TR-91-016

A Linear Time Optimal Via Assignment Algorithm for Three-Dimensional Channel Routing

A Linear Time Optimal Via Assignment Algorithm for Three-Dimensional Channel Routing

Jan-Ming Ho[1]
Institute of Information Sciences
Academia Sinica
R. O. C.

1

Abstract

A three-dimensional channel refers to a 3-D rectangular block with multiple routing layers. Terminals exist only on the top and the bottom layers and they form two well-aligned 2-D rectangular channels. In this paper, we consider a special version in which the three-dimensional channel contains only three layers. The routing algorithm is as follows. First, a channel routing algorithm is applied to both the top and the bottom layer to route the terminals belonging to the same net on the same layer. The second step is to form another channel routing problem as defined below. A net N is said to be an *inter-layer net* if it contains terminals on both the top and the bottom layers. Two via positions in the middle layer are selected for each inter-layer net. The first via is chosen from the position immediately below one of the terminals belonging to N, while the second is chosen from the position immediately above one of the terminals belonging to N. Notice that it defines a channel routing problem containing only two-terminal nets. The channel routing algorithm is then applied. In this paper, we present a linear time optimal via assignment algorithm for the second step decribed above such that number of incompletely routed nets are minimized.

[1]This work has been supported in part by the National Science Council, R.O.C., under Grant NSC 79-0404-E-001-01.

