A Graph-Theoretic Via Minimization Algorithm for Two-Layer Printed Circuit Boards

RUEN-WU CHEN, YOJI KAJITANI, SENIOR MEMBER, IEEE AND SHU-PARK CHAN, FELLOW, IEEE

Abstract—**Based on graph theory, an efficient via minimization algorithm for certain types of two-layer printed circuit boards is developed which can be executed in polynomial time. The algorithm yields solutions for routings with junctions of degrees varying from 2 to 8 and guarantees the minimum number of vias for routings with three or fewer line segments connected to each junction. Examples are given to illustrate various aspects of the algorithm. In addition, preassignment of line segments on a particular layer of the board due to certain prescribed board (or component) constraints is discussed.**

I. INTRODUCTION

THE following basic definitions, as presented in [1], facilitate the description of the via minimization problem.

A *component* is a cell which may be either a basic circuit element or a network device. A *net* is a collection of (straight) line segments that electrically connect a specified set of component pins. A *physical routing*, denoted by P, is a collection of all the nets interconnected on a PC board. In a physical routing, a point where some line segments located at the same layer of the board are connected together is referred to as an *intersection*; and a point, other than a component pin, at which some line segments on two different layers of the board are connected is called a *via*.

Current routing algorithms for two-layer printed circuit (PC) boards produce a large number of vias. This is due to the fact that most routing algorithms which solve the layer assignment problem efficiently rout all horizontal line segments on one layer of a board and all vertical line segments on the other layer [2]. It is desirable to minimize the number of vias in a PC layout, since the vias often contribute to failure of the board due to cracking. Furthermore, the extra through holes needed in making the vias add to the total cost of production.

Several algorithms have been proposed in the literature for the purpose of minimizing the number of vias under certain prescribed conditions. All of these algorithms begin by operating on a given physical routing in which the placement of the components and the routing of the nets have taken place. In 1971, Hasimoto and Stevens [3] presented an algorithm which attempted to reduce the number of vias by moving some line segments from one layer of the board to the other layer for a two-layer PC board. Their algorithm was restricted in the sense that only two line segments were allowed at each junction. In 1975, Sakamoto *et al.* [4] developed the Organized System for Automated Connection-Routing Algorithm (OSACA). System OSACA incorporated a procedure which minimizes vias in a local sense by moving a line segment from one layer to the other if it is unobstructed. Also in 1977, Servit [5] presented an algorithm for reducing the number of vias by duplicating certain line segments for short distances on both layers. Until recently, the via minimization problem was believed to be *NP*-complete even for the simplest case of the Hashimoto–Stevens type. Based on this belief, some approximate and heuristic algorithms have been developed. In 1979, Stevens and vanCleemput [6] introduced an approximate algorithm for via minimization in order to remove some of the restrictions imposed by other algorithms. In their algorithm, the location of the vias in the final solution is restricted to the points which are vias in a given physical routing. In 1981, Ciesielski and Kinnen [7] derived a layer assignment algorithm for routing. In their approach, the via minimization problem was formulated as a (0, 1)-integer programming problem and solved using the branch and bound technique. This approach is more general since it allows the degree of each via to vary between 2 and 8. However, the branch and bound technique will have a worst-case exponential order of complexity.

Kajitani has recently shown that the via minimization problem of the Hashimoto–Stevens type is not *NP*-complete [8]. However, there is a large number of constraints associated with each of the existing algorithms developed for solving the via minimization problem of the Hashimoto–Stevens type. At the outset of this research project it was believed that some of these constraints could be removed. The development of a more general algorithm which could be carried out in polynomial time was the primary objective.

In this paper, a polynomial time via minimization algorithm is developed for routings on a two-layer PC board. The starting point consists of a given physical routing which may have been generated by automatic or manual

Manuscript received March 10, 1982; revised June 15, 1982 and September 21, 1982.

R-W. Chen and S-P. Chan are with the Department of Electrical Engineering and Computer Science, University of Santa Clara, Santa Clara, CA 95053.

Y. Kajitani is with the Department of Electrical and Electronic Engineering, Tokyo Institute of Technology, O-okayama, Megura-ku, Japan.

0098-4094/83/0500-0284$01.00

routing techniques. Such a physical routing includes the specification of component locations, layout of the nets, initial layer assignments, and a set of initial vias. Prior to the execution of the algorithm an initial set of via candidates must be selected. This set of via candidates must include, but may not be limited to, the vias which appear in the given physical routing. Via candidates may be placed anywhere along any of the line segments as suggested in [7]. Therefore, global minimization, with respect to a given physical routing, can be achieved in principle. However, for the sake of clarity and simplicity of the examples in this paper, the via candidates are, in fact, restricted to the initial set of vias and intersections in the given physical routing. It is recognized that such a restriction will, in general, result in nonglobal minimal solutions with respect to a given physical routing. From a computational point of view there will be a tradeoff between the execution time and the achievement of a global minimal solution with respect to a given physical routing. In other words, the removal of the constraints on the number and the location of the via candidates will result in a more complex graph model and extension of the execution time.

In the event that the restriction on the location of the via candidates is removed, the algorithm yields solutions (not necessarily minimum) for a given physical routing with junctions of degrees varying from 2 to 8 and guarantees a global minimal solution with respect to a given physical routing if the degrees of the junctions in the given routing are at most 3.

II. Basic Definitions

In addition to the definitions presented in Section I, the following terms are defined.

In a physical routing, the line segments are said to be on the *same axis* if (a) they are on opposite layers of the board, and (b) they will coincide when one layer is projected directly onto the other. As an example, a physical routing P_1 along with the terms defined above is illustrated in Fig. 1. In the figure, line segments l_1, l_2, l_3, and l_4 connecting component pins A and F constitute a net. Also line segments l_5 and l_6 are on the same axis, and so are l_7 and l_8.

The *assignment of line segments* to complete a physical routing is the designation of each line segment with a particular layer (layer I or layer II) of the board. In general, different assignments of line segments will have different numbers of vias. Thus it is desirable to develop an algorithm that will generate an optimal assignment of line segments in a given physical routing which requires a minimum number of vias.

A *transient routing*, denoted by T, is an incomplete physical routing in which the placement of the components and the routing of the nets have taken place, whereas the line segments have not yet been assigned. The transient routing is obtained from a given physical routing P by superimposing all the line segments of one layer on the other layer. In a transient routing, a *crux*, x_i, is an intersection in a physical routing P which cannot be made a via

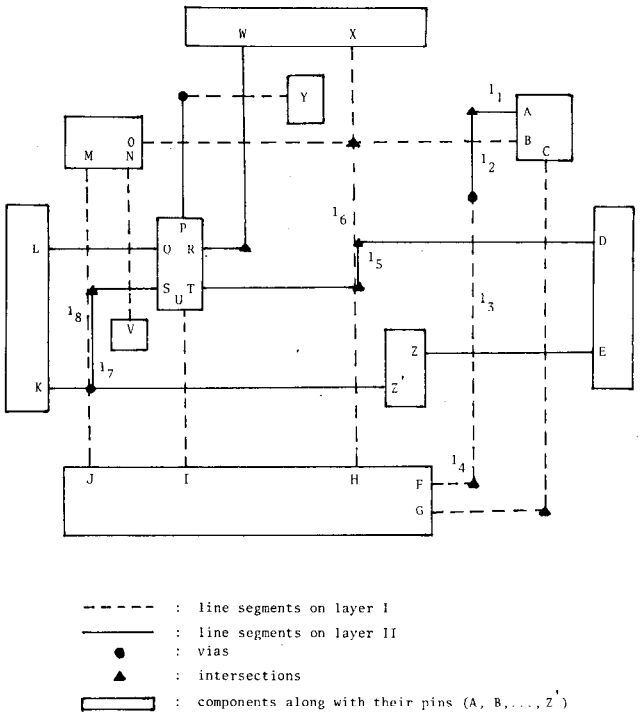


Fig. 1. Physical routing P_1.

(because of certain physical constraints). The region in a transient routing which covers two line segments on the same axis (on two different layers) between two cruxes is called a *crux zone*. Two or more crux zones may be combined to form an *extended* crux zone if they have common cruxes. If an intersection in P is not a crux, it is called a *via candidate*, denoted by h_i. A via in a physical routing P is also called a via candidate in the corresponding transient routing T. The region which covers two line segments on the same axis between two via candidates is referred to as a *via candidate zone* (*VCZ*). Two or more via candidate zones having common via candidates may be combined as an extended via candidate zone. Since a via can be chosen at *any point* within a via candidate zone, it follows that the entire zone is also referred to as a via candidate. Hence, each line segment in T must terminate at a component pin, a crux (zone), or a via candidate (zone).

As an example, the transient routing T_1 of a physical routing P_1 is shown in Fig. 2. In the figure, there is one crux zone (between cruxes x_1 and x_2) and 8 via candidates in which h_5 is a via candidate zone.

Note that as a starting point, a line segment l_i in T can be placed on either layer of the board, but all of its crossing line segments l_j's which belong to a different net in P must be placed on the other layer. According to this constraint, we shall introduce a term s_i, called the cluster which will play an important role in the via minimization algorithm (to be developed).

In the transient routing, a *cluster*, denoted by s_i, is a maximal set of mutually crossing line segments.

For example, in Fig. 2, if horizontal line segment l_1 is assigned to layer I of the board, then its crossing (vertical) line segments l_2 and l_3 must be assigned to layer II.

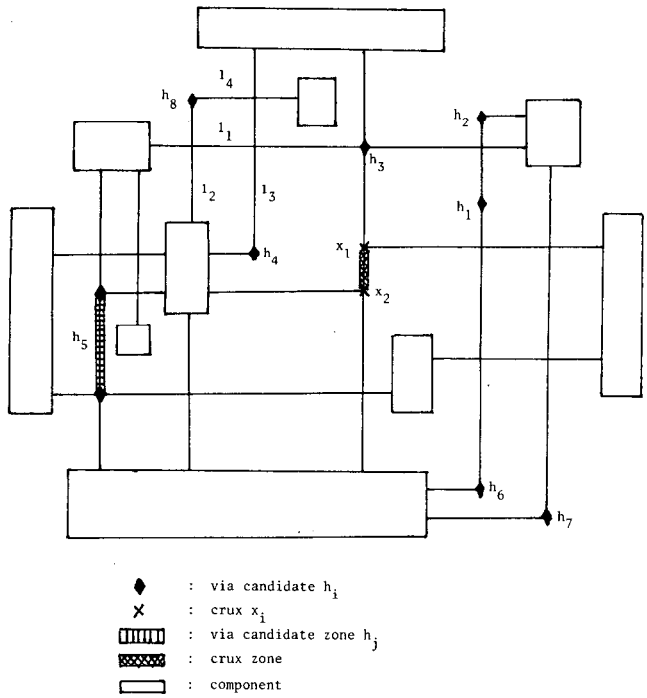


Fig. 2. Transient routing T_1 of P_1.

