VIA ASSIGNMENT IN SINGLE ROW ROUTING*

Jayaram Bhasker+ and Sartaj Sahni

University of Minnesota

ABSTRACT

We examine the via assignment problem that arises when the single row routing approach to the interconnection problem is used. Some new complexity results and two new heuristics are obtained. Experimental results establish the superiority of the new heuristics over earlier ones.

Keywords and Phrases

Single row routing, via assignment, complexity, heuristic.

* This research was supported in part by the National Science Foundation under grant MCS-83-05567.
+ Dr Jayaram Bhasker is currently with Computer Sciences Center, Honeywell Inc., 1000 Boone Ave North, Golden Valley, MN 55427.

1 SINGLE ROW ROUTING

The single row routing approach to the interconnection problem for multilayer printed circuit boards was proposed by So, [SO74]. In this approach, it is assumed that the pins and vias lie on the grid points of a unit grid (Figure 1.1). The vias occupy some number of columns on the right end of the board while the pins occupy the remaining columns. We shall use the term *point* to refer to both a via and a pin.

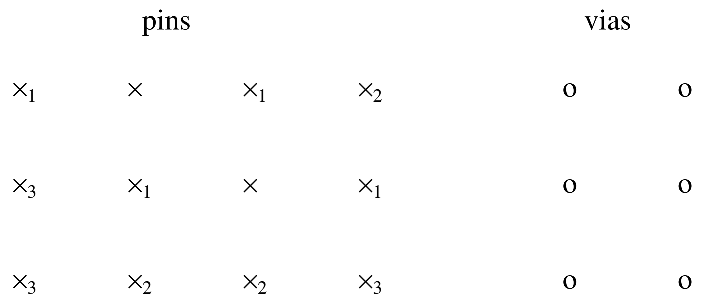

pins
vias
$\times_1$ $\times$ $\times_1$ $\times_2$ o o
$\times_3$ $\times_1$ $\times$ $\times_1$ o o
$\times_3$ $\times_2$ $\times_2$ $\times_3$ o o

Figure 1.1

The required interconnections are specified by means of a net list $N = \{N_1, N_2, ..., N_k\}$, where N_i is a subset of the pins and the N_i´s are pairwise disjoint. The single row routing approach imposes three restrictions on the routings that realize the net list N. These are :

(1) Each wire connects a pair of points that are either on the same row or on the same column.

(2) The layout of a wire connecting two points on the same row (column) is confined to the two wiring channels on either side of the row (column). Figure 1.2 shows such an example.

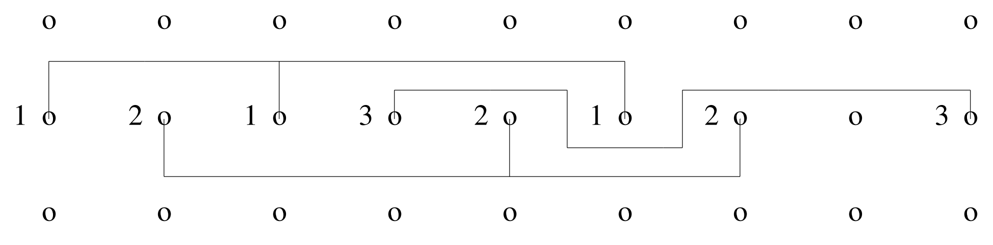


Figure 1.2: Single row routing.

(3) Each routing layer contains wires for only row connections or only column

connections.

The single row routing approach is traditionally implemented to comprise the following steps:

a) Via assignment -

In this step, each net is decomposed into a series of row and column interconnections. This is done using vias as necessary. Figure 1.3 shows the decomposition for the net list of Figure 1.1.

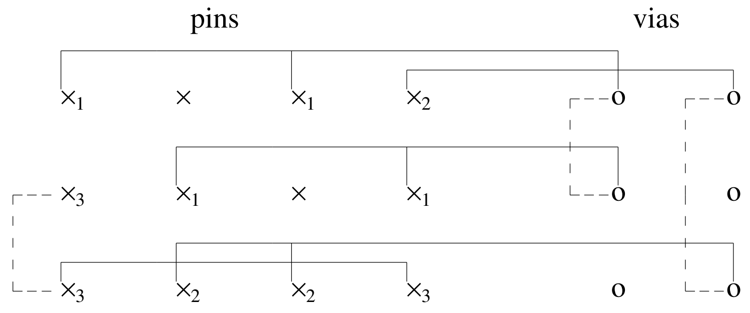


Figure 1.3: Via assignment of net list of Figure 1.1.

b) Via column permutation -

The via columns are permuted so as to minimize the number of connections that cross between any pair of adjacent via columns.

c) Layering -

The row (column) interconnects are partitioned into layers for realization.

d) Wire layout -

The row (column) interconnects assigned to each layer are laid out.

Each of the above four steps has recieved much attention in the literature. The via assignment problem has been studied in [TING79], [TSUK79], [GONZ83] and [RAGH84]. [TSUK79] has a slightly different model for the placement of vias. The via column permutation problem is identical to the board permutation problem. Two references are [COHO83] and [GOTO77]. The layering problem has been reported on in [TSUK83] and [HAN84a]. The wire layout problem has recieved maximum attention. Some references are [TING76], [RAGH83], [HAN84b], [HAN84c] and [TARN84].

In this paper, we shall be concerned with the via assignment problem only.

2 THE VIA ASSIGNMENT PROBLEM

Define a *g-node* to be a maximal subset of pins in a net that can be connected by row and column interconnects without using any vias. Figure 2.1 identifies the g-nodes of a sample net. It is easy to see that the g-nodes for any net are unique and may be determined quite easily.

The decomposition of the net list into a series of row and column interconnects is done by first obtaining the g-nodes for all the nets. Nets that have two or more g-nodes utilize vias to

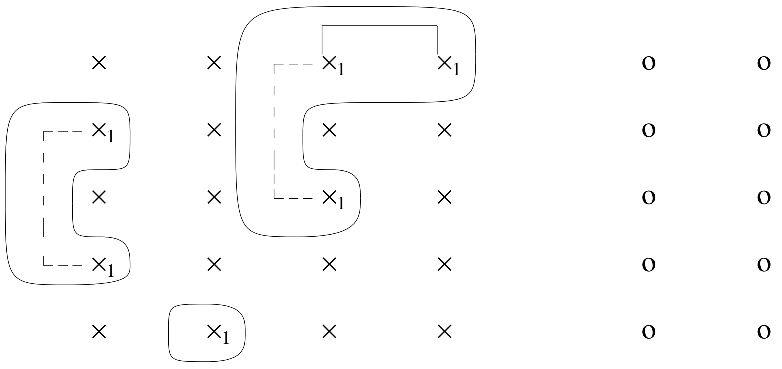


Figure 2.1: G-nodes.

connect the g-nodes together. The *via assignment problem* is that of assigning vias to nets so that all g-nodes of each net may be connected using row and column interconnects alone. It is desirable to accomplish this assignment in such a way so as to use the smallest number of via columns. The reason for this is that the size of the circuit board depends on the number of via columns. It is also desirable to use the smallest number of vias as vias generally cause a reliability problem.

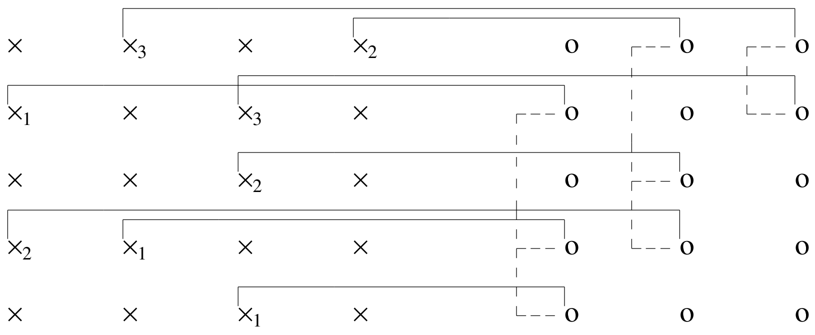


Figure 2.2: Decomposition of a net list instance when *RVIACOL* is used.

The via assignment problem has been studied under two models :

(1) *RVIACOL (restricted via column minimization)* -

In this model, the vias used to realize any one net must all come from the same via column. Figure 2.2 shows an optimal (i.e. least via columns) net decomposition using this model.

(2) *VIACOL* -

In this model, the vias used for any one net may be taken from more than one via column. Figure 2.3 shows an optimal decomposition for the nets of Figure 2.2 when this model is used.

The restricted via column minimization problem was shown to be NP-hard in [TING79] and [TSUK79]. A stronger result is obtained in [RAGH84]. Raghavan and Sahni [RAGH84] have shown that in the restricted model, deciding whether or not two via columns are sufficient, is NP-complete. They have also extended this result to the unrestricted model. A similar result for the unrestricted model appears in [GONZ83].