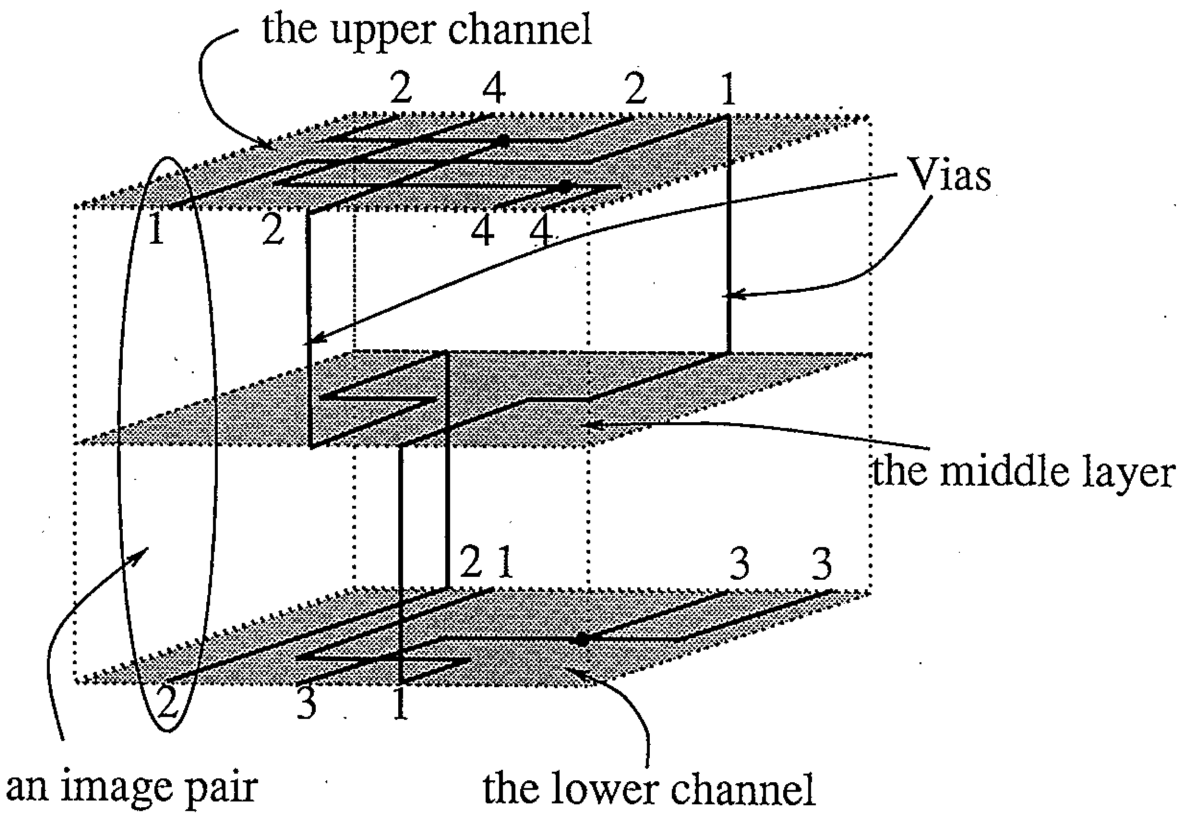

Figure 1: A three-dimensional channel; where nets 1 and 2 are inter-layer nets, while nets 3 and 4 are not.

1 Introduction

The three-dimensional channel routing problem has been studied by C.C. Tong and C.L. Wu [2]. In a two-dimensional channel routing problem, a channel is a rectangular region. Terminals to be connected by wires exist on the upper and the lower boundaries of the channel. Interior of the rectangle is used as the routing region. A three-dimensional channel (see figure 1) is defined as a multilayer rectangular block containing two well-aligned channels on the top and the bottom layers respectively. Terminals does not exist in the middle layers. A net specifies a set of terminals to be connected. A net is said to be an *inter-layer* net if it contains terminals on both the top and the bottom layers. Vias must be used to complete the connection of an inter-layer net. she subset N' of terminals of an inter-layer net N is called the *upper subnet* (*lower subnet*) of N if N' contains all the terminals of N on the top (bottom) layer. Both upper and lower subnets are also called *intra-layer* subnets. In [2], channels are defined as having three layers. Their routing model first performs channel routing algorithm on the top and the bottom layers to complete the connection of the terminals on the same layers. For the inter-layer nets, two vias, one for the upper subnet and the other for the lower subnet, are assigned to bring the subnets to the boundaries of the middle layer to form another two-dimensional channel routing problem. The via assigned to a upper (lower) subnet locates immediately below (above) one of its terminals. A terminal t' is said to be the *image* of the other terminal t if t and t' align on the same z-axis. Note that a terminal and its image cannot be assigned vias at the same grid point between them. This pair of terminals is said to form a *positional conflict* in [2]. The upper (lower) subnet of an inter-layer net N is said to be blocked if no via is assigned to it, i.e., each grid point immediately below (above) its terminals belongs to a via assigned to other nets. An inter-layer net is said to be *blocked* if either its upper subnet or its lower subnet is blocked. The objective of the optimal via assignment problem for three-dimensional channel routing, *OVA problem* for short, is to minimize the number of blocked intra-layer subnets.

In [2], a *constraint graph* is defined as follows. A vertex of the constraint graph is a terminal of an inter-layer net. An edge exists between two vertices if the corresponding terminals either belong to the same upper (or lower) subnet of an inter-layer net or form a positional conflict. Note that the construction of the constraint graph takes $O(n^2)$ time. Tong and Wu give characterizations of the constraint graph and presented an $O(n^2)$ algorithm for the *OVA problem.* In this paper, we present an algorithm for the *OVA problem* with linear time complexity. In section 2, we give a new formulation of the *OVA problem* based on the notion of a *constraint hyper-graph* and we also describe the linear-time *algorithm OVA*. Note that construction of the initial constraint hyper-graph takes only linear time. Correctness of the *algorithm OVA* is asserted in section 3. We also consider the objective of minimizing the number of incompletely routed nets. In section 4, we present a *modified algorithm OVA* which optimizes the new objective.

