a solution with the minimum number of layers can be obtained from it. However, minimizing density tends to minimize the number of layers. We assume that if a via in row i is assigned to net j, then there exists at least one pin from net j in row i. This assumption is not too restrictive because if vias are assigned using the VAS model, then the assumption is immediately satisfied. When vias are assigned according to MVAS model this condition may not be satisfied. However, if one solves the MVAS problem using the algorithm in [7], then the assumption holds. Since the algorithm in [7] is the currently best algorithm for the MVAS problem, it appears at this time that our assumption is not restrictive. Heuristics for the VPO and related problems appear in [2] and [6].

In this paper we study the VPO problem. We present an algorithm that generates a solution with a density which is not more than twice the optimal density d_{OPT}. Our technique consists of reducing the VPO problem to an integer linear max-flow problem. Since this problem has been shown to be *NP*-complete [15], it is unlikely that one can find an efficient algorithm for its solution [4]. However, if we relax the integer constraints in this linear max-flow problem, a solution may be obtained by standard linear programming algorithms [5] or by more recent algorithms [11], [12], [22]. The solution to the (non-integer) linear max-flow problem is not a solution to our VPO problem, but one can obtain a solution to our problem from it. The density for such a solution is shown to be within two times the density in an optimal solution. The time complexity for our algorithm is bounded by a low order polynomial in terms of r and n, and we believe it can be obtained quickly. In Section II we formally define our problem as well as the integer max-flow problem. In Section III we show that a restricted via placement decision problem can be solved by formulating it as an integer max-flow problem with bundle capacities. In Section IV we show that a solution with a density that is within two times the optimal density can be obtained from a solution to the (integer constraint relaxed) max-flow problem with bundle capacities. We show in Section V how the restrictions made in the previous sections can be handled by a more general algorithm. Therefore, our result holds for the unrestricted VPO problem.

II. Definitions and Problem Simplification

Let P be a multilayered printed circuit board. Board P consists of r rows and n columns (columns 1, 2, $\cdots$,

$n - v$ are pin columns and remaining v columns are via columns numbered $1, 2, \cdots, v$). At the intersection of a row with a column there is a point (pin or via). Each point belongs to at most one net. The nets are represented by sets $N_1, N_2, \cdots, N_m$, where N_i consists of the set of points in net i. Between every pair of adjacent pin columns there is a channel, and there is a channel to the left (right) of the first (last) pin column. We label the channels with the column number that appears immediately to the left of the channel, if there is one, and zero otherwise. Fig. 4 shows a problem instance with $r = 5$, $n = 11$, $v = 4$, $m = 6$, and the integer i at a grid point indicates that the point belongs to net N_i. The objective of the *VPO problem* is to place the via columns in the channels (any number of via columns per channel) in such a way that the density of the resulting problem is least possible. The *point density* of (point) p located at the intersection of row k with column l is defined as the number of nets with $s_i \leq l \leq e_i$ and $s_i \neq e_i$, where $s_i(e_i)$ is the leftmost (rightmost) column containing a point from net i in row k. The *row density* is the maximum of the point densities for the points in that row. The *density of a problem* is defined as the maximum of the row densities. Note that all via placements have exactly the same column density. We should also point out that each via column has at least two vias assigned to nets. If this is not the case, the via column may be deleted without modifying the pin connectivity.

Instead of solving the VPO problem directly, we solve a set of *via placement decision problems* (*VPD problems*) and then from their solutions we construct the solution to the optimization problem. The input to the VPD problem is identical to that for the VPO problem, except that there is another input value denoted by $d_{\max}$. The VPD problem consists of determining whether there is a placement, P, for the via columns with density, denoted by $d(P)$, at most $d_{\max}$. Let d_{curr} be the density of the pin columns (remember that via columns are not pin columns) in the VPD problem. Clearly, in any placement, P, of the via columns the resulting density $d(P)$ is at most $(d_{\text{curr}} + v)$. The decision problem can be solved a logarithmic number of times for $d_{\max}$ between d_{curr} and $(d_{\text{curr}} + v)$ until the optimal density bound is found. Thus by repeatedly solving the decision problem $\log(v)$ times, the optimal density is obtained. Once this optimal value is computed, an optimal placement is constructed from the "flow" values obtained by solving the decision problem (this will be explained later on). The main reason for introducing the via placement decision problem is to simplify our exposition. In Section V we show how to solve the via placement optimization problem directly.

Let us now consider the following two restrictions on a via placement problem.

R1) If a via in row i is assigned to net j, then no other via assigned to net j is located in row i.

R2) If there is a placement P for the via placement problem with density $d(P)$, then there is another placement P' with density $d(P') \leq d(P)$ in which the via columns that appear in each channel are placed in increasing order of their initial column ordering (index).

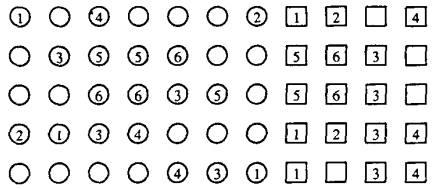


Fig. 4. MPCB routing problem.

In what follows we use the notation (i, j)-VPO (or (i, j)-VPD) to refer to a VPO (VPD) problem that satisfies restrictions *Ri* through *Rj*. It is important to remember that our final result is a polynomial time algorithm for the (unrestricted) VPO problem that generates a solution with density $\leq 2 * d_{\text{OPT}}$.

