An Exact Jumper-Insertion Algorithm for Antenna Violation Avoidance/Fixing Considering Routing Obstacles

Bor-Yiing Su, Yao-Wen Chang, *Member, IEEE*, and Jiang Hu, *Member, IEEE*

***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 postlayout 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.**

***Index Terms*—Antenna effect, design for manufacturability, physical design, reliability, routing.**

I. Introduction

As PROCESS technology enters the nanometer era, product reliability and manufacturing yield have become major concerns in the design and manufacturing of very large scale integrated circuits. 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 [4]. However, these plasma-based processes will charge conducting components of a fabricated structure. As a result, the accumulated charges may affect the quality of ICs. This is called the antenna effect.

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 [6], [8]. 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 [6], [8]. 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 Fig. 1 for an illustration). On the other hand, if the charged conductor layers are connected only to the gate oxide, Fowler–Nordheim tunneling current through thin oxide discharges such charges and causes damage to the thin oxide [9]; see Fig. 1(b) and (c). As shown in Fig. 1, interconnects are manufactured layer by layer. Before a conducting path to the diffusion is formed in metal-2-layer pattern etching [see Fig. 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 Fig. 1(c) (note that there will not be any antenna violation after a conducting path to the diffusion is formed).

The following are popular 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) Diode insertion: Fix those wires with antenna violations that have enough rooms for under-the-wire diode (or reverse diode) insertion. During wafer manufacturing, all the inserted diodes are floating (or ground). A diode or a reverse diode can be used to protect all input ports that are connected to the same output ports.
3) Embedded protection diode: Add protection diodes on every input port of a standard cell.
4) Antenna-aware routing: Since the antenna effect mainly occurs in the gate input (with high impedance) and seldom occurs in the diffusion output (with low impedance and drains out the plasma immediately) for current technology, an antenna-aware router can try to route the wire with a high antenna-strength ratio at the diffusion output and minimize the antenna-strength ratio at the gate input.

Manuscript received May 2, 2006; revised July 19, 2006. The work of B.-Y. Su and Y.-W. Chang was supported by the National Science Council of Taiwan, R.O.C., under Grant NSC 93-2815-C-002-046-E, Grant NSC 94-2215-E-002-005, and Grant NSC 94-2752-E-002-008-PAE. The work of J. Hu was supported by Semiconductor Research Corporation under Contract 2003-TJ-1124. This paper was recommended by Guest Editor P. H. Madden.

B.-Y. Su is with the Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: b90901130@ntu.edu.tw).

Y.-W. Chang is with the Graduate Institute of Electronics Engineering and also with the Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: ywchang@cc.ee.ntu.edu.tw).

J. Hu is with the Department of Electrical Engineering, Texas A&M University, College Station, TX 77843 USA (e-mail: jianghu@ece.tamu.edu).

Digital Object Identifier 10.1109/TCAD.2007.892338

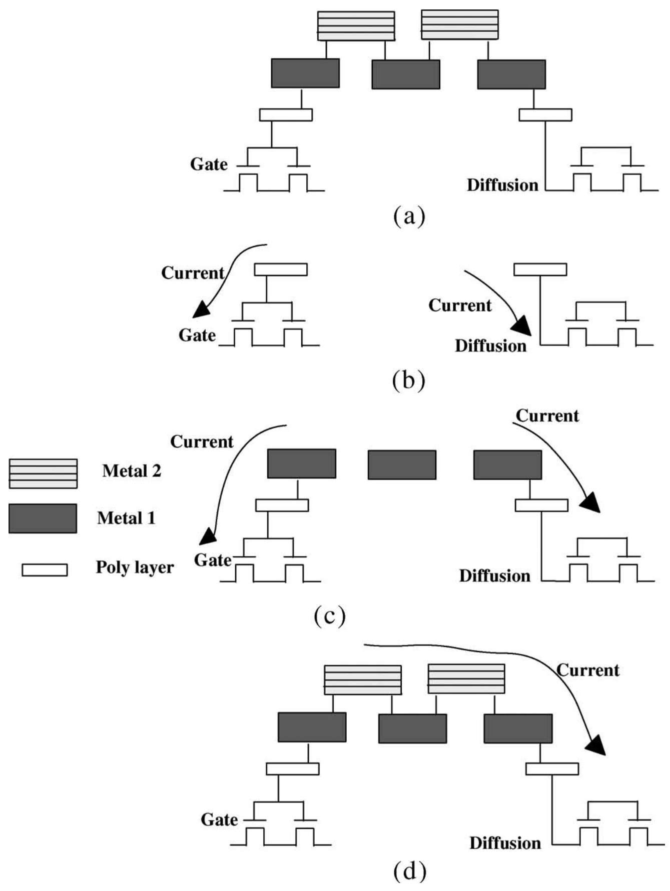

Fig. 1. Antenna effect. (a) Example routing. (b) Late stage of polylayer pattern etching of (a). Charge on the left polypattern is discharged through the gate while charge on the right polypattern is discharged through the diffusion. (c) Late stage of metal-1-layer pattern etching of (a). Charge on the 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.

Comparing the four methods, for method 3) 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 second and third methods, we need extra space in the chip to place the diodes. Because 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. In order to resolve this problem, the dynamic diode insertion during the placement with bounding-box-timing estimation is used to resolve this problem [10]. This functionality is embedded in many commercial placers. For the fourth method, an already complex router needs to additionally consider the antenna effect, and the technique is less effective for fixing the antenna violations. 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 Fig. 2. In Fig. 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 Fig. 2(b),

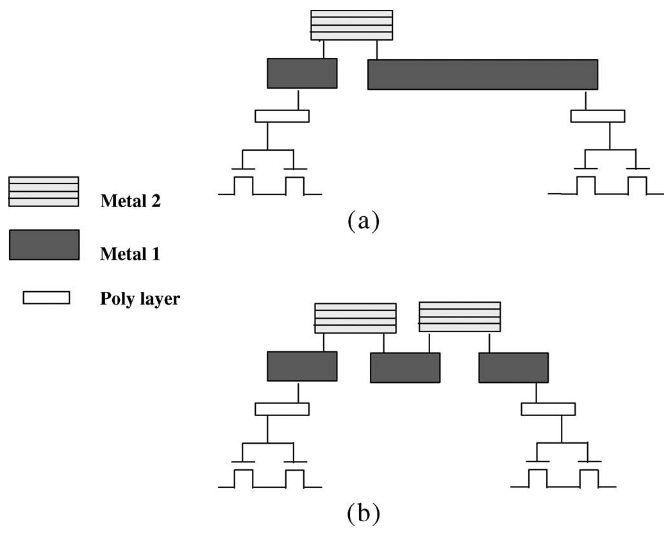

Fig. 2. Jumper insertion. (a) Stage before inserting a jumper. (b) Stage after inserting a jumper from metal 1 layer to metal 2 layer.