2 The Constraint Hyper-Graph and the OVA Algorithm

The constraint hyper-graph is a dynamic data structure used to describe the *OVA problem* at its current status. An intra-layer subnet is said to be *active* if no via has been assigned to it. A *super-node* is associated with each active intra-layer subnet and it can be implemented, say, as a linked list. Note that a super-node corresponds to a clique in the constraint graph defined in [2]. A terminal is said to be *active* if it belongs to an active inter-layer subnet and its adjacent grid position in the middle layer has not been assigned to any via. We associcate a *vertex* for each active terminal. An edge connecting two vertices is called a P-edge if the corresponding terminals form a positional conflict. Note that an edge is the candidate of a via position for its adjacent active intra-layer subnet. A via can always be assigned to a terminal, if the corresponding vertex does not have an edge connecting to it. In which case, the terminal is called a *leaf.* A constraint hyper-graph can thus be denoted as a triple $H = (V, N, E)$, where V denotes the set of vertices, N denotes the set of super-nodes, and E is the set of edges. Notice that N is a partition of V. The *associated constraint graph* of H is the graph $G_H = (V, E')$, where E' is the union of E and the edges contained in the cliques defined by super-nodes in N. Edges in $E' - E$ are referred to as the N-edges.

Initialization of the constraint hyper-graph H can be done as the following. We are going to use four arrays *U1, L1, U2*, and *L2* to store information of the terminals belonging to each side of the two channels, where *U1* and *U2* belong to the upper channel and *L1* and *L2* belong to the lower channel. Since a net is specified as a set of terminals in their coordinates in the channels, we can easily determine if a net is an inter-layer net and whether super-nodes should be created for it in time proportional to the size of the net; and information of each terminal can be written to the corresponding position in one of the four arrays in constant time as it is retrieved. *U1* and *L1* are then examined simultaneously to create the set of edges E and the set of vertices V. In this way, the constraint hyper-graph is created in a total of $O(n)$ time, where n is the number of grid point on each side of the upper and lower channels. Tong and Wu's algorithm constructs the *associated constraint graph* G_H explicitly. Since every intra-layer subnet constitutes a clique in G_H, the construction of G_H takes $O(n^2)$ time. Note that our data structure maintains the cliques implicitly in the linked-lists representing the super-nodes to keep the time complexity down to $O(n)$. In our paper, we are going to use the notations super-node and cliques interchangeable since they only differ in the implementation details.

The vertices are given priorities 1, 2 and 3 as the following. A vertex is given priority 1 if it is not contained in any P-edge. If a P-edge connects two vertices corresponding to terminals in the same inter-layer net, then the two vertices are also given priority 1. A vertex is given priority 2 if it correponding to the last terminal t left in the super-node containing t. All the other vertices are given priority 3. Note that priorities of the vertices can be computed in a total time of $O(n)$.

A variable *INCOMPLETE* is used to store the number of failures in assigning a via to an intra-layer subnet and is initialized to 0. *Algorithm OVA* then iteratively selects from V a vertex v of the lowest priority, assigns a via for the net containing t and updates the data structures, where t denotes the active terminal corresponding to v. Let s denote the super-node containing v. If v is a leaf, *algorithm OVA* assigns the adjacent via to v and calls *CLEAR(v)*. If v is not a leaf and the priority of v is 1, then *algorithm OVA* assigns the via associated with e, the edge connecting v and its image v', to v and e is deleted from E. Procedure *CLEAR(•)* is then performed on both v and v'. If the priority of v is 2 or 3, then *algorithm OVA* performs the following before calling *CLEAR(v)*. Let v' be the vertex connected to v by an edge e and s' be the super-node containing v'. If s' contains v' as its last vertex, then *INCOMPLETE* is incremented by one and s' is removed from N. If s' contains more than one vertex, then v' is deleted from s'. Then the edge e is deleted from E. The procedure *CLEAR(v)* (see figure 2) first examines every vertex $\overline{v}$ in s, $\overline{v} \neq v$. If $\overline{v}$ is a leaf, then it is simply deleted form s. Otherwise, let $\overline{v}'$ be the vertex connected to $\overline{v}$ by the edge $\overline{e}$. Then, $\overline{v}'$ is marked as a leaf with its priority being raised to 1 and the edge $\overline{e}$ is deleted from E. After all the other vertices in S is removed, v is deleted from s and s is deleted from N. The algorithm terminals when there is no vertex in V. Obviously, *algorithm OVA* runs in linear time.