Similarly l_4 must be placed on layer I (since l_4 crosses l_3). The set s_1 containing all of these (mutually crossing) segments (l_1, l_2, l_3, and l_4) constitutes a cluster. It (along with other clusters) is also illustrated by a shaded closed curve, as shown in Fig. 3.

Moreover, clusters can be classified with respect to a given via candidate into three different types: (a) an *H-type cluster with respect to* $h_i(x_i)$ which is a cluster that contains exactly one horizontal line segment connected to the via candidate h_i (a crux x_i); (b) a *V-type cluster with respect to* $h_j(x_j)$ which is a cluster that contains exactly one vertical line segment connected to the via candidate h_j (a crux x_i); and (c) a *VH-type cluster with respect to* h_k is a cluster s_u that has horizontal and vertical line segments connected to the via candidate h_k. Note that in (c), the via candidate h_k connected with two line segments in the same cluster s_u is called an *essential via* of s_u.

In Fig. 3, it is seen that the transient routing T is partitioned into a number of clusters of types (a), (b), and (c) as described above, each of which is with respect to a via candidate (a crux). In the figure h_8 and h_5 are essential vias of s_1 and s_2, respectively.

Furthermore, subsequent to the assignment of line segments, clusters in the transient routing T can be separated into two classes C_1 or C_2 with *Class* C_1 containing those clusters in which horizontal (vertical) line segments are placed on layer I (II) and *Class* C_2 containing those clusters in which horizontal (vertical) line segments are placed on layer II (I).

III. Graph G_T

In this section, a topological structure, called graph G_T will be constructed as a representation of the entire transient routing T. G_T will consist of a number of subgraphs each of which represents a via candidate h_i along with its adjoining clusters. Based on this graph-theoretic representation, an efficient algorithm will then be developed (in Section V) for via minimization of a given physical routing. The treatment of cruxes will be deferred until Section VII.

Consider any via candidate h_i in T. The connected subgraph, $G_E(h_i)$, can be defined and constructed according to the following cases.

Case 1: The clusters contain no essential vias, thus excluding the *VH*-type clusters in T. If there exist two line segments on the same axis in T, h_i will be a via candidate zone; otherwise the number of clusters adjoining h_i will be limited to four. The subgraph $G_E(h_i)$ with respect to the via candidate (zone) h_i along with its m adjoining clusters in T is constructed as follows:

Step 1: Create a vertex v_{s_j} for each cluster s_j among all the adjoining clusters at h_i.

Step 2: Starting with the vertex corresponding to the top (or the upper leftmost cluster if there exist more than one such cluster on the same level) cluster adjoining h_i, place the m vertices in a circular arrangement in the same sequence as the clusters appear around a via candidate in T. The starting vertex will be called the *top vertex.*

Step 3: Label each vertex according to the type of clusters it represents (i.e., *V*-type or *H*-type). If all the vertices are of the same type, insert a dummy vertex of the opposite type to the right of the top vertex; and increase m by one.

Step 4: Starting with the top vertex, scan the m vertices in the clockwise direction. If any two consecutive vertices are of the same type, stop scanning and go to Step 5. Otherwise, when the scanning completes the full sequence, go to Step 6.

Step 5: Starting with the vertex found in Step 4, number the m vertices from 1 to m in the clockwise direction. Then proceed to Step 7.

Step 6: Starting with the top vertex, number the m vertices from 1 to m in the clockwise direction.

Step 7: Let k denote the number associated with a given vertex. Starting with the vertex for which $k = 1$, create an edge between vertex k and the nearest vertex of the opposite type in the clockwise direction. Increase k by one, and continue this operation until $k = m$.

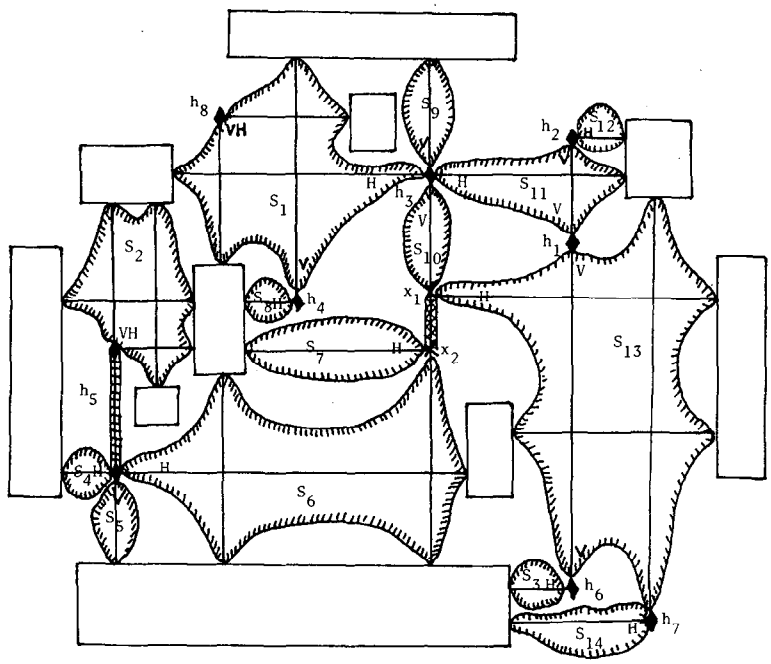
Thus for the m clusters adjoining h_i there are m vertices and $m-1$ edges in $G_E(h_i)$, called *similar edges*, all labeled e_{h_i}, with adjacent vertices corresponding to clusters of opposite types.

As an example of Case (1), clusters s_1, s_2, s_4, and s_3 in Fig. 4 are of types *V*, *H*, *V*, and *H*, respectively, with respect to h_2, and hence subgraph $G_E(h_2)$, shown in Fig. 5, contains vertices v_{s_1}, v_{s_2}, v_{s_4}, and v_{s_3}, interconnected by three similar edges e_{h_2}.

We shall now consider another example in which clusters adjoin a via candidate zone (VCZ).

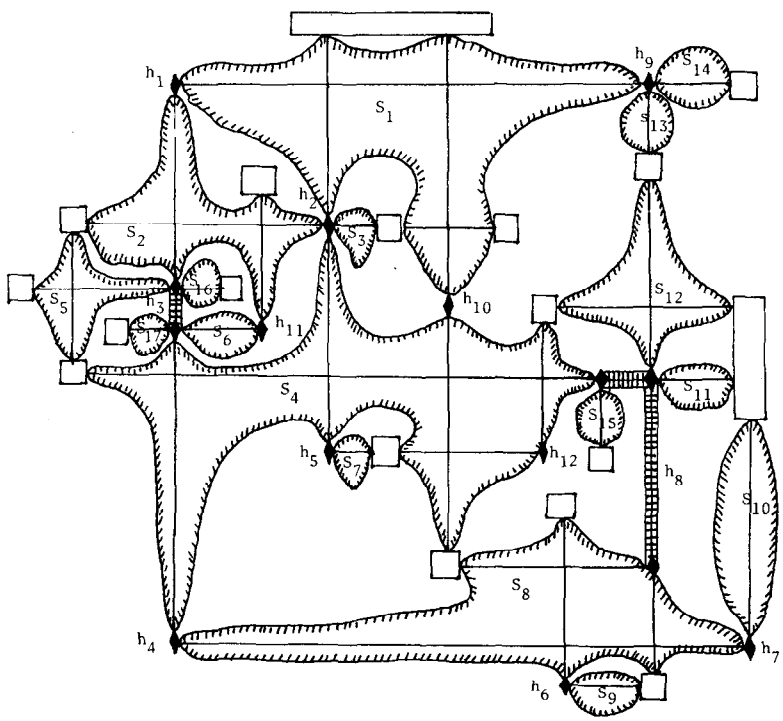
In Fig. 4, clusters s_2, s_5, s_{17}, s_4, s_6, and s_{16} adjoin VCZ h_3. The corresponding subgraph $G_E(h_3)$ is shown in Fig. 6. In Fig. 6(b), the number associated with each edge indicates the order in which the edges were created.

In the event that all clusters adjoining h_i are of the same type, subgraph $G_E(h_i)$ is constructed by introducing a



◆ : via candidate h_i
▥ : via candidate zone h_j
▨ : crux zone
◯ : cluster s_j
H : H-type cluster
V : V-type cluster
VH : VH-type cluster

Fig. 3. Partitioning T_1 into clusters.



◆ : via candidate h_i
▥ : via candidate zone h_j
◯ : cluster s_j

Fig. 4. Partitioning T_2 into clusters.

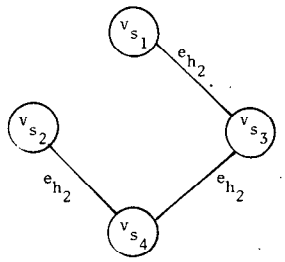


Fig. 5. Subgraph $G_E(h_2)$ of G_T.

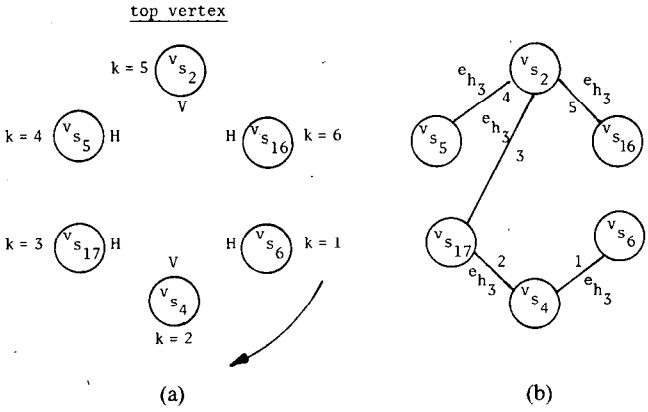


Fig. 6. Subgraph $G_E(h_3)$ of G_T. (a) Step 1 to 6. (b) Step 7.

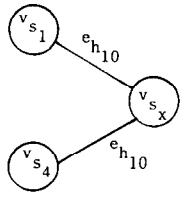


Fig. 7. Subgraph $G_E(h_{10})$ of G_T.

dummy vertex v_{s_x} (refer to Step 3). It corresponds to a "dummy" cluster s_x containing no line segments in T. Hence the addition of v_{s_x} will not affect the final solution.

For example, in Fig. 4, two V-type clusters s_1 and s_4 are joined at h_{10}. Thus in Fig. 7, a dummy vertex v_{s_x} of type H is created with similar edges $e_{h_{10}}$ linking it to v_{s_1} and v_{s_4}.

Case 2: The via candidate h_i is an essential via in T because of cluster s_u (since s_u is of type VH with respect to h_i). The subgraph $G_E(h_i)$ with respect to h_i is constructed as follows:

Step 1: If two or more clusters adjoin h_i, $G_E(h_i)$ is constructed as stated in Case (1) in which the VH-type cluster s_u is treated as either an H-type or a V-type cluster; otherwise proceed to Step 2.

Step 2: A self-loop[1] with label e_{h_i} is created at vertex v_{s_u}.

