An Optimal Jumper Insertion Algorithm for Antenna Avoidance/Fixing on General Routing Trees with Obstacles [*]

Bor-Yiing Su
Dept. of Electrical Engineering
National Taiwan University
Taipei 106, Taiwan
b90901130@ntu.edu.tw

Yao-Wen Chang
Dept. of Electrical Engineering &
Grad. Inst. of Electronics Engr.
National Taiwan University
Taipei 106, Taiwan
ywchang@cc.ee.ntu.edu.tw

Jiang Hu
Dept. of Electrical Engineering
Texas A&M University
College Station, TX 77843, USA
jianghu@ee.tamu.edu

ABSTRACT

We study in this paper the problem of jumper insertion on general routing (Steiner/spanning) trees with obstacles for antenna avoidance/fixing at the routing and/or post-layout stages. We formulate the jumper insertion for antenna avoidance/fixing as a tree-cutting problem and present the *first* optimal algorithm for the *general* tree-cutting problem. We show that the tree-cutting problem exhibits the properties of optimal substructures and greedy choices. With these properties, we present an $O((V+D)\lg D)$-time optimal jumper insertion algorithm that uses the least number of jumpers to avoid/fix the antenna violations on a Steiner/spanning tree with V vertices and D obstacles. Experimental results show the superior effectiveness and efficiency of our algorithm.

Categories and Subject Descriptors: B.7.2 [Integrated Circuits]: Design Aids

General Terms: Algorithms, Performance, Reliability

Keywords: Antenna Effect, Jumper Insertion, Routing

1. INTRODUCTION

As the process technology enters the nanometer era, product reliability and manufacturing yield have become major concerns in VLSI circuit design and manufacturing. The fine feature size of modern IC technologies is typically achieved by using plasma-based processes. In nanometer technology, more stringent process requirements cause some advanced high-density plasma reactors adopted in the production lines to achieve fine-line patterns [6]. However, these plasma-based processes will charge conducting components of a fabricated structure. As a result, the accumulated charges may affect the quality of IC's. This is called the *antenna effect.*

[*]Bor-Yiing Su and Yao-Wen Chang's work was partially supported by National Science Council of Taiwan under Grant No's. NSC 93-2815-C-002-046-E, NSC 94-2215-E-002-005, and NSC 94-2752-E-002-008-PAE; Jiang Hu's work was partially supported by SRC under contract 2003-TJ-1124.

During metallization, long floating interconnects act as temporary capacitors and accumulate charges gained from the energy provided by fabrication steps such as plasma etching. A random discharge of the floating node due to subsequent process steps could permanently damage transistors in the IC [8, 11]. For instance, the exposed polysilicon and metal structures connected to a thin-oxide transistor will collect charge from the processing environment (e.g., reactive ion etching) and damage the transistor when the discharging current flows through the thin oxide. The mechanism of antenna damage is not fully understood, but there is experimental evidence indicating when charging occurs and how it may affect the quality of gate oxide [8, 11]. Charging occurs when conductor layers not covered by a shielding layer of oxide are directly exposed to plasma. The amount of such charging is proportional to this plasma-exposed area. If conductor layers are connected to a diffusion layer pattern, such charges are discharged to the substrate through the diffusion; see Figure 1 for an illustration. On the other hand, if the charged conductor layers are connected only to the gate oxide, Fowler-Nordheim (F-N) tunneling current through thin oxide discharges such charges and causes damage to the thin oxide [8]; see Figures 1(b) and (c). As shown in Figure 1, interconnects are manufactured layer by layer. Before a conducting path to the diffusion is formed in metal 2 layer pattern etching (see Figure 1(d)), the interconnects in the poly and metal 1 layers might have accumulated so many charges that they cause damage on the gate in the left of Figure 1(c). (Note that there will not be any antenna violation after a conducting path to the diffusion is formed.)

There are three kinds of solutions to reduce the antenna effect [1]:

1. Jumper insertion: Break the signal wires with antenna violations and route them to the highest layers by jumper insertion. This reduces the charge amount for violated wires during manufacturing.

2. Embedded protection diode: Add protection diodes on every input port of a standard cell.

3. Diode insertion after placement and routing: Fix those wires with antenna violations that have enough rooms for "under-the-wire" diode insertion.

Comparing the three methods, for Method 2 of embedded protection diode, since these diodes are embedded and fixed, they consume unnecessary areas when there is no violation at the connecting wire. For the third method, we need extra space in the chip to place the diodes. Because

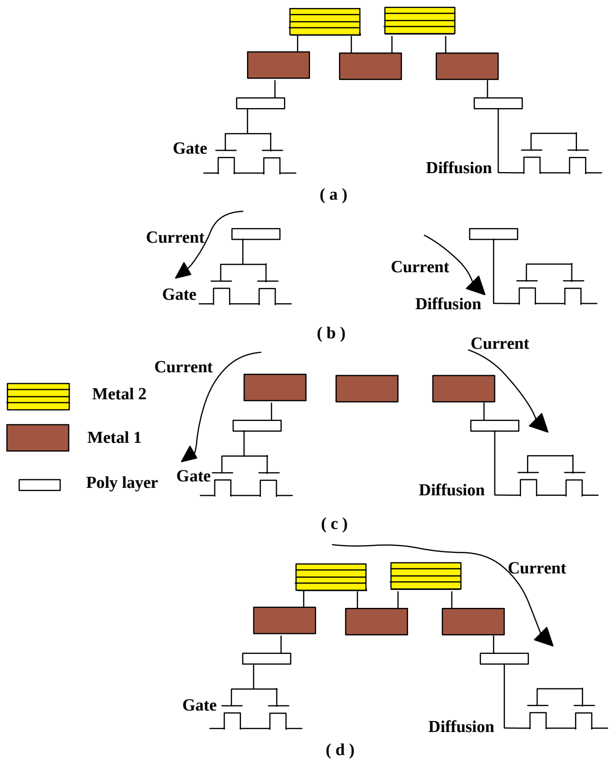


Figure 1: Antenna effect. (a) An example routing. (b) Late stage of poly layer pattern etching of Figure (a). Charge on the left poly pattern is discharged through the gate while charge on the right poly pattern is discharged through the diffusion. (c) Late stage of metal 1 layer pattern etching of Figure (a). Charge on the left metal 1 pattern is discharged through the gate while charge on the right metal 1 pattern is discharged through the diffusion. (d) Late stage of metal 2 layer pattern etching. Charges on all the metal 2 patterns are discharged through the diffusion.

the number of diodes needed for fixing antenna violations grows dramatically as the feature shrinks, it is hard to preserve enough space for diodes in nanometer IC designs. As a result, jumper insertion becomes one of the most popular approaches for avoiding/fixing antenna violations. The function of jumper insertion can be explained using Figure 2. In Figure 2(a), when the metal 1 layer is manufactured, the gate on the right might be damaged because the large area of the metal 1 interconnection can accumulate sufficient charges to damage the gate. However, if we insert a jumper to route the interconnect on the metal 2 layer as shown in Figure 2(b), the effective conductor layer becomes smaller. Therefore, the stored charge is not enough to damage the gate on the right, and thus we can avoid the antenna violation.