The *integer max-flow problem with bundle capacities* is defined as follows. As input we are given a directed graph $G = (N, E)$ with two special nodes labeled *source* and *sink*, and an edge e_0 directed from the sink to the source; a collection $S = (S_1, S_2, \cdots, S_l)$ of subsets of arcs; and a collection $C = (c_1, c_2, \cdots, c_l)$ of bundle capacities. Graph G must also satisfy the property that the only edge emanating from the sink and the only edge ending in the source is edge e_0. The flow problem consists of assigning a non-negative integer flow to each edge in E so as to maximize the flow along e_0 (i.e., from sink to source, which is identical to the flow from source to sink), provided that flow is conserved at each node (for each node, the total incoming flow must equal the total outgoing flow) and the bundle capacities are satisfied (for $1 \leq i \leq l$, the total flow assigned to the edges in S_i must not exceed the bundle capacity c_i). The *max-flow problem with bundle capacities* is defined similarly, except that the flows are allowed to be non-negative real numbers. The integer max-flow problem with bundle capacities has been shown to be *NP*-complete [15]; however, when the integer constraints are relaxed it is well known that the problem can be solved by formulating it as a linear programming problem which we know is polynomially solvable.

As an example, consider the integer max-flow problem defined over the directed graph depicted in Fig. 5. The edges are labeled with their corresponding flows. The collection S of arc bundles is $(\{(\text{source}, v_1), (\text{source}, v_2)\}, \{(\text{source}, v_2), (\text{source}, v_3)\}, \{(\text{source}, v_3), (\text{source}, v_1)\}, \{(v_1, \text{sink}), (v_2, \text{sink})\}, \{(v_2, \text{sink}), (v_3, \text{sink})\}, \{(v_3, \text{sink}), (v_1, \text{sink})\})$ and the collection of the bundle capacities is $C = (1, 1, 1, 1, 1, 1)$.

Clearly the maximum integer flow from source to sink, given that the above bundle capacities are satisfied, is one. The optimal flows are: either $f_0 = f_1 = f_4 = 1$ and all other f_i's zero; $f_0 = f_2 = f_5 = 1$ and all other f_i's zero; or $f_0 = f_3 = f_6 = 1$ and all other f_i's zero. After relaxing the integer constraints (the flows could be any non-negative real numbers) we obtain a max-flow problem with

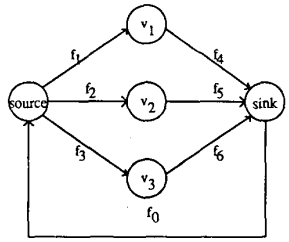


Fig. 5. Max-flow problem with bundle capacities.

maximum flow equal to 3/2. The optimal flow values are: $f_1 = f_2 = f_3 = f_4 = f_5 = f_6 = 0.5$ and $f_0 = 1.5$. Note that the maximum flow in an optimal solution for a max-flow problem is greater than or equal to the maximum flow in an optimal solution for the corresponding integer max-flow problem. The max-flow problem defined above is formulated as a linear programming problem as follows.

maximize f_0
subject to
/* conservation rules */
$f_0 - f_1 - f_2 - f_3 = 0; f_1 - f_4 = 0;$
$f_2 - f_5 = 0; f_3 - f_6 = 0; f_4 + f_5 + f_6 - f_0 = 0;$
/* bundle capacities */
$f_1 + f_2 \le 1; f_2 + f_3 \le 1; f_3 + f_1 \le 1;$
$f_4 + f_5 \le 1; f_5 + f_6 \le 1; f_6 + f_4 \le 1;$
/* flows must be non-negative real values */
$f_i \ge 0$

When there are max selectors, each element in a bundle is a set of edges rather than a single edge, and the restriction is that instead of adding the flow of the single edge, we take the maximum of the flow along the edges in the set. Formally, instead of being given a collection of subset of edges, we are given a collection S of sets whose elements are subsets of edges and a collection of capacities C. For example $S = (\{\{a, b, c\}, \{d, e\}\}, \{\{g, h\}, \{i\}\})$ and $C = (x, y)$. This is interpreted as follows.

$$\max\{f(a), f(b), f(c)\} + \max\{f(d), f(e)\} \le x;$$
$$\text{and } \max\{f(g), f(h)\} + f(i) \le y$$

where $f(x)$ is the flow along edge x. By relaxing the integer constraints, the problem can be formulated as a linear programming problem. In particular, the above constraints are modeled by introducing variables x_1, x_2 and x_3 together with the following inequalities in the linear program.

$$f(a) \le x_1; f(b) \le x_1; f(c) \le x_1$$
$$f(d) \le x_2; f(e) \le x_2$$
$$x_1 + x_2 \le x$$
$$f(g) \le x_3; f(h) \le x_3$$
$$x_3 + f(i) \le y$$
$$f(*) \ge 0$$
$$x_i \ge 0.$$

The *integer linear max-flow problem* is a max-flow problem with max selectors and bundle capacities in which a variable may be used as a bundle capacity. It is simple to show that the noninteger version of this problem can also be formulated as a linear programming problem. That is why we call the problem the linear max-flow problem.

III. Formulation of the (1, 1)-VPD Problem as an Integer Max-Flow Problem with Bundle Capacities

In this section we show that the (1, 1)-VPD problem can be formulated as an integer max-flow problem with bundle capacities. However, finding a solution to the above integral flow problem is computationally difficult. In Section IV, we show that a suboptimal solution to the (1, 1)-VPD problem can be obtained from an optimal solution to the relaxed max-flow problem with bundle capacities, which can be solved in polynomial time. We begin by formulating the (1, 2)-VPD problem as an integer max-flow problem with bundle capacities, and then we show how to formulate the (1, 1)-VPD problem.

Given a (1, 2)-VPD problem, we formulate it as an integer max-flow problem with bundle capacities. That is, given a (1, 2)-VPD problem X we construct an integer max-flow problem Y such that Y has an integer flow from source to sink with value equal to v if and only if X has a placement P with density $d(P) \le d_{\max}$. First let us construct a *flow graph* from the (1, 2)-VPD problem X. There are two special nodes labeled source and sink. For each via column $k (1 \le k \le v)$, we introduce the following set of nodes and edges. The set of nodes is $\{(i, j)_k \mid 1 \le i \le 2, 0 \le j \le n - v\}$. Assume that these nodes are laid along two parallel horizontal lines, with the node $(1, j)_k$ above $(2, j)_k$ and node $(i, j)_k$ to the left of $(i, j + 1)_k$. The (horizontal) edges in the flow graph are directed from node $(i, j)_k$ to node $(i, j + 1)_k$, for $1 \le i \le 2$, and $0 \le j < n - v$. The (vertical) edges are directed from node $(1, j)_k$ to node $(2, j)_k$, for $0 \le j \le n - v$. Two more edges, one from the node labeled source to the node $(1, 0)_k$, and the other from node $(2, n - v)_k$ to the node labeled sink are added to the flow graph. For example, consider the flow graph in Fig. 6 introduced for via column i in the problem given in Fig. 4.