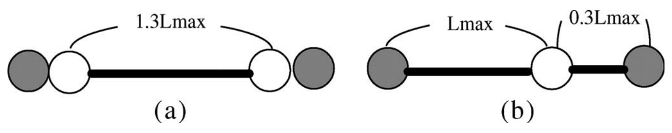

Fig. 3. Jumper insertion for a wire of 1.3 $L_{\max}$ long. (a) Two jumpers are needed for fixing the antenna violation if jumpers can be inserted only beside tree nodes, as the assumption made in [3]. (b) One jumper suffices to fix the antenna violation if jumpers can be inserted at an arbitrary position of the wire segment, as the assumption made in [7].

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 *et al.* in [3] 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 [7] 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 [3] for the relaxed problem. As an example shown in Fig. 3, the wire segment is of 1.3 $L_{\max}$ 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 [3] needs two jumpers to fix the antenna violation [see Fig. 3(a)] while a single jumper suffices for the work [7] to fix the violation [see Fig. 3(b)].

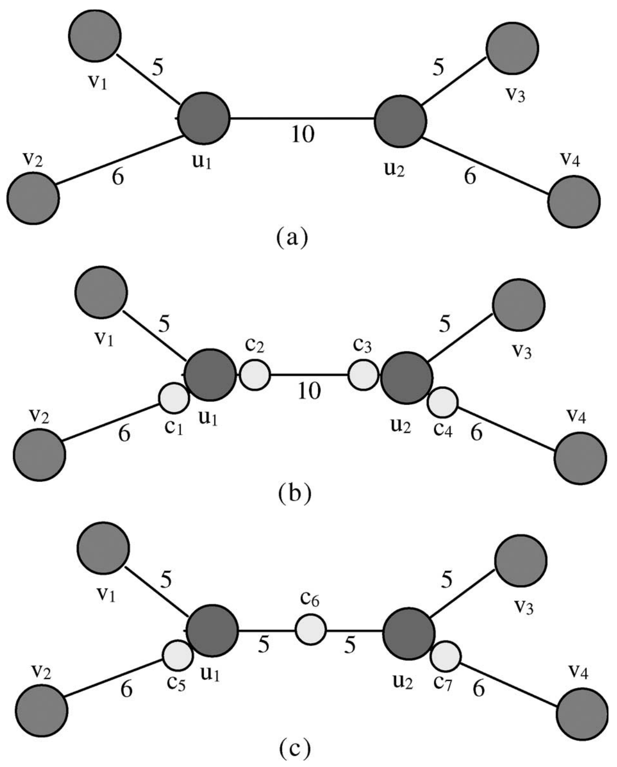

Fig. 4. (a) Routing tree with two sink nodes u_1 and u_2. (b) Work in [9] needs jumpers $c_1, \ldots, c_4$ to solve the antenna violations. (c) This paper needs only three jumpers c_5, c_6, and c_7 to satisfy the antenna rule.

Another recent work [9] by Wu *et al.* extends the work in [3] to handle either a spanning or a Steiner tree. With the implementation scheme proposed by Kundu and Misra [5], the work in [9] 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 Fig. 4(a), u_1 and u_2 are two sink nodes, the number beside each edge denotes the antenna charge weight (measured by the ratio of antenna strength (length, area, perimeter, etc.) to the gate size, wire length, wire area, and/or wire perimeter) and the maximum antenna weight that a sink node can bear is assumed to be ten. For the work in [9], since we cannot partition the tree into any subtree with the total weight equal to ten, 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 Fig. 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 Fig. 4(c).

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

TABLE I
FEATURES OF THE RELATED JUMPER INSERTION WORKS

	[4] ISPD'04	[13] ISPD'05	[10] 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

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, which could be prerouted nets, power/ground 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 I 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 postlayout 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 [2]. 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.

The remainder of this paper is organized as follows. Section II formulates the problem of jumper insertion on a Steiner/spanning tree with obstacles for antenna avoidance/fixing. Section III presents an optimal algorithm for the proposed problem. Section IV proves the optimality of the algorithm. Section V analyzes the complexity of the algorithm. Section VI extends the algorithm to the handling of the antenna-strength-to-gate-size model. Section VII reports the experimental results. Finally, the conclusions are given in Section VIII.

II. Problem Definition

Before formulating our problem, we first list the notation used in this paper in Table II for clarity.

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 ratio of antenna strength (length, area, perimeter, etc.) to the gate size, wire-area limit, wire-perimeter limit, wire-length limit, 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

TABLE II
NOTATION USED IN THE ALGORITHM

Notation	Meaning
$T = (V, E)$	The given routing tree.
V_G	Gate terminals.
V_N	Nodes that are not gate terminals.
E	The wires connecting the circuit terminals or junctions
L_{max}	The threshold of the routing tree.
$L(u)$	Sum of the effective edge weights on node u.
C	The set of cutting nodes.
c	Some cutting node.
$e(u_1, u_2)$	Edge between nodes u_1 and u_2.
$l(e)$	Weight of edge e.
$l(u_1, u_2)$	Weight of edge $e(u_1, u_2)$.
$p(u)$	Parent of node u.
D	The set of obstacles in the active layers.
F	The set of the forbidden regions.
$r(u, v)$	The node between nodes u and v and just beside the forbidden region covering node v.

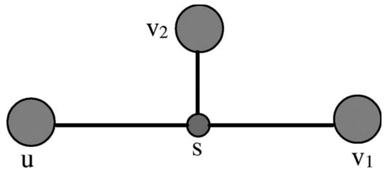

Fig. 5. u, v_1, and v_2 are gate terminals, and s is a Steiner point. The charges accumulated on edges $e(u, s)$, $e(s, v_1)$, and $e(s, v_2)$ will all cause antenna effect on u.

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 in Section VI that the modeling of the antenna-strength-to-gate-size ratio is still feasible, since our approach processes gate by gate for the antenna avoidance/fixing. Typically, a net is modeled as a routing tree, where a node in the tree denotes a circuit terminal/junction (a gate, diffusion, or 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 I, 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 the nodes represents all gate terminals, the set V_N of the nodes represents all Steiner points, the set E of the 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 (see Fig. 5 for an illustration). The charges accumulated on edges $e(u, s)$, $e(s, v_1)$, and $e(s, v_2)$ will all cause antenna effect on the gate terminal u (we shall focus on Steiner trees in the following discussions; the proposed method readily applies to spanning trees, for which $V_N = \emptyset$).