Although jumper insertion is currently a very popular approach for antenna avoidance/fixing, jumpers induce vias that will consume silicon areas and reduce circuit performance. Therefore, it is desired to fix antenna violations by using the minimum number of jumpers. The problem of jumper insertion on a routing tree for antenna avoidance has attracted much attention in the literature recently. Ho, Chang, and Chen in [4] propose an $O(V \lg V)$-time bottom-up approach to insert jumpers in a *spanning* tree of V vertices for antenna avoidance. The work assumes that each tree node corresponds to a gate terminal and inserts jumpers *only beside the tree nodes*; its optimality holds only for this special condition of inserting jumpers right beside the nodes of a spanning tree. There are two recent works that consider more general cases for jumper insertion on a general routing tree (could be a spanning or Steiner tree). The recent work [9] relaxes the constraint of inserting jumpers only beside the tree nodes, for which jumpers can be inserted at an arbitrary position of a tree edge. The work achieves the same time complexity as [4] for the relaxed problem. As an example shown in Figure 3, the wire segment is of $1.3L_{max}$ long, where L_{max} denotes the upper bound for antenna (i.e., any wire longer than L_{max} will violate the antenna rule). For this wire segment, the work in [4] needs two jumpers to fix the antenna violation (see Figure 3(a)) while a single jumper suffices for the work [9] to fix the violation (see Figure 3(b)).

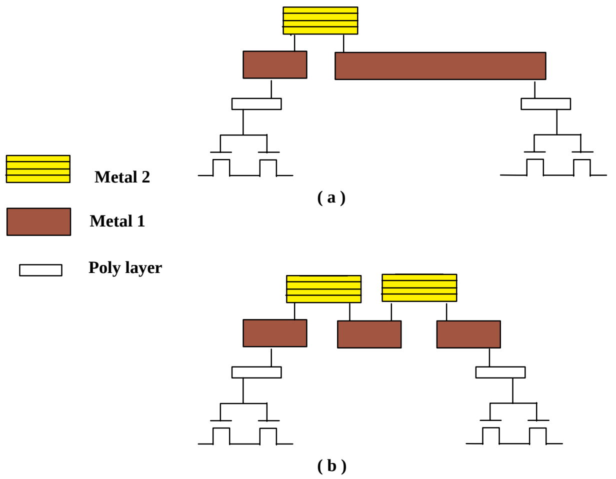


Figure 2: Jumper insertion. (a) Stage before inserting a jumper. (b) Stage after inserting a jumper from the metal 1 layer to the metal 2 layer.

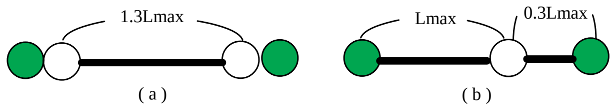


Figure 3: Jumper insertion for a wire of $1.3L_{max}$ long. (a) Two jumpers are needed for fixing the antenna violation if jumpers can be inserted only beside tree nodes, as the assumption made in [4]. (b) One jumper suffices to fix the antenna violation if jumpers can be inserted at an arbitrary position of the wire segment, as the assumption made in [9].

Another recent work [12] by Wu, Hu, and Mahapatra extends the work [4] to handle either a spanning or a Steiner tree. With the implementation scheme proposed by Kundu and Misra [7], the work [12] can achieve linear-time complexity for jumper insertion in a Steiner/spanning tree for antenna avoidance/fixing. To fix the antenna violation of a sink node (a gate terminal in this paper), the work first removes all subtrees around the node that violate the antenna rules. After all such subtrees are removed, if the sink still violates the antenna rule, the work will continually remove the heaviest branch from the sink until the antenna rules are satisfied. This approach is optimal *only for sink nodes* *alone*. For the case with two adjacent sink nodes, their method might not be optimal. As the routing-tree example shown in Figure 4(a), u_1 and u_2 are two sink nodes, the number beside each edge denotes the antenna charge weight (measured by the wire length, the wire area, and/or the wire perimeter), and the maximum antenna weight that a sink node can bear is assumed to be 10. For the work in [12], since we cannot partition the tree into any subtree with the total weight equal to 10, we will cut the heaviest edge near the sink node until the antenna rule is satisfied on u_1 and u_2. Thus, the edge $e(u_1, u_2) = 10$ will be removed first, and the work will insert four jumpers c_1, c_2, c_3, and c_4 as shown in Figure 4(b). Nevertheless, for this case, three jumpers suffice to solve the antenna violations; see the jumpers c_1, c_2, and c_3 shown in Figure 4(c).

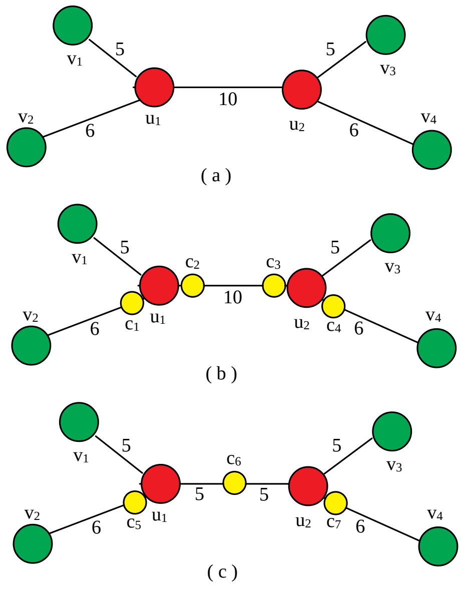


Figure 4: (a) A routing tree with two sink nodes u_1 and u_2. (b) The work in [12] needs jumpers $c_1, \ldots, c_4$ to solve the antenna violations. (c) Our work needs only three jumpers c_1, c_2, c_3 to satisfy the antenna rule.

For a jumper insertion algorithm to be practical, we shall work on general routing (Steiner or spanning) trees in which a tree node represents a gate terminal and a Steiner node represents a routing junction. We shall also allow a jumper to be inserted at an arbitrary position of a tree edge. Since jumper insertion routes a signal wire to the top-most layer, we must further consider the routing with obstacles in the *active layers*—the layers from the current routing layer up to the top-most layer; typical obstacles include power/ground nets, pre-routed nets, clock nets, etc. A jumper insertion algorithm that does not work on Steiner trees, allow arbitrary jumper insertion position, or consider routing obstacles cannot be practical for real applications.

