TR--89--002

THE VIA MINIMIZATION PROBLEM

THE VIA MINIMIZATION PROBLEM

TR-89-002

Y. S. Kuo

Institute of Information Science

Academia Sinica

1. Introduction

The automatic layout of integrated circuits or printed circuit boards is traditionally divided into two processes: placement and routing [25]. Via minimization is a postprocess after routing for eliminating unnecessary feedthrough vias. Many different models and algorithms have been proposed for the via minimization of two-layer and three-layer routing. This article reviews significant results among these works. The emphasis is placed on optimal algorithms that have sound theorectical basis and heuristic algorithms that seem to be effective in practice.

1.1. Motivation

Most existing routing algorithms route all horizontal wires on one of the two conducting layers and route all vertical wires on the other. Therefore, a large number of vias are introduced to interconnect the wire segments on different layers. Vias not only reduce the reliability and performance of the circuit, but also increase the manufacturing cost. Thus, it is desirable to have a postprocess after routing to eliminate unnecessary vias. Moreover, since vias are usually wider than wires, via minimization can be used to make room for uncompleted connections.

1.2. Layer Assignment

Given a layout where the positions of the modules and the wires are fixed, the via minimization problem is to assign the conducting layers to the wire segments so that the total number of vias required is minimized. Thus, the via minimization problem is also known as an optimal layer assignment problem. In this article, we shall use both terms interchangably.

Consider a layout such as in Fig. 1 where two conducting layers are used for routing. A net is a collection of wire segments that electrically connect a set of terminals. A layer assignment is an assignment of the two layers to all the wire segments such that no two wire segments in different nets that cross or overlap each other are assigned to the same layer. Fig. 1 shows a possible layer assignment. When two connected wire segments are assigned to different layers, a via must be introduced. Thus minimizing the number of vias is equivalent to minimizing the total layer changes in the layout. A new layer assignment for the nets in Fig. 1 is shown in Fig. 2 where the number of vias required has been reduced from 3 to 1.

Layer assignment has its major objective of minimizing the total number of vias. Frequently, it is also subject to some extra constraints due to practical considerations. For example, wire segments connecting to terminals may have to be assigned to a particular layer.

1.3. Two-Layer Via Minimization

The via minimization problem for two-layer routing has been studied extensively. Since the pioneer work of Hashimoto and Stevens in 1971 [9], many researchers have proposed different models and algorithms which can be classified into two categories: the optimal approach and the heuristic approach. The optimal approach aims to find the global minimum of the problem. Hashimoto and Stevens formulated the via minimization problem as a graph-theoretic maximum cut problem [9], though the latter is NP-hard for general graphs [7]. Kajitani identified the wire-segment clusters in a layout, and showed that the graph in Hashimoto's model is planar [12]. Thus optimal polynomial-time algorithms for via minimization were proposed based upon Hadlock's maximum cut algorithm for planar graphs [3][8][21][23]. The time complexity of these algorithms is $O(n^3)$ where n is the number of wire-segment clusters in the given layout. Recently, Kuo, et al. developed an $O(n^{1.5} log\ n)$ optimal algorithm by transforming the via minimization problem into finding a maximum weight matching of a planar graph [15]. Other than the graph-theorectic formulation, Ciesielski and Kinnen proposed an integer programming formulation of the via minimization problem. However, the time complexity of their algorithm is exponential [4].

In the heuristic approach, the optimality is compromized, but the flexibility and the speed are gained. An early heuristic algorithm proposed by Stevens and VanCleemput uses a force-directed constructive method

[26]. Chang and Du developed a layer assignment algorithm based upon properties of the bipartite graph [1]. Recently, Xiong and Kuh proposed an iterative improvement algorithm by swapping vertices of a graph [27]. The main advantage of these heuristic algorithms is that they can be easily modified to take extra constraints into consideration.

1.4. Three-Layer Via Minimization

The works that have been mentioned so far are about two-layer via minimization. The rapid advance in VLSI technology has made it possible to use three conducting layers for routing. Three-layer via minimization is still a rather new problem. It has been shown to be NP-hard and a heuristic algorithm was proposed by Chang and Du [2].

The via minimization problem considered in this article is sometimes referred to as a constrained via minimization problem since the geometry of a layout is given and fixed. The problem in which both the topology of the layout and the layer assignment are to be decided is referred to as an unconstrained via minimization problem or a topological via minimization problem [1][11][19][24]. The latter problem is beyond the scope of this article.

2. Problem Definition

2.1. Terminology

To precisely formulate the via minimization problem, let us introduce some terminologies. When defining these terminologies, the given layout is considered not to have any layer assignment yet. For the layout in Fig. 1, such a "layerless" layout is shown in Fig. 3.

Two pieces of wires in different nets are said to conflict with each other if they cross or overlap each other, or if they are too close to be in the same layer due to the design rules.

A wire segment is a maximal piece of wire that is in conflict with some other wires and that cannot accommodate any via. Since no via can be placed in a wire segment, a wire segment is a unit to be assigned a layer; it is meaningless to further divide a wire segment into smaller pieces.

A via candidate is a piece of wire that connects two or more wire segments (in the same net) and that can accommodate at least one via. A via candidate is a location where a via might be introduced. Since our goal is to minimize via usage, at most one via would be placed at a via candidate; there is no sense in switching layers twice or more along a via candidate.

The degree of a via candidate is the number of wire segments connecting to the via candidate.

For the layout in Fig. 3, wire segments are labeled by numbers 1 through 18; via candicates are labeled by lower-case letters a through k; and nets are labeled by upper-case letters A through G. As an illustration, net A contains wire segments 1, 4 and 7; wire segments 1 and 4 are connected by via candidate i; and wire segments 4 and 7 are connected by via candidate a. All via candidates in Fig. 3 are of degree 2. A typical via candidate of degree 3 is a T-shape connection which often appears in a multi-terminal net.

An essential via is a via placed at a particular via candidate which is present in any layer assignment. In Fig. 3, if no via could be placed at locations d and f due to the design rules, then wire segments 12 and 13 would be collapsed to a single wire segment, and so would be wire segments 17 and 18. Since wire segments 13(12) and 17(18) should be assigned to different layers, so should be wire segments 6 and 9. Therefore, a via must be introduced at location c to connect wire segments 6 and 9. Such a via would be an essential via.

The via minimization problem considered in this article is very general. Any location in a wire that can accommmodate a via must be part of a via candidate. In other words, vias can be placed at any appropriate locations so long as the design rules are not violated. On the other hand, the wires in a given layout do not have to lie on a rectangular grid. Such a layout is sometimes referred to as a gridless layout [21][23]. To identify wire segments and via candidates in a layout, what is required is a design rule checker which can report all pairs of wire segments which are too close to each other [28].

2.2. Extra Constraints

In addition to the constraint that conflicting wire segments are assigned to different layers, a layer assignment should also satisfy some extra constraints due to the design methodology and performance considerations. For instance, wire segments connecting to terminals are usually required to be on a particular layer. Power lines should always be assigned to the metal layer. The number of vias introduced in a net may have to be limited in order to avoid burdening a performance critical path. In summary, the following extra constraints should be associated with the via minimization problem [1].

- Some wire segments can be preassigned to certain layers.
- The numbers of vias introduced in certain nets can be limited under certain threshold values.

3. Graph Models

Theorectical models such as graphs and integer programming have been proposed for representing the via minimization problem. However, since integer programming is a NP-hard problem in general [7], no efficient via minimization algorithms were ever designed based upon the integer programming model. We only consider the various graph models. The graph is especially suitable for modeling the geometric relationship among the wire segments in a layout.

3.1. The Wire-Segment Graph

A given layout can be naturally represented as a graph called the wire-segment graph. For the moment, let us assume that all via candidates in the layout are of degree 2. Associated with each wire segment, there is a vertex in the wire-segment graph. There are two kinds of edges. A conflict edge connects two vertices whose corresponding wire segments are in conflict with each other. A continuation edge connects two vertices whose corresponding wire segments are incident to a common via candidate. The wire-segment graph associated with the layout in Fig. 3 is shown in Fig. 4.

The two-layer via minimization problem can be formulated as a graph coloring problem. A 2-coloring of the wire-segment graph is an assignment of two colors to its vertices so that no two vertices connected by conflict edges have the same color. It can be easily shown that, given a layerless layout, a layer assignment exists iff the associated wire-segment graph has a 2-coloring, i.e. is 2-colorable. In fact, a layer assignment uniquely determines a 2-coloring, and the converse is also true. Then, the two-layer via minimization problem is to find a 2-coloring of the wire-segment graph so that the number of continuation edges whose two end vertices have different colors is minimum. The layer assignment in Fig. 1 natually induces a 2-coloring for the graph in Fig. 4; the color assigned to a vertex is shown by the vertex.