For the moment let us assume that $S = (\{e_1\}, \{e_2\}, \cdots, \{e_s\})$ and $C = (1, 1, \cdots, 1)$, where $E = \{e_1, e_2, \cdots, e_s\}$ is the set of edges in the flow graphs and let $d_{\max} = 3$. Remember that we intend to construct Y (the integer max-flow problem with bundle capacities) in such a way that it has a flow from source to sink equal to v if and only if the VPD problem X has a placement P with density $d(P) \le d_{\max}$. Let us now consider a maximum integer flow for the above integer max-flow problem. From our formulation it is simple to verify that the maximum integer flow for our problem is four. Let us now consider the following optimal integer flow. Assign a flow of one to the vertical edges $((1, 0)_1, (2, 0)_1)$, $((1, 0)_2, (2, 0)_2)$, $((1, 1)_3, (2, 1)_3)$, and $((1, 2)_4, (2, 2)_4)$; a flow of one to all edges emanating from node source and

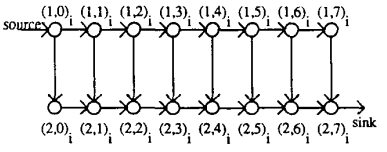


Fig. 6. Flow graph for via column i.

ending at node sink; and a flow of one to all other edges to form a legal flow, i.e., a flow that satisfies the flow conservation rules. Note that there is only one legal flow that satisfies the restrictions imposed above. The interpretation of the individual flows is as follows. The flow graph for via column k has a flow of one from source to sink. From the structure of the flow graph, the restriction that the flows are nonnegative integers and the restriction that all flows have a value of at most one, we know that exactly one vertical edge has a flow of one. Let the edge from vertex $(1, j)_k$ to $(2, j)_k$ be such an edge. This indicates that via column k is to be placed in channel j. The placement, P, generated by the above optimal flow is depicted in Fig. 7. Note that in channel number zero the two via columns are ordered according to their initial column order, i.e., via column 1 is to the left of via column 2.

For many problem instances any optimal integer flow generates a placement with density $\leq d_{\max}$; however, this is not true in general. Consider the following optimal flow. Assign a flow of one to the vertical edges $((1, 7)_1, (2, 7)_1)$, $((1, 7)_2, (2, 7)_2)$, $((1, 6)_3, (2, 6)_3)$, and $((1, 5)_4, (2, 5)_4)$; a flow of one to all edges emanating from source and ending in sink; and a flow of one to all the other edges to form a legal flow. Note that there is only one legal flow that satisfies the restrictions imposed above. The placement, P', generated by this optimal integer flow is depicted in Fig. 8. Clearly, the density in row 4 is greater than $d_{\max}$. Therefore the placement is not valid.

In order for our integer max-flow problem to model the (1, 2)-VPD problem, we need to introduce additional constraints, i.e.; more elements in S. In other words, our modeling so far allows all possible placements and what we need is to eliminate the placements that are not valid. Let us consider the via placement problem with one via column given in Fig. 9(a). The flow graph for it is given in Fig. 9(b). An optimal flow is illustrated in the same figure as follows: thicker edges indicate a flow of one, and thinner edges indicate a flow of zero. The resulting placement following our convention for the flow in the flow graph in Fig. 9(b) is given in Fig. 9(c), where the wiggle lines indicate the regions where the density increases by one because of the placement of the via column. The edges are labeled by the letters a–g and a'–g' in Fig. 9(b). In Fig. 9(d) we show the regions in the placement where the density increases, because of the flow along the edges labeled a–g and a'–g'. Clearly, there is a limit on the amount of density that a region can absorb. In order to facilitate the introduction of the additional constraints we make use of the bundle-capacity graph.

Given a via placement decision problem P, we construct G_v (*bundle-capacity graph*) as follows:

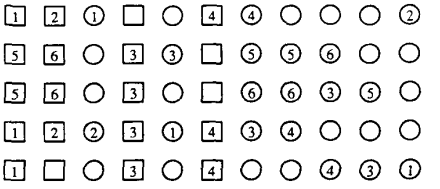


Fig. 7. Placement P.

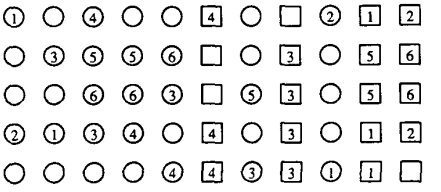


Fig. 8. Placement P'.

The set of vertices is defined by the set of grid points $\{(i, (j, k)) \mid 1 \leq i \leq r, 0 \leq j \leq n - v$, and $0 \leq k \leq v\}$.

The edges in the graph are defined as follows. For $1 \leq i \leq r$, $0 \leq j \leq n - v$, and $0 \leq k < v$, there is a directed edge from vertex $(i, (j, k))$ to vertex $(i, (j, k + 1))$. For $1 \leq i \leq r$ and $0 \leq j < n - v$, there is a directed edge from vertex $(i, (j, v))$ to vertex $(i, (j + 1, 0))$.

Associated with each edge in the bundle-capacity graph, there is an integer (the capacity) and a set of edges from the flow graphs (bundles).