In this paper, we consider the *general case* of inserting jumpers at arbitrary positions of tree edges with obstacles for antenna avoidance/fixing. (See Table 1 for the features considered by the recent jumper insertion algorithms.) We formulate the general jumper insertion for antenna avoidance (applicable at the routing stage) and/or fixing (applicable at the post-layout stage) with obstacles as a tree-cutting problem on a Stenier/spanning tree and present *the first* optimal algorithm for the *general* tree-cutting problem. We show that the tree-cutting problem exhibits the properties of optimal substructures and greedy choices. With these properties, a greedy algorithm suffices to find an optimal solution [3]. Based on the theory, we present an $O((V+D) \lg D)$-time optimal jumper insertion algorithm that uses the minimum number of jumpers to fix the antenna violations in a Steiner/spanning tree with V vertices and D obstacles. Experimental results show that our algorithm is very efficient and effective.

	[4] ISPD'04	[12] ISPD'05	[9] DAC'05	Ours
Optimal for the general routing tree?	No	No	No	Yes
Consider Steiner trees?	No	Yes	No	Yes
Allow arbitrary inserting positions?	No	Yes	Yes	Yes
Consider obstacles?	No	No	No	Yes

Table 1: Features of the related jumper insertion works.

The remainder of this paper is organized as follows. Section 2 formulates the problem of jumper insertion on a routing (Steiner or spanning) tree with obstacles for antenna avoidance/fixing. Section 3 presents an optimal algorithm for the proposed problem. Section 4 proves the optimality of the algorithm. Section 5 reports the experimental results. Finally, the conclusions are given in Section 6.

2. PROBLEM DEFINITION

To avoid/fix the antenna violation, we require that the total effective conductor connecting to a gate be less than or equal to a threshold, L_{max}. The threshold could be the wire length limit, the wire area limit, the wire perimeter limit, the ratio of antenna strength (length, area, perimeter, etc) to the gate size, or any model of the strength of antenna effect caused by conductors. For example, for wire area, we can simply compute the product of the wire length and the wire width (size); for the antenna-strength-to-gate-size ratio, we can simply model the antenna strength divided by the gate size as the edge weight. It will be clear later that the modeling of the antenna-strength-to-gate-size ratio is still feasible for uniform gate sizes since our approach processes gate by gate for the antenna avoidance/fixing. Typically, a net is modelled as a routing tree, where a node in the tree denotes a circuit terminal/junction (a gate, a diffusion, or a junction of interconnects) and an edge denotes the interconnection between two circuit terminals or junctions. Since the interconnection connecting to a diffusion terminal will not cause any antenna violation, as explained in Section 1, we shall focus on those connecting to gate terminals.

Let $T = (V = V_G \cup V_N, E)$ be a Steiner tree. The set V_G of nodes represents all gate terminals, the set V_N of nodes represents all Steiner points, the set E of edges denotes the wires connecting the circuit terminals or junctions, and an edge weight gives the measure of the wires with the same unit as L_{max}. Note that a Steiner point denotes a wire junction, which cannot help discharge the wire. For example, if L_{max} is a wire length limit, an edge weight denotes the wire length between two circuit terminals/junctions. If L_{max} is a wire area limit, the edge weight denotes the wire area. A gate will violate the antenna rule if the effective conductor incident on the gate (i.e., the *effective weight*—the sum of the weights of the edges incident on the corresponding node) is larger than L_{max}. To reduce the antenna effects on a gate, we can apply the technique illustrated in Figure 2 by adding a jumper on a wire connecting to the gate to reduce the effective conductor. This operation is modelled as adding a *cutting node* on the tree edge corresponding to the wire to reduce the effective edge weight associated with the gate node. As aforementioned, jumpers are implemented by vias which will consume silicon areas and reduce circuit performance. Therefore, it is desired to fix antenna violations by using the minimum number of jumpers. In other words, given a routing tree $(V = V_G \cup V_N, E)$ and an upper bound on the antenna L_{max}, we intend to add the minimum number of cutting nodes so that the effective edge weight associated with each node is smaller than L_{max}. Let $p(u)$ denote node u's parent. Let $L(u)$ denote the sum of the effective edge weights (wire lengths, wire areas, wire perimeter, strength ratio, etc) on node u. Let D be the set of obstacles in the active layers. (For simplicity, we focus our discussions on the rectangular obstacles. The jumper insertion algorithm to be presented in this paper readily applies to the problem with obstacles of arbitrary shapes, with additional procedures for obstacle identification.) The projection of the obstacles in D defines the *forbidden regions* for the edges and nodes of a routing tree for jumper insertion. Let F be the set of the forbidden regions. Given a node u (an edge or a tree segment e) of a routing tree, $f(u) = 1$ ($f(e) = 1$) if the node u (the edge/segment e) falls inside the forbidden regions ($u \in F$); $f(u) = 0$ ($f(e) = 0$), otherwise. With the definitions above, we can formulate the addressed problem as follows:

- Problem *JIROA* (*Jumper Insertion on a Routing tree with Obstacles for Antenna avoidance/fixing*): Given a routing tree $T = (V = V_G \cup V_N, E)$, an upper bound L_{max}, and a set D of rectangular obstacles, find the minimum set C of cutting nodes, $c \neq u$ for any $c \in C$ and $u \in V$, $f(c) = 0$ for any $c \in C$, so that $L(u) \leq L_{max}$, $\forall u \in V_G$.

Note that the routing tree in this formulation represents a net in any layout design stage, e.g., a net to be globally routed, a net after detailed routing (in the post-layout stage). Therefore, the JIROA problem is applicable to the antenna estimation in the global/detailed routing stage and the antenna violation fixing in the post-layout stage.

3. ALGORITHM FOR FINDING THE MINIMUM $|C|$

For the JIROA problem, we present in this section an $O((V{+}D) \lg D)$-time optimal algorithm, named *BUJIO* (*Bottom Up Jumper Insertion with Obstacles*), for finding the minimum cutting set C for a given routing (Steiner or spanning) tree $T = (V, E)$ with V nodes and D obstacles. (Note that we use V to denote the *set* or the *number* of nodes in a routing tree, which is common in the community of computer science; its meaning is clear from the context.) Algorithm BUJIO is summarized in Figure 5. Let $l(e)$ (or $l(u, v)$) be the weight (could be the wire length, wire area, wire perimeter, strength ratio, etc)) of the edge $e = (u, v)$ in T. In the BUJIO algorithm, we add the cutting nodes into the original tree in a bottom-up manner. We first define a *subleaf* node and an optimal replacement function $r(u, v)$ (see Figure 6 for an illustration) as follows:

DEFINITION 1. *A* subleaf *is a node for which all its children are leaf nodes, and if any of its children is a gate terminal, the edges between it and its children all have weights* $\leq L_{max}$.