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 (and so is the ratio of antenna strength to the gate size). 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

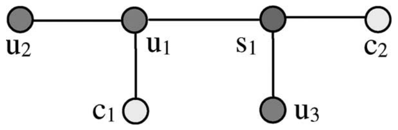

Fig. 6. u_1, u_2, and u_3 are gate terminals. s_1 is a Steiner point. c_1 and c_2 are cutting nodes. Therefore, the sum of effective edge weights of u_1, u_2, and u_3 are: $L(u_1) = l(u_1, u_2) + l(u_1, c_1) + l(u_1, s_1) + l(s_1, u_3) + l(s_1, c_2)$, $L(u_2) = l(u_1, u_2)$, $L(u_3) = l(u_1, s_1) + l(s_1, u_3) + l(s_1, c_2)$.

a gate, we can apply the technique illustrated in Fig. 2 by adding a jumper on a wire connecting to the gate to reduce the effective conductor. This operation is modeled 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 on the tree edges so that the effective edge weight associated with each node is smaller than $L_{\max}$. Let $p(u)$ denote the node u's parent. Let $L(u)$ denote the sum of the effective edge weights (strength ratio, wire lengths, wire areas, wire perimeter, etc.) on node u (see Fig. 6 as an illustration). 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 Jumper Insertion on a Routing tree with Obstacles for Antenna (JIROA) 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 on edges $c \neq u$ for any $c \in C$ and $u \in V$ and $f(c) = 0$ 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 postlayout stage). Therefore, the JIROA problem is applicable to the antenna estimation in the global/detailed routing stage and the antenna violation fixing in the postlayout stage.

III. 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 Bottom Up Jumper Insertion with Obstacles (BUJIO), 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

```
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              $More(T, C, u_p, totallen)$;
```

Fig. 7. Algorithm BUJIO deals with the leaf nodes first and, then, call Subroutines EqualLess and More to deal with the subleaf nodes.

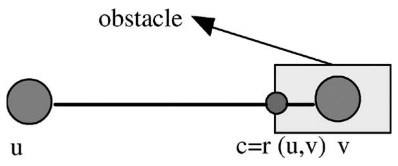

Fig. 8. Illustration of the $r(u, v)$ function, where c is just beside the forbidden region.

is common in the community of computer science; its meaning is clear from the context). Algorithm BUJIO is summarized in Fig. 7.

Let $l(e)$ (or $l(u, v)$) be the weight (could be the strength ratio, wire length, wire area, wire perimeter, 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 Fig. 8 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 and 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 (see Fig. 8).

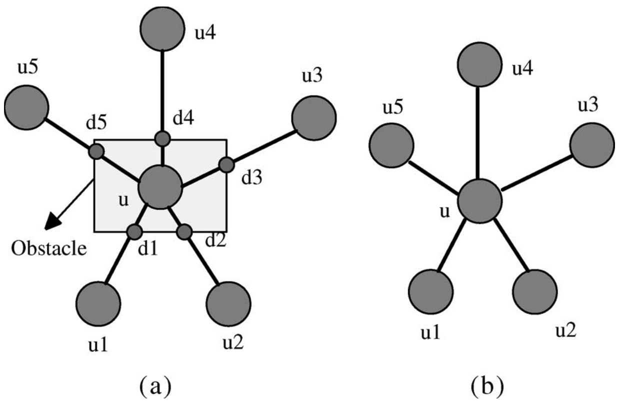

Fig. 9. Example reduction for Step 2). (a) Routing tree with a node u in an obstacle (denoted by the shaded region), where $d_1, d_2, \ldots, d_5$ are nodes on the tree edges, just beside the obstacle. (b) Reducing the tree of (a) by removing edges $e(u, d_1), e(u, d_2), \ldots, e(u, d_5)$ and assigning $w(u) = \sum_{i=1}^{5} l(e(u, d_i))$.

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: We 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. We use weight $w(u)$ to record such information. That is, $w(u) = \sum_{f(e)=1 \wedge e \text{ incident on } u} l(e)$. 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 Fig. 9 for an illustration). After this processing, we can insert jumpers just as the case without any obstacles.

Step 3) Lines 6–15 of Algorithm BUJIO: We deal with 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 $l(u, p(u)) + w(u) \leq L_{\max}$, the leaf node u satisfies the antenna

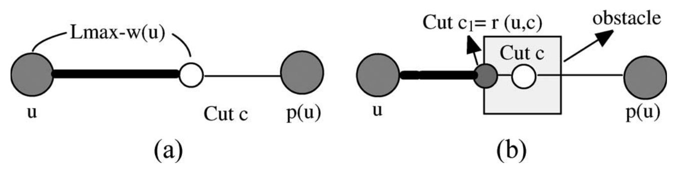

Fig. 10. Explanation of lines 6–15 in the BUJIO algorithm. (a) Cutting node c is the optimal one among the nodes on edge $e(u, p(u))$. (b) c_1 is the optimal substitute of c, where $c_1 = r(u, c)$ is just beside the forbidden region.

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 $L(u) \leq L_{\max}$. 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 Fig. 10(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 Fig. 10(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 Algorithm BUJIO: We deal with 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, none of them needs to satisfy the antenna rule. Therefore, we just combine u_p and its children into a new leaf node and modify 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 edge weights between the node and its children and the weights of its children. Let u_p be a subleaf node and $u_i, \forall 1 \leq i \leq k$, be its children. Let $totallen = \sum_{i=1}^{k} (l(u_i, u_p) + w(u_i))$ (see Fig. 11 as an illustration).

Case 4.1) $totallen + w(u_p) \leq L_{\max}$.

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 $totallen + w(u_p) + l(u_p, p(u_p)) \leq L_{\max}$, 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 contribute 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 Fig. 12(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 Fig. 12(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 Fig. 13).

Case 4.2) $totallen + w(u_p) > L_{\max}$.

For this case, we apply the More subroutine summarized in Fig. 14. 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

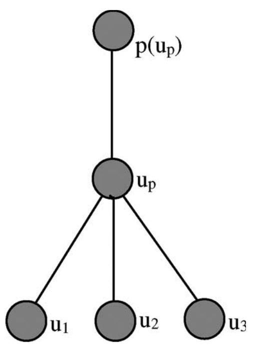

Fig. 11. u_p is a subleaf node. Therefore, $totallen = l(u_p, u_1) + w(u_1) + l(u_p, u_2) + w(u_2) + l(u_p, u_3) + w(u_3)$.