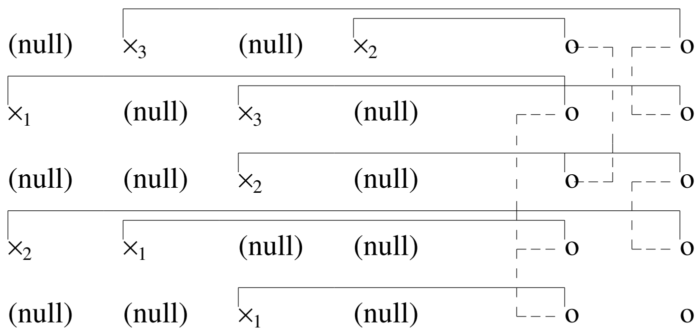

(null) x_3 (null) x_2 o o
x_1 (null) x_3 (null) o o
(null) (null) x_2 (null) o o
x_2 x_1 (null) (null) o o
(null) (null) x_1 (null) o o

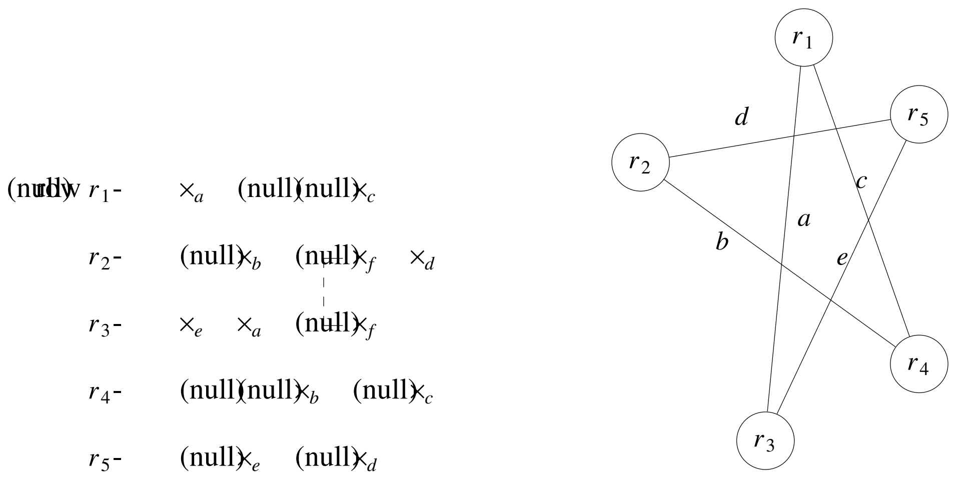
Figure 2.3: Decomposition of the instance of Figure 2.2 when *VIACOL* is used.

3 NEW COMPLEXITY RESULTS

The NP-complete(hard) results of [TING79], [TSUK79], [RAGH84] and [GONZ83] involve constructions in which nets may contain an arbitrarily large number of pins. This leaves open the question of whether the via assignment problem (whether restricted or not) remains difficult when the nets are constrained to contain a small number of pins. We examine this problem in this section.

Theorem 3.1: Let L be a net list in which each net consists of exactly two pins.
(a) One can determine in polynomial time whether or not two via columns suffice under the restricted model.
(b) Determining whether or not three via columns suffice under this model is NP-complete.
Proof:
(a) We construct a graph, G, that corresponds to L and the underlying grid. G contains one vertex for each row of the grid. The 2-pin nets in L are considered one by one. If the two pins in a net are in the same row or column, we disregard the net. Otherwise, draw an edge in G between the vertices representing the two distinct rows in which the two pins lie. Figure 3.1 gives an example construction.

It is easy to see that the minimum number of via columns needed for L is equal to the minimum number of colors needed to edge color G (note that in an edge coloring, edges incident on the same vertex must be assigned different colors). Let the minimum number of colors (and hence the minimum number of via columns) be *k*.


(null)row r_1- ×$_a$ (null)(null)×$_c$
r_2- (null)×$_b$ (null)×$_f$ ×$_d$
r_3- ×$_e$ ×$_a$ (null)×$_f$
r_4- (null)(null)×$_b$ (null)×$_c$
r_5- (null)×$_e$ (null)×$_d$
r_1
r_2
r_3
r_4
r_5
a
b
c
d
e

(a) Net list, L (b) Corresponding graph, G

Figure 3.1: Construction of graph, G, from 2-pin nets in L.

k is one iff no vertex in G has degree more than 1. If G has a vertex of degree > 2, then $k > 2$. If every vertex is of degree ≤ 2, then $k \le 2$ iff G contains no cycles of odd length (i.e. G is bipartite). Hence, the cases $k = 1$ and $k = 2$ may be detected in polynomial time.

(b) The case $k = 3$ is NP-complete. This follows from the result of [HOLY81] that 3-coloring the edges of a graph is NP-complete. To see this, let G = (V, E) be an arbitrary graph. We shall construct an instance *I*, of the via assignment problem that has the following properties :

(i) Each net in *I* has exactly two pins.

(ii) Three via columns suffice for *I* iff G can be 3-colored.

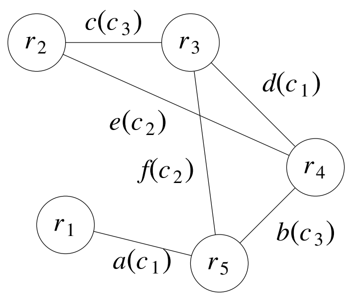
(iii) *I* can be constructed from G in polynomial time.

For each vertex $v \in$ V, *I* has a distinct grid row. For each edge $(u, v) \in$ E, define a two terminal net with pins on the rows corresponding to *u* and *v*. An example is given in Figure 3.2.

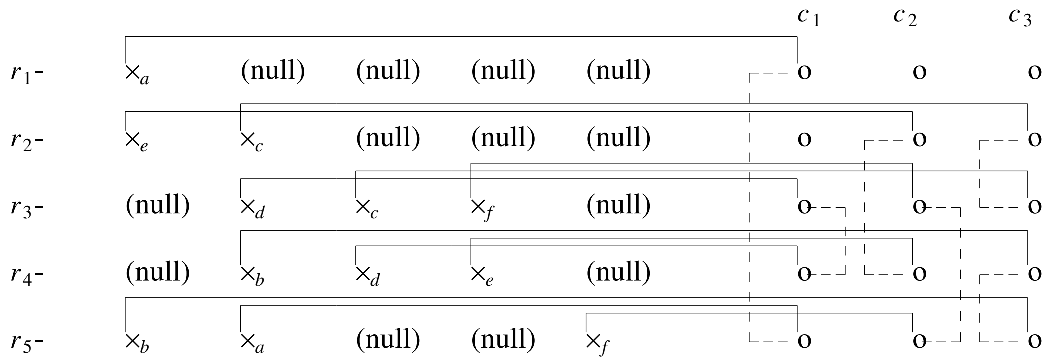
Clearly, three colors suffice for the edges of G iff three via columns suffice for the nets in *I*. □

Theorem 3.2: Let L be a net list in which each net consists of exactly two pins.
(a) There is a polynomial time algorithm to determine whether or not two columns suffice under the unrestricted model.
(b) Determining whether or not three via columns suffice is NP-complete.

Proof:
(a) From L, we obtain the graph G as in the proof of Theorem 3.1(a). If G has any vertex of degree ≥ 3, then two columns are not sufficient. So, assume that the degree of every vertex in G is ≤ 2. Under this assumption, G consists of a unique set of maximal paths and cycles. For each cycle of odd length, a vertex of degree zero is needed (see Example 3.1). Hence, two columns suffice iff the number of odd cycles does not exceed the number of vertices of zero degree. This is



(a) Graph, G = (V, E)


c_1 c_2 c_3
r_1- $\times_a$ (null) (null) (null) (null) o o o
r_2- $\times_e$ $\times_c$ (null) (null) (null) o o o
r_3- (null) $\times_d$ $\times_c$ $\times_f$ (null) o o o
r_4- (null) $\times_b$ $\times_d$ $\times_e$ (null) o o o
r_5- $\times_b$ $\times_a$ (null) (null) $\times_f$ o o o

(b) Corresponding instance, *I*

Figure 3.2: Construction of a net list, *I*, from the graph, G.

easily determined in polynomial time.

(b) This again, follows from [HOLY81]. The graphs constructed in [HOLY81] are cubic (i.e. every vertex has degree 3). Hence, the number of vertices is even (note that there are no cubic graphs with an odd number of vertices as the number of edges in such a graph is $3n/2$, where $n = |V|$). We use the same construction as in part (b) of Theorem 3.1. Since G is cubic, each row of the constructed instance *I* has three pins. The number of 2-pin nets is $|E| = 3|V|/2$. The minimum number of vias needed is $3|V|$ which is the total available in three via columns. This number suffices only if no net is assigned vias from different columns. Hence, for instances obtained from cubic graphs, the three via column problem is the same under both the restricted and unrestricted models. Consequently, there is an unrestricted three via column assignment iff there is a restricted three via column assignment iff the edges of the original cubic graph G are 3-colorable. Hence, the unrestricted three via column assignment problem for 2-pin nets is NP-complete. □

Example 3.1: Consider the net list of Figure 3.3(a). The corresponding graph is shown in Figure 3.3(b) and its unique paths and cycles are shown in Figure 3.3(c).