DEFINITION 2. *Let* u, v *be two adjacent nodes with* $f(v) = 1$. *Then* $r(u, v)$ *denotes the cutting node* c *on edge* $e = (u, v)$ *with* $f(c) = 0$ *and* $l(u, c)$ *being the maximum among every node on edge* $e(u, v)$. *In other words,* c *is just beside the forbidden region covering node* v.

```
Algorithm: $BUJIO(T, L_{max}, D, C)$
Input: $T = (V = V_G \cup V_N, E)$ /* The given tree. */
       $L_{max}$ /* Upper Bound on antenna */
       $D$ /* Set of obstacles. */
       $C$ /* Cutting set */
1   Sort the obstacles in $D$ by the $x$-axes and then $y$-axes.
2   for each node $u \in T$
3       $w(u) = \sum_{f(e)=1 \wedge e \text{ incident on } u} l(e)$;
4       Contract every edge $e$ incident on $u$ with $f(e) = 1$;
5   while $|V_G| > 0$
6       for each leaf node $u$ not having been processed
7           Mark $u$ as processed;
8           if $u \in V_G$ and $l(u, p(u)) + w(u) > L_{max}$
                Let $c$ be the node on the edge $e(u, p(u))$
                with $l(u, c) = L_{max} - w(u)$;
9               if $f(c) = 1$
10                  $c_1 \leftarrow r(u, c)$;
11                  $C \leftarrow C \cup \{c_1\}$;
12                  $T(V, E) \leftarrow T((V_G \setminus \{u\}) \cup (V_N \cup \{c_1\}),$
                            $E \setminus \{e(u, c_1)\})$;
13              else
14                  $C \leftarrow C \cup \{c\}$;
15                  $T(V, E) \leftarrow T((V_G \setminus \{u\}) \cup (V_N \cup \{c\}),$
                            $E \setminus \{e(u, c)\})$;
16      for each subleaf node $u_p \in T$
            Let $u_1, u_2, \ldots, u_k$ denote all children nodes of $u_p$;
17          $totallen \leftarrow \sum_{i=1}^{k} (l(u_p, u_i) + w(u_i))$;
18          if $u_p$ and all of its children are in $V_N$
19              $w(u_p) \leftarrow w(u_p) + totallen$;
20              $T(V, E) \leftarrow T(V \setminus \cup_{i=1}^{k}\{u_i\},$
                        $E \setminus \cup_{i=1}^{k}\{e(u_i, u_p)\})$;
21          else if $totallen + w(u_p) \leq L_{max}$
22              $EqualLess(T, C, u_p, totallen)$ ;
23          else
24              $More(T, C, u_p, totallen)$;
```

Figure 5: Algorithm BUJIO deals with the leaf nodes first, and then call Subroutines EqualLess and More to deal with the subleaf nodes.

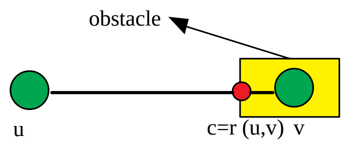


Figure 6: Illustration of the $r(u, v)$ function, where c is just beside the forbidden region.

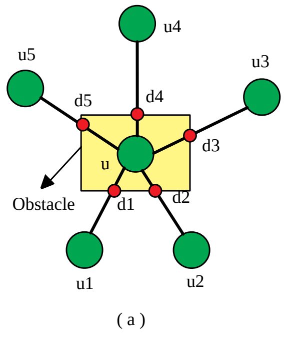
We derive the BUJIO algorithm based on the following four steps:

- Step 1 (line 1 of Algorithm BUJIO): Sort the obstacles in D by the x-axes and then the y-axes.

  With this process, we can determine $f(u)$ and $f(e)$ in $O(\lg D)$ time.

- Step 2 (lines 2–4 of Algorithm BUJIO): Compute the weight of every node.

  If a tree node $u \in V$ is in a forbidden region, some segments of the edges incident on u could also be in the forbidden region. Therefore, we cannot insert jumpers on these segments, and charges induced from these segments cannot be removed. Let the weight $w(u) = \sum_{f(e)=1 \wedge e \text{ incident on } u} l(e)$. Obviously, if $w(u) > L_{max}$ for some node u, the accumulated charges on u are over the upper bound that u can tolerate. For this case, we cannot prevent node u from antenna violation by jumper insertion alone. Otherwise, we can always find an optimal solution for jumper insertion. For the feasible case of a set of edges incident on a node u, we can reduce the problem with obstacles into the case without any obstacle by contracting the tree segments $e_1, e_2, \ldots, e_k$ inside the obstacles to node u and assigning $w(u) = \sum_{i=1}^{k} l(e_i)$. See Figure 7 for an illustration. After this processing, we can insert jumpers just as the case without any obstacles.



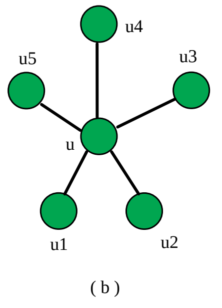


Figure 7: An example reduction for Step 2. (a) A routing tree with a node u in an obstacle (denoted by the shaded region), where $d_1, d_2, \ldots, d_5$ are nodes on the tree edges, just beside the obstacle. (b) Reducing the tree of (a) by removing edges $e(u, d_1), e(u, d_2), \ldots, e(u, d_5)$ and assigning $w(u) = \sum_{i=1}^{5} l(e(u, d_i))$.

- Step 3 (lines 6–15 of BUJIO): Handle every leaf node.

  In this step, our main goal is to prevent every leaf node from antenna violation. Obviously, if we have dealt with a leaf node, we need not consider it any more. Therefore, line 7 of the BUJIO algorithm marks these nodes to make sure that every leaf node is processed only once. If $l(u, p(u)) + w(u) \le L_{max}$, the leaf node u satisfies the antenna rule. Thus, we need not insert any cutting nodes. If $u \in V_N$, since u is not a gate terminal, we need not insert any cutting node, either. However, if $l(u, p(u)) + w(u) > L_{max}$ and $u \in V_G$, we must insert at least one cutting node to satisfy that $L(u) \le L_{max}$. We claim that $l(u, c) = L_{max} - w(u)$ (and thus $l(c, p(u)) = l(u, p(u)) - L_{max} + w(u)$) gives the best position for inserting the cutting node; see Figure 8(a) for an illustration. However, if $f(c) = 1$, we must find a position not in F. We claim that $c_1 = r(u, c)$ is the optimal substitute; see Figure 8(b) for an illustration. After adding jumper c or c_1 into C, we cut edge $e(u, c)$ or $e(u, c_1)$ from the tree T (lines 6–15 in BUJIO).