The vertex $(i, (0, 0))$ is introduced for convenience only. The vertex $(i, (j, 0))$ corresponds to the pin located at the intersection of row i with column j. For $1 \leq k \leq v$, the column $(*, (j, k))$ indicates the location that via column k will occupy if it is placed in channel j. Let $d(i, j)$ = the current density (i.e., considering only the pin columns) of row i after column j and before column $j + 1$, for $1 \leq j < n - v$, and let $d(i, 0) = d(i, n - v) = 0$. The capacity associated with the edge whose tail is located at vertex $(i, (j, k))$ is $d_{\max} - d(i, j)$. Later on, we add a set of edges from the flow graphs to the bundle associated with each edge. The set of additional constraints is defined as follows. For each edge in the bundle-capacity graph, the sum of the flow for the set of edges in the flow graphs assigned to the bundle is less than or equal to the capacity for that edge in the bundle-capacity graph.

The bundle-capacity graph for the via placement problem given in Fig. 4 with $d_{\max} = 3$ and without including the bundles is given by Fig. 10. The capacity of an edge in the bundle-capacity graph indicates that the density between two points may be increased by at most that amount.

Let us now add edges from the flow graphs to each bundle (associated with an edge) in the bundle-capacity graph. Consider via column k with its flow graph defined over the vertices $(1, j)_k$ and $(2, j)_k$, for $0 \leq j \leq n - v$. The start (end) point $s_i(e_i)$ is the leftmost (rightmost) pin that needs to be connected to the via on the ith row of via column k. By assumption such points exist (see comment just after the definition of the VPO problem in Section I).

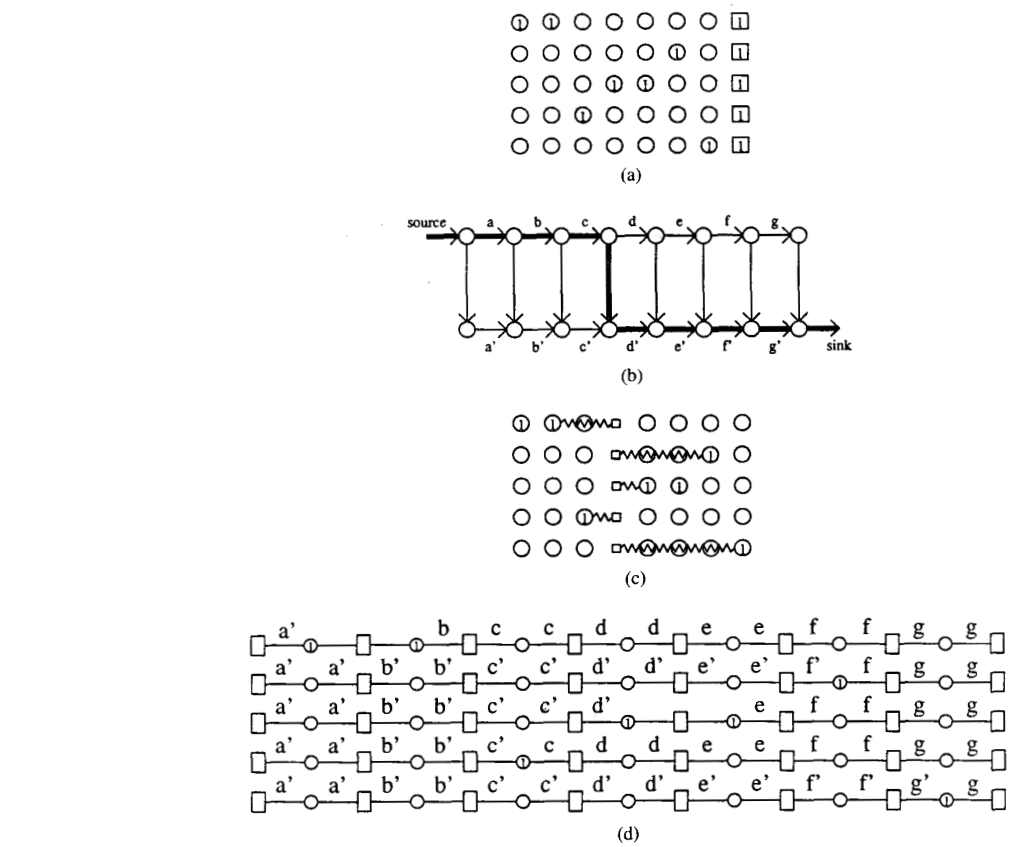


Fig. 9. (a) Via placement decision problem. (b) Flow graph for via column. (c) Placement for (a) obtained from the flow in (b). (d) Edges from the flow graph included in the bundles.

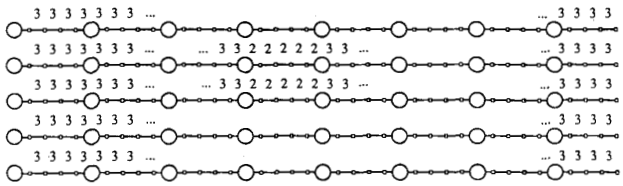
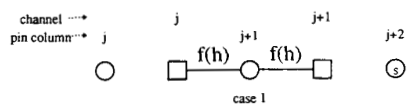


Fig. 10. Bundle-capacity graph constructed for the problem given in Fig. 3 with $d_{max} = 3$.

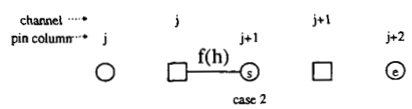
Let g represent edge $((1, j)_k, (1, j + 1)_k)$ and let h represent the edge $((2, j)_k, (2, j + 1)_k)$. We use $f(x)$ to represent the flow along edge x (see Fig. 11).

Clearly, if $f(g) = 1$, then via column k is not placed to the left of pin column $j + 1$ (i.e., it is not placed on channels $1, 2, \cdots, j$); and if $f(h) = 1$, then via column k is not placed to the right of pin column $j + 1$ (i.e., it is not placed on channels $j + 1, j + 2, \cdots$). By construction, either $f(g)$ is one or $f(h)$ is one, but not both. Now we determine in which edge (whose endpoints are vertices in the set $\{(i,(j, k)), \cdots, (i, (j + 1, k))\}$) bundle in the bundle-capacity graph these edges must be included. This decision is based on the location of s_i and e_i. There are six cases that need to be considered, depending on the values of s_i and e_i.