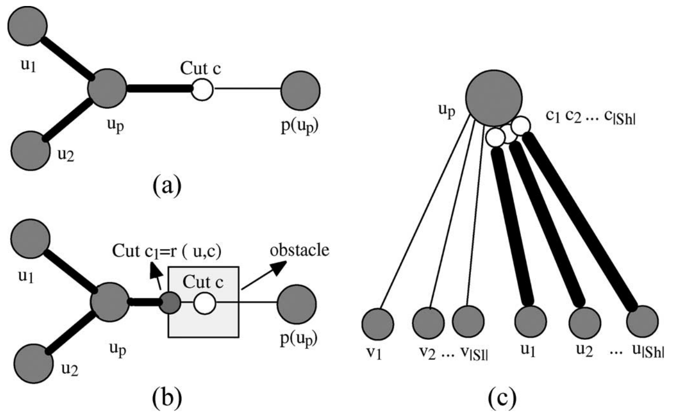

Fig. 12. (a) Illustration of the EqualLess subroutine. Here, $l(u_p, u_1) + w(u_1) + l(u_p, u_2) + w(u_2) + w(u_p) + l(u_p, c) = L_{\max}$. (b) Illustration of the EqualLess subroutine with the cutting node c in a forbidden region. Here, c_1 is the optimal substitute of c, where $c_1 = r(u_p, c)$ is just beside the forbidden region. (c) Illustration of the More subroutine.

```
Subroutine: EqualLess($T$, $C$, $u_p$, $totallen$)
1  if $p(u_p)$ does not exist
2      return;
3  if $totallen + w(u_p) + l(u_p, p(u_p)) \leq L_{max}$
4      if $u_p \in V_N$
5          $w(u_p) \leftarrow w(u_p) + totallen$;
6      if $u_p$ or any of its children is in $V_G$
7          $V_N \leftarrow V_N \setminus \{u_p\}$;
8          $V_G \leftarrow V_G \cup \{u_p\}$ and mark $u_p$ as unprocessed;
9      $T(V,E) \leftarrow T(V \setminus \cup_{i=1}^{k}\{u_i\}, E \setminus \cup_{i=1}^{k}\{e(u_i,u_p)\})$;
10 else
11     Let $c$ be the node on $e(u_p, p(u_p))$ with
       $l(c, u_p) + w(u_p) + totallen = L_{max}$;
12     if $f(c) = 1$
13         $c_1 \leftarrow r(u, c)$;
14         $C \leftarrow C \cup \{c_1\}$;
15         $T(V,E) \leftarrow T(V \setminus (\cup_{i=1}^{k}\{u_i\}) \cup \{c_1\} \setminus \{u_p\},$
                   $E \setminus \cup_{i=1}^{k}\{e(u_i,u_p)\} \setminus \{e(u_p,c_1)\})$;
16     else
17         $C \leftarrow C \cup \{c\}$;
18         $T(V,E) \leftarrow T(V \setminus (\cup_{i=1}^{k}\{u_i\}) \cup \{c\} \setminus \{u_p\},$
                   $E \setminus \cup_{i=1}^{k}\{e(u_i,u_p)\} \setminus \{e(u_p,c)\})$;
```

Fig. 13. Compute the case where $totallen + w(u_p) \leq L_{\max}$.

```
Subroutine: More($T$, $C$, $u_p$, $totallen$)
1  $S = \cup_{i=1}^{k}\{l(e(u_i,u_p)) + w(u_i)\}$;
2  $S_h = SPLIT(S, L_{max} - w(u_p))$;
   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
   $\{l(e(u_i,u_p)) + w(u_i)\} \in S_h$, $\forall 1 \leq i \leq |S_h|$;
3  $C \leftarrow C \cup \{c_i\}$ $\forall$ $1 \leq i \leq |S_h|$ ;
4  $T(V,E) \leftarrow T(V \setminus \cup_{i=1}^{|S_h|}\{u_i\}, E \setminus \cup_{i=1}^{|S_h|}\{e(u_i,u_p)\})$;
5  $minuslen \leftarrow \sum_{s \in S_h} s$;
6  $EqualLess(T, C, u_p, totallen - minuslen)$;
```

Fig. 14. Compute the case where $totallen + w(u_p) > L_{\max}$.

```
Subroutine: SPLIT($S$, $Bound$) [7]
1  if $|S| = 1$
2      if $\sum_{s \in S} s \leq Bound$
3          return $\emptyset$;
4      else
5          return $S$;
6  else
7      Median-find-and-halve($S$) and let $S_h$ be
       the higher half of $S$;
8      $W = \sum_{s \in S \text{ and } s \notin S_h} s$;
9      if $W = Bound$
10         return $S_h$;
11     else if $W < Bound$
12         return $SPLIT(S_h, Bound - W)$;
13     else ($W > Bound$)
14         return $SPLIT(S \setminus S_h, W) + S_h$;
```

Fig. 15. 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, $|S_h| \leq |S_l| \leq |S_h| + 1$.

the linear-time algorithm SPLIT presented in [5] 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 Fig. 15). The two subsets have three important properties: 1) for any $a \in S_l$ and any $b \in S_h$, we have $a \leq b$; 2) $\sum_{s \in S_l} s \leq L_{\max} - w(u_p)$; and 3) for any $b \in S_h$, we have $\sum_{s \in S_l} s + b > L_{\max} - w(u_p)$. Moreover, 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 $l(e(u_i, u_p)) + w(u_i) \in S_h$, $\forall 1 \leq i \leq |S_h|$ gives the best positions for inserting the cutting nodes; see Fig. 12(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).

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

IV. 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 [2]. 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 [2].

Theorem 1: The JIROA problem exhibits an optimal substructure.

Proof: We prove this property by contradiction. Given a tree $T = (V, E)$, suppose that the cutting set C is the optimal solution of the tree. Every cutting node in C cuts the given tree into two subtrees. Let some cutting node $c \in C$ cut T into subtrees T_1 and T_2. Let the cutting set $C_1 \subseteq C$ ($C_2 \subseteq C$) be the set of cutting nodes in T_1 (T_2). Thus, $C = C_1 \cup C_2 \cup \{c\}$. If C_1 does not form an optimal solution on T_1, let C_1' be the optimal solution on T_1, and thus, $|C_1'| < |C_1|$. Let $C' = C_1' \cup C_2 \cup \{c\}$. We have $|C'| = |C_1'| + |C_2| + 1 < |C_1| + |C_2| + 1 = |C|$. This contradicts the assumption of the optimality of set C. Therefore, 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., $L(u) \leq L_{\max}$, $\forall$ leaf nodes $u \in V_G$).

Proof: If $l(u, p(u)) \leq L_{\max}$, the leaf node u must satisfy the antenna rule, and thus, we do nothing. Otherwise, we must insert at least one cutting node between u and $p(u)$ to prevent u from antenna violation. The possible cutting range is represented by a thick line in Fig. 10(a). Let the optimal solution add the cutting node c' between u and u_p with $L(u) \leq L_{\max}$ and $L(p(u)) \leq L_{\max}$. We have $l(c, p(u)) = l(u, p(u)) - L_{\max} + w(u) \leq l(c', p(u))$. If we replace c' with c, the antenna rule that $L(u) \leq L_{\max}$ is satisfied and the antenna rule that $L(p(u)) \leq L_{\max}$ is more tightly satisfied. Therefore, among all nodes on the cutting range, c is the best position for adding a cutting node.

Moreover, if $f(c) = 1$, we must find a substitute for the cutting node c. The possible substitution range is represented by a thick line in Fig. 10(b). Let the optimal solution select the cutting node c'_1 between u and c with $L(u) \leq L_{\max}$ and $L(p(u)) \leq L_{\max}$. Because $l(c_1, u) \geq l(c'_1, u)$ and $l(c_1, p(u)) \leq l(c'_1, p(u))$, if we replace c'_1 with c_1, the antenna rule that $L(u) \leq L_{\max}$ is still satisfied and the antenna rule that $L(p(u)) \leq L_{\max}$ is more tightly satisfied. Therefore, among all possible substitution nodes on edge $e(u, c)$, c_1 is the best substitute. ■

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 $w(u_i)$: $\sum_{i=1}^{k}(l(u_p, u_i) + w(u_i)) + w(u_p) \leq L_{\max}$ 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., $L(u_p) \leq L_{\max}$, $\forall$ subleaf nodes $u_p \in V_G$ satisfying $\sum_{i=1}^{k}(l(u_p, u_i) + w(u_i)) + w(u_p) \leq L_{\max}$, where $u_p = p(u_i)$).