- Step 4 (lines 16–24 of BUJIO): Process every subleaf node.

  In this step, our main goal is to prevent every subleaf node from antenna violation. Moreover, we delete some nodes and edges to make each subleaf node a leaf node. (Note that as the edges are chopped off in tree cutting, the leaf nodes of the remaining tree might be Steiner or cutting nodes, which may not always correspond to gate terminals.) First of all, if u_p and all its children are in V_N, any of them needs not satisfy the antenna rule. Therefore, we just combine u_p and its children into a new leaf node and modify

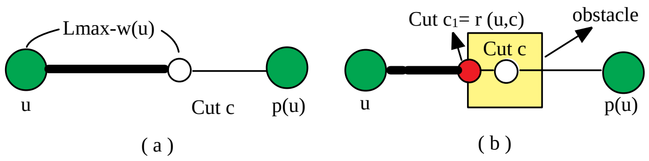


Figure 8: Explanation of lines 6–15 in the BUJIO algorithm. (a) The cutting node c is the optimal one among the nodes on edge $e(u, p(u))$. (b) c_1 is the optimal substitute of c, where $c_1 = r(u, c)$ is just beside the forbidden region.

  its weight as $w(u_p) + totallen$ (see lines 18–20 in Algorithm BUJIO). Then we classify the subleaf nodes into two cases by the sum of the weights between the node and its children and the weights of its children. Let u_p be a subleaf node and $u_i, \forall 1 \le i \le k$, be its children. Let $totallen = \sum_{i=1}^{k} (l(u_i, u_p) + w(u_i))$.

  Case 1: $totallen + w(u_p) \le L_{max}$

  We use the EqualLess subroutine to deal with this case. If u_p and its children form an isolated component, they must satisfy the antenna rule, and thus we are done with the subroutine. If $totallen + w(u_p) + l(u_p, p(u_p)) \le L_{max}$, u_p will not violate the antenna rule. If $u_p \in V_N$, it must be a Steiner node and all the edges between u_p and its children contributes to its weight. Thus we simply combine u_p and its children into a new leaf node and modify its weight as $w(u_p) + totallen$ (see lines 4–5 in Subroutine EqualLess). Moreover, if u_p or any of its children is in V_G, it means that the new leaf node u_p satisfies the antenna rule and thus we add u_p into V_G. Otherwise, we let u_p be its original type. If $totallen + w(u_p) + l(u_p, p(u_p)) > L_{max}$, we must add at least one cutting node c to prevent u_p from antenna violation. We claim that $l(c, u_p) + w(u_p) + totallen = L_{max}$ gives the best position for inserting the cutting node; see Figure 9(a). If $f(c) = 1$, however, we must find a position not in F. We claim that $c_1 = r(u_p, c)$ is the optimal substitute; see Figure 9(b) for an illustration. Therefore, we add c or c_1 into C, and cut u_p and all its children from the original tree T (lines 10–17 in EqualLess).

  Case 2: $totallen + w(u_p) > L_{max}$

  For this case, we apply the More subroutine. We first introduce the set $S = \cup_{i=1}^{k} \{l(e(u_i, u_p)) + w(u_i)\}$ from the subleaf node u_p and its k children. Then, we apply the linear-time algorithm SPLIT presented in [7] to split the set S into two disjoint subsets, S_h and S_l, where S_h is the higher subset, and S_l is the lower subset. (To make this paper self-contained, we also give the SPLIT algorithm in Figure 12.) The two subsets have three important properties: (1) for any $a \in S_l$ and any $b \in S_h$, we have $a \le b$; (2) $\sum_{s \in S_l} s \le L_{max}$; (3) for any $b \in S_h$, we have $\sum_{s \in S_l} s + b > L_{max}$. And the SPLIT algorithm will return the S_h subset. We claim that c_i on edge $e(u_i, u_p)$ with $l(c_i, u_p) = 0$ (and thus $l(c_i, u_i) = l(u_p, u_i)$) and $l(e(u_i, u_p)) + w(u_i) \in S_h$, $\forall 1 \le i \le |S_h|$ gives the best positions for inserting the cutting nodes; see Figure 9(c). Therefore, we add $c_1, \ldots, c_{|S_h|}$ into C, and cut $u_i, \ldots, u_{|S_h|}$ from the original tree T (lines 1–5 in More). Moreover, we call subroutine EqualLess to further reduce u_p into a new leaf node (line 6 in More).

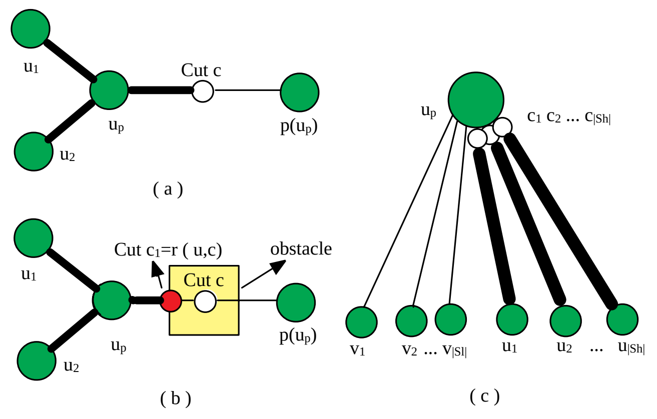


Figure 9: (a) Illustration of the *EqualLess* Subroutine. Here, $l(u_p,u_1)+w(u_1)+l(u_p,u_2)+w(u_2)+w(u_p)+l(u_p,c)=L_{max}$. (b) Illustration of the *EqualLess* subroutine with the cutting node c in a forbidden region. Here, c_1 is the optimal substitute of c, where $c_1=r(u_p,c)$ is just beside the forbidden region. (c) Illustration of the *More* subroutine.

When $|V_G|=0$, Algorithm BUJIO terminates and C gives a cutting set of the minimum size.

4. PROOF OF THE OPTIMALITY OF $|C|$

Algorithm BUJIO is greedy in nature. To prove that Algorithm BUJIO finds the optimal cutting set (of the minimum size), therefore, we can show that the JIROA problem exhibits *optimal substructure* and has the *greedy-choice property* [3]. A problem exhibits *optimal substructure* if an optimal solution to the problem contains within it optimal solutions to the subproblems; a problem has the *greedy-choice property* if a globally optimal solution can be arrived at by making a locally optimal (greedy) choice [3]. Due to the page limit, we only show the theoretical results and omit the proofs.