By associating positive weights to continuation edges, the wire-segment graph can be generalized to model a layout containing multi-terminal nets. The vertices of the graph still correspond to the wire segments. For a via candicate connecting k wire segments, the k corresponding vertices in the wire-segment graph form a complete subgraph (k-clique); each edge of the k-clique is a continuation edge with weight $1/(\lfloor k/2 \rfloor \lceil k/2 \rceil)$ where $\lfloor x \rfloor$ (resp. $\lceil x \rceil$) is the greatest (resp. smallest) integer smaller (resp. greater) than or equal to x. Fig. 5 illustrates such k-cliques for k = 2,3,4. For the wire-segment graph in Fig. 4, since k=2, all continuation edges have weight 1. Now the graph coloring problem can be restated as follows:

Minimum Weight Coloring (MWC):
Find a 2-coloring of the weighted wire-segment graph so that the total

weight of the continuation edges whose two end vertices have different colors is minimum.

Lemma 1. Given a layout where all via candidates are of degree no more than 3, an optimal layer assignment can be found by finding a minimum weight coloring of the weighted wire-segment graph.
Proof: This can be easily verified from Fig. 5(a)(b). Q.E.D.

In general, the via minimization problem is not equivalent to the MWC problem if the degrees of some via candidates are greater than 3. However, a minimum weight coloring is usually a good approximation to an optimal layer assignment.

In a given layout, a wire-segment cluster, or simply cluster, is a maximal set of mutually conflicting wire segments. For example, in Fig. 3, wire segments 4, 5, 6, 12 and 14 form one cluster; and wire segments 7, 8, 9 and 18 form another. In fact, a cluster in a layout corresponds to a connected component of the wire-segment graph due to the conflict edges, and vice versa. This can be seen in Fig. 4. From our earlier formulation, the layer assignment problem has a feasible solution iff the wire-segment graph is 2-colorable, or equivalently, each connected component of the graph is a bipartite graph. A graph is bipartite if its vertices can be partitioned into two sets so that no edge connects two vertices in the same set. Since we assume that a layer assignment is known, it can be claimed that all connected components of the wire-segment graph are bipartite graphs.

One can locate the essential vias in a layout by looking at the connected components of the wire-segment graph. Note that vertices of a connected component are partitioned into two sets, and vertices (i.e. wire segments) in different sets must be assigned different colors (i.e. layers). Therefore, a continuation edge connecting two vertices in the opposite sets of a connected component determines a via candidate where an essential via must be introduced [9]. Fig. 6 illustrates such a continuation edge.

3.2. The Cluster Graph

Given a layerless layout, once a wire segment in a cluster is assigned to a certain layer, layer assignment of the rest of the cluster is forced. In other words, there are only two possible ways to assign the wire segments in a cluster to two layers. On the other hand, wire segments in different clusters can be assigned to layers independently to each other. Thus, it is appropriate to treat all wire segments in a cluster as a whole. With a prescribed layer assignment, a cluster is said to be flipped over if all the wire segments in the cluster are reassigned to the opposite layers.

The clusters in a layout can form a graph called the cluster graph. For the moment, let us assume that all via candidates in the layout are of degree 2. Each vertex of the graph corresponds to a cluster, and two vertices are connected by an edge if their corresponding clusters are connected to at least a common via candidate. The cluster graph for the layout in Fig. 3 is shown in Fig. 7 where each vertex is labeled by using a representative in its corresponding cluster.

Assume that a layer assignment such as in Fig. 1 is known. Then associated with each edge e of the cluster graph is a weight w(e) defined as follows: Let v be the number of via candidates connecting the two clusters incident to e, and let δ be the number of vias introduced by the known layer assignment connecting the two clusters. Then

$$w(e) = \delta - (v - \delta).$$

Note that, if either one of the two clusters is flipped over, then the current δ vias in the known layer assignment can be eliminated, but another $v - \delta$ vias must be introduced. Thus, the weight w(e) indicates the via reduction that can be achieved due to flipping over either one of the two clusters. As an example, in Fig. 7, the weight for the edge connecting clusters 5 and 15 is 2 since for this edge $v = \delta = 2$ (with reference to Fig. 1). If one of the clusters 5 and 15 is flipped over, then the two vias connecting clusters 5 and 15 can be eliminated.

An arbitrary layer assignment L can be obtained from a known layer assignment L_0 by flipping over a set of clusters. Let $\delta(L)$ and $\delta(L_0)$ be the numbers of vias introduced by L and L_0 respectively, and let X be the set of clusters that are flipped over. Then

$$\delta(L) = \delta(L_0) - \sum_{e \in E(X,\bar{X})} w(e) \qquad (1)$$

where $E(X,\bar{X})$ is a cut separating X and $\bar{X}$, i.e. the set of edges connecting vertices in X and vertices not in X. (1) is due to the fact that for any two clusters both in X or both in $\bar{X}$, the via count between the two clusters remains unchanged, but for two clusters one in X and one in $\bar{X}$, the via count is reduced by w(e). In order to minimize the via count $\delta(L)$, we want to find a cut $E(X,\bar{X})$ which maximizes its cut capacity $\sum_{e \in E(X,\bar{X})} w(e)$, i.e. to find a maximum cut. Note that the edge weights w(e) can be positive or negative, but a maximum cut is always nonnegative since X can be $\emptyset$ and $\sum_{e \in E(X,\bar{X})} w(e) = 0$ for X $= \emptyset$. In case that a maximum cut has capacity 0, L_0 is an optimal layer assignment. Let us summarize what we have just described as a lemma.

Lemma 2. For a layout where all via candidates are of degree 2, an optimal layer assignment can be found by finding a maximum cut of the cluster graph.

For the cluster graph in Fig. 7, vertex sets $\{2,5,8\}$ and $\{11,15\}$ determine a maximum cut of capacity 2. Thus an optimal layer assignment can be obtained from the layer assignment shown in Fig. 1 by flipping over clusters 11 and 15. The resulting layer assignment is that in Fig. 2.

The cluster graph can also be constructed from the weighted wire-segment graph. First, the wire-segment graph is reassigned edge weights. For a continuation edge originally with weight w, the new weight is w or -w depending on if the two end vertices of the edge have different colors or have the same color. For the wire-segment graph in Fig. 4, the newly assigned edge weights are shown in Fig. 8. Then the cluster graph is constructed by contracting each connected component of the wire-segment graph to a single vertex, and collapsing all continuation edges incident to the same pair of connected components into a single edge whose weight is the sum of the weights of the collapsed edges. The cluster graph constructed from the graph in Fig. 8 is exactly that in Fig. 7. Since the weighted wire-segment graph is general enough to represent a layout containing multi-terminal nets, the cluster graph constructed this way is as general. The following theorem is a generalization of Lemma 2. It can be derived in the same way as Lemma 2. Thus the proof is omitted.

Theorem 1. For a given layout, a minimum weight coloring of the wire-segment graph can be found by finding a maximum cut of the cluster graph.

Corollary 1. Given a layout where all via candidates are of degree no more than 3, an optimal layer assignment can be found by finding a maximum cut of the cluster graph.

Proof: Combining Lemma 1 and Theorem 1. Q.E.D.

The via minimization problem has been reduced to the maximum cut problem, yet the latter is NP-hard for general graphs. Fortunately, Kajitani made an important observation that the cluster graph is usually planar.

Lemma 3. For a layout where all via candidates are of degree no more than 3, the associated cluster graph is planar.

Proof: Consider a layout such as in Fig. 3. By collapsing clusters into single vertices, and stretching connections between clusters due to via candidates (without altering the topology), the cluster graph can be drawn in the plane such that no two edges cross each other. Thus the cluster graph is a planar graph. Q.E.D.

The wire-segment graph and the cluster graph were initially defined in [9], and later refined in [3][23][26]. The exposition of these graph models in this section has basically followed [26], but is more general and complete.

3.3. The Via-Net Graph

In a layerless layout, a net is formed by wire segments interconnected by via candidates. It is natural to represent a net by a tree called the interconnection tree. There are two kinds of vertices in the tree corresponding to the wire segments and the via candidates respectively. For each incident pair of wire segment and via candidate in the layout, there is an edge in the tree connecting the two corresponding vertices. Since each net has an interconnection tree, there are as many interconnection trees as the nets. These interconnection trees can be connected by conflict edges. For each pair of conflicting wire segments, there is a conflict edge connecting their corresponding vertices in two separate interconnection trees. Thus the interconnection trees form a graph called the via-net graph. To distinguish the two kinds of edges, edges of the interconnection trees will be called interconnection edges. As an example, the via-net graph for the layout in Fig. 3 is shown in Fig. 9.

The concept of conflicting wire segments can be generalized to conflicting nets. Two nets are in conflict with each other if they contain a pair of conflicting wire segments. Conflicting nets can be represented by a graph called the net crossing graph. Each vertex of the graph corresponds to a net. Two vertices are connected by an edge if their corresponding nets are in conflict with each other. The net crossing graph for the layout in Fig. 3 is illustrated in Fig. 10. Note that the net crossing graph can be constructed from the via-net graph by contracting each interconnection tree to a single vertex and collapsing all conflict edges connecting the same pair of interconnection trees into a single edge. For example, vertices 1, i, 4, a and 7 in Fig. 9 are contracted to a single vertex (i.e. net) A in Fig. 10, and vertices 12, f and 13 are contracted to vertex D, etc.