For example, via candidate h_{12} and via candidate zone h_8 in Fig. 4 are essential vias of clusters s_4 and s_8, respectively. The corresponding subgraphs $G_E(h_{12})$ and $G_E(h_8)$ are shown in Figs. 8 and 9, respectively. In Fig. 8, $e_{h_{12}}$ is a self-loop; and in Fig. 9, a path consisting of similar edges

[1]The construction of a self-loop with label e_{h_i} in G_T will ensure that the via candidate h_i will be a via in the final routing. Further discussion may be found in Section IV.

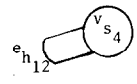


Fig. 8. Subgraph $G_E(h_{12})$ of G_T.

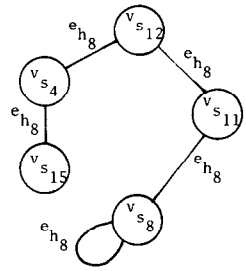


Fig. 9. Subgraph $G_E(h_8)$ of G_T.

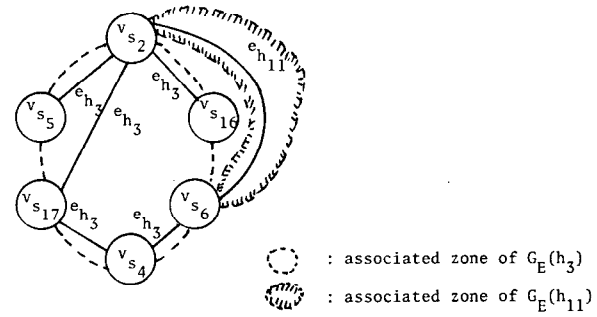


Fig. 10. The connections of subgraphs $G_E(h_3)$ and $G_E(h_{11})$.

e_{h_8}'s is constructed between vertices $v_{s_{15}}$ and v_{s_8} (of type V) and a self-loop is constructed at v_{s_8}.

Next, graph G_T is formed by joining together the n $G_E(h_i)$ subgraphs (where $i = 1,2,\cdots,n$) as follows:

Step 1: Enclose each subgraph $G_E(h_i)$ by a simple closed curve J_i with the enclosed region referred to as the *associated zone* of $G_E(h_i)$.

Step 2: Consider two subgraphs $G_E(h_i)$ and $G_E(h_j)$. If they have a set of *common* vertices (i.e., vertices with the same labels), they are to be connected by coalescing all common pairs of vertices in both graphs in such a manner that their associated zones will not overlap.

Step 3: Repeat Step 2 until all the n subgraphs have been treated as above.

Step 4: Remove all the n simple closed curves associated with the n subgraphs in the resultant graph, which yields G_T.

As an example, subgraphs $G_E(h_{11})$ and $G_E(h_3)$ obtained from T_2 of Fig. 4 are connected, as shown in Fig. 10. However, if the two subgraphs are joined, as shown in Fig. 11, they will be incorrectly connected since $e_{h_{11}}$ is inside the associated zone of $G_E(h_3)$, thus causing overlap between their associated zones.

As another example, consider the transient routing T_2 of Fig. 4 which has 17 clusters and 12 via candidates. The corresponding graph G_T is constructed by connecting all of the 12 $G_E(h_i)$ subgraphs and the result is shown in Fig. 12.

In order to make use of certain theorems which depend upon the planarity of G_T, we must first demonstrate the fact that G_T is planar.

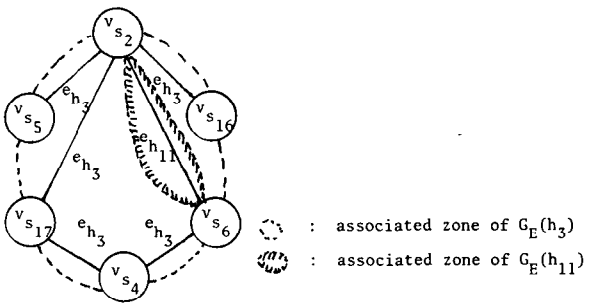


Fig. 11. The incorrect connections of subgraphs $G_E(h_3)$ and $G_E(h_{11})$.

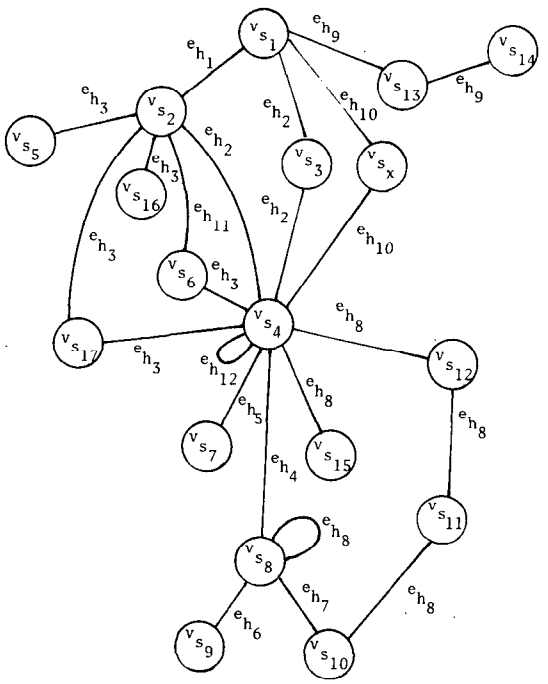


Fig. 12. Graph G_T of T_2.

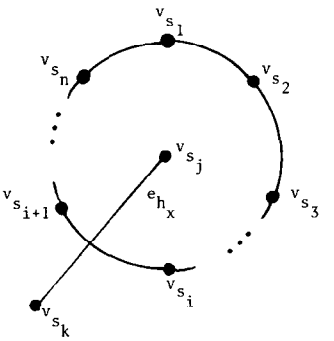


Fig. 13. An example of graph G_T.

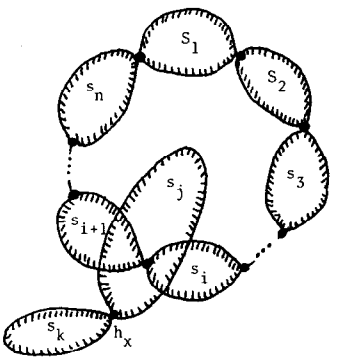


Fig. 14. Corresponding T of G_T in Case (a) of the proof of Theorem 1.

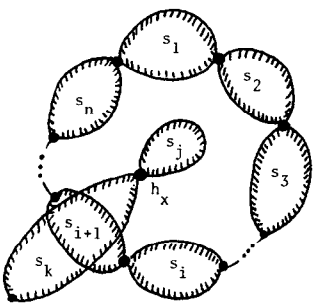


Fig. 15. Corresponding T of G_T in Case (b) of the proof of Theorem 1.

Consider the plane on which a given transient routing T is drawn. In the partitioned transient routing T (such as the one shown in Fig. 3), each line segment belongs to one and *only* one cluster. Thus the clusters are drawn on the plane such that no two overlap. This fact results in the following theorem.

Theorem 1: The graph G_T of a given transient routing T is planar.

Proof: For G_T to be nonplanar, there exists an edge e_{h_x} connected between some vertices v_{s_j} and v_{s_k} crossing some circuit C_i with v_{s_j} enclosed by C_i and v_{s_k} outside C_i (as shown in Fig. 13). We shall show that such a situation cannot exist as there cannot be such a one-to-one correspondence between G_T and the corresponding transient routing T.

Note that a vertex in G_T corresponds to a cluster in T and an edge between two vertices in G_T corresponds to a common via candidate between some two clusters in T. Thus in Fig. 13, if v_{s_j} and v_{s_k} are connected by edge e_{h_x} which crosses some circuit C_i, the two clusters s_j and s_k in T (corresponding to v_{s_j} and v_{s_k}, respectively, in G_T) must be joined by a common via candidate h_x satisfying one of the following cases:

Case (a): Cluster s_j overlaps some clusters corresponding to C_i, as shown in Fig. 14.

Case (b): Cluster s_k overlaps some clusters corresponding to C_i, as shown in Fig. 15; or

Case (c): Both clusters s_j and s_k are joined at a via candidate which is also a common via candidate to two of those clusters corresponding to C_i, as shown in Fig. 16.

However, both Cases (a) and (b) violate the properties of clusters that no two of them overlap; and in Case (c), the corresponding subgraph must contain three similar edges (denoted by e_{h_x}), as shown in Fig. 17, with some two of these edges contained in the circuit and no crossing, thus contradicting our assumption (as illustrated in Fig. 13). Hence G_T must be planar.

IV. Via Minimization Problem Analysis

It has been shown in the preceding section how a transient routing T can be represented by a planar graph G_T.

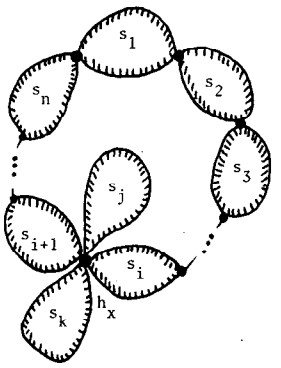


Fig. 16. Corresponding T of G_T in Case (c) of the proof of Theorem 1.

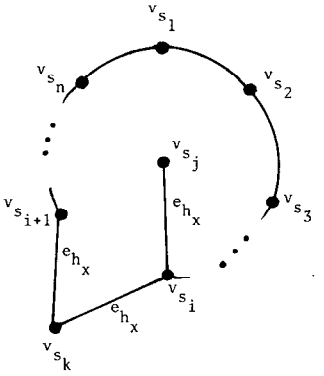


Fig. 17. The corresponding G_T of T shown in Fig. 16.

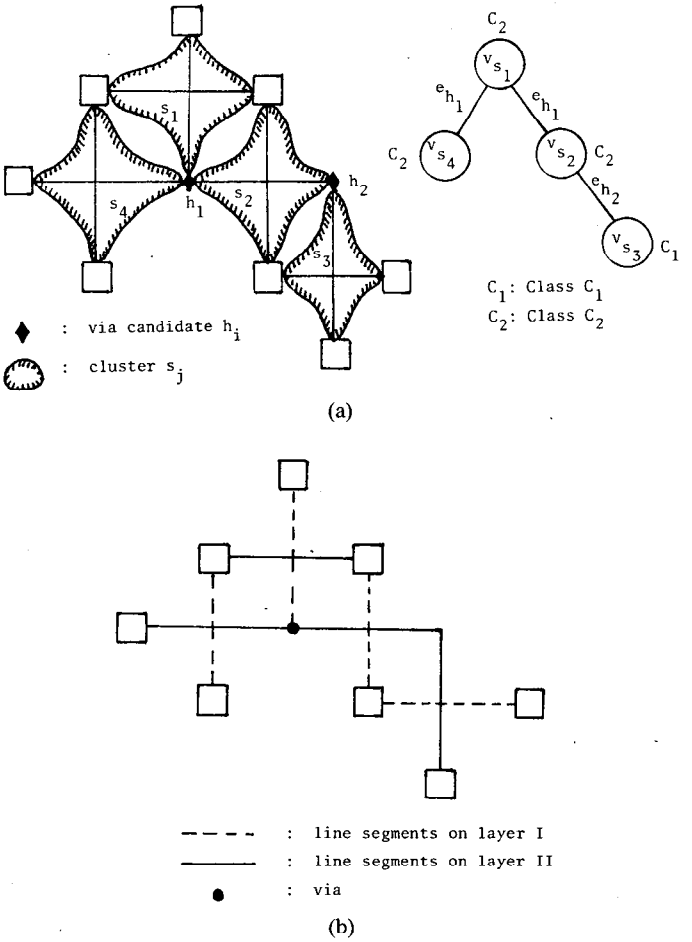


Fig. 18. (a) Transient routing T_3 and its corresponding G_T. (b) Final solution of physical routing P_3 corresponding to T_3.