Proof: If u_p and its children form an isolated component, they must satisfy the antenna rule, and thus, we do nothing. If $\sum_{i=1}^{k}(l(u_i, u_p) + w(u_i)) + w(u_p) + l(u_p, p(u_p)) \leq L_{\max}$ is satisfied, u_p satisfies the antenna rule, and thus, we need no cutting node. Otherwise, because $\sum_{i=1}^{k}(l(u_i, u_p) + w(u_i)) + w(u_p) + l(u_p, p(u_p)) > L_{\max}$, we need to insert at least one cutting node to maintain $L(u_p) \leq L_{\max}$. Therefore, the least number of cutting nodes is one. The possible cutting range is represented by a thick line in Fig. 12(a). Suppose that the optimal solution adds the cutting node c' other than c to maintain $L(u_p) \leq L_{\max}$ and $L(p(u_p)) \leq L_{\max}$. We have $l(c, p(u_p)) = l(u_p, p(u_p)) - l(c, u_p) \leq l(c', p(u_p)) = l(u_p, p(u_p)) - l(c', u_p)$. If we replace c' with c, $L(u_p) \leq L_{\max}$ is satisfied and $L(p(u_p)) \leq L_{\max}$ is satisfied more tightly. Therefore, c is the best position for adding the cutting node.

Moreover, if $f(c) = 1$, we must find a substitute for cutting node c. The possible substitution range is represented by a thick line in Fig. 12(b). Let the optimal solution select the cutting node c'_1 between u_p and c with $L(u_p) \leq L_{\max}$ and $L(p(u_p)) \leq L_{\max}$. Because $l(c_1, u_p) \geq l(c'_1, u_p)$ and $l(c_1, p(u_p)) \leq l(c'_1, p(u_p))$, if we replace c'_1 with c_1, the antenna rule that $L(u_p) \leq L_{\max}$ is still satisfied and the antenna rule that $L(p(u_p)) \leq L_{\max}$ is more tightly satisfied. Therefore, among all possible substitution nodes, c_1 is the best substitute. ■

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., $L(u_p) \leq L_{\max}$, $\forall$ 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 $u_p = p(u_i)$).

Proof: By the proof of [5], we know that the set $S = S_l \cup S_h$ has the following three properties.

Step 1) For any $a \in S_l$ and any $b \in S_h$, we have $a \leq b$.
Step 2) $\sum_{s \in S_l} s \leq L_{\max} - w(u_p)$.
Step 3) For any $b \in S_h$, we have $\sum_{s \in S_l} s + b > L_{\max} - w(u_p)$.

Using the properties above, we can easily verify that S_l is the set with $\sum_{s \in S_l} s \leq L_{\max} - w(u_p)$, and the size of the set is maximized. Moreover, if two or more sets have the same maximum size and satisfy the same summation rule, S_l is the one with the minimum $\sum_{s \in S_l} s$ value. Since $\sum_{s \in S_l} s \leq L_{\max} - w(u_p)$, we need no cutting nodes inside the set, but we must add cutting nodes on every edge in S_h. Since S_l's size is maximized, the minimum number of cutting nodes is $|S| - |S_l| = |S_h|$. Moreover, every edge $e(u_i, u_p)$ with $\{l(e(u_i, u_p)) + w(u_i)\} \in S_h$ needs a cutting node. The cutting range of every edge $e(u_i, u_p)$ is represented by the thick line shown in Fig. 12(c). Let the optimal solution choose the set S'_l (and $S'_h = S \setminus S'_l$) with the optimal size $|S_l|$ such that the optimal solution do not add any jumper on the edge $e(u'_i, u_p)$ so that $\{l(e(u'_i, u_p)) + w(u'_i)\} \in S'_h$. Also, let the optimal solution select $|S_h|$ cutting nodes $c'_1, c'_2, \ldots, c'_{|S_h|}$. Since $l(c_i, u_p) = 0 \leq l(c'_i, u_p), \forall 1 \leq i \leq |S_h|$ and $\sum_{s \in S_l} s \leq \sum_{s' \in S'_l} s'$, $L(u_p) = \sum_{s \in S_l} s + w(u_p) + \sum_{i=1}^{|S_h|} 0 + l(u_p, p(u_p)) \leq L'(u_p) = \sum_{s' \in S'_l} s' + w(u_p) + \sum_{i=1}^{|S_h|} l(c'_i, u_p) + l(u_p, p(u_p))$. Thus, if we replace each c'_i by c_i, the antenna rule on u_p is satisfied more tightly. Therefore, $c_1, \ldots, c_{|S_h|}$ are the best positions for adding the cutting nodes. As a result, we can cut the original tree T and call Subroutine EqualLess (line 6 in More) to further reduce u_p into a leaf node. ■

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

Theorem 2: The BUJIO algorithm finds an optimal solution.

Proof: By Lemmas 2 and 3, lines 16–21 of Algorithm BUJIO exhibit the greedy-choice property on subleaf nodes. Moreover, by Lemma 1, lines 6–15 of Algorithm BUJIO also have the greedy-choice property on leaf nodes. Therefore, Algorithm BUJIO has the greedy-choice property on both leaf nodes and subleaf nodes. Since Algorithm BUJIO has the greedy-choice property and the JIROA problem has optimal substructure (by Theorem 1), Algorithm BUJIO finds an optimal cutting set. ■

V. Complexity Analysis

We analyze the time and space complexity of Algorithm BUJIO in this section.

A. Time Complexity

In Step 1), it takes $O(D \lg D)$ time to sort the obstacles, where D is the number of obstacles. The BUJIO algorithm applies a bottom-up method to find the optimal solution on the routing tree. In Step 2) of the BUJIO algorithm, it needs $O(V+E)$ time to compute the weights of all nodes. Since $O(E) = O(V)$ in a tree, this step requires $O(V)$ time. In Steps 2) and 3) of the BUJIO algorithm, we consider each leaf and each subleaf node only once. Since every node in the tree might be a subleaf and might be cut into a leaf, we traverse each node at most twice. When we traverse a leaf node, we need at most $O(\lg D)$ times to check whether the cutting node c is in the forbidden region or not. When we traverse subleaf nodes using Subroutine EqualLess, at most we also need $O(\lg D)$ time to check whether the cutting node c is in the forbidden region or not. In Subroutine More, we use the linear-time SPLIT algorithm to find the set S_h, and the algorithm requires constant time on each node. Then, we need constant time to cut the tree and call Subroutine EqualLess. Thus, the More subroutine requires constant time on each node. To sum up, we traverse each node at most twice, and in each traversal, we compute each node at most $O(\lg D)$ times. Therefore, the total time complexity of the BUJIO algorithm is $O(D \lg D) + O(V) + O(V \lg D) = O((V+D) \lg D)$.