The via-net graph and the net crossing graph can represent not only a layerless layout, but also a layout containing some vias. Consider a layout such as in Fig. 2. The via or vias introduced in a net divides the net into two or more subnets. Each subnet can be treated as a net. Then the via-net graph and the net crossing graph can be defined similarly as before. For the layout in Fig. 2, net F is split into two subnets consisting of wire segment 18 and wire segments 16 and 17 respectively. The associated via-net graph is shown in Fig. 11. The net crossing graph is shown in Fig. 12 where vertices 18 and 16 represent the two subnets.

The via-net graph and the net crossing graph can be updated dynamically as vias are introduced in a layout. When a via is introduced in a net at a via candidate, the vertex corresponding to the via candidate and its incident interconnection edges are removed from the via-net graph. In other words, the interconnection tree corresponding to the net is broken to subtrees corresponding to the subnets. On the other hand, the vertex in the net crossing graph corresponding to the split net is replaced by two or more vertices, each corresponding to a newly generated subtree or subnet. The edges connecting to each new vertex are determined by traversing its corresponding subtree in the via-net graph. For example, the via-net graph in Fig. 11 is constructed from that in Fig. 9 by removing vertex d and its incident edges. The net crossing graph in Fig. 12 is constructed from that in Fig. 10; vertex F is replaced by two vertices 18 and 16.

When solving the via minimization problem, the best one could hope is that the net crossing graph is 2-colorable. In this case, each net can be assigned to a single layer such that conflicting nets are assigned to different layers. Thus a layer assignment to all the wire segments can be obtained without introducing any via. Otherwise, some vias must be introduced. The vias split the nets into subnets. In a layer assignment such as Fig. 2, conflicting subnets must be assigned to different layers. Thus its associated net crossing graph (eg. Fig. 12) is 2-colorable. Conversely, if the nets are split into subnets so that the resulting net crossing graph is 2-colorable, then a layer assignment can also be obtained. Therefore, the via minimization problem can be formulated as follows: Introduce the minimum number of vias in a layout in order to split the nets into subnets so that the final net crossing graph is 2-colorable (i.e. bipartite) [1].

4. Optimal Approach

Many optimal algorithms for two-layer via minimization have been proposed. In this section, we only describe the one recently developed by Kuo, et al. [15]. This algorithm is based upon the cluster graph model in Sec. 3.2, and is at present the most efficient optimal algorithm from the theorectical point of view.

4.1. Problem Transformations

Let G = (V,E) be the cluster graph for a layout which contains no via candidate of degree greater than 3. Then G is planar and each edge of E has an associated real-valued weight. By introducing a series of transformations, it can be shown that a maximum cut of G can be found by finding a minimum complete matching in a planar graph G' constructed from G.

Without loss of generality, we assume that G is connected. (Otherwise a maximum cut can be found by finding maximum cuts in individual connected components.) We first triangulate G by adding some new edges. A triangulation G_t = (V,E_t) of G is a connected planar graph embedded in the plane satisfying

(i) $E \subseteq E_t$,

(ii) Each vertex of G_t has degree at least 2,

(iii) Each face of G_t is enclosed by a simple cycle of three edges, and

(iv) Any two faces of G_t share at most one edge.

We assign zero weight to each new edge in E_t - E. As an example, a triangulation of the planar graph in Fig. 7 is shown in Fig. 13.

Lemma 4. A maximum cut of G = (V,E) corresponds to a maximum cut of G_t = (V,E_t), and vice versa.

Proof: Obvious. Q.E.D.

Consider a geometric dual $G_d = (V_d, E_d)$ of $G_t = (V, E_t)$ [5][18]. G_d can be constructed from G_t as follows: Consider an embedding of G_t in the plane. Associated with each face of G_t, there is a vertex in G_d. For each edge shared by two faces of G_t, there is an edge in G_d connecting the two corresponding vertices. A geometric dual of the planar graph shown in Fig. 13 is illustrated in Fig. 14. We assign to each edge of E_d the same weight as its corresponding edge of E_t. In general, a geometric dual of a planar graph is a multigraph. However, due to the construction of G_t, G_d contains no self-loops and parallel edges, and G_d is regular.

Lemma 5. $G_d = (V_d, E_d)$ is a cubic planar graph. (A graph is cubic if each vertex of the graph is of degree 3.)

Proof: G_d contains no self-loops and parallel edges since G_t satisfies (ii) and (iv). Thus G_d is a graph. Each vertex of V_d is of degree 3 since G_t satisfies (iii). The planarity of G_d is due to the fact that G_d is a geometric dual of G_t. Q.E.D.

In $G_d = (V_d, E_d)$, an edge set $D \subseteq E_d$ is said to be even-degree if each vertex of V_d is incident to an even number of edges in D. The weight of an even-degree edge set D is the total weight of the edges in D. Since $G_d = (V_d, E_d)$ is a geometric dual of $G_t = (V, E_t)$, there is a one-to-one correspondence between edges of E_t and edges of E_d. This induces a natural correspondence between the cuts of G_t and the even-degree edge sets of G_d.

Lemma 6. A cut of $G_t = (V, E_t)$ corresponds to an even-degree edge set of $G_d = (V_d, E_d)$, and vice versa.

Lemma 6 is a well known result in the planar graph theory. Please refer to [5][18] for its proof. Here we simply illustrate this lemma by an example. For the graph in Fig. 13, consider the vertex set $X = \{8, 11\}$. X determines a cut $E(X, \bar{X})$ where

$$E(X, \bar{X}) = \{(8,2), (8,5), (8,15), (11,2), (11,5), (11,15)\}.$$

In the dual graph (Fig. 14), the edge set that corresponds to $E(X, \bar{X})$ is

$$\{(E,D), (D,A), (A,E), (F,C), (C,B), (B,F)\}\ .$$

Apparently, this edge set is even-degree, and consists of two simple cycles.

From Lemma 6, finding a maximum cut of G_t is equivalent to finding a maximum (weight) even-degree edge set of G_d. The following lemma characterizes a maximum even-degree edge set of G_d.

Lemma 7. Let D be a maximum even-degree edge set of $G_d = (V_d, E_d)$.

Then D is either empty or a union of vertex-disjoint nonnegative cycles.

Proof: Assume $D \neq \emptyset$. Since G_d is a cubic graph, each vertex of G_d is adjacent to 0 or 2 edges in D. Thus D is a union of vertex-disjoint cycles in G_d. The claim then follows from the fact that D is maximum. Q.E.D.

To find a maximum even-degree edge set of $G_d = (V_d, E_d)$, we construct a graph G' = (V',E') from G_d. Each vertex v of G_d is replaced by a 3-clique in G' and each edge e of G_d has a surrogate in G' as depicted in Fig. 15. For the cubic planar graph in Fig. 14, the constructed graph is illustrated in Fig. 16. Define the edge weights of G' as follows: the surrogate of each edge $e \in E_d$ has the same weight as e; and all new edges in 3-cliques have zero weights. Note that the constructed G' is still planar for its topology remains the same as that of G_d.

A matching M of graph G' = (V',E') is a set of edges no two of which have a common vertex. If $|M| = |E'|/2$, then M is called a complete matching. A maximum weight matching (resp. minimum complete matching) is a matching (resp. complete matching) of G' whose total weight is maximum (resp. minimum).

Lemma 8. Let $M \subseteq E'$ be a minimum complete matching of $G' = (V', E')$. Then $E_d - M$ is a maximum even-degree edge set of $G_d = (V_d, E_d)$.

Proof: Let $M \subseteq E'$ be any complete matching of G'. Consider any vertex v of G_d and its associated 3-clique in G' as in Fig. 15. Then M either contains all three edges e_1, e_2 and e_3 in G', or contains exactly one edge among the three. In the former case, v has degree 0 in the subgraph of G_d induced by $E_d - M$; and in the latter case, v has degree 2 in the subgraph. Thus $E_d - M$ is an even-degree edge set of G_d. Conversely, let D be any even-degree edge set of G_d. As shown in Lemma 7, D is either empty or a union of vertex-disjoint cycles. From the construction of G', one can observe that there exists a complete matching M of G' such that $D = E_d - M$. (Such a complete matching is illustrated in Fig. 16 with respect to an even-degree edge set shown in Fig. 14.) Clearly the weight of $E_d - M$ is maximum iff the weight of M is minimum. Q.E.D.

4.2. The Optimal Algorithm

Let us summarize the algorithm for optimal layer assignment and analyze its time complexity.

Input: The cluster graph $G_c = (V_c, E_c)$ for a given layout.

Output: A maximum cut of G_c.

Algorithm MaxCut

1. Decompose G_c into connected components. For each connected Component = (V,E), do steps 2 to 5.

2. Construct a triangulation $G_t = (V, E_t)$ of G = (V,E) by adding new edges to G.

3. Construct a geometric dual $G_d = (V_d, E_d)$ of $G_t = (V, E_t)$.

4. Construct the planar graph G' = (V',E') from G_d. Each vertex of G_d is replaced by a 3-clique in G'.

5. Find a minimum complete matching M of G'. M determines a maximum even-degree edge set $E_d - M$ of G_d which corresponds to a maximum cut of G_t and thus a maximum cut of G = (V,E).