By 2-coloring the edges on the paths and even cycles, a via assignment for the corresponding nets is obtained. For the odd cycles, a via assignment is possible only if a vertex of degree zero is available. Using such a vertex, an odd cycle can, in effect, be transformed into an even

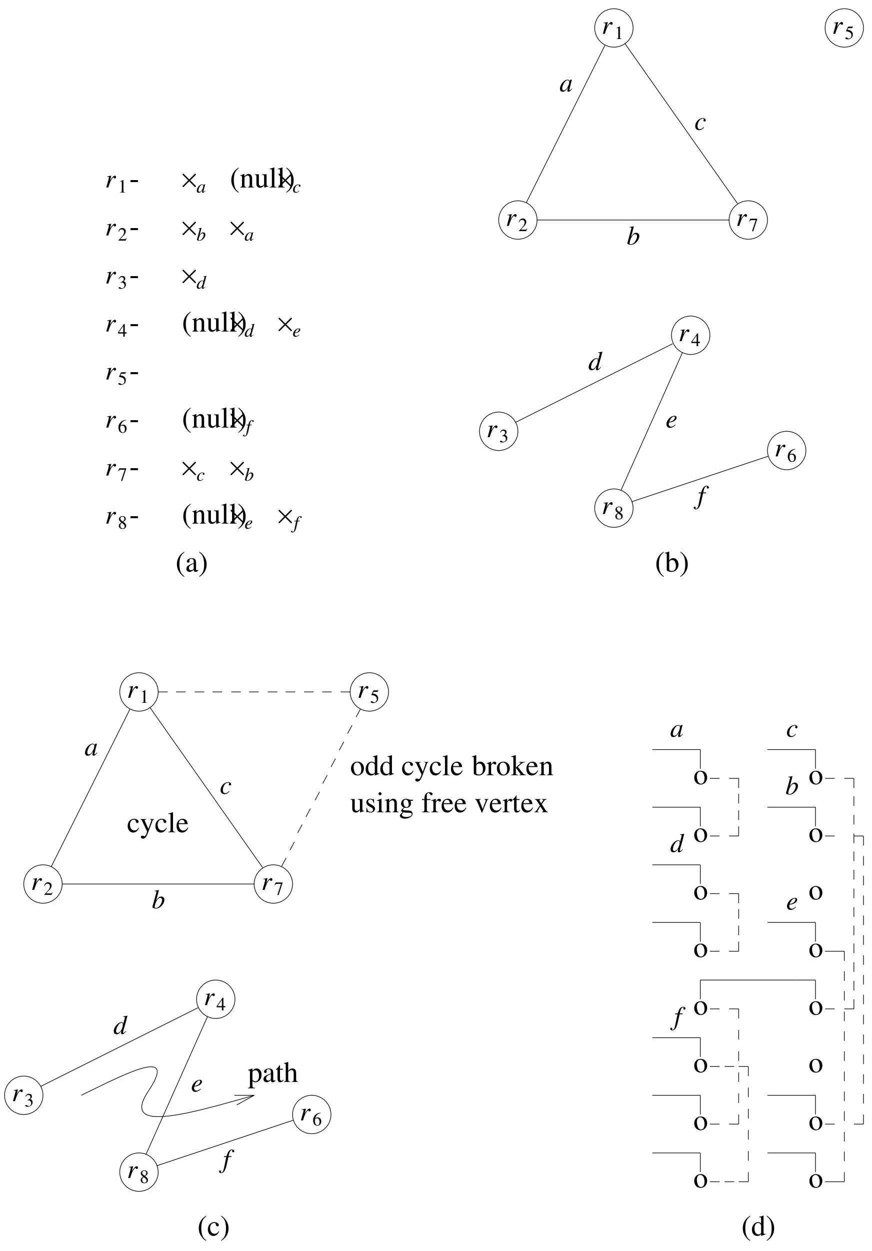

r_1- $\times_a$ (null) $\times_c$
r_2- $\times_b$ $\times_a$
r_3- $\times_d$
r_4- (null) $\times_d$ $\times_e$
r_5-
r_6- (null) $\times_f$
r_7- $\times_c$ $\times_b$
r_8- (null) $\times_e$ $\times_f$
(a)
(b)
cycle
odd cycle broken using free vertex
path
(c)
(d)

Figure 3.3: Breaking up of odd cycles to obtain a 2-column via assignment.

cycle. Figure 3.3(d) shows the resulting via assignment. □

The proofs given in [RAGH84] for the NP-completeness of the two column restricted and unrestricted via assignment problem involve net lists in which a g-node may contain many pins. Our next theorem examines the complexity of the two via column problem under the constraint that each g-node contains exactly one pin.

Theorem 3.3: Let L be a net list in which each g-node consists of exactly one pin.
(a) There is a polynomial time algorithm to determine whether or not two columns suffice in the restricted model.
(b) For the unrestricted model, determining whether or not two columns suffice is NP-complete.

Proof:

(a) Remove from L, any net that contains only one g-node. If any row now contains more than two g-nodes, then the net list cannot be realized using two via columns (recall that each g-node consists of exactly one pin). Assume that no row contains more than two g-nodes. Construct the graph G = (V, E) as follows. With each net in L, identify a unique vertex in V. There is an edge (u, v) in E iff the nets identified with u and v have g-nodes on the same row. Clearly, if two nets have g-nodes on the same row and each g-node consists of exactly one pin, then in the restricted model the two nets must be assigned to different via columns. Hence, two via columns suffice iff G is bipartite. This is easily determined in polynomial time.

(b) It is easy to see that this problem is in NP. So, we need merely show that it is NP-hard. For this, we use the NP-complete problem [GARE79]: Minimum Node Deletion Bipartite Subgraph (MNDBS). In this problem, we are given an undirected graph G = (V, E) and an integer k. We are required to determine if there is a subset $U \in V, |U| \leq k$, such that the subgraph induced by (V − U) is bipartite.

From any undirected graph G and any k, we may construct an instance I of the two column unrestricted via assignment problem such that every g-node in I has exactly one pin and I is realizable using two columns iff the answer to the MNDBS problem for (G, k) is yes.

I is obtained in the following way. First delete from G, vertices with degree ≤ 1. Let the resulting graph be $G' = (V', E')$. The number of rows in I is $(|E'| + k)$. Identify each of the first $|E'|$ rows with a distinct edge in E′. Each $v \in V'$ defines a net of I. This net has a pin in each row that corresponds to an edge that is incident on v. Each pin of a net is in a different column. So, each pin defines a g-node. Note that each of the first $|E'|$ rows has exactly two pins on it. The remaining k rows have no pins. Figure 3.4 shows an example construction with $k = 4$.

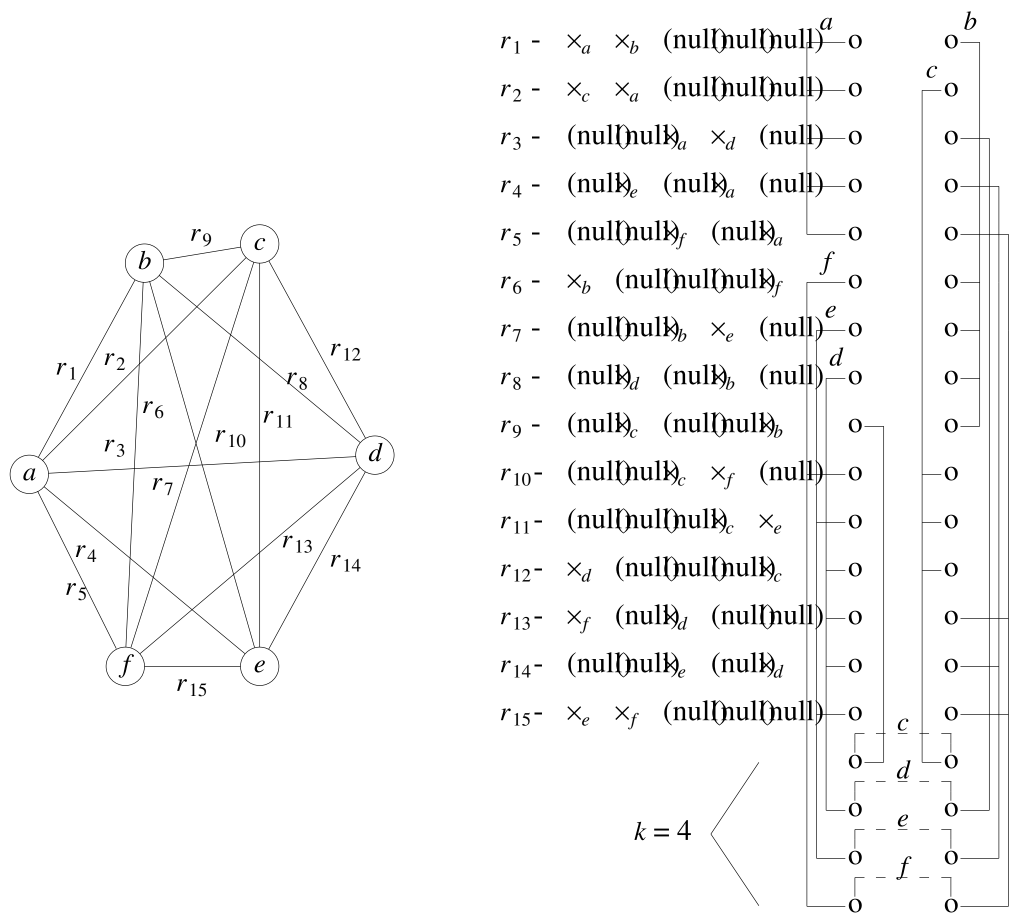
In every two column assignment of vias, the two pins on each of the first $|E'|$ rows must be assigned to the two vias at the end of the row. The k pairs of vias at the bottom may be assigned to upto k nets to allow these nets to use vias from both columns. If a two column assignment exists, then deletion of the vertices corresponding to the nets (upto k) assigned to the bottom k via pairs results in a bipartite graph. Similarly, if there is a U with $|U| \leq k$ such that (V − U) defines a bipartite subgraph, then assigning the bottom $|U|$ pairs to the nets corresponding to vertices in V will result in a two column via assignment. □

4 EXISTING VIA ASSIGNMENT HEURISTICS

In this section, we briefly review two existing heuristics for the via assignment problem. The first of these is due to Ting, Kuh and Vincentelli [TING79] and the second to Gonzalez [GONZ83]. Both heuristics use the unrestricted model for via assignment.