Case 1: $j + 2 \leq s_i \leq e_i$: In this case h is added to the bundle associated with the edges whose endpoints are vertices in the set $\{(i, (j, k)), \cdots, (i, (j + 1, k))\}$. That is, the density in the region from vertex $(i, (j, k))$ to vertex $(i, (j + 1, k))$ will be increased by one only when via column k is placed to the left of channel $j + 1$.

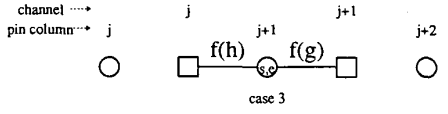


Case 2: $j + 1 = s_i < e_i$: In this case h is added to the bundle associated with the edges whose endpoints are located in the set $\{(i, (j, k)), \cdots, (i, (j + 1, 0))\}$. That is, the density in the region from vertex $(i, (j, k))$ to vertex $(i, (j + 1, 0))$ will be increased by one only when via column k is placed to the left of channel $j + 1$.

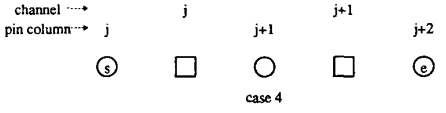


Case 3: $j + 1 = s_i = e_i$: In this case h is added to the bundle associated with in the edges whose endpoints are located in the set $\{(i, (j, k)), \cdots, (i, (j + 1, 0))\}$,

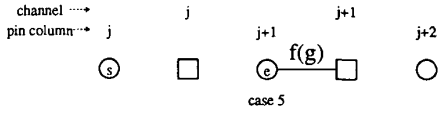
and g is added to the bundle associated with the edges whose endpoints are located in the set $\{(i, (j + 1, 0)), \cdots, (i, (j + 1, k))\}$. That is, the density in the region from vertex $(i, (j, k))$ to vertex $(i, (j + 1, 0))$ will be increased by one only when via column k is placed to the left of channel $j + 1$, and the density in the region from vertex $(i, (j + 1, 0))$ to vertex $(i, (j + 1, k))$ will be increased by one only when via column k is placed to the right of channel j.



Case 4: $s_i < j + 1 < e_i$*:* In this case neither g nor h is added to any bundle.



Case 5: $s_i < j + 1 = e_i$*:* In this case g is added to the bundle associated with in the edges whose endpoints are located in the set $\{(i, (j + 1, 0)), \cdots, (i, (j + 1, k))\}$. That is, the density in the region from vertex $(i, (j + 1, 0))$ to vertex $(i, (j + 1, k))$ will be increased by one only when via column k is placed to the right of channel j.



Case 6: $s_i \leq e_i \leq j$*:* In this case g is added to the bundle associated with in the edges whose endpoints are located in the set $\{(i, (j, k)), \cdots, (i, (j + 1, k))\}$. That is, the density in the region from vertex $(i, (j, k))$ to vertex $(i, (j + 1, k))$ will be increased by one only when via column k is placed to the right of channel j.

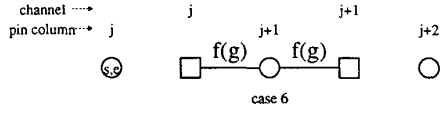


Fig. 12 shows some of the edges in the bundle-capacity graph where edges from the flow graphs have been added to their bundles. The new constraints are defined for each edge in the bundle-capacity graph as the sum of the flows for the edges in the bundle must not exceed the capacity. After adding these constraints it is simple to prove Lemma 2.1.

Lemma 2.1: The (1, 2)-VPD problem X has a placement P with density $d(P) \leq d_{max}$ iff the integer max-flow problem with bundle capacities Y constructed from it has a flow from source to sink equal to v.

Proof: By the above discussion. □

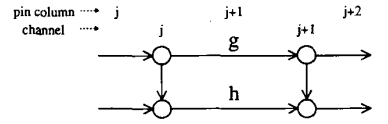
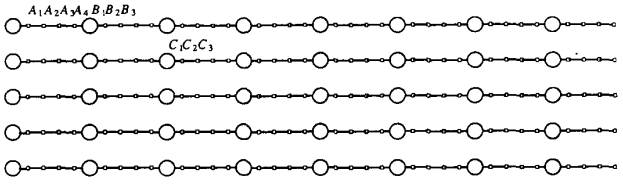


Fig. 11. Flow graph.



Examples of Bundles

$A_1 = \{((2, 0)_1, (2, 1)_1)\}$

$A_2 = \{((2, 0)_1, (2, 1)_1), ((2, 0)_2, (2, 1)_2)\}$

$A_3 = \{((2, 0)_1, (2, 1)_1), ((2, 0)_2, (2, 1)_2)\}$

$A_4 = \{((2, 0)_1, (2, 1)_1), ((2, 0)_2, (2, 1)_2), ((2, 0)_4, (2, 1)_4)\}$

$B_1 = \{((1, 0)_1, (1, 1)_1), ((2, 0)_2, (2, 1)_2), ((2, 0)_4, (2, 1)_4)\}$

$B_2 = \{((1, 1)_1, (1, 2)_1), ((2, 0)_2, (2, 1)_2), ((2, 0)_4, (2, 1)_4)\}$

$B_3 = \{((1, 1)_1, (1, 2)_1), ((2, 1)_2, (2, 2)_2), ((2, 0)_4, (2, 1)_4)\}$

$C_1 = \{((2, 1)_1, (2, 2)_1), ((2, 1)_2, (2, 2)_2), ((1, 1)_3, (1, 2)_3)\}$

$C_2 = \{((2, 2)_1, (2, 3)_1), ((2, 1)_2, (2, 2)_2), ((1, 1)_3, (1, 2)_3)\}$

$C_3 = \{((2, 2)_1, (2, 3)_1), ((2, 2)_2, (2, 3)\quad ((1, 1)_3, (1, 2)_3)\}$

Fig. 12. Examples of bundles for the (1, 3)-PVD problem constructed for the instance given in Fig. 3.