6. Combining the maximum cuts for individual connected components, we have a maximum cut $G_c = (V_c, E_c)$.

Let n_c and n be the numbers of vertices of G_c and G respectively. Since G_c is planar, due to Euler's formula [5], G_c has $O(n_c)$ edges. Thus Step 1 can be computed in $O(n_c)$ time by using a simple depth-first search. Applying Euler's formula repeatedly, we can see that each of the graphs G, G_t and G_d has O(n) vertices and O(n) edges. G' also has O(n) vertices and O(n) edges for each 3-clique in G' contains 3 vertices and 3 edges. Each of Steps 2 and 3 takes O(n) time since they can be carried out by embedding G and G_t in the plane and identifying their faces [10]. That Step 4 takes O(n) time and Step 6 takes $O(n_c)$ time is obvious. We have argued that all steps execpt Step 5 takes linear time in the worst case. Below we shall sketch an $O(n^{1.5} log\ n)$ algorithm for Step 5.

A minimum complete matching of G' = (V',E') can be found by finding a maximum weight matching of the same graph except that the weight w(e) of each edge $e \in E'$ must be replaced by a new weight $W - w(e)$ where W is a large constant. (The negation of w(e) converts a minimization problem to a maximization problem. The added large constant W forces the obtained matching to be complete.) Lipton and Tarjan have designed an $O(n^{1.5} log\ n)$ algorithm for finding a maximum weight matching of a planar graph by applying the planar separator theorem [16][17]. Thus we have the following lemma.

Lemma 9. Finding a minimum complete matching of a planar graph G' can be done in $O(n^{1.5} log\ n)$ time.

Theorem 2. Given a planar graph $G_c = (V_c, E_c)$, Algorithm MaxCut can find a maximum cut of G_c in $O(n_c^{1.5} log\ n_c)$ time. In other words, for a layout where all via candidates are of degree no more than 3, an optimal layer assignment can be found in $O(n_c^{1.5} log\ n_c)$ time where n_c is the number of clusters in the layout.

5. Heuristic Approach

In this section, two fast heuristic algorithms for two-layer via minimization are reported. Even though no theory can guarantee their optimality in any way, experiments have demonstrated that these algorithms are very effective in generating near-optimal solutions. Moreover, these algorithms have the advantage that they can be easily modified to handle extra constraints.

5.1. The Net Split Algorithm

Chang and Du developed a layer assignment algorithm by using the net split formulation described in Sec.3.3 [1]. In this formulation, vias are introduced in a layout to split nets into subnets so that the final net crossing graph is bipartite. A bipartite graph has the following well known property [5].

Lemma 10. A graph is bipartite iff it contains no odd cycle, i.e. cycle of odd length.

Due to Lemma 10, vias should be introduced to eliminate odd cycles in the net crossing graph. The following example illustrates how the net crossing graph is modified when a via is introduced. Consider the layout in Fig. 3 and its associated net crossing graph in Fig. 10. Since there are many odd cycles in Fig. 10 passing through vertex F, a via should be introduced in net F, say at location d (Fig. 3), to break these odd cycles. Then net F is split into two subnets as in Fig. 2, and the resulting net crossing graph is that shown in Fig. 12. Note that vertex F in Fig. 10 has been split into two vertices in Fig. 12. This breaks all the odd cycles.

An intuitive method of breaking all the odd cycles in the net crossing graph is by generating all of them. Since a graph can contain an exponential number of odd cycles, this method is not practical. Thus a heuristic approach is taken that only generates all the 3-cycles, i.e. cycles of length three. Vias are first introduced to break all the 3-cycles in the net crossing graph. If there are still odd cycles left, these remaining odd cycles are then broken by using fundamental sets of cycles.

Consider a connected graph G = (V,E). A spanning tree of G is a tree that is a subgraph of G and contains every vertex of G. Let (V,T) be a spanning tree of G. Then any edge not in T will create exactly one cycle when added to T. Such a cycle is a member of the fundamental set of cycles of G with respect to T. Without causing any confusion, we will simply call cycles in the fundamental set of cycles with respect to a spanning tree "fundamental cycles". It is well known that the fundamental cycles can generate all the cycles of a graph [5][18].

Lemma 11. Any cycle of a graph can be expressed as the ring sum of a set of fundamental cycles. (The ring sum of two edge sets consists of the edges in either one edge set but not in both.)

Corollary 2. A graph is bipartite iff each fundamental cycle is of even length.

Proof: The "only if" part is obvious. Since the ring sum of two or more even edge sets (i.e. edge set containing an even number of edges) is an even edge set, the "if" part follows directly from Lemma 11. Q.E.D.

Now we can outline the net split algorithm as follows:

Algorithm NetSplit

1. Construct the initial via-net graph and the net crossing graph.
2. Find all 3-cycles in the net crossing graph. Introduce a set of vias to split the nets such that all 3-cycles are broken. Modify the via-net graph and the net crossing graph to reflect the split of nets.
3. Find a fundamental set of cycles for the net crossing graph [5][22]. Introduce a via in the net which is involved in the largest number of fundamental cycles of odd length. Modify the via-net graph and the net crossing graph accordingly. This process is repeated until all fundamental cycles are of even length.
4. The net crossing graph is bipartite. A layer assignment is obtained by finding a 2-coloring of the net crossing graph.

It should be noted that all computations in Steps 2, 3 and 4 are performed on the via-net graph and the net crossing graph. Also, recall that these two graphs can be maintained dynamically as vias are introduced in the layout (Sec. 3.3). Since Step 2 is the most crucial part of Algorithm NetSplit (The majority of odd cycles in the net crossing graph are broken in Step 2), we will explain it in more detail.

The following simple algorithm has been used to generate all the 3-cycles. For each vertex, consider any two vertices adjacent to the vertex. If they are adjacent to each other, then the three vertices form a 3-cycle. In this way, all the 3-cycles can be generated.

For each cycle in the net crossing graph, there exists at least one corresponding cycle in the via-net graph. Such a cycle in the via-net graph can be considered to induce the cycle in the net crossing graph. For example, the 3-cycle (B,F,D) in Fig. 10 is induced by cycle (5,b,8,18,d,17,13,f,12) in Fig. 9. The latter cycle consists of three parts: (5,b,8) in net B, (18,d,17) in net F; and (13,f,12) in net D. These three parts are connected by conflict edges. As shown in Fig. 3, such a cycle in the via-net graph usually corresponds to a "loop" in the layout [1]. A cycle in the net crossing graph may be induced by more than one (usually a small number) cycle in the via-net graph. Thus, to break a 3-cycle in the net crossing graph, it is necessary to generate all the cycles in the via-net graph that induce the 3-cycle. This can be done by traversing the three interconnection trees in the via-net graph corresponding to the three vertices (nets) that form the 3-cycle.

A cycle in the via-net graph is broken if a via candidate in the cycle is chosen to hold a via. Thus, after generating all the cycles in the via-net graph that induce the 3-cycles in the net crossing graph, we want to select the minimium number of via candidates such that every cycle generated contains at least a selected via candidate. This problem is equivalent to the subset cover problem which has been shown to be NP-hard [7]. A simple approximation method is to select the via candidate that is contained in the maximum number of cycles repeatedly until every cycle contains a selected via candidate. When a via candidate is selected, all the cycles containing the via candidate are excluded from further consideration.

5.2. The Vertex Swapping Algorithm

Vertex swapping has been a well known technique for min-cut network partitioning [6][13]. It is not surprising that the same technique can be applied to a similar problem of finding a maximum cut of a weighted graph. Since the via minimization problem can be formulated as a maximum cut problem, a heuristic layer assignment algorithm has been developed by using vertex swapping [27].

Consider the cluster graph G = (V,E) for a given layout. Each edge e has a real-valued weight w(e). The goal is to maximize the cut capacity

$\sum_{e \in E(A,B)} w(e)$ where (A,B) is a partition of V. Starting with an initial partition of V into two sets, the algorithm tries to increase the cut capacity by a series of vertex exchanges between the two sets. The algorithm stops when no further improvement is possible.

Let A and B be any partition of V. For any vertex a in A, define an internal connection I(a) and an external connection E(a) by

$$I(a) = \sum_{(a,x) \in E, x \in A} w(a,x),$$
$$E(a) = \sum_{(a,y) \in E, y \in B} w(a,y).$$

Similary, define I(b) and E(b) for any $b \in B$. Let $D(v) = I(v) - E(v)$ for any $v \in V$. It can be verified that D(v) is precisely the gain in cut capacity if v is moved from its current set to the other set. After a vertex moving, the D values can be recalculated easily. If vertex a in A is moved to B, then the new D values D' can be calculated by

$$\begin{aligned} D'(a) &= -D(a), & & \\ D'(x) &= D(x) - 2w(a,x) & & x \in A - \{a\}, (a,x) \in E, \qquad (2) \\ D'(y) &= D(y) + 2w(a,y) & & y \in B, (a,y) \in E. \end{aligned}$$

The D values can be recalculated similary if a vertex $b \in B$ is moved to A.