4.1 Heuristic of [TING79]

This heuristic assigns vias to nets one net at a time. Via columns are labeled old or new depending on whether or not any via from the column has already been assigned. The following procedure is used to assign vias to the net currently being considered :



r_1 - $\times_a$ $\times_b$ (null)(null)(null)
r_2 - $\times_c$ $\times_a$ (null)(null)(null)
r_3 - (null)(null)$\times_a$ $\times_d$ (null)
r_4 - (null)$\times_e$ (null)$\times_a$ (null)
r_5 - (null)(null)$\times_f$ (null)$\times_a$
r_6 - $\times_b$ (null)(null)(null)$\times_f$
r_7 - (null)(null)$\times_b$ $\times_e$ (null)
r_8 - (null)$\times_d$ (null)$\times_b$ (null)
r_9 - (null)$\times_c$ (null)(null)$\times_b$
r_{10}- (null)(null)$\times_c$ $\times_f$ (null)
r_{11}- (null)(null)(null)$\times_c$ $\times_e$
r_{12}- $\times_d$ (null)(null)(null)$\times_c$
r_{13}- $\times_f$ (null)$\times_d$ (null)(null)
r_{14}- (null)(null)$\times_e$ (null)$\times_d$
r_{15}- $\times_e$ $\times_f$ (null)(null)(null)
$k = 4$
a
b
c
d
e
f

Figure 3.4: Example construction of net list, *I*, from graph, *G´*, with *k* = 4.

Step 1: Determine the g-nodes of the net. If the number of g-nodes is less than two, then no vias are needed and we proceed to the next net (if any).

Step 2: Assume that the number of g-nodes is greater than 1. Determine an old via column to which a maximum number of g-nodes connect. Ties are broken arbitrarily.

Step 3: If all g-nodes of this net are connected, proceed with the next net (if any) from Step 1. If some unconnected g-nodes remain for this net, perform a breadth first search beginning at the column selected above to an unconnected g-node (the search may utilize only unused vias in old via columns).

Step 4: If the search of Step 3 reaches an unconnected g-node, then a via column connecting the maximum number of unconnected g-nodes is selected by a backtrack process. The newly connected g-nodes are labeled connected. Go to Step 3.

Step 5: The current net cannot be connected using old via columns alone. So, add a new via column and assign vias from this column only to connect the net. Label this column old. Proceed with the next net (if any) from Step 1.

Example 4.1: An eleven net example is shown in Figure 4.1(a). Assume that the nets are considered for via assignment in the order 1,2,....,11. As no net has two pins in the same column or row, each pin is a g-node. Nets 1 and 2 can be realized using vias from column 1. The g-nodes of net 3 cannot be connected using vias from column 1 alone and so a new via column is added. Net 3 is assigned the first 5 vias in column 2. At this time, we have two old columns and their status is as in Figure 4.1(b).

Net 4 is considered next. Both old columns connect two g-nodes of net 4. Assume that column 2 is selected in Step 2. A breadth first search from column 2 gets us to column 1 (via the row connection of row 7) which in turn gets us to the remaining g-nodes using column connections. Column 1 is selected in Step 4 and the net 4 via assignment completed as in Figure 4.1(c). Figure 4.1(d) shows the via assignment obtained for all 11 nets. □

In our later use of this heuristic, we make the following assumptions about how some of the steps are implemented :

1. Nets are considered in non-increasing order of their number of g-nodes.
2. When a g-node is being assigned a via, all pins in the g-node are considered. An available via from any of the rows on which there is a pin in the g-node may be assigned to the g-node (Figure 4.2). For each of these assignable vias, we determine its *demand.* This is the number of remaining g-nodes (from other remaining nets) to which it may be assigned. A via with the least demand is assigned to the g-node.

4.2 Heuristic of [GONZ83]

Gonzalez [GONZ83] has developed a polynomial time heuristic for the unrestricted via assignment problem that uses at most three times the minimum possible number of via columns. This heuristic consists of the following three steps :

Step I: (a) From the given via assignment instance, construct a bipartite graph, G = (S $\cup$ T, E), where S contains one vertex for each g-node and T contains one vertex for each row of the pin grid. $(s, t) \in$ E iff $s \in$ S, $t \in$ T and the g-node represented by s has a pin in the row represented by T.

(b) Define a *complete s-matching*, I, of G to be a subset of E such that every vertex in S is adjacent to exactly one edge in I. Let M(I) be the maximum number of edges in I that are adjacent to any vertex in T. An *optimal complete s-matching* (*ocs*) is a complete s-matching with minimum M(I). Find an *ocs* for G. Let M(I) for this *ocs* be α. Clearly, α is a lower bound on the number of via columns needed.

Step II: The *ocs* obtained in Step I is used to construct a restricted via assignment instance in which each net has exactly two pins. This restricted instance has the property that from any via assignment for the instance, a via assignment that uses the same number of columns can be constructed for the original unrestricted instance. Let g_i be the number of g-nodes in net i of the original instance, $1 \le i \le n$. The restricted instance contains $n' = (\sum_{i=1}^{i=n} g_i - n)$ 2-pin nets, $2n'$ pin columns, and as many pin rows as the original instance. For each net i in the original instance, there are $(g_i - 1)$ nets in the restricted instance. Let $p_1, p_2, \ldots, p_{g_i}$ be the g_i g-nodes of net i. Construct one net for each of the $(g_i - 1)$ pairs (p_1, p_2), (p_2, p_3),..., $(p_{g_i - 1}, p_{g_i})$. The net for (p_k, p_{k+1})

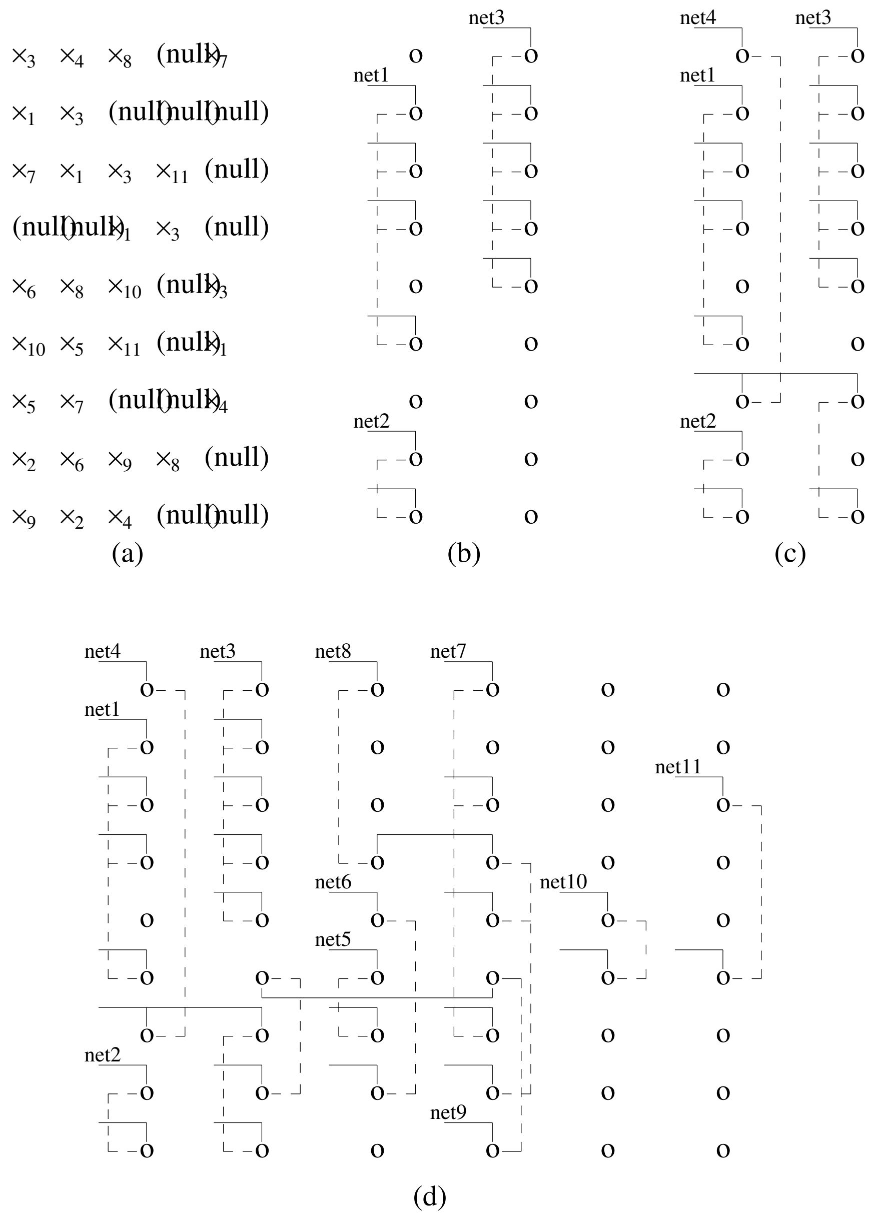


Figure 4.1: [TING79] heuristic on an 11-net instance.

consists of one pin in each of the rows to which p_k and p_{k+1} are matched in the *ocs*. All pins are in distinct columns.

Step III: The via assignment problem constructed above consists only of 2-pin nets. No row has more than 2α nets. This problem may be transformed into an equivalent edge coloring problem as in Section 3. The edges may be colored in polynomial time using at most 3α colors using the algorithm of Vizing [BOND76] (note that the

			vias	demand (for g-node of net1)	
$\times_1$	(null)	(null)	ϕ	0	
$\times_1$	$\times_2$	$\times_3$	o	2	ϕ : used via
$\times_1$	(null)	(null)	ϕ	0	o : unused via
$\times_1$	(null)	$\times_2$	o	1	

Figure 4.2: Demand.

degree of the constructed graph is at most 2α). From the edge coloring, a via assignment for the restricted instance is obtained and from this a via assignment for the original problem obtained. The maximum number of via columns used is 3α.