We will demonstrate next how the via minimization problem can be attacked by solving an equivalent graph-theoretic problem.

Consider a given transient routing T; separate all the clusters in T into two classes C_1 and C_2 as defined in Section II. Similarly, all the vertices of G_T will also be divided into two classes C_1 and C_2. Note that assigning all the vertices of G_T to two classes, C_1 and C_2, is equivalent to assigning all the line segments of the transient routing to the two layers of the board.

Next note that in G_T, an edge e_{h_i} connecting two vertices v_{s_p} and v_{s_q} will correspond to a via in the physical routing if and only if v_{s_p} and v_{s_q} are assigned to the same class (C_1 or C_2). For example, in Fig. 18(a), v_{s_1}, v_{s_2}, and v_{s_4} have been assigned to class C_2; hence, two similar edges with label e_{h_1} of Fig. 18(a) correspond to one via as shown in Fig. 18(b). If G_T is a bipartite graph, the physical routing will contain no vias. In general, G_T will not be a bipartite graph. Hence, vias do exist, and the total number of vias in a physical routing P is equal to the number N of *distinct* labels used in identifying similar edges in G_T which connect vertices of the same class. The number N is referred to as the *Boolean cardinality* (or simply the *cardinality*) of edges in G_T. For example, in Fig. 19, the total number of edges corresponding to vias is 3. However, the edges with the same label represent a single via. Hence, the Boolean cardinality of this set of edges is 1.

Next, the following definitions will be introduced and used in our theoretical development of the via minimization algorithm. A circuit that consists of an odd number of edges is referred to as an *odd circuit*. A set of edges whose removal leaves a subgraph free of odd circuits is called an *odd circuit cover* (*OCC*). Such a subgraph is always bipartite [8]. Thus it follows from the discussion in the preceding paragraph that the total number of vias in a physical routing P is equal to the cardinality of edges in an OCC of G_T. Among all possible OCC's in G_T, those with minimum cardinality of edges will be called *minimum odd circuit covers* (*MOCC's*). Consequently the problem of finding the minimum number of vias in P is equivalent to the problem of finding an MOCC of G_T. Note that the *noncircuit edges* (i.e., those edges not contained in any circuit) in G_T may be removed since they will not create any vias, and

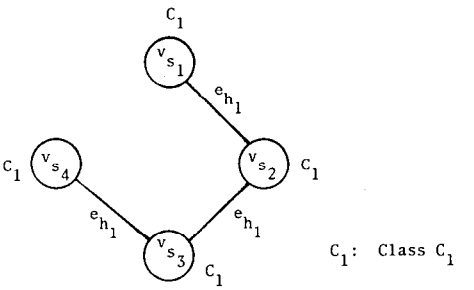


Fig. 19. An example of the cardinality of edges.

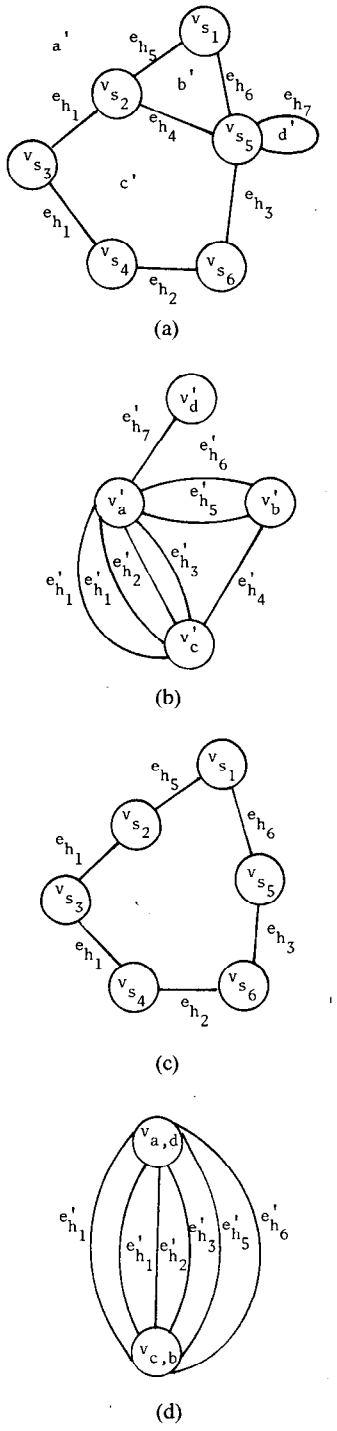


Fig. 20. An example of graph G_T and its dual G'_T with an MOCC and an MOVP. (a) Graph G_T. (b) Dual graph G'_T. (c) Resultant subgraph of G_T after removing edges e_{h_4} and e_{h_7}. (d) Resultant subgraph of G'_T after contracting edges e'_{h_4} and e'_{h_7}.

hence their removal will not affect the final assignment of vias.

Note also that it is more convenient to work with the dual graph G'_T of G_T in order to obtain an MOCC in G_T since in determining an MOCC, one must first identify the odd circuits in G_T, and in so doing it is easier to deal with the degrees of the vertices in G'_T than to work with the lengths of the circuits in G_T. The subgraph of G'_T which corresponds to the subgraph $G_E(h_i)$ of G_T will be called $G'_E(h_i)$. The edges of $G'_E(h_i)$ are also called the similar edges with respect to h_i. For example, the dual graph G'_T of a given G_T, shown in Fig. 20(a), is illustrated in Fig. 20(b). In G'_T, subgraph $G'_E(h_1)$ has two vertices, v'_a and v'_c, and two similar e'_{h_1} edges.

In G'_T, a set of edges whose contraction leaves a subgraph free of vertices of odd degree (hereafter referred to as *odd vertices*) is called an *odd vertex pairing* (OVP). It has been shown that an edge set is an OCC of a planar graph G if and only if the corresponding edge set in its dual graph G' is an OVP of G' [9], [10]. Thus the total number of vias in P is equal to the cardinality of edges in an OVP of the dual graph G'_T. Also, a *minimum odd vertex pairing* (*MOVP*) is defined as an OVP with the minimum cardinality of edges among all possible OVP's in a graph G. Consequently, the problem of finding an MOCC of G_T is equivalent to the problem of determining an MOVP of G'_T. For example, in Fig. 20(a), the set of edges $\{e_{h_4}, e_{h_7}\}$ of G_T is an MOCC because its removal leaves the resultant subgraph bipartite, and is shown in Fig. 20(c). The set of edges $\{e'_{h_1}, e'_{h_7}\}$ in G'_T of Fig. 20(b) is the corresponding edge set of $\{e_{h_4}, e_{h_7}\}$ in G_T. Hence, in G'_T, $\{e'_{h_4}, e'_{h_7}\}$ is an MOVP whose contraction leaves the resultant subgraph free of odd vertices, as illustrated in Fig. 20(d).

To form an OVP, one need only contract those paths which connect the odd vertices (refer to Fig. 20(d)). Furthermore, it has been shown that for any graph an MOVP (with minimum cardinality of edges) consists of a collection of paths P having odd vertices as endpoints. Each odd vertex is used once and only once as an endpoint [9]. Consequently, an MOVP of G'_T can be determined by finding a collection of paths P with the minimum cardinality of edges.

V. An Algorithm for Minimizing the Number of Vias

In this section an algorithm for minimizing the number of vias for a given physical routing will be developed. It will be shown that, under certain cases, the algorithm will provide the minimum number of vias.

A method for finding an MOVP in G'_T which can be executed in polynomial time will now be given. In order to facilitate the discussion, the following terms are introduced.

A *matching* is a set of edges which have no common vertices. A *full matching* in a graph G is a matching containing all the vertices of G.

Also a *minimum* (*maximum*) *weighted matching* in a weighted graph is defined as a full matching for which the sum of the weights of the edges is minimum (maximum). A path connecting endpoints v_a and v_b in graph G with minimum cardinality of edges is referred to as a *shortest path* between v_a and v_b.

Next, to develop the method for determining an MOVP in G'_T, a complete weighted graph G_w is defined and constructed as follows:

Step 1: Represent each of the odd vertices in G'_T by a vertex in G_w.

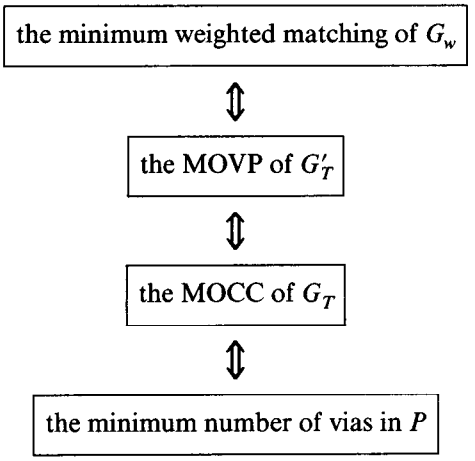
Step 2: For each pair of odd vertices v'_a and v'_b in G'_T, create an edge e_i in G_w between the corresponding vertices v_a and v_b, respectively.

Step 3: Assign the weight w_i to each edge e_i where w_i is the cardinality of edges of a shortest path connecting endpoints v'_a and v'_b in G'_T.

Note that the number of odd vertices in any graph is always even. Hence, the number of vertices in G_w will be even. Further, each edge e_i in G_w will correspond to a shortest path between some odd vertices in G'_T. It follows, from the discussion in Section IV, that a full matching in G_w corresponds to an OVP in G'_T. Consequently, the problem of finding an MOVP of G'_T can be translated into that of finding a minimum weighted matching of G_w. Each edge e_i in the minimum weighted matching of G_w corresponds to a shortest path P_i connecting v'_a and v'_b in G'_T ($i = 1,, \cdots, t$ where t is one half of the total number of vertices in G_w). If there is more than one such path between v'_a and v'_b, choose one of them arbitrarily. The edges contained in P_i belong to an MOVP. It follows that an MOVP consists of the set of edges contained in $P_1 \cup P_2 \cup \cdots \cup P_t$.

Now, the algorithm for minimizing the number of vias for a given physical routing P is stated as follows:

The overall methodology for via minimization is summarized below.



Now, the algorithm for minimizing the number of vias for a given physical routing P is stated as follows:

Step 1: Obtain the transient routing T from the given physical routing P by superimposing all the line segments of one layer on the other layer.

Step 2: Partition T into clusters.

Step 3: Construct the corresponding graph G_T of T as described in Section III.