Let us state the vertex swapping algorithm as follows:

Algorithm VertexSwap

1. Construct an initial partition of the graph. Compute D(v) for all $v \in V$.

2. Determine a sequence of vertices $v_1, v_2, ..., v_m$ whose total gain $\sum_{i=1}^{m} D(v_i)$ is positive. The vertices v_i's are selected such that

$$D(v_1) = max\{D(v) \mid v \in V\},$$

$$D(v_i) = max\{D(v) \mid v \in V - \{v_1, v_2, ..., v_{i-1}\}\}, \ i > 1.$$

Note that each time after a vertex is selected, all the D values must be recalculated using (2) before the next vertex can be selected. If a vertex sequence with positive total gain is found, then move the vertices from their current sets to their opposite sets. Repeat this process until such a vertex sequence can not be found.

6. Three-Layer Via Minimization

Like two-layer via minimization, three-layer via minimization can also be formulated as a coloring problem with respect to the wire-segment graph. However, there is an additional constraint peculiar to three-layer via minimization. For example, in Fig. 17, wire segments 1 and 2 connect to each other at location a. Wire segment 3 in a different net overlaps wire segment 2 horizontally. If a via is placed at location a, then segment 3 cannot be on the second layer. Otherwise, the three wire segments will be all connected together due to the punch-through via.

Because of its similarity with the graph three-coloring problem which is NP-complete [7], the three-layer via minimization problem has been shown to be NP-complete, and a fast heuristic algorithm was proposed [2]. Starting with a given layer assignment, the algorithm tries to eliminate vias one after another by reassigning layers to the wire segments. A given layout with layer assignment is represented by a via-net graph though only vias in the given layout are considered as via candidates. In other words, vertices of the via-net graph now correspond to vias and subnets in the layout.

The algorithm determines if a via can be eliminated or not by using information local to the via. For a given via v, let LEVEL(1) be the set of subnets incident to v, and let LEVEL(j), $j > 1$, be the set of subnets which are in conflict with subnets in LEVEL(j-1) and which are not in

EVEL(2),..., LEVEL(j-1). In the via-net graph, vertices cor-
subnets in LEVEL(j) are j distance away from via v. Thus
EL(j) can be easily computed by a breadth first search through
aph.

rithm tries to eliminate a via by first considering the possi-
ent of all subnets in LEVEL(1). If this fails, it considers the
EVEL(1) and LEVEL(2). This process is repeated and every
he subnets in the next level will be taken into account. It is
ere are too many combinations to try all possibilities. Thus,
he algorithm has been limited to consider only up to level 2.

some valuable
G' in Sec.4.1

References

1. K. C. Chang and D. H. Du, Efficient algorithms for layer assignment problem, IEEE Trans. on Computer-Aided Design, vol. CAD-6, no. 1, January 1987, pp. 67-78.

2. K. C. Chang and H. C. Du, Layer assignment problem for three-layer routing, IEEE Trans. on Computers, vol. 37, no. 5, May 1988, pp. 625-632.

3. R. W. Chen, Y. Kajitani, and S. P. Chan, A graph-theoretic via minimization algorithm for two-layer printed circuit board, IEEE Trans. Circuits Syst., vol. CAS-30, no. 5, May 1983, pp. 284-299.

4. M. J. Ciesielski and E. Kinnen, An optimum layer assignment for routing in IC's and PCB's, Proc. 18th Design Automation Conf., 1981, pp. 733-737.

5. N. Deo, Graph Theory with Applications to Engineering and Computer Science, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1974.

6. C. M. Fiduccia and R. M. Mattheyses, A linear-time heuristic for improving network partitions, Proc. 19th Design Automation Conf., 1982, pp.175-181.

7. M. R. Garey and D. S. Johnson, Computers and Intractability, a

Guide to the Theory of NP-completeness, W. H. Freeman and Company, San Francisco, 1979.

8. F. Hadlock, Finding a maximum cut of a planar graph in polynomial time, SIAM J. Computing, vol.4, no. 3, Sept. 1975, pp. 221-225.

9. A. Hashimoto and J. Stevens, Wire routing by optimizing channel assignment within large apertures, Proc. 8th Design Automation Workshop, 1971, pp. 155-169.

10. J. Hopcroft and R. Tarjan, Efficient planarity testing, JACM, vol. 21, no. 4, 1974, pp.549-568.

11. C. P. Hsu, Minimum-via topological routing, IEEE Trans. Computer-Aided Design, vol. CAD-2, no. 4, Oct. 1983, pp. 235-246.

12. Y. Kajitani, On via hole minimization of routing in a 2-layer board, Proc. IEEE 1980 Int. Conf. Circuits Computers, June 1960, pp. 295-298.

13. B. W. Kernighan and S. Lin, An efficient heuristic procedure for partitioning graphs, Bell Syst. Tech. J., vol. 49, Feb. 1970, pp. 291-307,.

14. R. Kolla and P. Molitor, A note on hierarchical layer assignment, to appear in INTEGRATION, the VLSI Journal.

15. Y. S. Kuo, T. C. Chern and W. Shih, Fast algorithm for optimal layer assignment, Proc. 25th Design Automation Conf., 1988, pp. 554-559.

16. R. J. Lipton and R. E. Tarjan, A separator theorem for planar graphs, SIAM J. Appl. Math., vol. 36, 1979, pp. 177-189.

17. R. J. Lipton and R. E. Tarjan, Applications of a planar separator theorem, SIAM J. Comput, vol. 9, 1980, pp. 615-627.

18. C. L. Liu, Introduction to Combinatorial Mathematics, McGraw-Hill Inc., New York, 1968.

19. M. Marek-Sadowska, An unconstrained topological via minimization problem for two-layer routing, IEEE Trans. Computer-Aided Design, vol. CAD-3, no. 3, July 1984, pp. 184-190,.

20. P. Molitor, Via minimization for systolic arrays, private communication, 1988.

21. N. J. Naclerio, S. Masuda and K. Nakajima, Via minimization for gridless layouts, Proc. 24th Design Automation Conf., 1987, pp. 159-165.

22. K. Paton, An algorithm for finding a fundamental set of cycles of a graph, Comm. ACM, Vol. 9, No. 9, Sept. 1969, pp. 514-518,.

23. R. Y. Pinter, Optimal layer assignment for interconnect, J. VLSI and Computer Systems, vol. 1, no. 2, 1984, pp. 123-137.

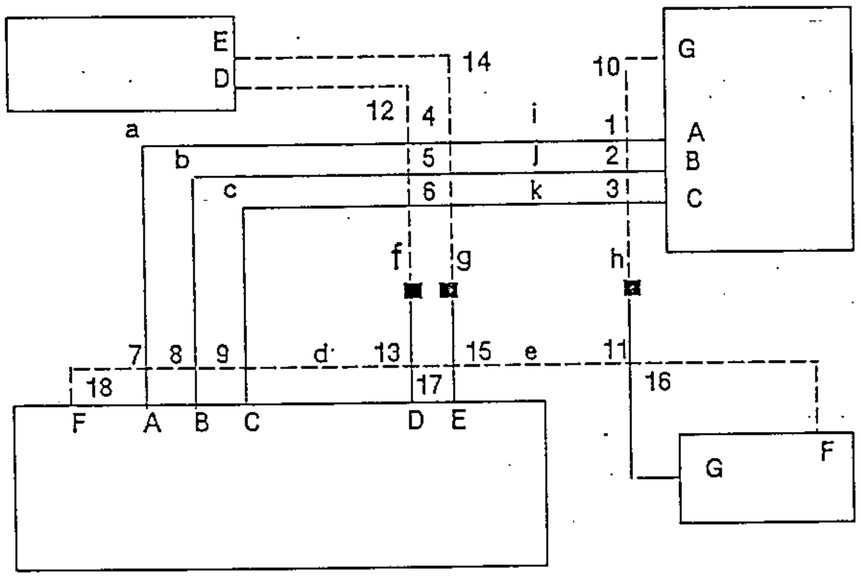
24. M. Sarrafzadeh and D. T. Lee, A new approach to topological via minimization, Technical Report CIMS-87-01, Northwestern University, 1987.

25. J. Soukup, Circuit Layout, Proceedings of the IEEE vol. 69, Oct. 1981, pp. 1281-1304.

26. K. R. Stevens and W. M. VanCleemput, Global via elimination in generalized routing environment, Proc. Int. Symp. on Circuits and Systems, 1979, pp. 689-692.

27. X. Xiong and E. S. Kuh, The constrained via minimization problem for PCB and VLSI design, Proc. 25th Design Automation Conf., 1988, pp. 573-578.