Theorem 1 *Algorithm OVA runs in $O(n)$ time.*

In the $O(n^2)$ time algorithm of Tong and Wu's, most of the time is spent in identifying and breaking loops in the graph obtained by contracting the cliques in G_H. Our result shows

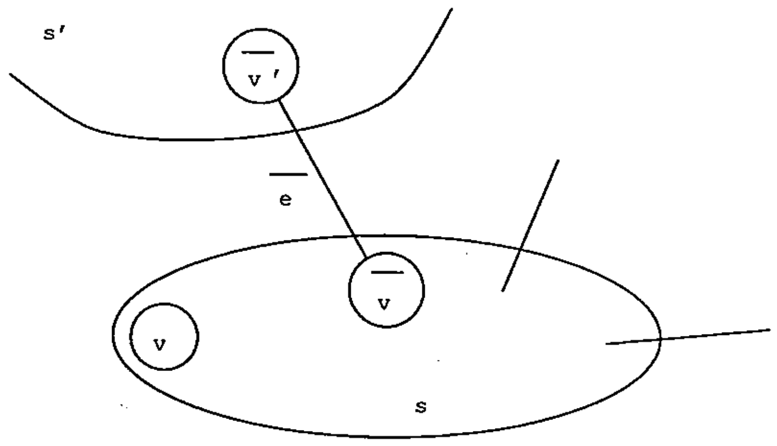

Figure 2: The procedure *CLEAR(v)*.

that these steps are unnecessary.

3 Correctness of Algorithm OVA

Let $H = (V, N, E)$ be the constraint hyper-graph describing a given three-dimensional channel routing problem, and $G_H = (V, E')$ be the associated constraint graph. Let $G_1, G_2, \ldots, G_l$ be the connected components of G_H. Tong and Wu [2] has shown that they can always assign vias to the intra-layer nets contained in each connected component G_i such that at most one failure occurs. *Algorithm OVA* basically takes advantage of this characterization. To assert the correctness of *algorithm OVA*, we first point out when a failure in assigning vias should occur.

Definition 1 *A connect component G_i is said to be* distinguished *if G_i has the following properties:*

(1) G_i does not have a leaf nor a pair of images belonging to the same net;

(2) G_i is a tree of the cliques corresponding to the intra-layer subnets.

Lemma 1 *There is at least one failure in assigning vias to a distinguished connected component G_i.*

Proof: Let q be the number of cliques in G_i. In other words, G_i contains q intra-layer subnets. Since the cliques form a tree, there are $q-1$ P-edges. Also note that each P-edge allows at most one via, thus at least one of the intra-layer subnets fails to attaining a via.

$\diamond$

Notice that *algorithm OVA* always assigns vias to the intra-layer nets of G_i with at most one failure if G_i is distinguished.

Lemma 2 *Algorithm OVA assigns vias to the intra-layer nets of G_i with at most one failure if G_i is distinguished.*

Proof: We shall prove by induction on the number of vertices contained in G_i. According to the definition of a distinguished component, the minimum size of G_i is two, v and v', each belonging to a distinct super-node s and s', respectively. Obviously only one of the intra-layer subnets corresponding to s and s' will get the via while the other fails. Now, let's assume that for every distinguished component G' containing less than k vertices, there is at most one failure in assigning vias to the intra-layer nets of G' by *algorithm OVA*, where k is the size of G_i and $k > 2$. Since G_i is distinguished, a clique s at the leaf of G_i must be contain only one vertex v. By our priority scheme, v gets the priority 2. Also notice that the priority of a vertex w at an internal clique of G_i gets priority 3. Thus the first vertex selected by *algorithm OVA* has a priority 2. Let's denote this vertex as v and v' as the vertex connected to v by a P-edge e. Denote s' as the super-node containing v'. If s' contains v' as the only vertéx, since G_i is a connected component, we immediately induce that there are two vertices in G_i. Which contradicts our previous assumption of $k > 2$. Thus s' contains at least two vertices (see figure 3). In processing v, *algorithm OVA* will assign the via associated with e to v and it will delete v' from s'. Obviously, G_i remains distinguished at its new state and its size is decreased by two. Thus, we can apply the principle of induction to conclude that *algorithm OVA* will process G_i with a single failure.