Now let us relax (*R*2), the "ordering" assumption when placing more than one via column in the same channel. That is, there might not be an optimal placement with the property that all the via columns assigned to the same channel are placed in order of their initial column index. Remember that we refer to this problem as the (1, 1)-VPO or the (1, 1)-VPD problem. To deal with this situation we modify the flow graphs and the bundle-capacity graph by introducing additional nodes and edges.

The new flow graphs are constructed as follows. There are two special nodes labeled source and sink. For each via column k we introduce the following set of nodes and edges. The set of vertices is $\{(1, (j, t))_k, (2, (j, t))_k \mid 0 \leq j \leq n - v$, and $1 \leq t \leq v\}$. Assume that these nodes are laid along two parallel horizontal lines, with node $(1, (j, t))_k$ above $(2, (j, t))_k$, and node $(i, (j, t))_k$ to the left of node $(i, (j', t'))_k$ if $(j - 1)*v + t < (j' - 1)*v + t'$. The (horizontal) edges are directed from vertex $(i, (j, t))_k$ to vertex $(i, (j, t + 1))_k$, for $1 \leq i \leq 2$, $0 \leq j \leq n - v$ and $1 \leq t < v$. There are also (horizontal) edges directed from vertex $(i, (j, v))$ to vertex $(i, (j + 1, 1))$, for $1 \leq i \leq 2$ and $0 \leq j < n - v$. The (vertical) edges are directed from vertex $(1, (j, t))_k$ to vertex $(2, (j, t))_k$, $0 \leq j \leq n - v$ and $1 \leq t \leq v$. Two more edges, one from the vertex labeled source to vertex $(1, (0, 1))_k$, and the other from vertex $(2, (n - v, v))_k$ to the vertex labeled sink are added to the flow graph. The meaning associated with the new flow graph is the following. If the vertical edge with flow of one is the edge oriented from node $(1, (j, t))_k$ to node $(2, (j, t))_k$, then via column k is placed on channel j in its tth position. As we shall see

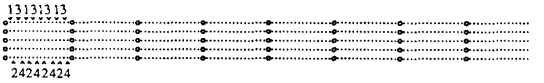


Fig. 13. Bundle-capacity graph for the (1, 2)-VPD problem.

later each via column has v positions where it can be placed in channel j.

The bundle-capacity graph G_v is now defined as follows:

The set of vertices is defined by the set of grid points $\{(i, (j, k)) \mid 1 \leq i \leq r, 0 \leq j \leq n - v$, and $0 \leq k \leq v^2\}$.

The edges in the graph are defined as follows. For $1 \leq i \leq r$, $0 \leq j \leq n - v$, and $0 \leq k < v^2$, there is an edge from vertex $(i, (j, k))$ to vertex $(i, (j, k + 1))$. For $1 \leq i \leq r$ and $0 \leq j < n - v$, there is a directed edge from vertex $(i, (j, v^2))$ to vertex $(i, (j + 1, 0))$.

Associated with each edge, there is an integer (the capacity) and a set of edges from the flow graphs (bundles).

The vertex $(i, (0, 0))$ is introduced for convenience only. The vertex $(i, (j, 0))$ corresponds to the pin located at the intersection of row i with column j. For $1 \leq k \leq v$ and $1 \leq l \leq v$, the column $(*, (j, (l - 1)v + k))$ indicates the location that via column k occupies if it is assigned to its lth place in channel j. Note that now there are v different locations in a channel where each via column can be placed. This allows arbitrary placement order for the via columns in each channel. Let $d(i, j)$ be the current density (i.e., considering only the pin columns) of row i after column j and before column $j + 1$, for $1 \leq j < n - v$, and let $d(i, 0) = d(i, n - v) = 0$. The capacity for the edge with tail at vertex $(i, (j, k))$ is $d_{\max} - d(i, j)$. The set of additional constraints is defined as follows. For each edge in the bundle-capacity graph, the sum of the flows for the set of edges in the flow graphs assigned to the bundle is less than or equal to the capacity for that edge in the bundle-capacity graph. Fig. 13 shows the nodes introduced for the problem given in Fig. 4.

The edges in the flow graph that are associated with each bundle in the bundle-capacity graph are defined by following a procedure similar to the one for the (1, 2)-VPD problem. From our construction rules it is simple to prove the following lemma.

Lemma 2.2: The (1, 1)-VPD problem X has a placement P with density $d(P) \leq d_{\max}$ iff the integer max-flow problem with bundle capacities constructed from it has a flow from source to sink equal to v.

Proof: By the above discussion. □

IV. Suboptimal Solution for the (1, 1)-VPO Problem

In the previous section, we presented a method for solving the (1, 1)-VPD problem by reducing it to the integer max-flow problem with bundle capacities. Since this integer max flow problem is computationally intractable, we relax the integer constraints by allowing real numbers as flows. In Section II we showed how such a problem can be solved in polynomial time by formulating it as a linear programming problem. In this section we show how to generate from the solution to the max-flow problem with bundle capacities a suboptimal solution for the (1, 1)-VPO problem. The suboptimal solution has density less than or equal to $2 * d_{\text{OPT}}$, where d_{OPT} is the density of an optimal solution.