28. K. Yoshida, Layout verification, in Advances in CAD for VLSI, vol. 4, ed. T. Ohtsuki, North-Holland, 1986.



solid line : layer 1
dotted line : layer 2

Fig. 1. A possible layer assignment

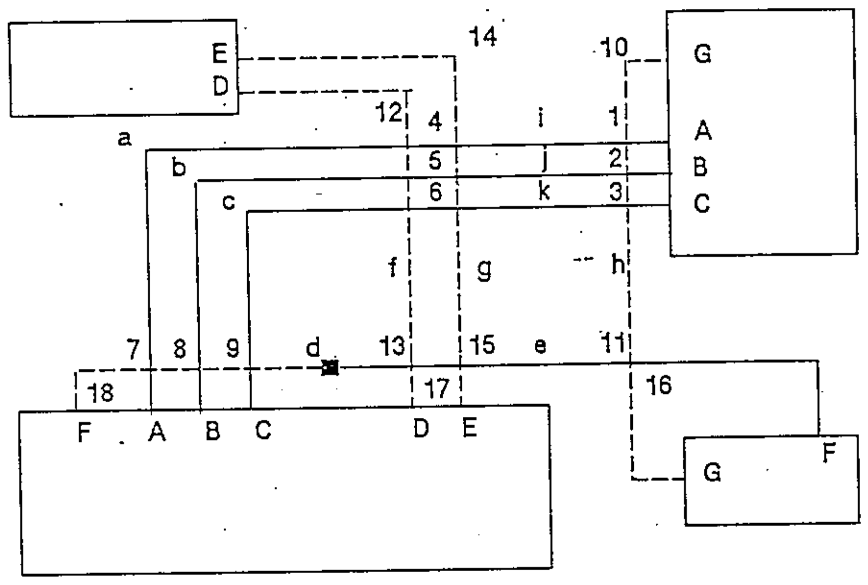


Fig. 2. Optimal layer assignment

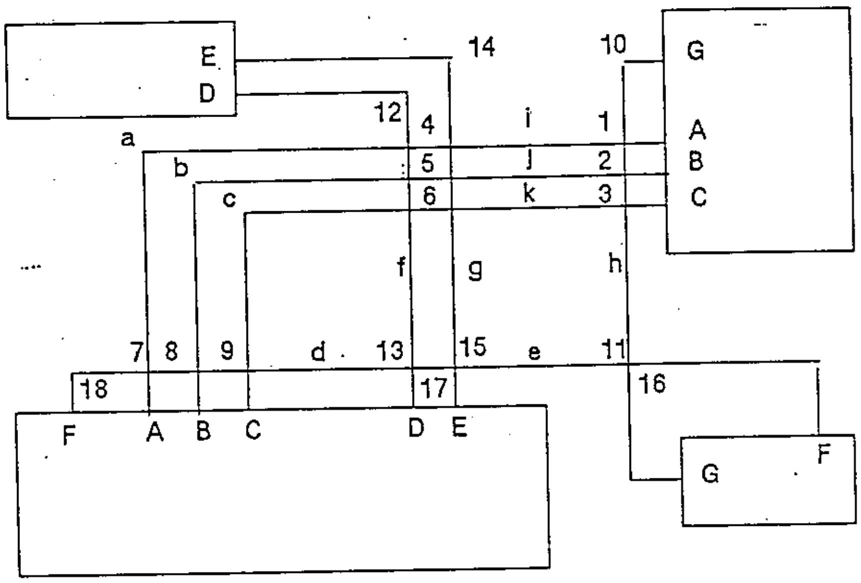
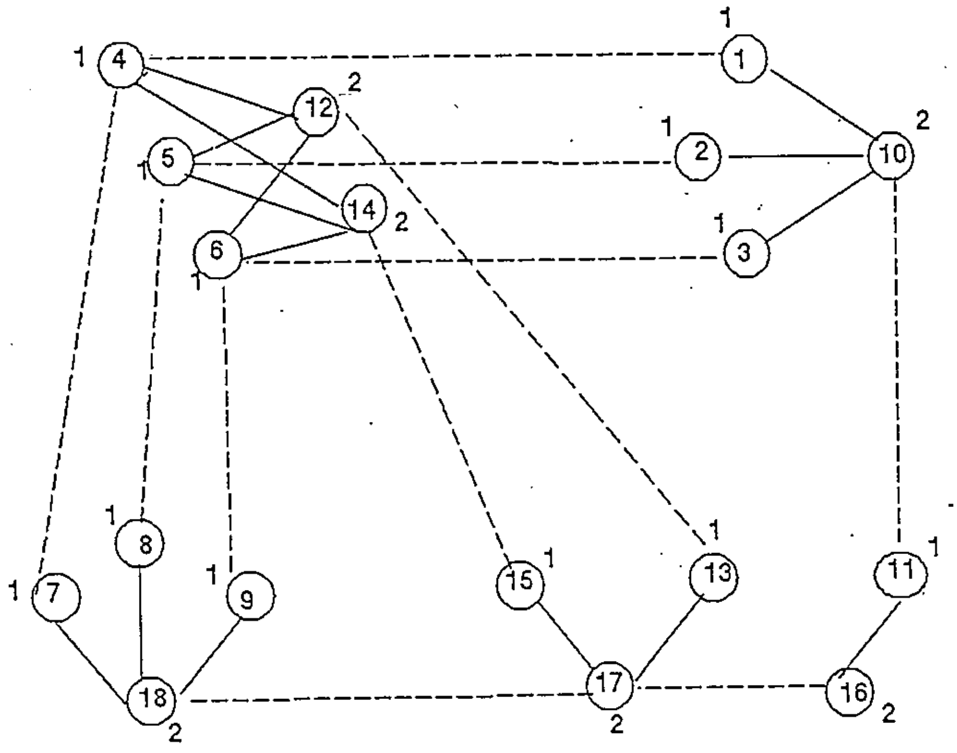


Fig. 3. A layerless layout



Solid lines are conflict edges.
Dotted lines are continuation edges.

Fig. 4. Wire – segment graph

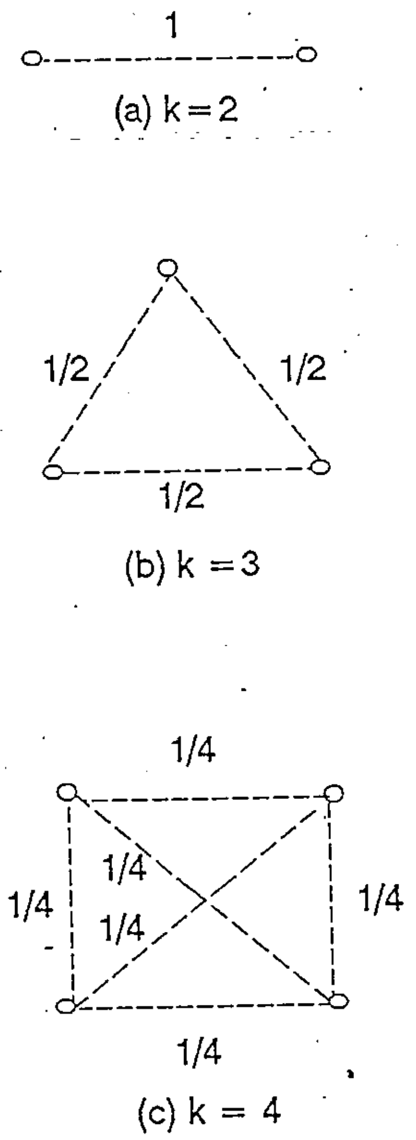
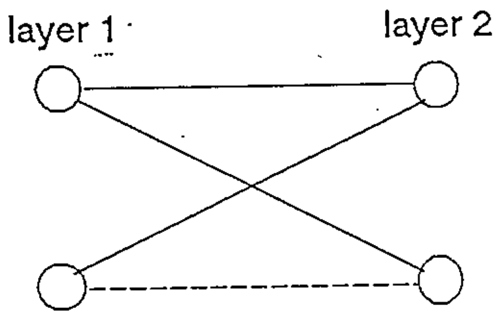