$\diamond$

Lemma 3 *If there is a vertex with priority 1 in the connected component G_i, then vias can be assigned to the intra-layer nets of G_i without failure.*

Proof: This can be proved by using induction on the number of vertices in G_i. If v is the only vertex in G_i, then it must be the leaf and we can assign a via to v without conflict.

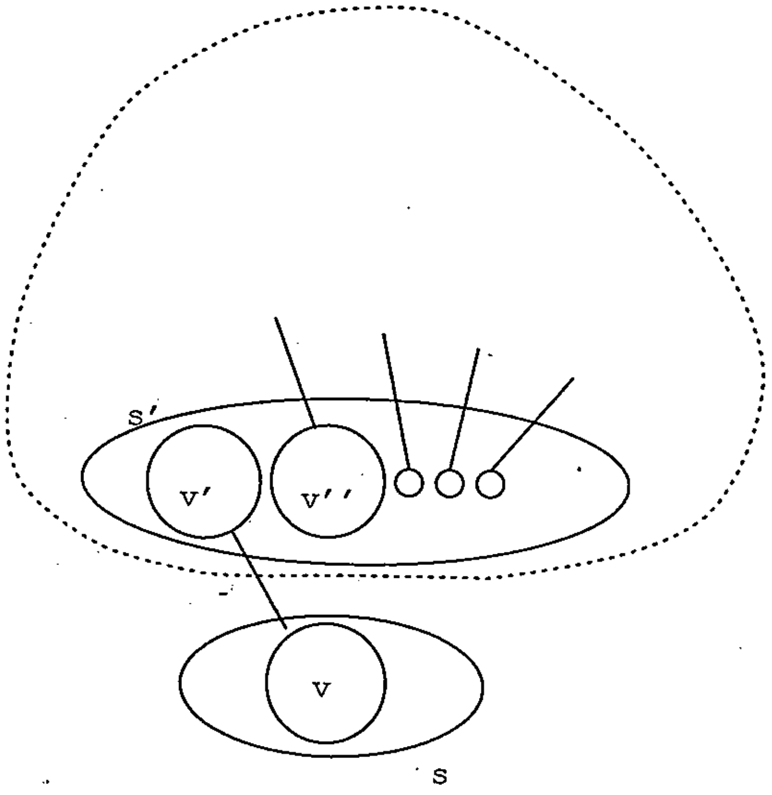

Figure 3: Illustration of the processing of a distinguished G_i.

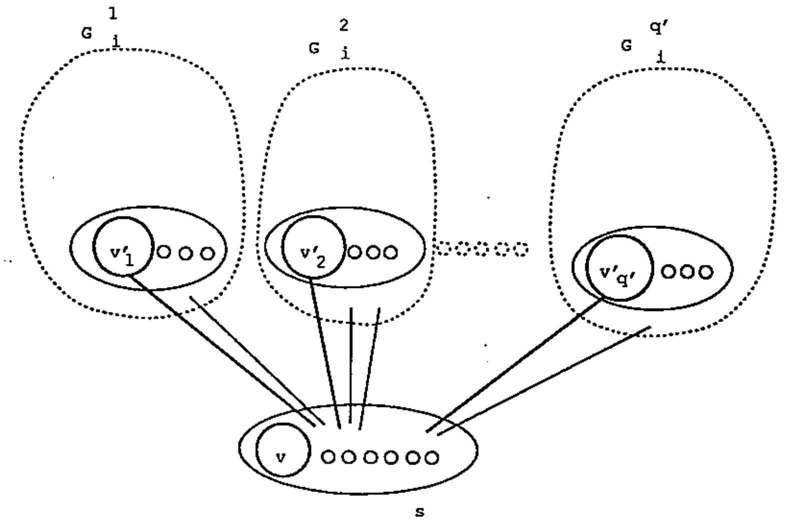

Figure 4: Illustration of the proof of lemma 3-(1).