Step 4: Obtain the graph $\bar{G}_T$ by contracting all the noncircuit edges (if any) in G_T.

Step 5: Obtain the dual graph $\bar{G}'_T$ of $\bar{G}_T$.

Step 6: Scan all the odd vertices of $\bar{G}'_T$, find a shortest path between each pair of odd vertices, and construct the complete weighted graph G_w.

Step 7: Find a minimum weighted matching of G_w obtained from Step 6.

Step 8: Obtain the set of edges, MOVP, in $\bar{G}'_T$ which corresponds to the minimum weighted matching found in Step 7.

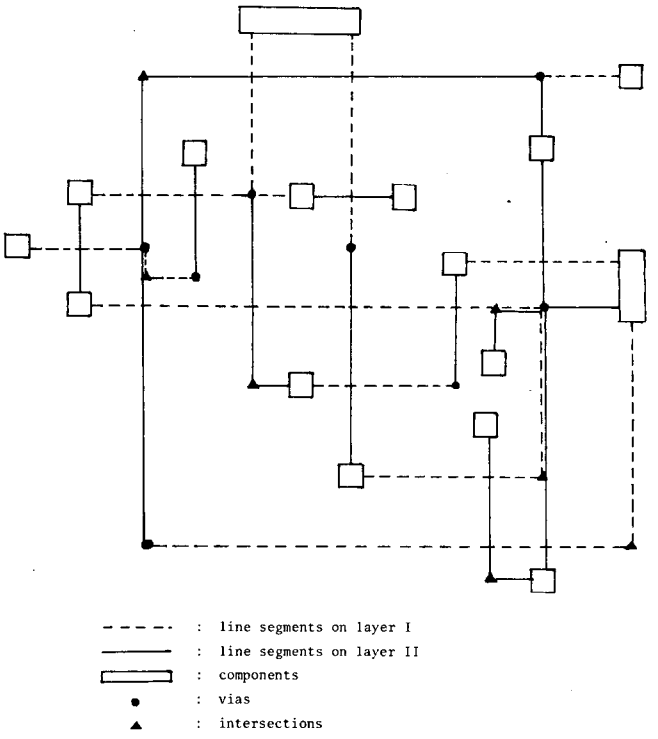


Fig. 21. Physical routing P_4.

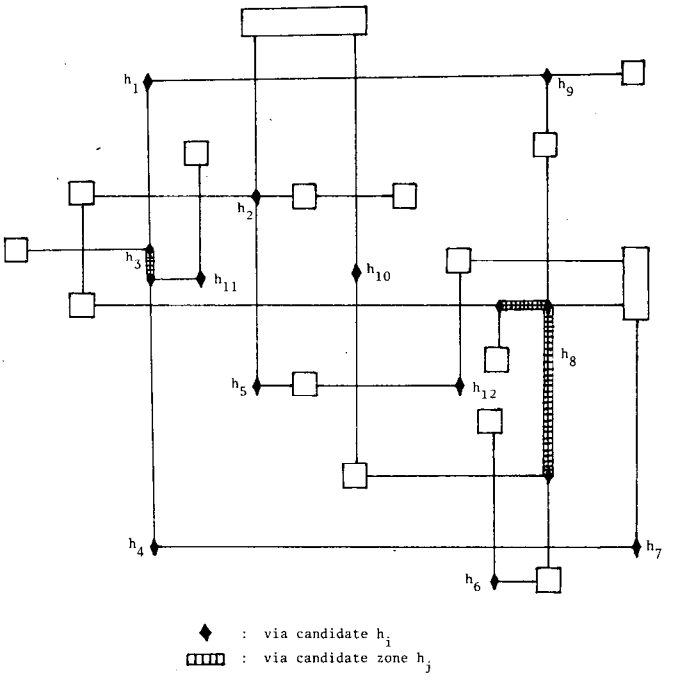


Fig. 22. Transient routing T_4 of P_4.

Step 9: In G_T, remove set of edges, MOCC, which corresponds to the MOVP found in Step 8 such that the resultant graph is bipartite. If any similar edge e'_{h_i} is contained in the MOVP of $\bar{G}'_T$, all the corresponding edges e_{h_i} in G_T are also removed.

Step 10: Starting from any arbitrary vertex, assign class C_1 or C_2 alternately, to each adjacent vertex in each connected bipartite subgraph.

Step 11: Assign each line segment of each cluster in T to a particular layer of the board according to the class

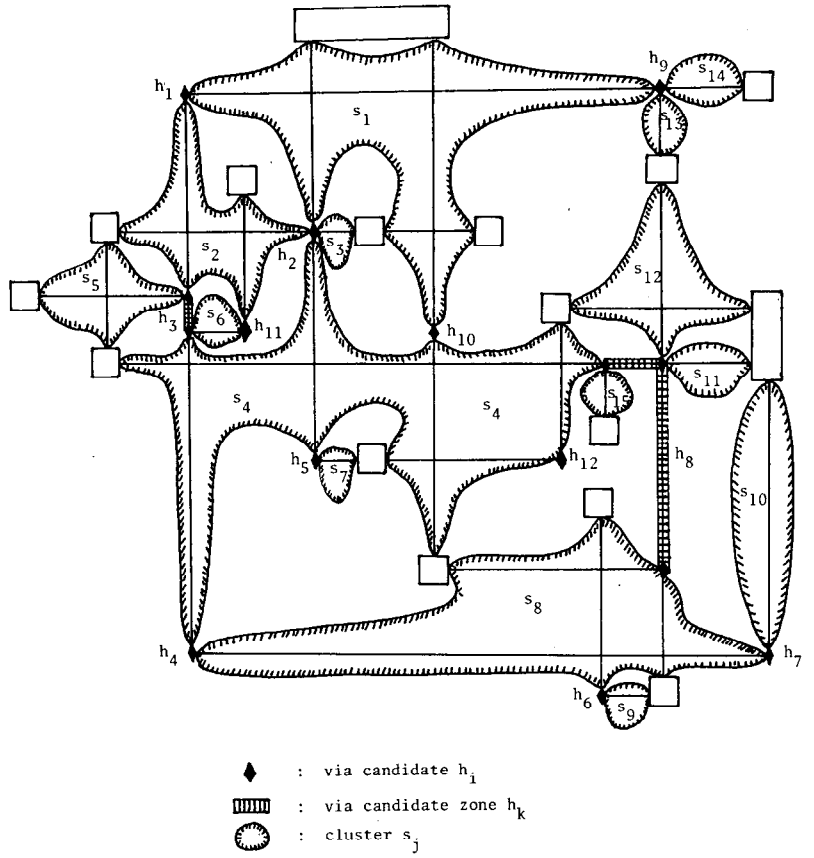


Fig. 23. Partitioning T_4 into clusters.

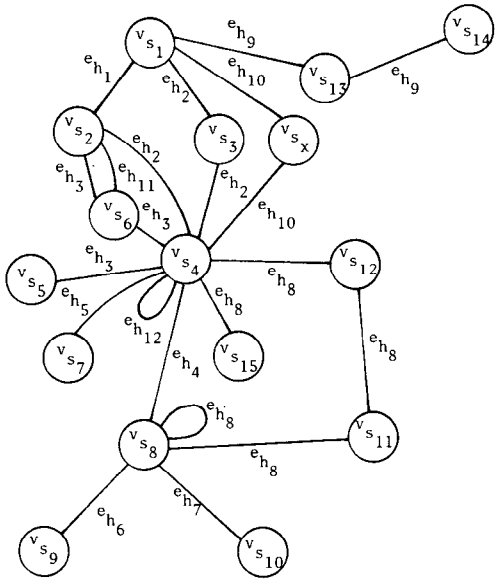


Fig. 24. Graph G_T of T_4.

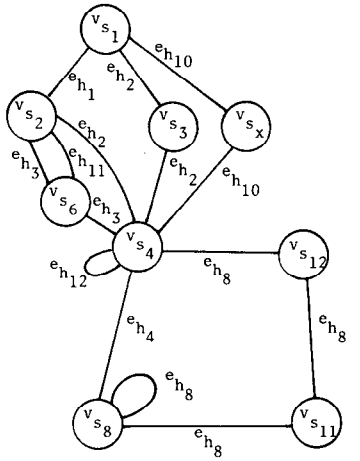


Fig. 25. Graph $\bar{G}_T$ of graph G_T.

assignment in Step 10, assign two same axis line segments in each VCZ (if any), and the process is complete.

The following example illustrates the steps of the algorithm.

Example 1: Minimize the number of vias in a given physical routing P_4, as shown in Fig. 21.

Step 1: Transient routing T_4 of P_4 is obtained, as shown in Fig. 22.

Step 2: Transient routing is partitioned into clusters, as shown in Fig. 23.

Step 3: Graph G_T of T_4 is obtained, as illustrated in Fig. 24.

Step 4: Graph $\bar{G}_T$ of G_T is obtained, as illustrated in Fig. 25.

Step 5: Dual graph $\bar{G}'_T$ of $\bar{G}_T$ is obtained, as shown in Fig. 26.

Step 6: Graph G_w of $\bar{G}'_T$ is obtained and is illustrated in Fig. 27.

Step 7: The minimum weighted matching $\{\overline{v_c v_e}, \overline{v_g v_f}\}$ is found.

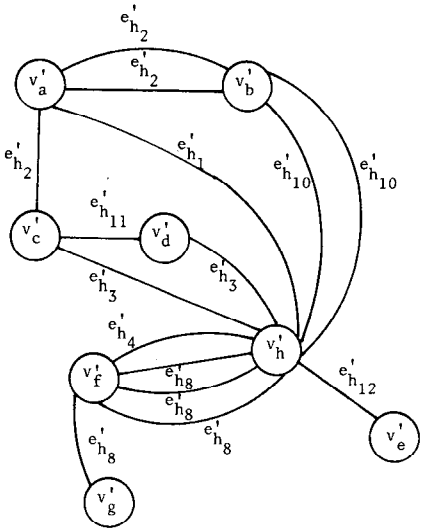


Fig. 26. Dual graph $\bar{G}'_T$ of $\bar{G}_T$.

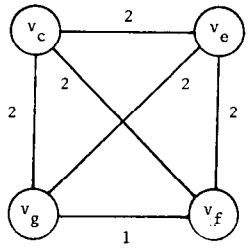


Fig. 27. Graph G_w of $\bar{G}'_T$.

Step 8: The edges $\overline{v_c v_e}$ and $\overline{v_g v_f}$ of G_w found in Step 7 correspond to two shortest paths between odd vertices, v'_c and v'_e, v'_g, and v'_f, in $\bar{G}'_T$. Thus the MOVP $\{e'_{h_3}, e'_{h_8}, e'_{h_{12}}\}$ of $\bar{G}'_T$ is obtained.

Step 9: The MOCC $\{e_{h_3}, e_{h_8}, e_{h_{12}}\}$ of G_T is obtained. Edge $e_{h_{12}}$ (self-loop), and all the similar edges e_{h_3}'s and e_{h_8}'s (with respect to h_3 and h_8, respectively) of G_T are removed, as shown in Fig. 28.