Fig. 5. Continuation edges with weights



solid line: conflict edge
dotted line: continuation edge

Fig. 6. A continuation edge determining an essential via

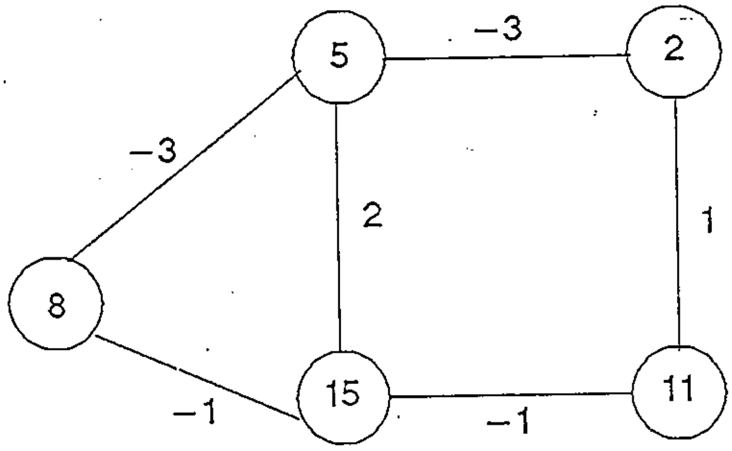
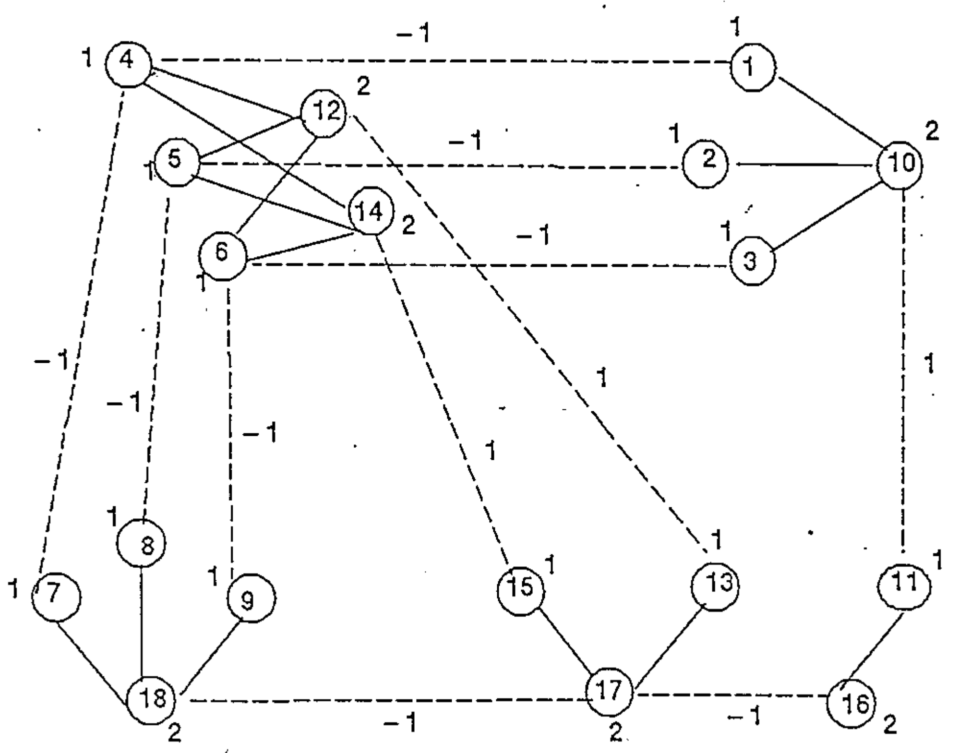
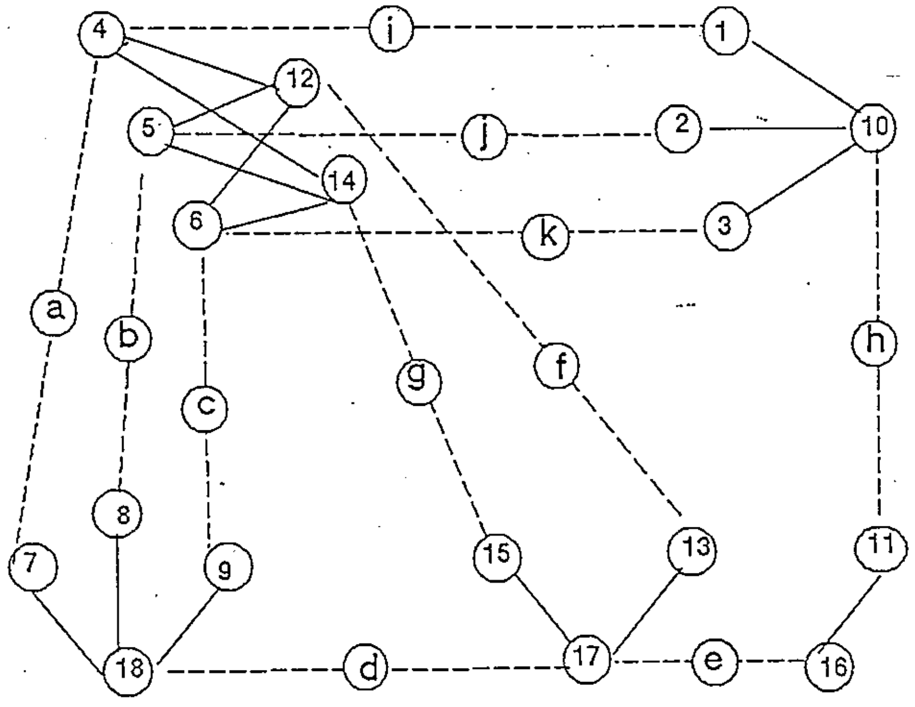


Fig. 7. Cluster graph G



Solid lines are conflict edges.
Dotted lines are continuation edges.

Fig. 8. Wire – segment graph



Solid lines are conflict edges.
Dotted lines are interconnection edges.

Fig. 9. Via – net graph

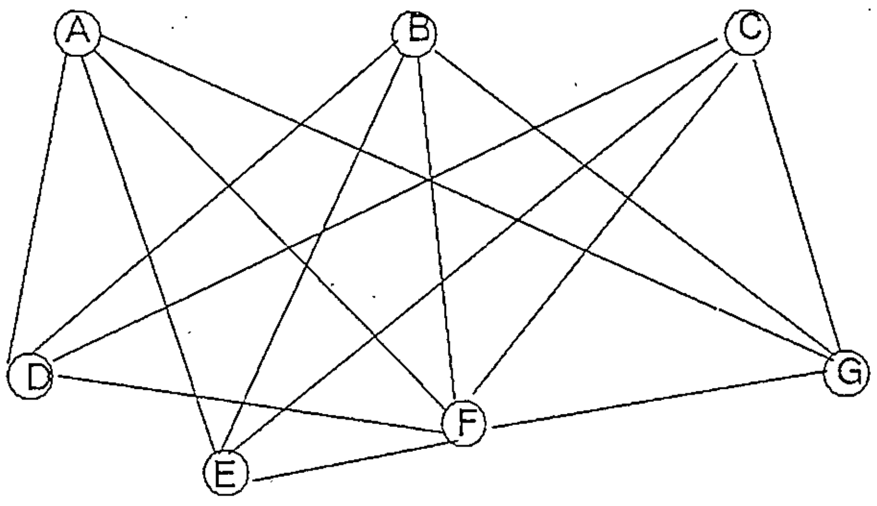
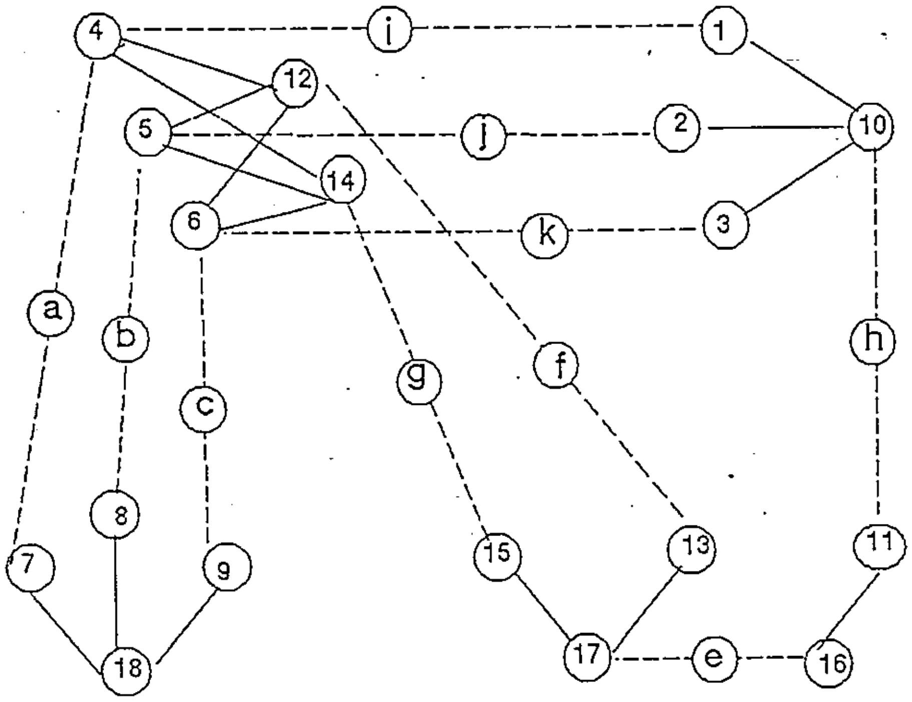


Fig. 10. Net crossing graph



Solid lines are conflict edges.
Dotted lines are interconnection edges.