Theorem 1. *The JIROA problem exhibits an optimal substructure.*

Now we show that the JIROA problem has the greedy-choice property, and Algorithm BUJIO finds the best solution in each step. First, we show that Algorithm BUJIO has the greedy choice property among all leaf nodes. Then, we show that BUJIO has greedy choice property among all subleaf nodes.

Lemma 1. *Lines 6–15 of Algorithm BUJIO finds the best cutting set of the minimum size so that every leaf node $u \in V_G$ satisfies the antenna rule (i.e., $L(u) \le L_{max}, \forall$ leaf nodes $u \in V_G$).*

We proceed to show that lines 16–21 in BUJIO finds the best cutting set of the minimum size for each subleaf node u_p. In this step, we classify the subleaf nodes into two cases based on the sum of the weights between u_p and its children u_i and u_i's weight records $w(u_i)$: $\sum_{i=1}^{k}(l(u_p,u_i)+w(u_i))+w(u_p) \le L_{max}$ and $\sum_{i=1}^{k}(l(u_p,u_i)+w(u_i))+w(u_p) > L_{max}$. Therefore, we show that each case is with the greedy-choice property, and we find the best cutting set for each case.

Lemma 2. *Subroutine EqualLess finds the best cutting set of the minimum size so that every subleaf node $u_p \in V_G$ satisfies the antenna rule (i.e., $L(u_p) \le L_{max}, \forall$ subleaf nodes $u_p \in V_G$ satisfying $\sum_{i=1}^{k}(l(u_p,u_i)+w(u_i))+w(u_p) \le L_{max}$, where $u_p = p(u_i)$).*

```
Subroutine: EqualLess($T$, $C$, $u_p$, $totallen$)
if $p(u_p)$ does not exist
    return;
if $totallen + w(u_p) + l(u_p, p(u_p)) \le L_{max}$
    if $u_p \in V_N$
        $w(u_p) \leftarrow w(u_p) + totallen$;
    if $u_p$ or any of its children is in $V_G$
        $V_N \leftarrow V_N \setminus \{u_p\}$;
        $V_G \leftarrow V_G \cup \{u_p\}$ and mark $u_p$ as unprocessed;
    $T(V,E) \leftarrow T(V \setminus \cup_{i=1}^{k}\{u_i\}, E \setminus \cup_{i=1}^{k}\{e(u_i,u_p)\})$;
else
    Let $c$ be the node on $e(u_p, p(u_p))$ with
    $l(c,u_p) + w(u_p) + totallen = L_{max}$;
    if $f(c) = 1$
        $c_1 \leftarrow r(u,c)$;
        $C \leftarrow C \cup \{c_1\}$;
        $T(V,E) \leftarrow T(V \setminus (\cup_{i=1}^{k}\{u_i\}) \cup \{c_1\} \setminus \{u_p\}$,
                  $E \setminus \cup_{i=1}^{k}\{e(u_i,u_p)\} \setminus \{e(u_p,c_1)\})$;
    else
        $C \leftarrow C \cup \{c\}$;
        $T(V,E) \leftarrow T(V \setminus (\cup_{i=1}^{k}\{u_i\}) \cup \{c\} \setminus \{u_p\}$,
                  $E \setminus \cup_{i=1}^{k}\{e(u_i,u_p)\} \setminus \{e(u_p,c)\})$;
```

Figure 10: Compute the case where $totallen \le L_{max}$.

```
Subroutine: More($T$, $C$, $u_p$, $totallen$)
$S = \cup_{i=1}^{k}\{l(e(u_i,u_p)) + w(u_i)\}$;
$S_h = SPLIT(S, L_{max})$;
Let $c_i$ be the nodes between $u_p$ and $u_i$
with $l(c_i,u_p) = 0$, $l(c_i,u_i) = l(u_p,u_i)$ and
$\{l(e(u_i,u_p)) + w(u_i)\} \in S_h$, $\forall 1 \le i \le |S_h|$;
$C \leftarrow C \cup \{c_i\}\ \forall\ 1 \le i \le |S_h|$ ;
$T(V,E) \leftarrow T(V \setminus \cup_{i=1}^{|S_h|}\{u_i\}, E \setminus \cup_{i=1}^{|S_h|}\{e(u_i,u_p)\})$;
$minuslen \leftarrow \sum_{s \in S_h} s$;
$EqualLess(T, C, u_p, totallen - minuslen)$;
```

Figure 11: Compute the case where $totallen > L_{max}$.

Lemma 3. *Subroutine More finds the best cutting set of the minimum size so that every subleaf node $u_p \in V_G$ satisfies the antenna rule (i.e., $L(u_p) \le L_{max}, \forall$ subleaf nodes $u_p \in V_G$ satisfying $\sum_{i=1}^{k}(l(u_p,u_i)+w(u_i))+w(u_p) > L_{max}$, where $u_p = p(u_i)$).*

Based on the above theorem and lemmas, we have the following theorem:

Theorem 2. *Algorithm BUJIO optimally solves the JIROA problem in $O((V+D)\lg D)$ time using $O(V)$ space.*

5. EXPERIMENTAL RESULTS

We implemented the BUJIO algorithm in the C++ language on a 2.4 GHz Intel Pentium PC with 256 MB memory under the Windows XP operating system. Since no previous work in the literature considers jumper insertion on a routing tree with obstacles, we extended the ISPD-05 work by Wu et al. [12], the DAC-05 work [9] by Su and Chang, and the ISPD-04 work [4] by Ho et al. to handle obstacles and made comparisons with our BUJIO algorithm. For the jumper insertion algorithms presented at ISPD-05, DAC-05, and ISPD-04, we just follow their procedures to insert

```
Subroutine: SPLIT($S$, $Bound$) [7]
if $|S| = 1$
    if $\sum_{s \in S} s \leq Bound$
        return $\emptyset$;
    else
        return $S$;
else
    Median-find-and-halve($S$) and let $S_h$ be
    the higher half of $S$;
    $W = \sum_{s \in S \ and \ s \notin S_h} s$;
    if $W = Bound$
        return $S_h$;
    else if $W < Bound$
        return $SPLIT(S_h, Bound - W)$;
    else ($W > Bound$)
        return $SPLIT(S \setminus S_h, W) + S_h$;
```

Figure 12: Return the required subset S_h from S. Median-find-and-halve(S) finds the median m of set S and partitions S into two subsets S_l and S_h, where each element in S_l is $\leq m$ and each element in S_h is $\geq m$. Moreover, $|S_h| \leq |S_l| \leq |S_h| + 1$.

jumpers. If the position for jumper insertion is in a forbidden region (an obstacle), we use the same optimal substitute presented in this paper to insert the jumper. We call the extended work as ISPD-05e, DAC-05e, and ISPD-04e, respectively. Moreover, since our BUJIO and the ISPD-05 algorithms are designed for Steiner trees while the DAC-05 and the ISPD-04 ones are for minimum spanning trees, we generated two sets of different trees based on the same gate terminals and tested the algorithms on the corresponding trees.