Our procedure is defined as follows. The solution to the (1, 1)-VPO problem is obtained by solving a set of (1, 1)-VPD problems. Each (1, 1)-VPD problem is formulated as an integer max-flow problem with bundle capacities. The integer constraints in the max-flow problem are relaxed and the problem is formulated as a linear programming problem and solved. If the (1, 1)-VPD problem has a yes answer, then the relaxed max-flow problem has a flow from source to sink equal to v; however, when the (1, 1)-VPD problem has a no answer, the relaxed max-flow problem may or may not have a flow from source to sink equal to v. In this section we prove that when the relaxed max-flow problem constructed from a (1, 1)-VPD problem (with $d_{\max} = d$) has a flow from source to sink equal to v, then the via placement problem has a placement P with density $d(P) \leq 2 * d$ and such a placement can be easily obtained from an optimal flow in the relaxed max-flow problem. Therefore, after a logarithmic search, we obtain a placement with density $d(P) \leq 2 * d$ and we know that there is no placement for the via placement problem with density less than d.

Let us now show that when the relaxed max-flow problem constructed from a (1, 1)-VPD problem (with $d_{\max} = d$) has a flow from source to sink equal to v, then the via placement problem has a placement with density $2 * d$ and such a placement can be easily obtained from an optimal flow in the relaxed max-flow problem. If the optimal flow obtained has the property that all the flow values are integers, then we are done, and we have a placement with density $d = d_{\text{OPT}}$. In general this is not going to be the case. For example consider the flow graph for via column k, given in Fig. 14 (the flow values are assigned to the edges in the flow graphs).

If we follow our previous interpretation it suggests that fractions of the via column are to be placed at different locations, this is why we call it a *pseudo-placement*. This is not allowed in our via placement problem. However, from the structure of the flow graph we know that there is at least one path from the source to the sink that goes through the flow graph for via column k with the property that each horizontal edge has a flow greater than or equal to 0.5. Any one of these paths is selected to generate our solution to the via placement problem, i.e., the via column will be placed at the position indicated by the vertical edge in this path.

Now the question is what is the density of the placement obtained this way? The answer is simple. With the non-integral flow, the density of the pseudo-placement

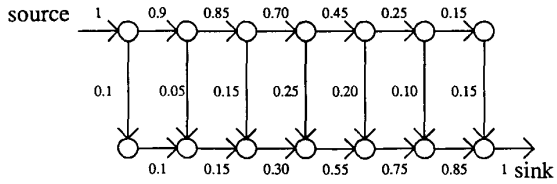


Fig. 14. Nonintegral solution for the flow graph of via column k.

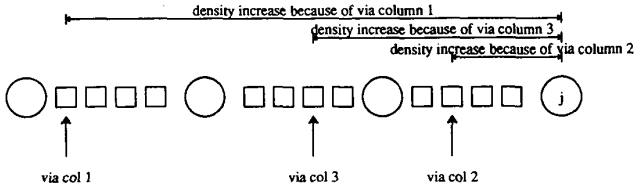


Fig. 15. Placement.

was at most d. Since we have at most doubled the flow along each edge in a bundle and each wire in the final layout is represented by a flow in a bundle, then the density of the placement that we construct is at most $2 * d \leq 2 * d_{\text{OPT}}$.

Lemma 4.1: Our algorithm generates a placement for the (1, 1)-VPO problem with density at most $2 * d_{\text{OPT}}$.

Proof: By the above discussion. □

V. Approximation Algorithm for the VPO Problem

In the previous sections we presented an algorithm that generates an approximate solution to the (1, 1)-VPO problem with a density which is at most two times the density in an optimal solution. However, we assumed that the via placement problem satisfies restriction $R1$. In this section we show how to generate a suboptimal solution for the unrestricted via placement problem. Restriction $R1$ can be eliminated by introducing max selectors. These max selectors will not affect our approximation bound, they will just increase the time complexity bound. Remember that $R1$ is the restriction that if a via in row i is assigned to net j, then no other via assigned to net j is located in row i.

Let us now explain how to solve the VPD problem by introducing max selectors. Suppose that three via columns have a via assigned to the same net N_j on some row i. Assume that our previous algorithm generates the placement given by Fig. 15, where the node labeled j is the leftmost pin for net N_j.

Note that the density increases by three in some regions in Fig. 16 because of net N_j. This is incorrect, it should be at most one in all these regions. This problem can be handled by selecting the maximum of the three flows. Our rule to handle this situation is defined as follows: whenever we deal with a bundle associated with an edge in the bundle-capacity graph, we find all the flows that are generated because of the same net and then compute the max of all of them. Hence the use of max selectors. It is simple to verify that restriction $R1$ can be eliminated by the introduction of appropriate max selectors.

Theorem 5.1: The VPD problem X has a placement P with density $d(P) \leq d_{\max}$ iff the integer max-flow problem with max selectors and bundle capacities Y constructed from it has a flow from source to sink equal to v.

Proof: By the above discussion. □

Using the techniques discussed in the previous section, we have an approximation algorithm for the VPO problem. With the non-integral flow, the density of the pseudo-placement was at most d. Since we have at most doubled the flow along each edge in a bundle and each wire in the final layout is represented by a flow in a bundle, then the density of the placement that we construct is at most $2 * d \leq 2 * d_{\text{OPT}}$.

Theorem 5.2: Our algorithm generates a placement for the VPO problem with density at most $2 * d_{\text{OPT}}$.

Proof: By the above discussion. □

VI. Discussion

It is simple to verify that the linear programming problems required to solve the VPO problem have a number of variables and equations that are bounded above by a polynomial in r and n. Also, the amount of space required to represent each linear programming problem is bounded above by a polynomial in r and n. Since the more recent algorithms that solve linear programming problems take time which is bounded by a polynomial in the problem size, we know that the total time complexity for our algorithm is polynomial. For brevity, we do not include a sharper time complexity bound.

Our algorithms can be trivially modified so that instead of solving $\log(v)$ LP problems, one needs to solve only one LP problem. This is accomplished by placing the objective function as a constraint that must equal v, and minimizing the required density $d_{\max}$. Note that many inequalities need to be reformulated in terms of the computed density.

In the introduction we mentioned that the VPO problem in which we allow a via from net j to be assigned to row i even though there is no pin from net j in row i is not interesting. However, this problem is important because a good approximation algorithm for it implies a good approximation algorithm for the board permutation problem, the min-cut linear arrangement problem, and the column permutation problem.