B. Space Complexity

All we need to save are the tree T, the weight records w, and the cutting set C. A tree needs only $O(V+E) = O(V)$ space. The weight records need $O(V)$ space. Moreover, according to the algorithm, we add at most one cutting node for each leaf node. Therefore, we need $O(V)$ space to keep the set C. Thus, the total space complexity is $O(V)$.

Theorem 3: Algorithm BUJIO optimally solves the JIROA problem in $O((V+D) \lg D)$ time using $O(V)$ space, where V is the number of vertices in the given routing tree and D is the number of obstacles.

VI. Extensions

For the JIROA problem, the BUJIO algorithm is mainly based on the wire length, perimeter, and area measures for the antenna-strength model. We discuss in this section for solving the antenna problem under the antenna-strength-to-gate-size ratio model.

A. Problem Definition

Let u be a gate terminal. Let $A(u)$ denotes the area of the gate terminal u. Let $R_{\max}$ be the ratio upper bound. That is, for a gate terminal u, if the total wire length (perimeter or area) associated with u divided by $A(u)$ is larger than $R_{\max}$, then u will be damaged because of its antenna effect. With these definitions, we reformulate the problem as follows. Problem JIROA under the antenna-strength-to-gate-size ratio model (JIROA-R): Given a routing tree $T = (V = V_G \cup V_N, E)$, an upper bound $R_{\max}$, 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 $L(u)/A(u) \leq R_{\max}$, $\forall u \in V_G$.

B. Algorithm for Solving the JIROA-R Problem

The JIROA-R problem can be solved by the BUJIO algorithm with minor modifications. For the JIROA-R problem, we present here an $O((V+D) \lg D)$-time optimal algorithm, named Algorithm BUJIO under the antenna-strength-to-gate-size ratio model (BUJIO-R), for finding the minimum cutting set C for a given routing (Steiner or spanning) tree $T = (V, E)$ with V nodes and D obstacles. The only modification of BUJIO-R from BUJIO can be explained as follows: In BUJIO-R, we compute the accumulated edge weight as in BUJIO. To make some gate terminal u satisfy the antenna rule, however, we must let the antenna ratio associated with u smaller or equal to $R_{\max}$ (i.e., $L(u)/A(u) \leq R_{\max}$). In order to describe the BUJIO-R algorithm, we need to define a function here.

Definiton 3: Let u_p be a subleaf node and $u_1, u_2, \ldots, u_k$ denote the children nodes of u_p. If u_p is a Steiner point, and some of u_p's children are gate terminals, the function $mA(u_p)$ defines the minimum gate size among u_p's children. That is, $mA(u_p) = \min\{A(u_i) : u_i \text{ is a gate terminal } 1 \leq i \leq k\}$.

Algorithm BUJIO-R is summarized in Fig. 16 and explained below.

Steps 1) and 2) Lines 1–4 of Algorithm BUJIO-R: The same as BUJIO.

Step 3) Lines 6–15 of Algorithm BUJIO-R: Deal with every leaf node. We use $(l(u, p(u)) + w(u))/A(u)$ (instead of $l(u, p(u)) + w(u)$) in line 8 to check whether node u satisfies the antenna rule or not. Moreover, if u violates the antenna rule, we have to find a cutting node c such that u just reaches the antenna upper bound $R_{\max}$. That is, we make $(l(u, c) + w(u))/A(u) = R_{\max}$. Therefore, c is at the position with $l(u, c) = R_{\max} \times A(u) - w(u)$. The other processes in this part remain the same.

Step 4) Lines 16–27 of BUJIO-R: Deal with every subleaf node.

We use $(totallen + w(u_p))/A(u_p)$ (instead of $totallen + w(u_p)$) in line 24 as the classification. However, if u_p is a Steiner point and some of its children are gate terminals, we must redefine the $A(u_p)$ value. We make the whole subtree rooted at u_p as a group that needs to satisfy the antenna rules. Moreover, according to the antenna-strength-to-gate-size ratio model, the gate terminal with the smallest gate size is the weakest one to suffer the antenna violation. Therefore, if the gate terminal of the smallest gate size among the whole subtree satisfies the antenna rule, then all the gate terminals connecting to u_p must satisfy the rule. Thus, we make $A(u_p) \leftarrow mA(u_p)$ (lines 22–23 of Algorithm

```
Algorithm: $BUJIO\text{-}R(T, R_{max}, D, C)$
Input: $T = (V = V_G \cup V_N, E)$ /* A given tree. */
       $R_{max}$ /* 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      $w(u) = \sum_{f(e)=1 \wedge e \ incident \ on \ u} l(e)$;
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 $(l(u, p(u)) + w(u))/A(u) > R_{max}$
               Let $c$ be the node on the edge $e(u, p(u))$
               with $l(u, c) = R_{max} \times A(u) - w(u)$;
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
22             if $u_p$ is a Steiner point
23                 $A(u_p) \leftarrow mA(u_p)$;
24             if $(totallen + w(u_p))/A(u_p) \le R_{max}$
25                 $EqualLess\text{-}R(T, C, u_p, totallen)$ ;
26             else
27                 $More\text{-}R(T, C, u_p, totallen)$;
```

Fig. 16. Algorithm BUJIO-R deals with the leaf nodes first and, then, call Subroutines EqualLess-R and More-R to deal with the subleaf nodes.

```
Subroutine: EqualLess-R($T$, $C$, $u_p$, $totallen$)
1  if $p(u_p)$ does not exist
2      return;
3  if $(totallen + w(u_p) + l(u_p, p(u_p)))/A(u_p) \le R_{max}$
4      if $u_p \in V_N$
5          $w(u_p) \leftarrow w(u_p) + totallen$;
6      if $u_p$ or any of its children is in $V_G$
7          $V_N \leftarrow V_N \setminus \{u_p\}$;
8          $V_G \leftarrow V_G \cup \{u_p\}$ and mark $u_p$ as unprocessed;
9      $T(V, E) \leftarrow T(V \setminus \cup_{i=1}^{k} \{u_i\}, E \setminus \cup_{i=1}^{k} \{e(u_i, u_p)\})$;
10 else
11     Let $c$ be the node on $e(u_p, p(u_p))$ with
       $l(c, u_p) = R_{max} \times A(u_p) - w(u_p) - totallen$;
12     if $f(c) = 1$
13         $c_1 \leftarrow r(u, c)$;
14         $C \leftarrow C \cup \{c_1\}$;
15         $T(V, E) \leftarrow T(V \setminus (\cup_{i=1}^{k} \{u_i\}) \cup \{c_1\} \setminus \{u_p\},$
                   $E \setminus \cup_{i=1}^{k} \{e(u_i, u_p)\} \setminus \{e(u_p, c_1)\})$;
16     else
17         $C \leftarrow C \cup \{c\}$;
18         $T(V, E) \leftarrow T(V \setminus (\cup_{i=1}^{k} \{u_i\}) \cup \{c\} \setminus \{u_p\},$
                   $E \setminus \cup_{i=1}^{k} \{e(u_i, u_p)\} \setminus \{e(u_p, c)\})$;
```