Step 10: To each adjacent vertex is assigned class C_1 or C_2 alternatively in each bipartite subgraph, as shown in Fig. 28. Isolated vertices may be assigned to classes arbitrarily.

Step 11: The final assignment for P is shown in Fig. 29. Note, any point in a VCZ can be a via.

Thus three vias (h_3, h_8, and h_{12}) are necessary for the given physical routing P_4.

The algorithm given above does not guarantee the minimum number of vias for all cases because Step 7 carries out the summing of weights arithmetically, and it cannot determine the *Boolean* cardinality of edges in a given path. Thus nonminimal solutions will result in the cases in which any two paths in an MOVP have similar edges in common.

For example, the graph, as shown in Fig. 30, is the dual graph $\bar{G}'_T$ of a given physical routing P (omitted here due to its complexity). In this figure, there are 8 odd vertices v'_a, v'_b, v'_c, v'_d, v'_e, v'_f, v'_g, and v'_h. Edges labeled e'_{h_1} and e'_{h_2} are the only similar edges. The corresponding complete

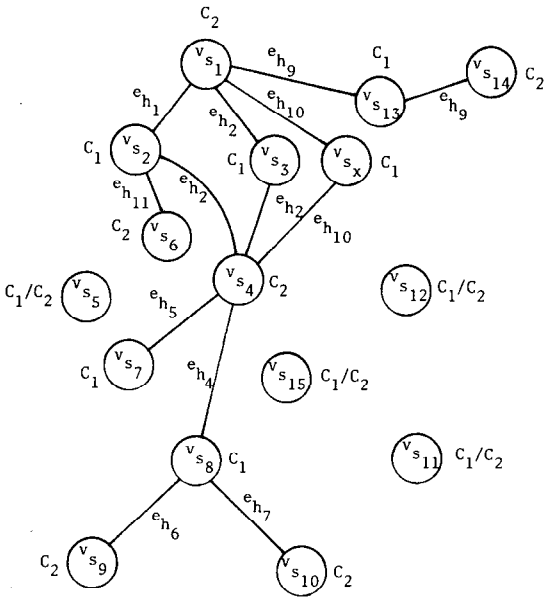


Fig. 28. Bipartite subgraph after removing edges $e_{h_{12}}$, e_{h_8}, and e_{h_3}.

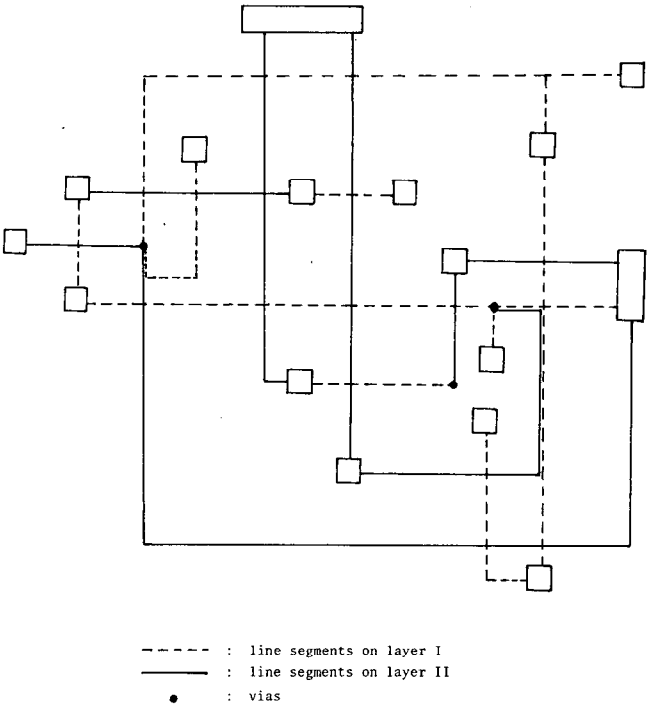


Fig. 29. Final assignment of P_4.

weighted graph G_w is illustrated in Fig. 31. In the figure, only the weights of edges less than 5 are shown for clarity. The minimum weighted matching of G_w, $\langle \overline{v_a v_b}, \overline{v_c v_e}, \overline{v_d v_f}, \overline{v_g v_h} \rangle$, has weight 13. However, there exists another matching, M, in G_w, $\langle \overline{v_a v_c}, \overline{v_e v_g}, \overline{v_b v_d}, \overline{v_f v_h} \rangle$, sum of whose weights is 14 (not minimum). The *Boolean* cardinality of edges of the OVP in $\bar{G}'_T$ which corresponds to matching M of G_w is 12, one less than the minimum, 13. Therefore, the solution obtained from the proposed algorithm is not optimal.

However, it is worth noting that this example is somewhat artificial. In this case the nonminimal solution requires

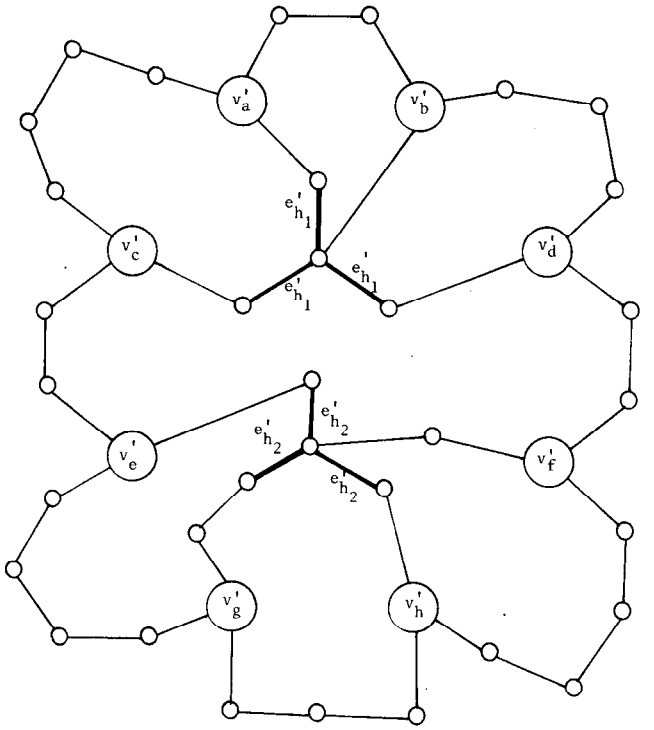


Fig. 30. An example of $\bar{G}'_T$ corresponding to P.

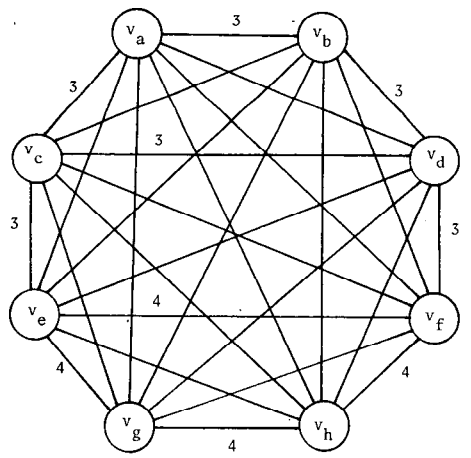


Fig. 31. Graph G_w of $\bar{G}'_T$, as shown in Fig. 30.

13 vias (other than 12, the minimum number), only one more via than that required in the minimal solution. Further, it is believed that such a case (with a nonminimal solution) will not be encountered frequently in actual layout applications.[2]

As mentioned earlier, since a nonminimal solution occurs when there exist common similar edges between two paths in a minimum odd vertex pairing (MOVP) in $\bar{G}'_T$, the dual of $\bar{G}_T$, we state the following theorem giving the condition in which the algorithm will provide an optimal solution.

Theorem 2: For a given physical routing, the proposed algorithm always provides the minimum number of vias when every subgraph $G'_E(h_i)$ of $\bar{G}'_T$ has three or few vertices.

Proof: We shall prove the theorem by contradiction. However, to facilitate our discussion in the proof we shall first define the following terms.

For an OVP, O_i, in $\bar{G}'_T$, $O_i = \{P_1^i, P_2^i, \cdots, P_t^i\}$ where each P^i term is the set of all edges in a shortest path between some two odd vertices in $\bar{G}'_T$ and t is one half of the total number of odd vertices in $\bar{G}'_T$.

$$U(O_i) \triangleq P_1^i \cup P_2^i \cup \cdots \cup P_t^i.$$

$BC\{U(O_i)\} \triangleq$ the Boolean cardinality of the edges in $U(O_i)$, and

$\Sigma O_i \triangleq$ the sum of the Boolean cardinalities of all the p^i terms in O_i.

Since, for any given OVP, we have

$$\Sigma O_i \geqslant BC\{U(O_i)\}. \tag{1}$$

based on the above definitions.

Note that the solution generated by the proposed algorithm with the number of vias equal to ΣO_i is minimum when the equality sign holds in (1), i.e., $\Sigma O_i = BC\{U(O_i)\}$; and is nonminimal otherwise. In the following, we shall show that $\Sigma O_i = BC\{U(O_i)\}$ must hold when each $G'_E(h_i)$ of $\bar{G}'_T$ contains three or fewer vertices.

Let $O_j = \{P_1^j, P_2^j, \cdots, P_t^j\}$ be one of those OVP's having the minimum $BC\{U(O_i)\}$, namely an MOVP, among all the OVP's. Also assume that O_j has the minimum ΣO_j among all the MOVP's.

Next, consider an OVP, O_m, in $\bar{G}'_T$ which is not an MOVP but has the minimum ΣO_m among all the OVP's. It follows that

$$BC\{U(O_m)\} > BC\{U(O_j)\} \tag{2}$$

and

$$\Sigma O_m \leqslant \Sigma O_j. \tag{3}$$

Also,

$$\Sigma O_m \geqslant BC\{U(O_m)\}. \tag{4}$$

From (2), (3), and (4) we have

$$\Sigma O_j > BC\{U(O_j)\}. \tag{5}$$

This exists only when O_j includes at least two paths, say P_1^j and P_2^j, which have common similar edges e'_{h_i}.

Since each subgraph $G'_E(h_i)$ has at most 3 vertices, there always exists another pair of paths, P_1^n and P_2^n, such that one of them, say P_1^n, bypasses $G'_E(h_i)$, as shown in Fig. 32. It is seen that the similar edges e_{h_i}'s are no longer common to paths P_1^n and P_2^n.

Hence, another OVP exist in $\bar{G}'_T$, namely

$$O_n = \{P_1^n, P_2^n, P_3^j, \cdots, P_t^j\} \tag{6}$$

[2]A survey was conducted among some electronic firms in the Bay Area (GTE-Lenkurt, GTE-Sylvania, Computer Vision, and Silvar-Lisco) and in Japan (Nippon Electric Co. and Sony). Also, detailed inspections were made of a number of PC layouts at several PC board manufacturers in the Bay Area (Printex, Norcal, M. B. C., and San Jose Circuits). The results were that the potentially nonminimal cases (i.e., each via candidate of degree 4 or more) occurred in less than 1 percent of all (over 2000) the layouts examined.