Another important feature of our algorithm is that it can be easily modified so that via columns are not placed in certain channels. This is desirable when the channel is in the middle of a chip. Note that in this case it is impossible to move only half of the chip.

Acknowledgment

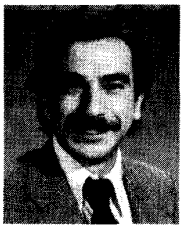
The authors would like to thank Professor Sing-Ling Lee and Ms. Connie Pun for their valuable comments and suggestions concerning this paper.

References

[1] J. Bhasker and S. Sahni, "Via assignment in single row routing," in *Proc. Foundations of Software Technology and Theoretical Computer Science*, pp. 154–176, Dec. 1986.

[2] J. Cohoon and S. Sahni, "Heuristics for the board permutation problem," in *Proc. 1983 IEEE ICCAD Conf.*, pp. 81–83.
[3] M. Garey and D. Johnson, "The complexity of near-optimal graph coloring," *J. ACM*, vol. 23, no. 1, pp. 43–49, Jan. 1976.
[4] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: Freeman, 1979.
[5] S. Gass, *Linear Programming*. New York: McGraw Hill, 1969.
[6] S. Goto, I. Cederbaum, and B. Ting, "Suboptimum solution to the back-board ordering with channel capacity constraint," *IEEE Trans. Circuits Syst.*, vol. CAS-24, pp. 645–652, Nov. 1977.
[7] T. Gonzalez, "An approximation algorithm for the via assignment problem," *IEEE Trans. Computer-Aided Design*, vol. CAD-3, pp. 257–264, Oct. 1984.
[8] T. Gonzalez and S. Kurki-Gowdara, "An efficient algorithm to minimize the number of layers in single row routing problems with fixed street capacity," *IEEE Trans. Computer-Aided Design*, vol. 7 pp. 420–424, Mar. 1988.
[9] S. Y. Han and S. Sahni, "Layering algorithms for single row routing," in *Proc. 22nd Design Automation Conf.*, pp. 516–522, 1984.
[10] S. Han and S. Sahni, "Single row routing on narrow streets," *IEEE Trans. Computer-Aided Design*, vol. CAD-5, pp. 235–241, July 1986.
[11] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica*, vol. 4, pp. 373–395, 1984.
[12] L. G. Khachiyan, "A Polynomial Algorithm in Linear Programming," Doklady Akademiia Nauk SSSR, 244:S, pp. 1093–1096, 1979, translated in Soviet Mathematics Doklady, 20:1, pp. 1–12, 1979.
[13] E. Kuh, T. Kashiwabara, and T. Fujisawa, "On optimal single-row routing," *IEEE Trans. Circuits Syst.*, vol. CAS-26, pp. 361–368, June 1979.
[14] R. Raghavan and S. Sahni, "Optimal single row router," *IEEE Trans. Computers*, vol. C-32, pp. 209–220, Mar. 1983.
[15] S. Sahni, "Computationally Related Problems," *SIAM J. Computing*, vol. 3, no. 4, pp. 262–279, Dec. 1974.
[16] S. Sahni, A. Bhatt, and Raghavan, R., "The complexity of design automation problems," in *Proc. 19th Design Automation Conf.*, June 1981.
[17] H. So, "Some theoretical results on the routing of multilayer printed-wiring boards," in *Proc. IEEE Symp. on Circuits and Systems*, pp. 296–303, 1974.
[18] B. Ting, E. Kuh, and A. Sangiovanni-Vincentelli, "Via assignment problem in multilayer printed circuit board," *IEEE Trans. Circuits Syst.*, vol. CAS-26, pp. 261–271, Apr. 1979.
[19] T. Tarng, M. Marek-Sadowska, and E. S. Kuh, "An efficient single-row routing algorithm," *IEEE Trans. Computer-Aided Design*, vol. CAD-3, pp. 178–183, July 1983.
[20] S. Tsukiyama, I. Shirakawa and S. Asahara, "An algorithm for the via assignment problem in multilayer backboard wiring," *IEEE Trans. Circuits and Syst.*, vol. CAS-26, pp. 369–377, June 1979.
[21] S. Tsukiyama, E. S. Kuh, and I. Shirakawa, "On the layering problem on multilayer PWB wiring," *IEEE Trans. Computer-Aided Design*, vol. CAD-2, pp. 30–38, January 1983.
[22] P. M. Vaidya, "An algorithm for linear programming which requires $O(((m + n)n^2 + (m + n)^{1.5}n))$ arithmetic operations," in *Proc. 19th Annual ACM Symp. Theory of Computation*, pp. 29–38, May 1987.

*

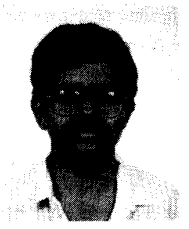


Teofilo F. Gonzalez received the B.S. degree in computer science from the Instituto Technologico de Monterrey and the Ph.D. degree in computer science from the University of Minnesota, Minneapolis, in 1972 and 1975, respectively.

He has been with the University of Oklahoma, the Pennsylvania State University, the Instituto Technologico de Monterrey and the University of Texas at Dallas. He is now Professor of Computer Science at the University of California, Santa Barbara. His major research interests are in the design and analysis of algorithms, computational aspects of computer-aided design, and scheduling theory.

Dr. Gonzalez is a member of the Association for Computing Machinery, and the Operations Research Society of America.

*



Shashishekhar Kurki-Gowdara received the B.E. degree in electrical and electronics engineering from Bangalore University, India, and the M.S. degree in Computer Science from the University of California at Santa Barbara, in 1980 and 1982, respectively. He is currently studying for the Ph.D. degree in Computer Science at the University of California at Santa Barbara.

His research interests include design and analysis of algorithms, VLSI routing and placement algorithms, and computability theory.

Mr. Kurki-Gowdara is a member of the Association for Computing Machinery and the IEEE Computer Society.