While the heuristic described above has the desirable characteristic that it guarantees to use no more than three times the optimal number of via columns, it generates assignments that are generally inferior to those obtainable by more direct methods (such as [TING79]). The reasons for this poor behavior in practice, are :

1. An *ocs* constrains the rows from which vias can be assigned. There may be no optimal via assignment available from an *ocs*. Consider the bipartite graph of Figure 4.3. All edges define the *ocs*, except for a choice between edge *a* or *b*. If edge *b* exists in the possible *ocs*, the best via assignment uses four via columns. Using edge *a*, a three column assignment is possible.

2. The method suggested in [GONZ83] to construct the 2-pin net instance of Step II can result in instances that have as many as 2α nets on a row. This may be larger than the total number of nets in the original instance. Clearly, there are always solutions that use no more via columns than the total number of nets. Figure 4.4 shows an *ocs* and the corresponding restricted instance created in Step II. Since the maximum number of nets on a row directly affects the final solution, we may employ a heuristic to obtain a better decomposition of the g-nodes $p_1, p_2, \ldots, p_{g_i}$ than the one currently employed (i.e., (p_1, p_2), (p_2, p_3),...., (p_{g_i-1}, p_{g_i})). Every decomposition of the form $(p_{\sigma(1)}, p_{\sigma(2)})$, $(p_{\sigma(2)}, p_{\sigma(3)})$,...... where σ is a permutation of (1,2,.....,g_i) is a valid decomposition for Step II. A good decomposition is obtained by taking into account the demands placed by all nets on a particular row. For the example of Figure 4.4, a better restricted instance is shown in Figure 4.5.

3. Finally, in Step III, the edge coloring heuristic employed has a tendency to use 3β/2 colors often (β is the degree of the multigraph created in Step III) even though fewer will often suffice.

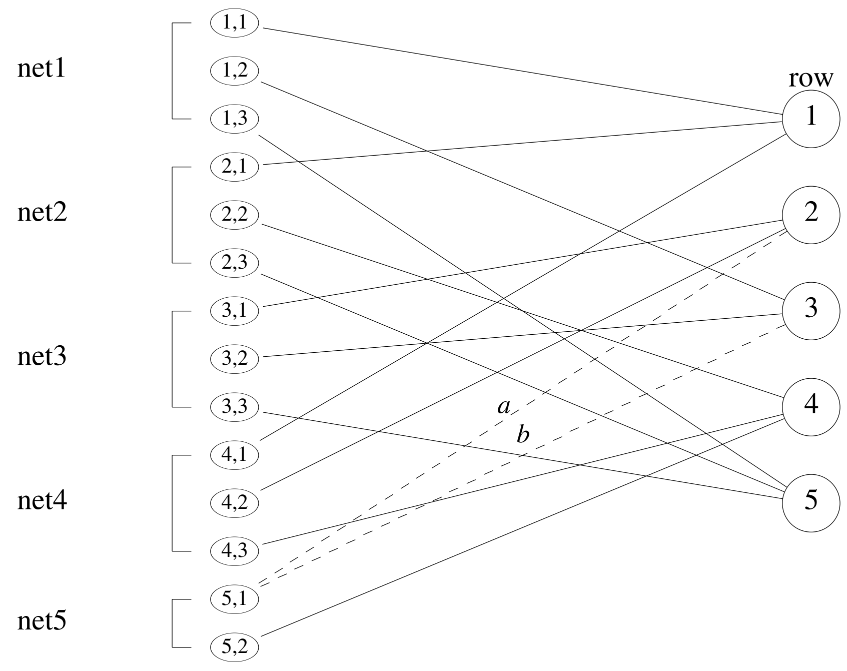


Figure 4.3: An *ocs*.

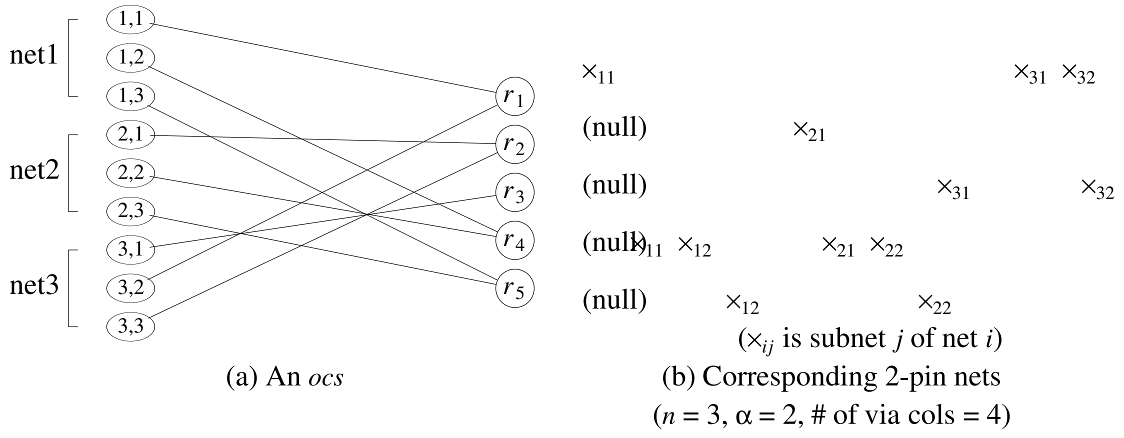

net1
net2
net3
1,1
1,2
1,3
2,1
2,2
2,3
3,1
3,2
3,3
r_1
r_2
r_3
r_4
r_5
$\times_{11}$ $\times_{31}$ $\times_{32}$
(null) $\times_{21}$
(null) $\times_{31}$ $\times_{32}$
(null) $\times_{11}$ $\times_{12}$ $\times_{21}$ $\times_{22}$
(null) $\times_{12}$ $\times_{22}$
($\times_{ij}$ is subnet j of net i)
(a) An *ocs*
(b) Corresponding 2-pin nets
(n = 3, α = 2, # of via cols = 4)

Figure 4.4: Construction of net list instance from an *ocs* using [GONZ83].

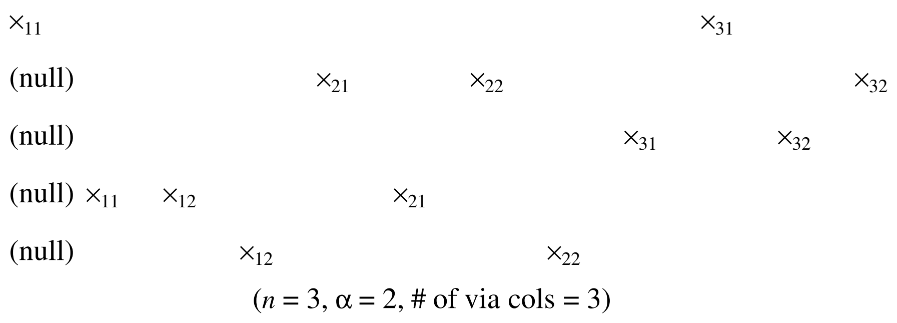

x_{11} x_{31}
(null) x_{21} x_{22} x_{32}
(null) x_{31} x_{32}
(null) x_{11} x_{12} x_{21}
(null) x_{12} x_{22}
($n = 3$, $\alpha = 2$, # of via cols = 3)

Figure 4.5: Net list instance of Figure 4.4(a) using a better method.

5 NEW HEURISTICS

In this section, we propose two new heuristics for the via assignment problem as well as a heuristic for the Step II decomposition of [GONZ83].

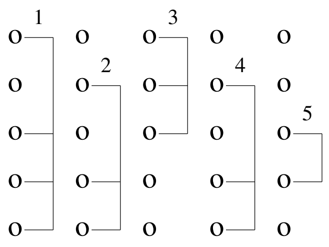
5.1 Heuristic-1

This heuristic is quite similar to that of [TING79]. It employs the assumptions made at the end of Section 4.1 about the order in which the nets are considered for via assignment and also the selection of vias for each g-node. The essential difference between the two heuristics comes about in the case of nets that cannot be realized using old via columns alone. While the heuristic of [TING79] assigns all the vias for this net from a new via column, our heuristic utilizes as many as possible from the old columns and gets the remainder from a new via column. Figures 5.1 and 5.2 show cases where the new strategy outperforms that of [TING79].

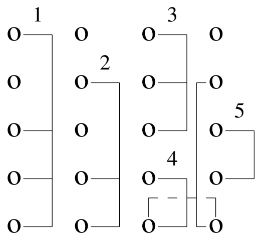
A detailed description of Heuristic-1 is provided below. The procedure to merge sort and that to determine the g-nodes are quite standard and are omitted.