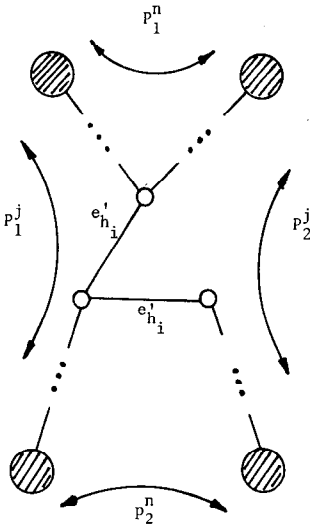


Fig. 32. An example of subgraph $G'_E(h_i)$ with 3 vertices.

which is obtained by replacing P_1^j by P_1^n and P_2^j by P_2^n in O_j. From Fig. 32,

$$BC\{U(O_j)\} = BC\{U(O_n)\} \tag{7}$$

and

$$\Sigma O_n = \Sigma O_j - 1. \tag{8}$$

Equation (7) shows that O_n is an MOVP. However, from (8), we have

$$\Sigma O_n < \Sigma O_j \tag{8}$$

which contradicts our initial assumption that ΣO_j is minimum among all MOVP's. It follows that O_n must be an MOVP with the minimum ΣO_n consisting of a set of shortest paths in which no two of them contain any common similar edges (i.e., $\Sigma O_n - BC\{U(O_n)\} = 0$). Hence, O_n (a solution generated by the algorithm with the minimum ΣO_n) corresponds to an optimal solution. This completes the proof.

In Example 1, the algorithm ensures the minimum number of vias because each $G'_E(h_i)$ of $\bar{G}_T$ has at most 3 vertices although in the given physical routing there exists a via candidate (h_8) of degree 6.

In general, the routing in which each via candidate h_i is of degree 3 or less, the corresponding $G'_E(h_i)$ has at most three vertices. Therefore, the following corollary is evident and the proof is omitted.

Corollary 1. In a transient routing T, if each via candidate is of degree 3 or less, the algorithm always provides the minimum number of vias.

VI. Complexity of the Algorithm

In the proposed algorithm given earlier, each step can be carried out using techniques that can be executed in polynomial time. In Step 2, each line segment is searched once. Thus the total time spent on Step 2 is $O(L^2)$ where L is the total number of line segments in T. In Step 3, G_T is obtained and is represented by its incidence matrix. This process is implemented by the procedure OM in program GRAPH-GT [11]. Inspection of the listing of OM shows the algorithm has a worst-case time bound of $O(v^3)$ where v is the total number of via candidates in T. In Step 4, G_T is obtained by contracting the noncircuit edges within G_T. The process is implemented by the procedure REDUCE in program GRAPH-GT. Inspection of the listing of REDUCE shows the total time spent on Step 4 is $O(v^3)$ where v is the number of via candidates in T. In Step 5, the dual graph $\bar{G}'_T$ of $\bar{G}_T$ can be obtained directly from the mesh matrix of $\bar{G}_T$ which in duality is an incidence matrix of $\bar{G}'_T$. The process is implemented by the procedure MESH in program GRAPH-GT. Inspection of the listing of procedure MESH shows the overall time spent on Step 5 is $O(v)$ where v is the total number of via candidates in T. In Step 6, Dijkstra's algorithm [12] can be applied to the graph which is obtained from $\bar{G}'_T$ by adding the similar edges e_{h_i}'s, if any, to make each $G'_E(h_i)$ a complete subgraph in $\bar{G}'_T$ to form a new graph $\bar{G}^*_T$, the minimum path length between any two vertices is equal to the shortest path between the two vertices. Hence, the time spent on Step 6 is $O(n^3)$ where n is the number of vertices in $\bar{G}'_T$. In Step 7, the task of finding a minimum weighted matching of G_w may be posed as a maximum matching problem by assigning the weight $\bar{w}_i = K - w_i$ to each edge, e_i, in G_w, where K is any integer greater than the maximum weight of all edges in G_w, and w_i is the weight of edge e_i in G_w. The resulting complete weighted graph will be denoted $\bar{G}_w$. Consequently, with this new assignment of weights to the edges of G_w, the problem of finding a minimum weighted matching in G_w is equivalent to that of finding a maximum weighted matching in $\bar{G}_w$, which can, in turn, be solved by applying the well-known algorithm of Edmonds [13]. The complexity of the algorithm of Edmonds is $O(m^3)$ where m is the number of all odd vertices in $\bar{G}'_T$. In Step 8, the determination of the MOVP from $\bar{G}'_T$ requires negligible execution time. In Step 9, each edge of G_T is scanned once, therefore, the time spent on Step 9 is $O(v)$ when v is the number of via candidates in T. In Step 10, the total time spent on class assignment to the vertices of G_T is $O(u)$ where u is the number of vertices in G_T. In Step 11, since each line segment is scanned once, the total time spent is $O(L^2)$ (similar to Step 2). Therefore, the overall computational complexity of the algorithm is of order equal to $\max\{O(n^3), O(v^3), O(L^2)\}$.

VII. Considerations on Preassignment

In some instances, certain requirements are placed on physical routings; e.g., some line segments may have to be placed on a particular layer of the board due to prescribed board or component constraints. This requirement will be referred to as the *preassignment of the line segments* (or *preassignment* for short). The preassignment of certain line segments to particular layers of the board is equivalent to assigning the clusters which contain these line segments to

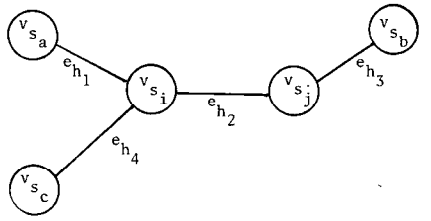


Fig. 33. An example of a graph G_T.

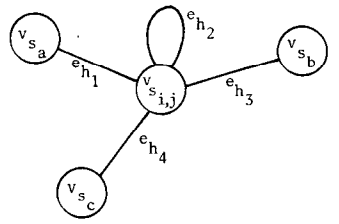


Fig. 34. Modified graph G_T of Fig. 33 of Case (a).

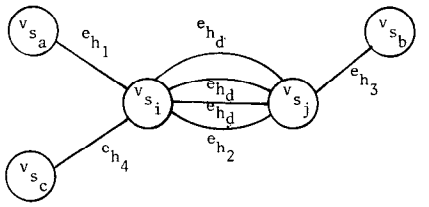


Fig. 35. Modified graph G_T of Fig. 33 of Case (b).

a particular class (class C_1 or C_2). Only two cases need be considered:

Case (a): The preassignment requires that a pair of cluster, say s_i and s_j, be assigned to the *same class* in the transient routing T, and

Case (b): The preassignment requires that s_i and s_j be assigned to two *different classes* in T.

After constructing G_T, preassignment is accomplished by modifying the graph G_T as follows:

Case (a): Coalesce the vertices v_{s_i} and v_{s_j} in G_T which correspond to clusters s_i and s_j, respectively.

Case (b): Add D edges[3] between vertices v_{s_i} and v_{s_j}. The edges in G'_T which correspond to these parallel edges will form a path whose length is large enough to preclude selection as a shortest path in G'_T. Therefore, v_{s_i} and v_{s_j} will always be assigned to different classes.

For example, consider the graph G_T in Fig. 33. If the preassignment involving v_{s_i} and v_{s_j} corresponds to Case (a), G_T is modified as shown in Fig. 34; if the preassignment of v_{s_i} and v_{s_j} corresponds to Case (b), G_T is modified as shown in Fig. 35.

There is an important case associated with preassignment which may arise in a given physical routing P, namely the existence of cruxes. When constructing the graph G_T, we regard a crux zone, x_i, as a via candidate zone and construct the graph G_T as described in Section III. Since vias are not allowed in crux zones, the class

[3]It is sufficient to let D equal the total number of edges in G_T.

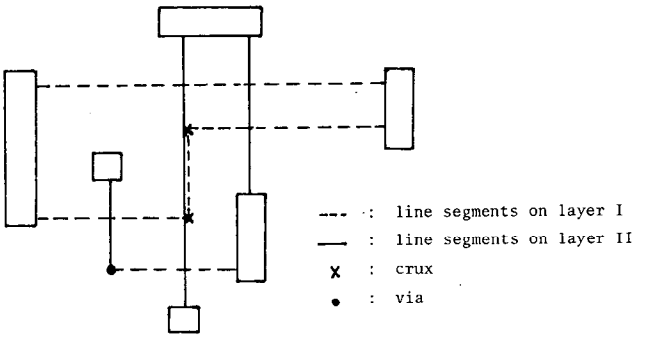


Fig. 36. Physical routing P_5.

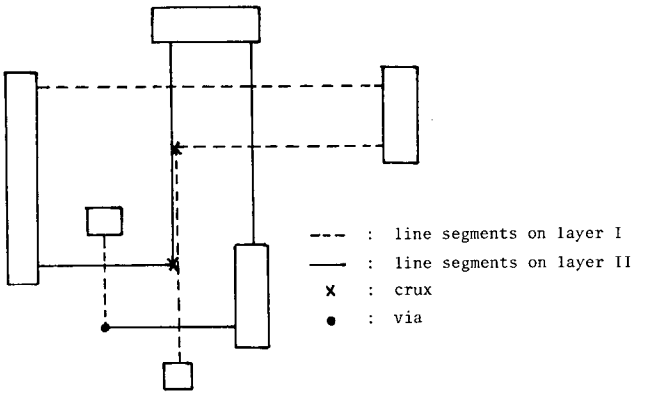


Fig. 37. Physical routing P_6.

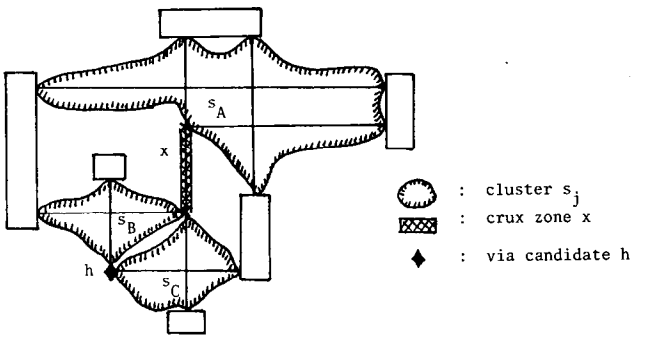


Fig. 38. Transient routing T of $P_5(P_6)$.

assignment of the clusters adjoining a crux zone may be thought of as a form of preassignment.

The specific preassignment will depend on the structure of the physical routing. The clusters which adjoin each crux zone must be observed in order to determine whether the preassignment corresponds to Case (a) or to Case (b). The graph G_T will then be modified accordingly, i.e., coalescing vertices or adding parallel edges. Furthermore, all similar edges in the modified form of G_T, which correspond to the crux zone x_i, will be removed.