To conduct the experiment, we first generated gate terminals on grid planes of the dimension $10^4 \mu m \times 10^4 \mu m$ and randomly placed rectangular obstacles of various sizes on the planes. Then we constructed minimal Steiner trees and minimum spanning trees based on the gate terminals.

Two experiments on the effects of varying L_{max} and varying node quantity were conducted. To focus on the evaluation of the existing algorithms, without loss of generality, we assume that the antenna bound L_{max} is measured by wire length. Table 2 shows the number of jumpers required for fixing all antenna violations for a routing tree with 10000 nodes and 500 obstacles by changing L_{max} from 220 μm to 320 μm. Column 1 gives the L_{max} value, and Columns 2, 3, 5, and 7 list the numbers of jumpers required ($\#J$) for fixing the antenna violations for each L_{max} for the BUJIO, the ISPO-05e, the DAC-05e, and the ISPD-04e algorithms, respectively. Columns 4, 6, and 8 give the percentages of additional jumpers required ($More$) for the respective ISPD-05e, the DAC-05e, and the ISPD-04e algorithms over BUJIO to fix all antenna violations, i.e., $More$ = (#Jumpers of the algorithm − #Jumpers of BUJIO)/ #Jumpers of BUJIO.

It is not surprising that BUJIO requires fewer jumpers than the ISPD-05e algorithm. However, their difference is not very significant for this set of test cases. The reason is that the ISPD-05e algorithm behaves very similarly to BUJIO for this set of test cases. Only when the gate terminals are adjacent to each other, the ISPO-05e algorithm adds more jumpers than BUJIO, as the case shown in Section 1. The case is rare for random designs, and thus the numbers of jumpers required for the two algorithms are close. Even though the results of the two algorithms are close, nevertheless, the BUJIO algorithm can always find the optimal solution while the ISPD-05e cannot; this optimality significantly differentiates our BUJIO algorithm from the previous work. We shall show that BUJIO can significantly outperform the ISPD-05e one for some non-random designs.

It is obvious that BUJIO may need much fewer jumpers than the DAC-05e and the ISPD-04e algorithms. The reduction comes from two parts: (1) BUJIO works on Steiner trees while the DAC-05e and the ISPD-04e algorithms work on minimum spanning trees. A minimal Steiner tree intrinsically has smaller wirelength and thus needs fewer jumpers to fix the antenna violations than those of a minimum spanning tree. (2) More importantly, BUJIO is much more effective than the DAC-05e and the ISPD-04e algorithms. As shown in Table 2, the improvements range from 17% to 57%, much more than the 10% wirelenth difference between the Steiner tree (728568 unit long) and the spanning tree (801302 unit long).

L_{max} (um)	BUJIO	ISPD-05e		DAC-05e		ISPD-04e	
	$\#J$	$\#J$	More	$\#J$	More	$\#J$	More
220	1533	1537	0.3%	1806	17.8%	2359	53.9%
230	1341	1348	0.5%	1607	19.8%	2020	50.6%
240	1144	1149	0.4%	1378	20.5%	1742	52.3%
250	961	966	0.5%	1210	25.9%	1501	56.2%
260	823	827	0.5%	1050	27.6%	1279	55.4%
270	706	708	0.3%	919	30.2%	1109	57.1%
280	599	601	0.3%	779	30.1%	928	54.9%
290	517	518	0.2%	649	25.5%	767	48.4%
300	434	435	0.2%	562	29.5%	645	48.6%
310	366	366	0.0%	488	33.3%	543	48.4%
320	305	305	0.0%	413	35.4%	458	50.2%

Table 2: Comparisons of the numbers of jumpers required for BUJIO, ISPD-05e, DAC-05e, and ISPD-04e to fix all antenna violations based on a routing tree of 10000 nodes and 500 obstacles.

The previous experiment shows that the ISPD-05e algorithm can achieve comparable performance to our BUJIO. In order to test the robustness of the ISPD-05e algorithm (and the DAC-05e and ISPD-04e ones), we constructed a set of test cases based on that shown in Figure 4. We first generated a test case of 5000 nodes as usual. After a Steiner tree and a minimum spanning tree have been constructed, we selected some gate terminals at the same position in both trees and modified the selected nodes in the same way. Let node u in Figure 13(a) be a selected node. We added a subtree rooted at node u as shown in Figure 13(b). The subtree has similar topology as that of the routing tree shown in Figure 4. It is clear that BUJIO needs only 4 jumpers for the subtree as shown in Figure 13(c) while the ISPD-05e algorithm needs 5 jumpers as illustrated in Figure 13(d) to fix the antenna violations. We generated the test cases based on this expansion method.

For the experiments shown in Table 3, we constructed a corresponding test case for each L_{max} value based on the previously mentioned expansion method. The experimental results show that BUJIO can outperform the ISPD-05e algorithm by an average improvement of about 27%. The results reveal that the ISPD-05e algorithm is not effective for such test cases. Similar results can be observed from the experiments shown in Table 4, for which we constructed a corresponding test case for each given number of gate terminals (number of nodes) based on the aforementioned expansion method.

Moreover, the DAC-05e algorithm also outperforms the ISPD-05e one for the two sets of test cases. The reasons are two-fold: (1) The minimum spanning tree and Steiner tree are the same for the appended subtrees, and (2) the DAC-05e algorithm adds only 4 jumpers on each subtree while the

ISPD-05e adds 5 jumpers. As a result, when the number of the appended subtrees increases, the DAC-05e algorithm requires much fewer jumpers than the ISPD-05e one. So each of the ISPD-05e and the DAC-05e algorithms have its own strengths and weaknesses. Each of them might be effective for some cases but perform poorly for other cases. No matter what test case is considered, however, BUJIO always finds the *optimal* solution.