$\times_1$	(null)	$\times_3$	(null)
$\times_3$	$\times_2$	$\times_4$	(null)
$\times_5$	(null)	$\times_1$	$\times_3$
$\times_4$	$\times_1$	$\times_5$	$\times_2$
(null)	$\times_4$	$\times_2$	$\times_1$

(a) Net List



(b) [TING79] heuristic
(5 via cols)

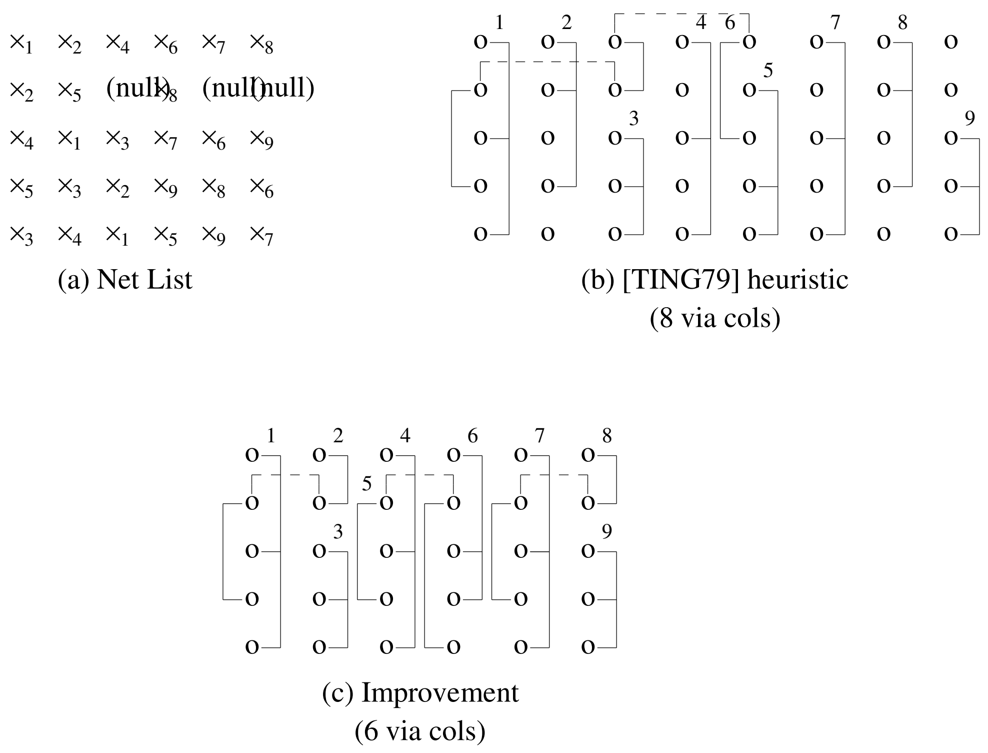


(c) Improvement
(4 via cols)

Figure 5.1: Comparison of Heuristic-1 with [TING79] heuristic.

Algorithm for Heuristic-1 :

```
Step 1  : Input net data;
Step 2  : Determine the g-nodes;
Step 3  : Sort nets on [# of g-nodes] using merge sort;
Step 4  : Current via column = 1;
Step 5  : Map in first net into current via column :
            FOR each g-node of this net
              Determine via with least demand
            ENDFOR;
Step 6  : DO (till Step 21) for each net in sorted order -
Step 7  :   FOR each via col (i.e. from 1 to current via col)
              Determine rows reachable by via col;
              Determine g-nodes reachable by these rows;
            ENDFOR; (* keep track of via col which reaches
                       the maximum number of g-nodes *)
Step 8  :   Put via col (which reaches max # of g-nodes) into stack;
            Mark via col as visited;
Step 9  :   WHILE [(stack of via columns is not empty) AND (net not fully netted yet)],
            DO (till Step 15) -
```



(a) Net List
(b) [TING79] heuristic (8 via cols)
(c) Improvement (6 via cols)

Figure 5.2: Another instance comparison of Heuristic-1 with [TING79] heuristic.

```
Step 10 :   Get via col from stack which reaches max # of g-nodes;
Step 11 :   FOR each g-node reached by this via col
              Determine via with least demand;
              Determine via cols reachable by this g-node;
              Put in stack if via col has not been visited yet;
              Mark them as visited;
            ENDFOR;
Step 12 :   Determine other via cols reachable by the
            via col (the one found in Step 10) ;
Step 13 :   Put them in stack if not visited yet;
            Mark them as visited;
Step 14 :   IF a reachable via col has already been visited but is reached by a
            via col using a row with least demand, THEN
              Mark via col visited using this least demand row vias;
            ENDIF;
            (* Note : If a via col has been reached by a g-node, this takes
            priority over all cases where via col is reached from another via col,
            because only one extra via may be used instead of two *)
Step 15 :  ENDWHILE; (*Step 9*)
Step 16 :   IF (net has not been netted yet), DO (till Step 20) -
Step 17 :     Get a new via col (becomes current via col);
Step 18 :     Map remaining g-nodes into this new via col :
                FOR each g-node
                  Determine via with least demand
                ENDFOR;
Step 19 :     Determine a row with least demand which connects the g-nodes
              (mapped in Steps 9 - 15) to this new via col;
Step 20 :   ENDIF; (*Step 16*)
Step 21 : ENDDO; (*Step 6*)
(* End of algorithm *)
```

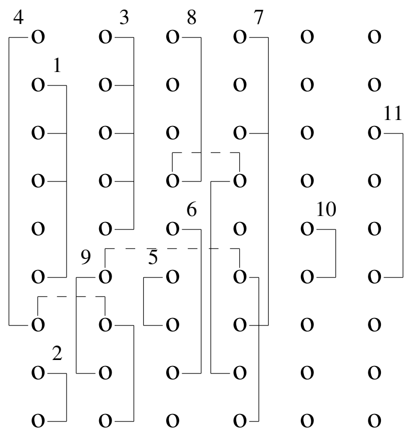
5.2 Heuristic-2

In this heuristic, each time a new via column is introduced, an attempt is made to meet the via needs of as many nets as possible. Nets are considered for via assignment in nonincreasing order of their number of g-nodes. When the total needs of no more nets can be met from the new via column, all remaining nets are reconsidered for via assignments from multiple via columns. This is done in a manner similar to Heuristic-1. However, no new via columns are introduced. When the needs of no additional nets can be totally met using the old via columns, a new via column is introduced and the process repeated.

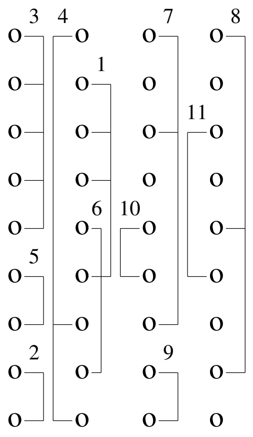
Figure 5.3 shows the result of applying this strategy to the eleven net example of Figure 4.1(a). The new via assignment uses 4 via columns rather than 6 as used by the heuristic of [TING79]. Another example is shown in Figure 5.4. The details of Heuristic-2 are provided below.

$\times_3$	$\times_4$	$\times_8$	(null)	(null)
$\times_1$	$\times_3$	(null)	(null)	(null)
$\times_7$	$\times_1$	$\times_3$	$\times_{11}$	(null)
(null)	(null)	$\times_1$	$\times_3$	(null)
$\times_6$	$\times_8$	$\times_{10}$	(null)	$\times_3$
$\times_{10}$	$\times_5$	$\times_{11}$	(null)	$\times_1$
$\times_5$	$\times_7$	(null)	(null)	$\times_4$
$\times_2$	$\times_6$	$\times_9$	$\times_8$	(null)
$\times_9$	$\times_2$	$\times_4$	(null)	(null)

(a) Net List



(b) [TING79] heuristic
(6 via cols)



(c) Improvement
(4 via cols)

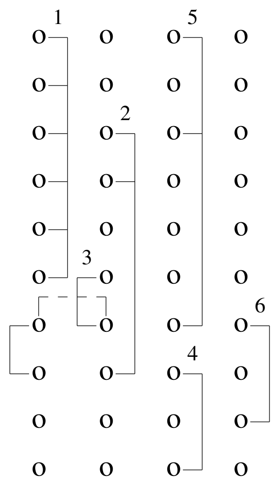
Figure 5.3: Comparison of Heuristic-2 with [TING79] heuristic.

Algorithm for Heuristic-2 :

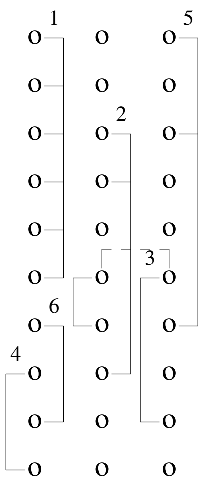
```
Step 1  : Input net data;
Step 2  : Determine g-nodes;
Step 3  : Sort nets on [# of g-nodes] using merge sort;
Step 4  : REPEAT (till Step 19) until all nets are done -
Step 5  :   Get new via column;
Step 6  :   REPEAT (till Step 10) for each net in sorted order which (have not
            been netted yet) and (which can be mapped into current via column) -
Step 7  :     Map in net into current via column :
```

(null)	$\times_1$	$\times_5$	(null)	(null)	(null)	(null)
(null)	(null)	$\times_1$	(null)	(null)	(null)	(null)
(null)	$\times_5$	(null)	$\times_1$	$\times_2$	(null)	(null)
(null)	(null)	(null)	$\times_2$	$\times_1$	(null)	(null)
(null)	(null)	(null)	(null)	(null)	(null)	$\times_1$
$\times_1$	(null)	(null)	$\times_3$	(null)	(null)	(null)
(null)	$\times_6$	(null)	$\times_5$	(null)	(null)	$\times_3$
(null)	(null)	$\times_4$	(null)	$\times_3$	(null)	$\times_2$
(null)	(null)	(null)	$\times_6$	(null)	(null)	(null)
(null)	(null)	(null)	(null)	$\times_4$	(null)	(null)

(a) Net List



(b) [TING79] heuristic
(4 via cols)



(c) Improvement
(3 via cols)

Figure 5.4: Another instance comparison of Heuristic-2 with [TING79] heuristic.

```
             FOR each g-node of this net
               Determine via with least demand
             ENDFOR;
Step 8  :   Mark net as done;
Step 9  :   FOR via rows used by this net
               Determine other nets in demand of these vias;
```

```
            Mark them as unmappable into current via col;
          ENDFOR;
Step 10 : ENDREPEAT; (* Step 6 *)
Step 11 : Reset nets (not yet netted) as mappable;
Step 12 : IF a row has no free vias
            Determine nets in demand of vias on this row;
            Mark them as unmappable;
          ENDIF;
Step 13 : (* Trying to map nets into old via columns *)
          REPEAT (till Step 17) for each net in sorted order which (have not
          been netted yet) and (which can be mapped) -
Step 14 : Same as Steps 7 to 15 of Heuristic-1; (* these are the steps where a
          net is tried to be netted into the old via columns *)
Step 15 :   IF (net has not been netted yet) THEN
              Ignore g-nodes already mapped in Step 14
            ELSEIF (net has been netted) THEN
              Mark net as done
            ENDIF;
Step 16 :   IF (net has been netted)
              Determine rows which have no vias left;
              Find other nets in demand of these rows;
              Mark them as unmappable;
            ENDIF;
Step 17 : ENDREPEAT; (* Step 13 *)
Step 18 : Mark nets (not yet netted) as mappable;
Step 19 : ENDREPEAT; (* Step 4 *)
(* End of algorithm *)
```

5.3 Heuristic for Step II of [GONZ83]

In Step II of [GONZ83], the *ocs* obtained in Step I is used to obtain an instance with two pins per net. This instance has the property that a via assignment for the instance is also a via assignment for the original instance. We propose the following heuristic for this step.