Now, let's assume that the lemma is true for every connected component having less than k vertices, where $k > 1$ is the number of vertices in G_i. Now let's consider the first vertex v in G_i selected by *algorithm OVA*. v is either a leaf or is contained in an image pair both belonging to the same inter-layer net according to the definition of priority 1.

(1) v is a leaf. Then a via can be assigned to v without conflict and we can apply the procedure *CLEAR (v)*. Denote s as the super-node containing v. Note that s contains at least two vertices. After applying *CLEAR (v)*, G_i is decomposed into smaller components $G_i^1, G_i^2, \ldots, G_i^{q'}$ (see figure 4). Each G_i^j contains at least a leaf v'_j which is the image of a vertex $v_j \neq v$ containing in s. Note that the size of G_i^j is smaller than k. By the hypothesis of induction, vias are also assigned to G_i^j without failures.

(2) v is contained in a P-edge $e = (v, v')$, and both v and v' belong to the same inter-layer net. Denote s and s' as the super-nodes containing v and v' repectively. Note that e can be (a) contained in a cycle of G_i or (b) otherwise (see figure 5). *algorithm OVA* will assign the via associated with e to v, and perform *CLEAR(•)* on both v and v' and yields several connected components $G_i^1, G_i^2, \cdots, G_i^{q'}$ in both cases (a) and (b). Similar to the discussions in (1), we know that the components each contains at least a leaf and size of each G_i^j is smaller than k. Again, by the hypothesis of induction, vias are successfully assigned to G_i^j.