Consider two physical routings, P_5 and P_6, shown in Figs. 36 and 37, respectively. The transient routing T and the graph G_T which correspond to P_5 and P_6 have the same structure. They are shown in Figs. 38 and 39, respectively. Examination of Fig. 36 indicates that clusters s_A, s_B, and s_C, of Fig. 38 belong to the same class.

On the other hand, Fig. 37 indicates that clusters s_B and s_C belong to one class while cluster s_A belongs to the other

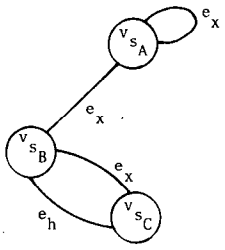


Fig. 39. Graph G_T of T shown in Fig. 38.

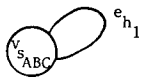


Fig. 40. Modified G_T of P_5 shown in Fig. 36.

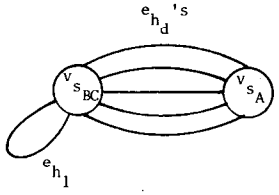


Fig. 41. Modified G_T of P_6 shown in Fig. 37.

class. Hence, the modified G_T's which correspond to P_5 and P_6 are different as illustrated in Figs. 40 and 41, respectively.

In general, coalescing vertices and adding edges in G_T may result in a nonplanar graph. In that case Step 5 of the via minimization algorithm will not be applicable. However, if the preassignment involves adjacent vertices only, the modified G_T will remain planar.

A given physical routing with arbitrary preassignments may not result in a global minimal solution. Without modifying G_T, apply the via minimization algorithm directly through Step 10. Then in the bipartite subgraph which results from Step 10, assign those vertices (each of which corresponds to a cluster containing those preassigned line segments) to a particular class. Next proceed to Step 11. It follows that the number of vias will be increased by the Boolean cardinality of edges which connect vertices of the same class in the graph as described above.

It has been shown that the via minimization problem of the Hashimoto–Stevens type can be translated into the problem of finding a maximum cut in a planar graph, which, in turn, can be solved in polynomial time by using Hadlock's algorithm [9]. Furthermore, any planar graph G has a corresponding physical routing P of the Hashimoto–Stevens type [9]. Assume that a preassignment exists; i.e., some clusters are assigned to the same class. The modified graph G, due to the preassignment, is obtained by coalescing vertices in G. This may result in a nonplanar graph. Consequently, the problem of finding the minimum number of vias in P can be translated into that of finding a maximum cut in G. In general, G may be nonplanar. Since the problem of finding a maximum cut in a general graph is *NP*-complete [14], it follows that a problem involving an arbitrary preassignment cannot be solved in polynomial time even in the case of the simplest type of problems, e.g., problems of the Hashimoto–Stevens type.

VIII. Concluding Remarks

Based on the graph-theoretic model, an algorithm for minimizing the number of vias in two-layer PC boards has been developed. Examples have been presented to illustrate various aspects of the algorithm. Further, it has been shown that the algorithm is of a polynomial order of complexity and always results in the minimum number of vias for routings whose via candidates are of degree 3 or less (under the assumption that no restrictions are imposed on the location of the via candidates): In actual applications, routings containing via candidates of degree 4 or more are rarely encountered. Therefore, the algorithm provides the minimum number of vias for most practical problems.

The algorithm developed in this paper represents a significant improvement over the most recent algorithms which have appeared in the literature [6], [7]. For the algorithm developed by Stevens and vanCleemput [6], the location of the via candidates is restricted to a given set of vias. Their algorithm does not allow for the introduction of additional via candidates which might achieve a net reduction in the number of vias that appear in the final routing. The proposed algorithm in this paper is more general in that the constraints imposed on the positions of the via candidates are removed. The algorithm developed by Ciesielski and Kinnen [7] represents a layer assignment algorithm. Although it is more general in that via candidates of degree 3 or more are considered and may handle preassignment of the line segments, it cannot be executed in polynomial time. The algorithm developed herein has been shown to be executable in polynomial time. Furthermore, it has been demonstrated that the solutions are minimal in the event that all via candidates are of degree 3 or less. Finally, it is hoped that this work will stimulate further research efforts to solve the via minimization problem for multilayer PC boards with algorithms that can be executed in polynomial time.

Acknowledgment

The authors wish to thank Stanley L. Basin, adjunct professor, Department of Applied Mathematics, University of Santa Clara, for his helpful discussion on this subject.

References

[1] R. Chen, Y. Kajitani, and S. Chan, "On the via minimization problem for the two-layer printed circuit board," in *Conf. Rec. 15th Asilomar Conf. on Circuits, Systems, and Computers*, pp. 22–26, 1982.

[2] M. A. Breuer, *Design Automation of Digital System*, vol. 1, Englewood Cliffs, NJ: Prentice Hall, 1972, pp. 321–329.

[3] A. Hashimoto and J. Stevens, "Wire routing by optimizing channel assignment within large apertures," in *Proc. 8th Design Automation Workshop*, pp. 155–169, June 1971.

[4] A. Sakamoto, *et al.*, "OSACA: A system for automated routing on two-layer printed wiring boards," in *USA–Japan Design Automation Symp.*, pp. 100–107, July 1975.

[5] M. Servit, "Minimizing the number of feedthroughs in two-layer printed boards," *Digital Processes*, vol. 3, pp. 177–183, Mar. 1977.

[6] K. R. Stevens and W. M. van Cleemput, "Global Via Elimination in Generalized Routing Environment," in *Proc. 1979 ISCAS*, pp. 689–692, July 1979.
[7] M. J. Ciesielski and E. Kinnen, "An optimum layer assignment for routing in ICs and PCBs," in *Proc. 18th Design Automation Conf.*, pp. 733–737, June 1981.
[8] Y. Kajitani, "On via hole minimization of routing on a 2-layer board," in *Proc. 1980 ICCC*, pp. 295–298, Oct. 1980.
[9] F. Hadlock, "Finding a maximum cut of a planar graph in polynomial time," *SIAM J. Comp.*, vol. 4.3, pp. 221–225, Sept. 1975.
[10] K. Aoshima and M. Iri, "Comment on F. Hadlock's paper: 'Finding a maximum cut of a planar graph in polynomial time,'" *SIAM J. Comp.*, vol. 6, no. 1, pp. 86–87, Mar. 1977.
[11] R. Chen, "Topological considerations of the via minimization problem for two-layer printed circuit boards," Ph.D. dissertation, Univ. of Santa Clara, CA 1982.
[12] N. Deo, *Graph Theory with Application to Engineering and Computer Science*, Englewood Cliffs, NJ: Prentice-Hall, 1974, pp. 292–301.
[13] J. Edmonds, "Maximum matching and a polyhedron with 0,1-vertices," *J. Res. Nat. Bur. Stand. B 69*, pp. 125–130, June 1965.
[14] R. Karp, "Reducibility among combinatorial problems," in *Complexity of Computations*, (R. E. Miller and J. W. Thatcher, eds.), New York: Plenum, pp. 85–103, 1972.

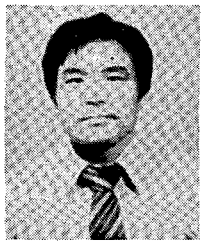
✤

Ruen-Wu Chen was born in Sichuan, China, on May 9, 1949. He received the B.S. degree in electrical engineering and computer science, and the M.S. and Ph.D. degrees, both in electrical engineering, from the University of Santa Clara, in 1974, 1976, and 1982, respectively. He was awarded a Teaching Fellowship by the University of Santa Clara from 1978 to 1980 while studying toward the Ph.D. degree in electrical engineering at the University.

In September 1980, he joined the faculty in the Electrical Engineering and Computer Science Department at the University of Santa Clara, where he is presently Visiting Assistant Professor. His major interests are in network theory, topological analysis and synthesis, integrated circuit design, and automated design.

Dr. Chen is a member of Tau Beta Pi and Eta Kappa Nu.

✤



Yoji Kajitani (SM'80) was born in 1941. He received his B.E., M.E., and D.E. degrees in electrical engineering from Tokyo Institute of Technology, Japan, in 1964, 1966, and 1966, respectively.

Since 1973, Dr. Kajitani has been an Associate Professor in the Department of Electrical and Electronic Engineering at Tokyo Institute of Technology. His research is in the area of graph theory and its applications.

Shu-Park Chan (S'62–M'63–SM'69–F'00) was born in Canton, China, on October 10, 1929. He received the B.Sc. degree in electrical engineering from the Virginia Military Institute in 1955, the M.S. and Ph.D. degrees, both in electrical engineering from the University of Illinois, Urbana-Champaign, in 1957 and 1963, respectively.

From 1957 to 1963, he taught at V. M. I. first as Instructor in Electrical Engineering and Mathematics and then as Assistant Professor of Mathematics. On leave of absence from V. M. I., he was awarded a University Fellowship by U. of I. from 1959 to 1960 while studying toward the Ph.D. degree in electrical engineering at the University. He taught as Instructor in Electrical Engineering in that department from 1960 and worked in the Coordinated Science Laboratories as Research Associate from 1961 to 1963 at the University. Since February 1963, he joined the faculty in the Electrical Engineering and Computer Science Department at the University of Santa Clara, where he is presently Professor and Chairman. His major interests and publications have been in network and system theory, topological analysis and synthesis, computer-aided circuit design, and applied mathematics. He is author of *Introductory Topological Analysis of Electrical Networks* (Holt, Rinehart and Winston, 1969), co-author of *Analysis of Linear Networks and Systems — A Matrix-Oriented Approach with Computer Applications* (Addison-Wesley, 1972) and *Introduction to the Applications of the Operational Amplifier* (Academic Cultural Co., 1974) and editor and major contributor of *Network Topology and its Engineering Applications* (National Taiwan University Press, 1975). In September 1973, he accepted the Special Chair Professorship at the Electrical Engineering Department of the National Taiwan University for a full year while taking a sabbatical leave from the University of Santa Clara. On July 1, 1980, at the invitation from the Academia Sinica, Peking, he spent three months in China, touring eleven cities and gave a series of special lectures on applied graph theory and topological applications to eighty some college teachers, scientists, and engineers. From October 1, 1980, he was with the University of Hong Kong as an Honorary Professor in the Electrical Engineering Department for six months. In 1982, he organized and chaired the Summer Institute on Graph Theory and Applications at the Graduate School of Academia Sinica in Beijing, China, based on the book he edited, *Graph Theory and Applications* (Science Press, Beijing, 1982). He then received the Honorary Professorship bestowed by Anhuei University, Hefei, China.

Dr. Chan is a member of Tau Beta Pi, Eta Kappa Nu, Pi Mu Epsilon, Phi Kappa Phi, and Sigma Xi, and a member of American Association of the Advancement of Science, and of the American Society of Engineering Education. He is a past chairman of the Circuit Theory Group of the San Francisco Section IEEE. He is listed in 'American Men and Women of Science,' 'Who's Who in America,' 'International Scholars Directory,' 'Dictionary of International Biography and Who's Who in the World.' Professor Chan is also president of the Chinese Arts and Culture Institute, and has been actively involved in promoting and organizing activities and projects relative to Chinese arts and culture in the United States and the Far East.