Fig. 17. Compute the case where $(totallen + w(u_p))/A(u_p) \le R_{\max}$.

```
Subroutine: More-R($T$, $C$, $u_p$, $totallen$)
1  $S = \cup_{i=1}^{k} \{l(e(u_i, u_p)) + w(u_i)\}$;
2  $S_h = SPLIT(S, R_{max} \times A(u_p) - w(u_p))$;
   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
   $\{l(e(u_i, u_p)) + w(u_i)\} \in S_h$, $\forall 1 \le i \le |S_h|$;
3  $C \leftarrow C \cup \{c_i\}$ $\forall$ $1 \le i \le |S_h|$ ;
4  $T(V, E) \leftarrow T(V \setminus \cup_{i=1}^{|S_h|} \{u_i\}, E \setminus \cup_{i=1}^{|S_h|} \{e(u_i, u_p)\})$;
5  $minuslen \leftarrow \sum_{s \in S_h} s$;
6  $EqualLess\text{-}R(T, C, u_p, totallen - minuslen)$;
```

Fig. 18. Compute the case where $(totallen + w(u_p))/A(u_p) > R_{\max}$.

BUJIO-R). Furthermore, if u_p satisfies the antenna rule, the smallest gate terminal in the subtree will also satisfy the antenna rule and so will other gate terminals in the subtree.

We discuss the processing in three cases as follows.

Case 4.1) The same as the BUJIO algorithm.

Case 4.2) $(totallen + w(u_p))/A(u_p) \le R_{\max}$, for this case, we apply the EqualLess-R subroutine; see Fig. 17. We use $(totallen + w(u_p) + l(u_p, p(u_p)))/A(u_p)$ (instead of $totallen + w(u_p) + l(u_p, p(u_p))$) in line 3 to check whether node u_p satisfies the antenna rule or not. Moreover, if u_p violates the antenna rule, we must find a cutting node c such that u_p just reaches the antenna upper bound $R_{\max}$. That is, we have to make $(totallen + w(u_p) + l(c, u_p)))/A(u_p) = R_{\max}$. Therefore, c is at the position with $l(c, u_p) = R_{\max} \times A(u_p) - w(u_p) - totallen$. The other processes in this part remain the same.

Case 4.3) $(totallen + w(u_p))/A(u_p) > R_{\max}$, for this case, we apply the More-R subroutine; see Fig. 18. In the BUJIO algorithm, the accumulated edge weight that node u_p can bear is $L_{\max} - w(u_p)$. However, in the BUJIO-R algorithm, we must have $L(u_p)/A(u_p) \le R_{\max}$. Therefore, the amount of additional edge weight that u_p can bear is $R_{\max} \times A(u_p) - w(u_p)$. Thus, we use $R_{\max} \times A(u_p) - w(u_p)$(instead of $L_{\max} - w(u_p)$) in line 2 as the upper bound for the SPLIT

TABLE III

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 10 000 NODES AND 500 OBSTACLES. NOTE THAT EVEN THOUGH THE RESULTS OF THE ISPD-05e ALGORITHM IS CLOSE TO OUR BUJIO ALGORITHM, 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

L_{max}	BUJIO	ISPD-05e		DAC-05e		ISPD-04e	
(um)	$\#J$	$\#J$	$\%More$	$\#J$	$\%More$	$\#J$	$\%More$
220	1533	1537	0.26%	1806	17.81%	2359	53.88%
230	1341	1348	0.52%	1607	19.84%	2020	50.63%
240	1144	1149	0.44%	1378	20.45%	1742	52.27%
250	961	966	0.52%	1210	25.91%	1501	56.19%
260	823	827	0.49%	1050	27.58%	1279	55.41%
270	706	708	0.28%	919	30.17%	1109	57.08%
280	599	601	0.33%	779	30.05%	928	54.92%
290	517	518	0.19%	649	25.53%	767	48.36%
300	434	435	0.23%	562	29.49%	645	48.62%
310	366	366	0.00%	488	33.33%	543	48.36%
320	305	305	0.00%	413	35.41%	458	50.16%

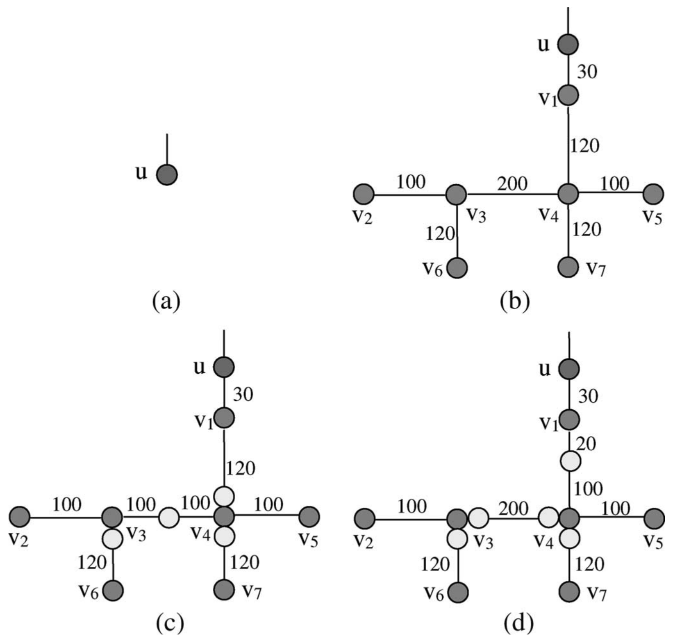

Fig. 19. (a) Selected node u. (b) Subtree appended to node u. (c) BUJIO adds four jumpers on the subtree. Here, $L_{\max} = 200$ μm. (d) ISPD-05e algorithm adds five jumpers on the subtree.

subroutine. The other processes in this part remain the same.

It is clear from the above discussion that the underlying ideas of Algorithm BUJIO-R are the same as those of Algorithm BUJIO. Therefore, the optimality proof of Algorithm BUJIO-R for the JIROA-R problem is similar to that of the BUJIO algorithm. Furthermore, all the modifications in the BUJIO-R algorithm do not influence the running time and the memory requirement. Thus, the time complexity and the space complexity of the BUJIO-R algorithm remain the same as BUJIO.

Theorem 4: Algorithm BUJIO-R optimally solves the JIROA-R problem in $O((V + D) \lg D)$ time using $O(V)$ space, where V is the number of vertices in the given routing tree and D is the number of obstacles.

VII. 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.* [9], the DAC-05 work [7] by Su and Chang, and the ISPD-04 work [3] 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 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 $10^4 \times 10^4$ μm 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 III shows the number of jumpers required for fixing all antenna violations for a routing tree with 10 000 nodes and 500 obstacles by changing $L_{\max}$ from 220 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.