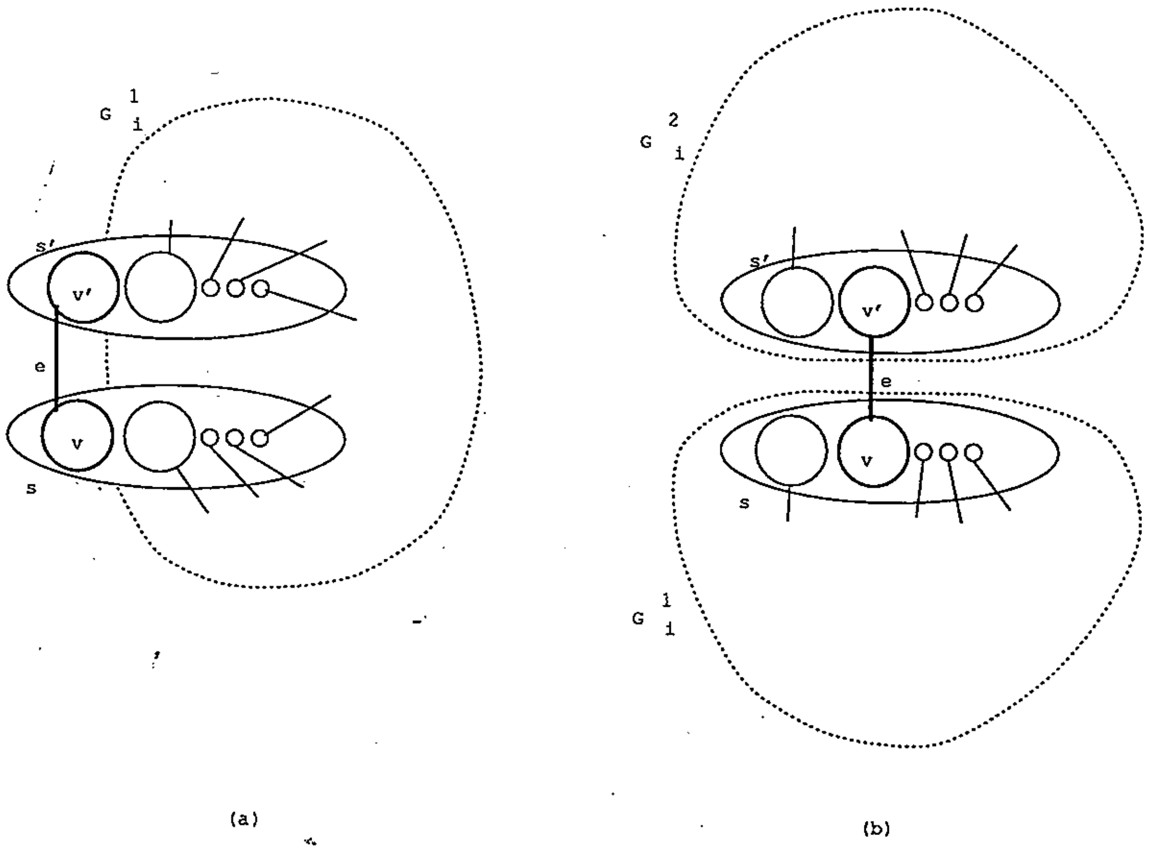

Figure 5: Illustration used in the proofs of lemma 3-(2) and 4.

◇

Lemma 4 *Algorithm OVA assigns vias to G_i optimally.*

Proof: Equivalently, we show that *algorithm OVA* assigns vias to G_i with

(1) one failure if G_i is distinguished;

(2) zero failure if otherwise.

Again, we are going to use induction on the number of vertices in G_i. It's obvious that the equivalent statement is satisfied when the size of G_i is one or two. Now, we assume that the equivalent statement is satisfied by every connected component G' of size less than k, the size of G_i. (1) is the result of lemma 1 and 2. If G_i contains a leaf or a P-edge connecting two vertices belonging to the same inter-layer net, then (2) is satisfied as asserted by lemma 3. Thus, we can assume G_i as having neither a leaf nor a P-edge connecting two vertices belonging to the same inter-layer net, and G_i is not distinguished. We are going to show that G_i satisfies (2) under these assumptions. First, we know that the priority of a vertex of G_i must be either 2 or 3. If G_i contains a vertex of priority 2, i.e., the super-node s contain v has only one vertex. Then similar to the arguments we used in the proof of lemma 2, we conclude that *algorithm OVA* will recursively delete such vertices until none is left. Note that G_i is still connected and still holds the three properties of G_i mentioned previously, i.e. it does not have neither a leaf nor a P-edge connecting two vertices belonging to the same inter-layer net, and it is not distinguished. *Algorithm OVA* will now select a vertex v with priority 3. Denote v' as the vertex connected to v by a P-edge e. Note that the priority of v' is also 3. Let s and s' denote the super-nodes containing v and v' respectively. e can be (a) contained in a cycle containing both s and s'; or (b) otherwise (see figure 5). Since s' contains no less than two vertices by the property of G_i, the via associated with e can be assigned to v without conflict. Furthermore, since s also contains no less than two vertices, after applying *CLEAR(v)*, the connected component G_i^1 (illustrated in figure 5-(a) and 5-(b)), contains at least one leaf. Lemma 3 shows that *Algorithm OVA* will successfully assign vias to all the intra-layer subnets in G_i^1. While for the connected component G_i^2, since its size is now smaller than k, the size of the previous G_i, *Algorithm OVA* will also successfully assign vias to all the intra-layer subnets in G_i^2 by the hypothesis of induction.

◇

Summarizing the above lemmas, we have the following theorem.