Fig. 11. Via – net graph after via introduction

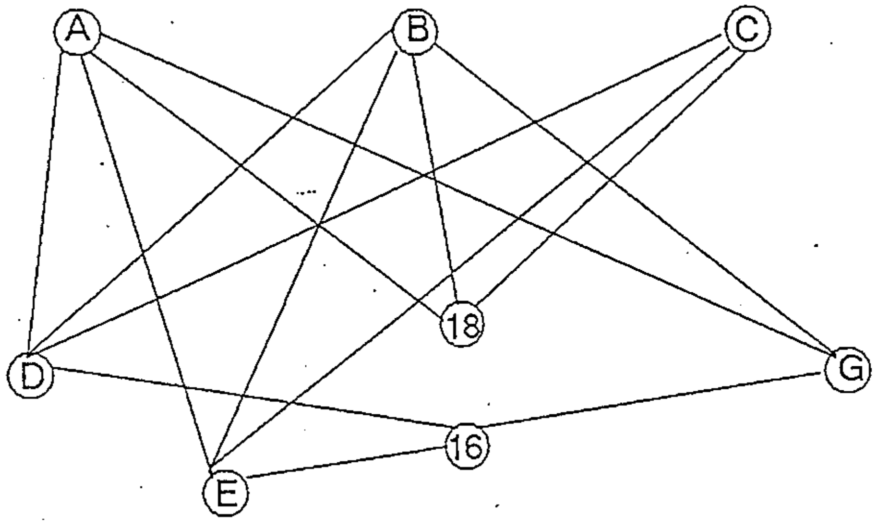
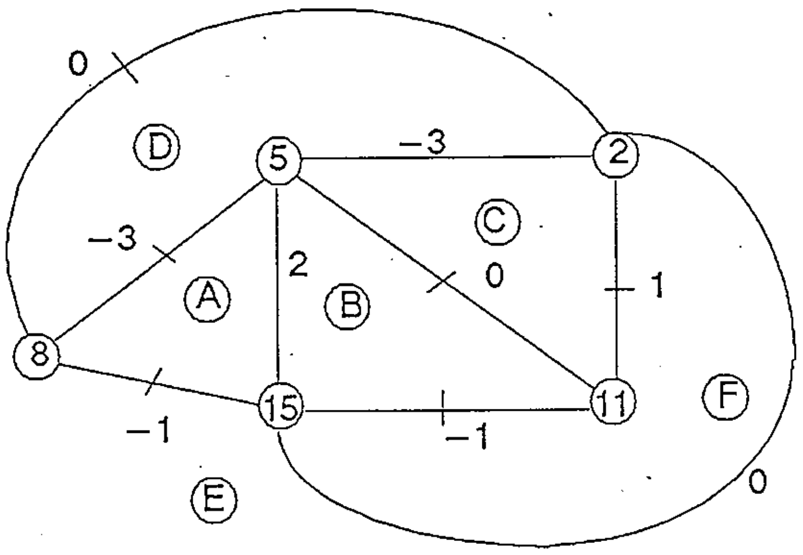
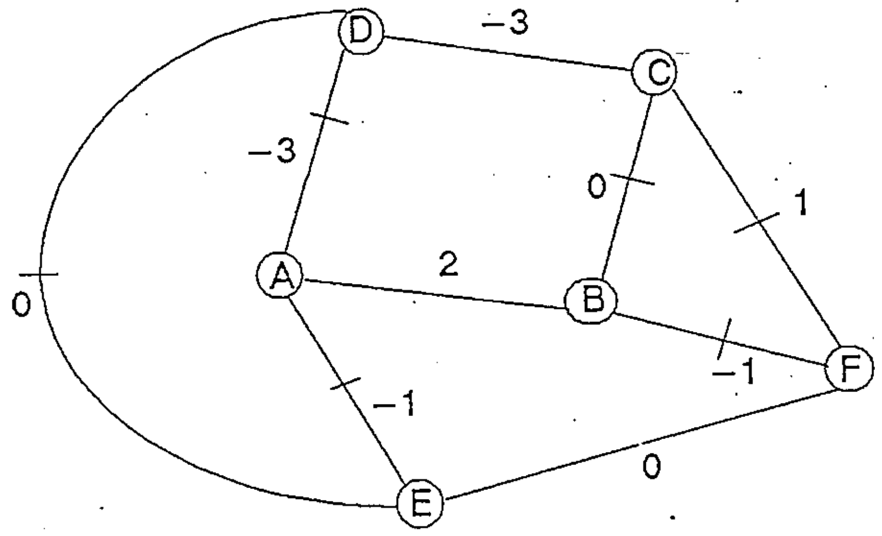


Fig. 12. Net crossing graph after net split



(Edges marked with short bars form a cut.)
Fig. 13. Triangulation G_t of G



(Edges marked with short bars
form an even – degree edge set.)

Fig. 14. Geometric dual G_d of G_t

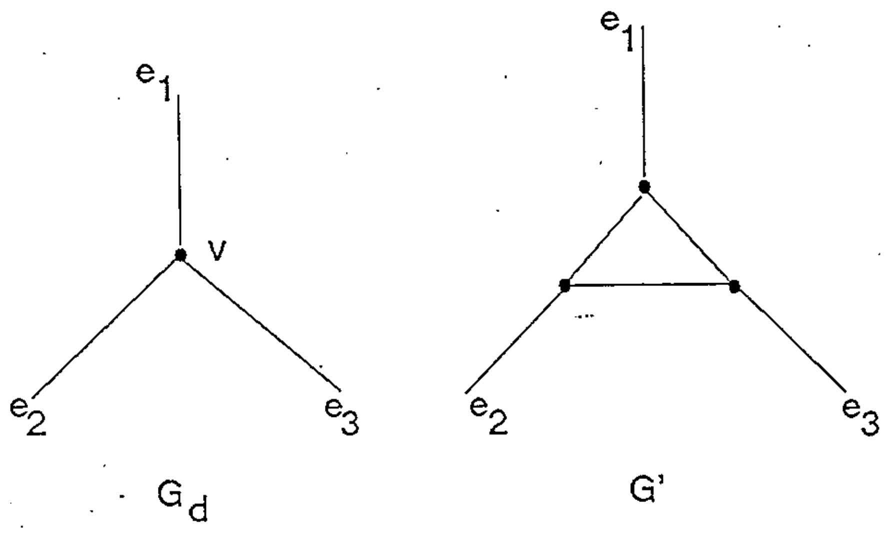
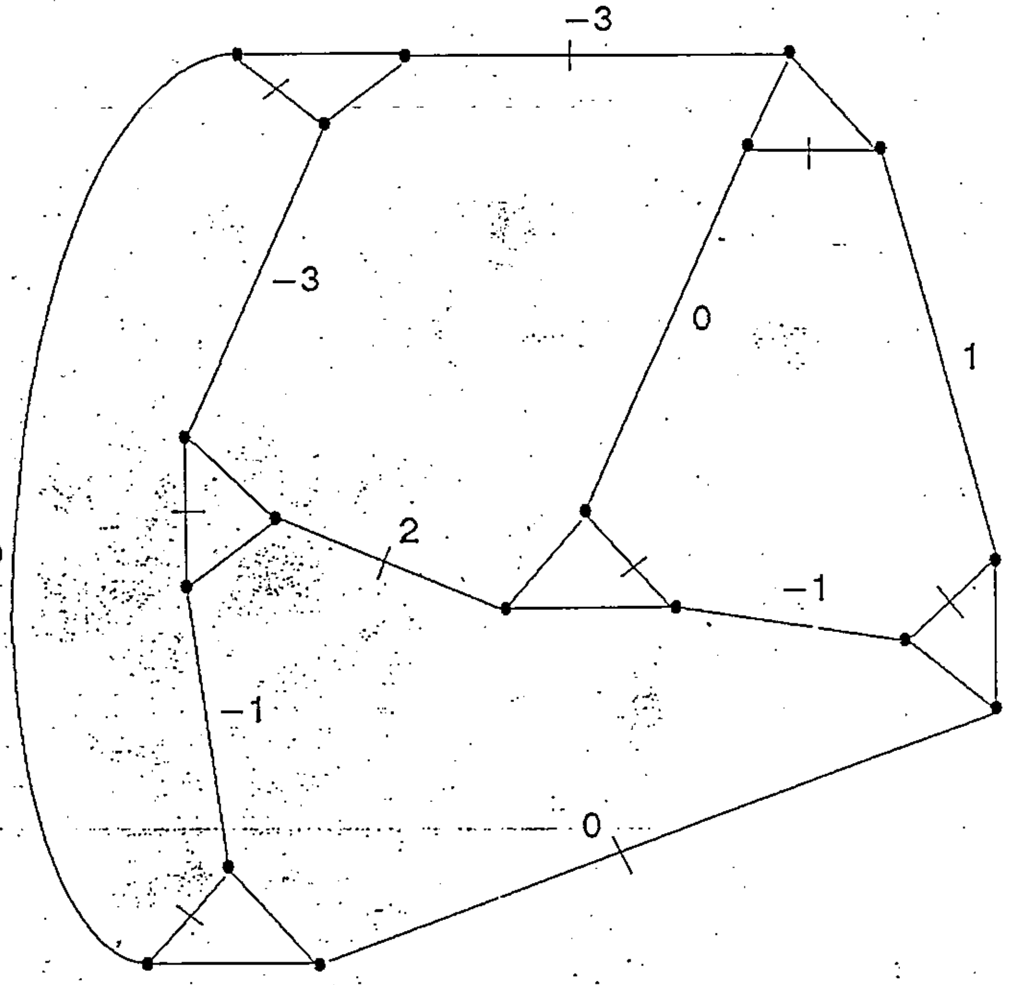


Fig. 15. The 3 – clique substituting a vertex



(Edges marked with short bars form a complete matching.)

Fig. 16. Graph G' constructed from G_d

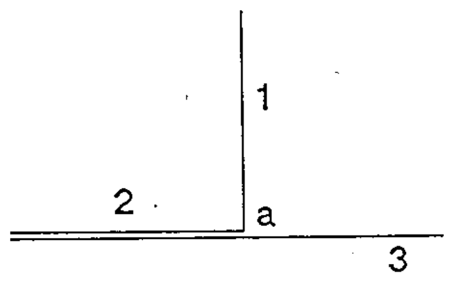


Fig. 17. Additional constraint