Step 1: Let d_i be the degree of row i in the *ocs*, i.e. it is the number of g-nodes that have been matched to vias in row i. Let n', n and g_i be as in Section 4.3. Let $m(q)$ be the row to which the g-node, q, is matched in the *ocs*.

Step 2: Discard nets that have only one g-node. Put each g-node of the remaining nets into a distinct cluster. For each cluster i, let $r(i)$ be a least degree row which contains a node. Initially, if q is the sole g-node in the cluster, then q is a g-node and $r(i) = m(q)$. Assign each g-node q, to a pin in the row $m(q)$. All pins are in distinct columns.

```
Step 3:  FOR $a$ = 1 TO $n'$ DO
             Let $c$ be the cluster with least $d(r(c))$.
             Let $e$ be the cluster from the same net as $c$ with maximum $d(r(e))$ and $c \neq e$.
             CASE
                 : $c$ and $e$ both contain only one g-node : The two g-nodes define a two pin
                 net;
                 : $c$ contains one g-node and $e$ more than one : Add a pin to row $r(e)$. This
```

```
            pin and the g-node of $c$ define a two pin net;
            : $c$ contains more than one g-node and $e$ one : Add a pin to row $r(c)$. This
            pin and the g-node of $e$ define a two pin net;
            : both $c$ and $e$ contain more than one g-node : Add a pin to each of the
            rows $r(c)$ and $r(e)$. These pins define a new two pin net.
        ENDCASE;
        Combine both clusters into one. If there is only one remaining cluster for the
        net, discard it.
        Update $r(\ )$ values.
    ENDFOR;
    { Note : All pins are added to new columns }
```

The above heuristic attempts to minimize the maximum number of pins assigned to a row. As an example, consider the two net instance of Figure 5.5. This figure shows the *ocs* obtained in Step I of [GONZ83].

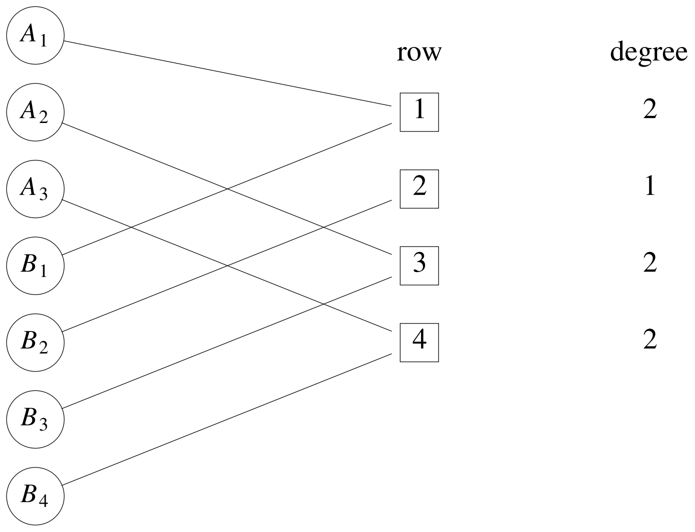


Figure 5.5: *ocs* of a two net instance.

The number of g-nodes in net A is $g_A = 3$ and that in net B is $g_B = 4$. So, $n = 2$, $n' = 3 + 4 - 2 = 5$. An initial assignment of g-nodes to pins in the two pin instance is shown in Figure 5.6. The cluster with least $d(r(\ ))$ is $\{B_2\}$. The one in net B with maximum $d(r(\ ))$ is $\{B_1\}$ (or $\{B_3\}$ or $\{B_4\}$). These get combined to give the cluster $\{B_1, B_2\}$. No new pins are added. Next, the cluster $\{B_1, B_2\}$ is combined with $\{B_3\}$ (or $\{B_4\}$). A pin is added to row 2. At this point, we have two 2-pin nets (Figure 5.7).

Now each cluster has $d(r(\ )) = 2$. So any two clusters from either net A or B can be combined next. Assume that $\{B_1, B_2, B_3\}$ and $\{B_4\}$ are combined. A possible result is shown in Figure 5.8. Now B has only one cluster and only net A remains.

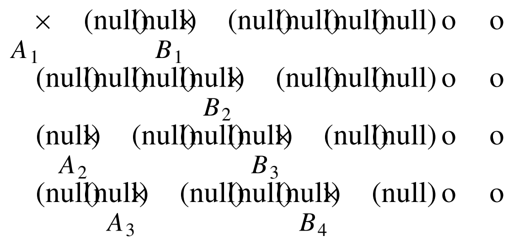

× (null)(null)× (null)(null)(null)(null) o o
A1 B1
(null)(null)(null)(null)× (null)(null)(null) o o
B2
(null)× (null)(null)(null)× (null)(null) o o
A2 B3
(null)(null)× (null)(null)(null)× (null) o o
A3 B4

Figure 5.6: Initial assignment of g-nodes to pins.

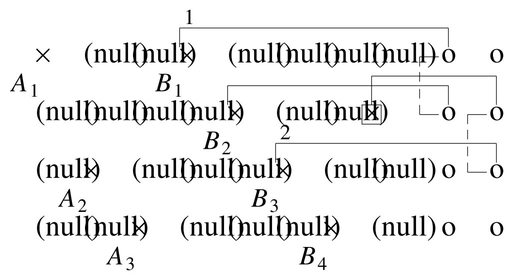


Figure 5.7

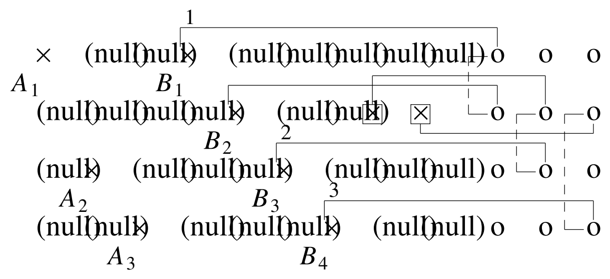


Figure 5.8

5.4 Complexity analysis

Let m, p and q respectively, denote the number of via columns, grid rows and g-nodes per net. Let r be the average number of pins in each g-node. Heuristic-1 takes $O(mpq)$ time to assign vias to a net while Heuristic-2 takes $O(m^2pq)$ time per net. It should be noted that these are worst case times and that in practice, one expects significantly better performance. The heuristic originally proposed in [TING79] takes $O(mp^2 + pm^2 + mqr)$ time per net. The modified version stated here can be implemented in $O(mpq)$ time. The heuristic of [GONZ83] takes $O(p^{\frac{1}{2}} Q^{\frac{5}{2}} \log Q)$, where Q is the total number of g-nodes.

6 EXPERIMENTAL RESULTS

We compared the performance (both time needed and number of via columns) of Heuristics 1,2 and [TING79]. The algorithms were coded in Pascal and run on a SUN workstation. Test data was generated randomly. To get some indication of how close the solutions are to optimal, we also computed a lower bound on the number of via columns needed. The lower bound computed is α, the degree of *ocs* obtained in [GONZ83].

In order to determine the reasonableness of 3α via columns, we also compute some upper bounds on the number of via columns needed. One obvious bound is n, the number of nets. Another and better upper bound is obtained by constructing the following graph G = (V, E). V has one vertex for each net. Consider the *ocs* obtained in Step I of [GONZ83]. There is an edge $(i, j) \in$ E iff nets i and j both have a g-node matched to a common row in the *ocs*. Let δ be the maximum degree in G. The vertices of G can be colored using at most $(\delta + 1)$ colors. Also, every coloring of G corresponds to a solution to the restricted via column assignment problem. Hence, $(\delta + 1)$ is another upper bound on the minimum number of via columns needed.

n	Heur-1		Heur-2		[TING79]		Lower bound			Upper bound			
	m	t	m	t	m	t	α	d	d_{our}	3α	$(\delta+1)$	$\frac{3d}{2}$	$\frac{3d_{our}}{2}$
10	4	2	4	1	4	2	4	8	4	12	9	12	6
20	6	4	6	3	6	4	6	10	6	18	20	15	9
50	10	17	10	7	10	15	10	20	12	30	45	30	18
70	14	34	14	11	14	34	14	26	16	42	61	39	24
80	16	68	16	16	16	63	16	30	18	48	70	45	27
90	16	56	16	17	16	51	16	32	20	48	77	48	30
100	18	81	18	18	18	84	18	35	22	54	85	53	33

Table 6.1: (p = 100, q = 12).