Theorem 2 *Algorithm OVA optimally assigns vias to the intra-layer subnets of a three-dimensional channel routing problem such that the number of intra-layer subnets failed in getting a via to the middle layer is minimized.*

4 The Modified Algorithm OVA'

In this section, we study the problem of optimally assigning vias to the inter-layer nets such that the number of unrouted nets is minimized. As we've seen in the above lemmas, *algorithm OVA* yields one failure in assigning vias to the intra-layer subnets of a connected component G_i if only if G_i is *distinguished.* An extra unrouted net is incurred if the two intra-layer subnets s_1 and s_2 of an inter-layer net N happen to appear in two distinct distinguished components G_i and $G_j, i \neq j$, while *algorithm OVA* fails to assign vias to intra-layer subnets s and s' respectively, where s and s' belongs to different inter-layer nets. Whenever one such pair is selected simultaneously, the number of unrouted nets is decreased by one. We readily recognize that the problem of minimizing the number of unroutable nets is equivalent to that of finding the *maximum cardinality matching.* We'll first present a formal discussion of the approach, then present the algorithm.

First, we are going to construct a *component graph* $G^c = (V^c, E^c)$, where a vertex v_i in V^c denotes a distinguished component and an edge $e = (v_i, v_j)$ in E^c denotes the set of inter-layer nets with one of its intra-layer subnet in G_i and the other in G_j. Let $M \subseteq E$ be the maximum cardinality matching of G^c. For every edge $e = (v_i, v_j) \in M$, let $s = s_1 \cup s_2$ denote one of the inter-layer nets associated with e, where s_1 in G_i and s_2 in G_j denote the two intra-layer subnets of s. We can modify G_i and G_j by removing s_1 and s_2 respectively which yields several connected components. By removing s_1 and G_i, for each edge $e' = (v, v')$ connecting a vertex v in s_1 to another vertex $v' \in V$, the vertex v' is now designated as a leaf whose priority becomes 1 and the edge e' is removed from E, and we also remove the vertex v from V and s_1 from N. Here $H = (V, N, E)$ is the hyper-graph representing the OVA problem. Each of the connected components, say G_i yielded by removing the super-nodes (the intra-layer subnets) contains at least one leaf and no more failures will be incurred in future assignments of the vias to G_i.

We can now present the *algorithm OVA'* as the following.

1. Perform a depth-first search to identify the connected components of the associated constraint graph G_H;
2. Run *algorithm OVA* to identify those connected components which are distinguished;
3. Construct the *component graph* G^c;
4. Compute the maximum cardinality matching M of G^c;
5. Remove the selected super-nodes from the hyper-graph H according to M;
6. Perform another phase of *algorithm OVA* to assign the vias.

Proof of correctness is not difficult and is omitted.

Theorem 3 *Algorithm OVA minimizes the number of unrouted nets for the three-dimensional channel routing problem.*

Note that every step except step 4 of *algorithm OVA'* takes $O(n)$ time and step 4 can be performed in $O(m^{1/2}n_s)$ time [1], where n_s and m denote the number of edges and the number of vertices in G^c respectively.

Theorem 4 *Algorithm OVA' runs in $O(n + m^{1/2}n_s)$ time, where n_s and m denote the number of edges and the number of vertices in G^c respectively.*

5 Conclusion

In this paper, we've studied the problem of three-dimensional channel routing. Note that the projections of the problem on the top and the bottom layers are each two-dimensional channel routings. The intra-layer subnets can thus be connected simultaneously using traditional (two-dimensional) channel routing algorithms. To electrically connect the intra-layer subnets belonging to the same inter-layer net, we can assign proper vias to one of the terminals per intra-layer subnet to connect it vertically to the middle layer. This defines another (two-dimensional) channel routing problem in the middle layer in which only two-terminal nets are given. Unfortunately, in general, the vias can not be completely assigned to all the intra-layer subnets. We presented a linear time algorithm to minimize the number of failures in doing via assigment. We also presented an almost linear time algorithm to minimize the number of incomplete nets under this routing method.

References

[1] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Inc., 1982. p. 247.

[2] C.C. Tong and C.L. Wu. Optimizing positional conflicts in three-dimensional channel routing. manuscript, 1990.