L_{max} (um)	BUJIO	ISPD-05e		DAC-05e		ISPD-04e	
	$\#J$	$\#J$	More	$\#J$	More	$\#J$	More
220	4274	5455	27.6%	4557	6.6%	5943	39.1%
230	4157	5311	27.8%	4421	6.4%	5762	38.6%
240	4056	5174	27.6%	4305	6.1%	5604	38.2%
250	3972	5061	27.4%	4187	5.4%	5455	37.3%
260	3889	4951	27.3%	4076	4.8%	5299	36.3%
270	3814	4844	27.0%	3983	4.4%	5157	35.2%
280	3728	4738	27.1%	3895	4.5%	5028	34.9%
290	3657	4637	26.8%	3809	4.2%	4915	34.4%
300	3605	4541	26.0%	3728	3.4%	4796	33.0%
310	3542	4467	26.1%	3653	3.1%	4695	32.6%
320	3489	4401	26.1%	3587	2.8%	4593	31.6%

Table 3: Comparisons of the numbers of jumpers required for BUJIO, ISPD-05e, DAC-05e, and ISPD-04e for fixing all antenna violations based on 11 test cases with routing trees of 10000 nodes and 200 obstacles.

$\#node$	BUJIO	ISPD-05e		DAC-05e		ISPD-04e	
	$\#J$	$\#J$	More	$\#J$	More	$\#J$	More
10000	4548	5844	28.5%	4881	7.3%	6391	40.5%
20000	7329	9188	25.4%	7634	4.2%	9800	33.7%
30000	10002	12414	24.1%	10246	2.4%	12899	29.0%
40000	12400	15375	24.0%	12647	2.0%	15753	27.0%
50000	15113	18754	24.1%	15283	1.1%	19013	25.8%
60000	17686	22019	24.5%	17877	1.1%	22256	25.8%
70000	20355	25369	24.6%	20487	0.7%	25525	25.4%
80000	23167	28888	24.7%	23217	0.2%	28950	25.0%
90000	25938	32373	24.8%	25985	0.2%	32426	25.0%
100000	28735	35884	24.9%	28741	0.0%	35889	24.9%

Table 4: Comparisons of the numbers of jumpers required for BUJIO, ISPD-05e, DAC-05e, and ISPD-04e for fixing all antenna violations based on 10 test cases with routing trees of 200 obstacles each and $L_{max} = 200\mu m$.

The empirical running time for the four methods are close to linear. In particular, BUJIO requires only 1.74 sec to find an optimal solution for a routing tree of 0.1 million nodes, with $L_{max} = 200\mu m$ and 500 obstacles on each plane (while the ISPD-05e, DAC-05e, and ISPD-04e algorithms need 1.74 sec, 1.53 sec, and 1.53 sec, respectively). Therefore, BUJIO can handle a test case of a very huge number of nodes in very short running time.

6. CONCLUSION

We have presented an $O((V+D)\lg D)$-time optimal jumper insertion algorithms for avoiding/fixing antenna violations on a Steiner/spanning tree of V nodes with D obstacles. It is the *first* optimal algorithm for the *general* tree-cutting problem. Empirical results have shown that our algorithms approach linear and obtain solutions of very high quality. Our work can be applied to any Steiner/spanning trees (could be a net to be globally routed or a net after detailed routing) and thus readily be incorporated into a global router for antenna effect avoidance or a post-layout optimizer for antenna violation fixing. Future work lies in the integration of jumper and diode insertion for antenna violation avoidance/fixing.

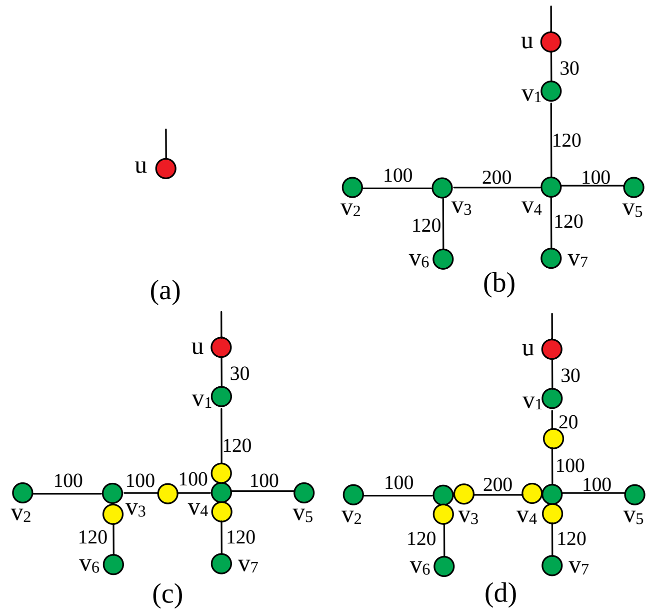


Figure 13: (a) A selected node u. (b) The subtree appended to node u. (c) BUJIO adds 4 jumpers on the subtree. Here, $L_{max} = 200\mu m$. (d) The ISPD-05e algorithm adds 5 jumpers on the subtree.

7. REFERENCES

[1] P. H. Chen, S. Malkani, C.-M. Peng, and J. Lin, "Fixing antenna problem by dynamic diode dropping and jumper insertion", *Proc. ISQED*, pp 275–282, 2000.

[2] Z. Chen and I. Koren, "Layer reassignment for antenna effect minimization in 3-layer channel routing", *Proc. Workshop on DFT*, pp 77–85, 1996.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd Ed., McGraw-Hill Book Co., 2001.

[4] T.-Y. Ho, Y.-W. Chang, and S.-J. Chen, "Multilevel routing with antenna avoidance," *Proc. ISPD*, pp. 34–40, April 2004.

[5] L.-D. Huang, X. Tang, H. Xiang, D. F. Wong, and I.-M. Liu, "A polynomial time optimal diode insertion/routing algorithm for fixing antenna problem," *Proc. DATE*, pp. 470–475, 2002.

[6] S. Krishnan, et. al., "Assessment of charge-induced damage to ultra-thin gate MOSFETs", *Proc. ITEM*, pp. 445–448, 1997.

[7] S. Kundu and J. Misra, "A lineartree partitioning algorithm," *SIAM J. of Computing*, vol. 6, no. 1, pp. 151–154, March 1977.

[8] H. Shin, C. -C. King, and C. Hu, "Thin oxide damage by plasma etching and ashing process", *Proc. IRPS*, pp. 37–41, March–April, 1992.

[9] B.-Y. Su and Y.-W. Chang, "An optimal jumper insertion algorithm for antenna effect avoidance/fixing," *Proc. DAC*, pp. 325–328, June 2005.

[10] K. P. Wang, M. Marek-Sadowska, and W. Maly, "Layout design for yield and reliability," *Proc. PDW*, pp. 190–197, April 1996.

[11] H. Watanabe, et.al., "A wafer level monitoring method for plasma-charging damage using antenna PMOSFET test structure," *IEEE Trans. Semiconductor Manufacturing*, vol. 10, no. 2, pp. 228–232, May 1997.

[12] D. Wu, J. Hu, and R. Mahapatra, "Coupling aware timing optimization and antenna avoidance in layer assignment," *Proc. ISPD*, pp. 20–27, April 2005.