The results of our experiments are shown in Tables 6.1 - 6.6. The number of pin columns used was 100. t is time specified in seconds. Tables 6.1 - 6.5 also provide some lower and upper bounds on the number of via columns needed for the respective instances. From the lower bound α, it is evident that for each instance reported in Tables 6.1 - 6.4, an optimal via assignment was

q	Heur-1		Heur-2		[TING79]		Lower bound			Upper bound			
	m	t	m	t	m	t	α	d	d_{our}	3α	$(\delta+1)$	$\frac{3d}{2}$	$\frac{3d_{our}}{2}$
3	7	12	7	4	7	12	7	10	7	21	13	15	11
6	10	20	10	6	10	19	10	18	10	30	33	27	15
9	12	29	12	9	12	26	12	24	14	36	53	36	21
12	16	47	17	13	16	43	16	30	18	48	69	45	27
15	20	99	20	20	20	87	20	40	22	60	77	60	33
18	21	81	21	24	21	95	21	42	26	63	80	63	39

Table 6.2: (p = 100, n = 80).

p	Heur-1		Heur-2		[TING79]		Lower bound			Upper bound			
	m	t	m	t	m	t	α	d	d_{our}	3α	$(\delta+1)$	$\frac{3d}{2}$	$\frac{3d_{our}}{2}$
50	25	55	24	11	24	61	24	46	32	72	80	69	48
70	19	41	19	11	19	49	19	38	24	57	80	57	36
90	20	77	20	14	20	68	20	40	20	60	72	60	30
100	16	48	16	15	16	57	16	30	18	48	74	45	27
110	15	47	15	15	15	45	15	30	17	45	67	45	26

Table 6.3: (q = 12, n = 80).

obtained by at least one of the three heuristics (and most of the time by all three). The column labeled d, gives us the degree of the two pin per net instance generated by Step II of [GONZ83]. d_{our} is the d of the similar instance generated using the heuristic of Section 5.3. As can be seen, d is considerably larger than d_{our}. Further, both are often larger than α.

The solutions produced by the two heuristics developed here as well as those produced by the modified version of the heuristic of [TING79], generally use the same number of via columns. However, the heuristic developed in Section 5.2 takes significantly less time than the other two heuristics.

The solutions obtainable by [GONZ83] may use as many as $(3 * \alpha)$ via columns. $(3 * \alpha)$ is often a better upper bound than $(\delta + 1)$. For the instances generated, [GONZ83] generates solutions that use between d and $3d/2$ (or d_{our} and $3d_{our}/2$) via columns. Clearly, on our test instances,

Heur-1		Heur-2		[TING79]		Lower bound			Upper bound			
m	t	m	t	m	t	α	d	d_{our}	3α	$(\delta+1)$	$\frac{3d}{2}$	$\frac{3d_{our}}{2}$
11	18	11	5	11	18	11	16	11	33	36	24	17
9	12	9	5	9	15	9	17	10	27	39	26	15
9	18	9	5	10	16	9	18	10	27	40	27	15
16	49	16	11	16	42	16	32	16	48	50	48	24
15	36	15	12	15	34	15	30	17	45	50	45	26

Table 6.4: (p = 100, n = 50).

n	Heur-1		Heur-2		[TING79]		Lower bound			Upper bound			
	m	t	m	t	m	t	α	d	d_{our}	3α	$(\delta+1)$	$\frac{3d}{2}$	$\frac{3d_{our}}{2}$
60	37	21	38	32	39	18	36	72	56	108	60	108	84
70	41	33	38	18	40	25	37	74	57	111	70	111	86
80	44	56	42	19	45	36	41	80	61	123	80	120	92
90	45	91	44	67	50	43	41	82	64	123	90	123	96
100	51	56	46	28	52	65	44	82	66	132	100	123	99
120	49	93	47	67	53	126	45	85	68	135	120	128	102
140	53	96	49	17	53	91	45	87	68	135	139	131	102

Table 6.5: (p = 20).

the heuristic of [GONZ83] will generate significantly inferior solutions. However, it should be noted that none of the three heuristics tested have a better worst case performance bound than that of [GONZ83].

The instances of Table 6.5 differ from the remaining instances. Tables 6.1 - 6.4 were obtained with instances in which the number of nets (n) were generally less than the number of rows (p). The instances used for Table 6.5 contain significantly more nets than rows. Here, none of the three heuristics produced solutions with value equal to α. Heuristic-2 generally performed better than Heuristic-1 which in turn generally performed better than [TING79].

We also experimented with one instance that had 175 nets and 20 rows and another that had 225 nets and 20 rows (number of pin columns = 200). For both of these instances, Heuristic-2

n	Heur-1		[TING79]	
	m	*t*	*m*	*t*
175	84	819	94	407
225	97	392	103	655

Table 6.6

required more time than we could spend. However, Heuristic-1 and [TING79] were able to obtain solutions in less than 15 minutes each. The results are summarized in Table 6.6. Once again the solutions obtained by Heuristic-1 are superior to those obtained by [TING79].

7 CONCLUSIONS

Based upon the experiments carried out by us, we conclude that Heuristic-2 generally obtains better solutions than does Heuristic-1. Further, both heuristics generally outperform that of [TING79]. The worst case time complexity of Heuristic-2 is, however, larger than that of Heuristic-1. So, for certain instances it may not be practical to run Heuristic-2 to completion and Heuristic-1 will have to be used instead.

8 REFERENCES

BOND76 Bondy J. and U.S.R.Murthy, *Graph Theory with Applications*, Elsevier North Holland Inc, 1976.

COHO83 Cohoon J. and S.Sahni, *Heuristics for the Board Permutation Problem*, Proc. 1983 IEEE ICCAD Conf.

FIOR77 Fiorini S. and R.J.Wilson, *Edge-Coloring of Graphs*, Research notes in Mathematics, 16, Pitman, London, 1977.

GARE79 Garey M.R. and D.S.Johnson, *Computers and Intractibility : A Guide to the Theory of NP-Completeness*, W.H.Freeman, San Francisco, 1979.

GONZ83 Gonzalez T., *An Approximation Algorithm for the Via Assignment Problem*, ICCAD, 1983, pp 125-128.

GOTO77 Goto S., I.Cederbaum and B.Ting, *Suboptimal Solution of the Backboard Ordering with Channel Capacity Constraints*, IEEE Trans Circuits and Systems, Nov 1977, pp 645-652.

HAN84a Han S. and S.Sahni, *Layering Algorithms for Single Row Routing*, University of Minnesota, Technical report, June 1984.

HAN84b Han S. and S.Sahni, *Single Row Routing in Narrow Streets*, IEEE Trans CAD, CAD-3, 3, 1984, pp 235-241.

HAN84c Han S. and S.Sahni, *A Fast Algorithm for Single Row Routing*, University of Minnesota, Technical report, Jan 1984.

HOLY81 Holyer I., *The NP-Completeness of Edge-Coloring*, Siam J. Computing, Vol 10, No 4, Nov 1981, pp 718-720.

HOPC73 Hopcroft J. and R.Karp, *An $n^{5/2}$ Algorithm for Maximum Matching in Bipartite Graphs*, Siam J. Computing, 1973, pp 225-231.

HORO78 Horowitz E. and S.Sahni, *Fundamentals of Computer Algorithms*, Computer science press, Potomac, MD, 1978.

KUH79 Kuh E., T.Kashiwabara and T.Fujisawa, *On Optimal Single Row Routing*, IEEE Trans Cicuits and Systems, Vol CAS-26, No 6, June 1979.

RAGH83 Raghavan R. and S.Sahni, *Single Row Routing*, IEEE Trans Computers, C-32, 3, 1983, pp 209-220.

RAGH84 Raghavan R. and S.Sahni, *The Complexity of Single Row Routing*, IEEE Trans Circuits and Systems, CAS-31, 5, 1984, pp 462-472.

SO74 So H.C., *Some Theoritical Results on the Routing of Multilayer Printed Wiring Boards*, Proc IEEE Symp Circuits and Systems, 1974, pp 296-303.

SYSL83 Syslo M.M., N.Deo and J.S.Kowalik, *Discrete Optimization Algorithms*, Prentice Hall Inc, NJ, 1983.

TARN84 Tarng T., M.Marek-Sadowska and E.S.Kuh, *An Efficient Single Row Routing Algorithm*, IEEE Trans CAD, CAD-3, 3, 1984, pp 178-183.

TING76 Ting B., E.S.Kuh and I.Shirakawa, *The Multilayer Routing Problem : Algorithms and Necessary and Sufficient Conditions for the Single-Row Single-Layer Case*, IEEE Trans Circuit and Systems, CAS-23, Dec 1976, pp 768-778.

TING78 Ting B.S. and E.S.Kuh, *An Approach to the Routing of Multilayer Printed Circuit Boards*, Proc IEEE Symp Circuits and Systems, 1978, pp 902-911.

TING79 Ting B.S., E.S.Kuh and A.Sangiovanni-Vincentelli, *Via Assignment Problem in Multilayer Printed Circuit Board*, IEEE Trans Circuits and Systems, Vol CAS-26, Apr 79, pp 261-271.

TSUK79 Tsukiyama S., I.Shirakawa and S.Asahara, *An Algorithm for the Via Assignment Problem in Multilayer Backboard Wiring*, IEEE Trans Circuits and Systems, Vol CAS-26, Jun 79, pp 369-377.

TSUK83 Tsukiyama S., E.S.Kuh and I.Shirakawa, *On the Layering Problem of Multilayer PWB Wiring*, IEEE Trans CAD, CAD-2, 1, Jan 83, pp 30-38.