TABLE IV
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 10 000 NODES AND 200 OBSTACLES

L_{max}	BUJIO	ISPD-05e		DAC-05e		ISPD-04e	
(um)	$\#J$	$\#J$	$\%More$	$\#J$	$\%More$	$\#J$	$\%More$
220	4274	5455	27.63%	4557	6.62%	5943	39.05%
230	4157	5311	27.76%	4421	6.35%	5762	38.61%
240	4056	5174	27.56%	4305	6.14%	5604	38.17%
250	3972	5061	27.42%	4187	5.41%	5455	37.34%
260	3889	4951	27.31%	4076	4.81%	5299	36.26%
270	3814	4844	27.01%	3983	4.43%	5157	35.21%
280	3728	4738	27.09%	3895	4.48%	5028	34.87%
290	3657	4637	26.80%	3809	4.16%	4915	34.40%
300	3605	4541	25.96%	3728	3.41%	4796	33.04%
310	3542	4467	26.12%	3653	3.13%	4695	32.55%
320	3489	4401	26.13%	3587	2.81%	4593	31.64%

TABLE V
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 $L_{\max} = 200$ μm

	BUJIO	ISPD-05e		DAC-05e		ISPD-04e	
$\#node$	$\#J$	$\#J$	$\%More$	$\#J$	$\%More$	$\#J$	$\%More$
10000	4548	5844	28.50%	4881	7.32%	6391	40.52%
20000	7329	9188	25.36%	7634	4.16%	9800	33.72%
30000	10002	12414	24.12%	10246	2.44%	12899	28.96%
40000	12400	15375	23.99%	12647	1.99%	15753	27.04%
50000	15113	18754	24.09%	15283	1.12%	19013	25.81%
60000	17686	22019	24.50%	17877	1.08%	22256	25.84%
70000	20355	25369	24.63%	20487	0.65%	25525	25.40%
80000	23167	28888	24.69%	23217	0.22%	28950	24.96%
90000	25938	32373	24.81%	25985	0.18%	32426	25.01%
100000	28735	35884	24.88%	28741	0.02%	35889	24.90%

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 I. 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 nonrandom 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 III, the improvements range from 17% to 57%, much more than the 10% wirelenth difference between the Steiner tree (728 568 units long) and the spanning tree (801 302 units long).

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 Fig. 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 Fig. 19(a) be a selected node. We added a subtree rooted at node u as shown in Fig. 19(b). The subtree has similar topology as that of the routing tree shown in Fig. 4. It is clear that BUJIO needs only four jumpers for the subtree, as shown in Fig. 19(c), while the ISPD-05e algorithm needs five jumpers, as illustrated in Fig. 19(d), to fix the antenna violations. We generated the test cases based on this expansion method.

For the experiments shown in Table IV, we constructed a corresponding test case for each $L_{\max}$ value based on the previously mentioned expansion method. The experimental results show that BUJIO outperforms 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 V, for which we constructed a corresponding test case for each given number of gate terminals (number of nodes) based on the aforementioned expansion method. Note that we tested on larger problem sizes (node numbers) to examine the runtime differences more closely. For the problems with typical sizes (node numbers), we conducted another experiment, as shown in Table VI. The results also show that the BUJIO algorithm still outperforms the other three algorithms for those practical problem sizes.

TABLE VI
COMPARISONS OF THE NUMBERS OF JUMPERS REQUIRED FOR BUJIO, ISPD-05e, DAC-05e, AND ISPD-04e FOR FIXING ALL ANTENNA VIOLATIONS BASED ON TEST CASES WITH ROUTING TREES OF 200 OBSTACLES EACH

node number	L_{max} (*um*)	BUJIO $\#J$	ISPD-05e $\#J$	ISPD-05e $\%More$	DAC-05e $\#J$	DAC-05e $\%More$	ISPD-04e $\#J$	ISPD-04e $\%More$
10	50	16	16	+ 0.00%	18	+12.50%	18	+12.50%
	100	16	16	+ 0.00%	18	+12.50%	18	+12.50%
	150	15	15	+ 0.00%	17	+13.33%	18	+20.00%
	200	14	14	+ 0.00%	16	+14.29%	18	+28.57%
30	50	50	53	+ 6.00%	56	+12.00%	58	+16.00%
	100	44	47	+ 6.82%	50	+13.64%	54	+22.73%
	150	37	41	+10.81%	42	+13.51%	48	+29.73%
	200	34	35	+ 2.94%	36	+ 5.88%	42	+23.53%
50	50	87	88	+ 1.15%	94	+ 8.05%	98	+12.64%
	100	71	77	+ 8.45%	78	+ 9.86%	90	+26.76%
	150	59	69	+16.95%	62	+ 5.08%	71	+20.34%
	200	49	61	+24.49%	52	+ 6.12%	65	+32.65%
100	50	148	154	+ 4.05%	166	+12.16%	181	+22.30%
	100	116	133	+14.66%	126	+ 8.62%	150	+29.31%
	150	89	108	+21.35%	98	+10.11%	120	+34.83%
	200	68	86	+26.47%	76	+11.76%	96	+41.18%
150	50	228	237	+ 3.95%	247	+ 8.33%	272	+19.30%
	100	159	193	+21.38%	177	+11.32%	221	+38.99%
	150	107	133	+24.30%	126	+17.76%	158	+47.66%
	200	76	99	+30.26%	91	+19.74%	119	+56.58%
200	50	280	307	+ 9.64%	306	+ 9.29%	344	+22.86%
	100	188	231	+22.87%	211	+12.23%	255	+35.64%
	150	120	151	+25.83%	144	+20.00%	182	+51.67%
	200	78	104	+33.33%	93	+19.23%	126	+61.54%
250	50	346	388	+12.14%	379	+ 9.54%	428	+23.70%
	100	227	272	+19.82%	248	+ 9.25%	307	+35.24%
	150	140	184	+31.43%	167	+19.29%	226	+61.43%
	200	89	114	+28.09%	106	+19.10%	146	+64.04%

For the experiments shown in Table IV, 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 V, 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 twofold: 1) The minimum spanning tree and Steiner tree are the same for the appended subtrees and 2) the DAC-05e algorithm adds only four jumpers on each subtree while the ISPD-05e adds five 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. Therefore, each of the ISPD-05e and the DAC-05e algorithms has its own strengths and weaknesses. Each of them might be effective for some cases but performs poorly for other cases. No matter what test case is considered, however, BUJIO always finds the optimal solution.

Table VII shows the CPU times required for antenna fixing on routing trees of the numbers of nodes ranging from 10 000 to 100 000, with $L_{\max} = 200$ μm and 500 obstacles on each plane. Column 1 gives the numbers of nodes in the routing trees. Because the numbers of nodes are so huge, the program spent most of the CPU times in reading the input files. Therefore, we list the CPU times for reading the input files and running the algorithms separately in order to examine the time complexity more closely. The second column (File) gives the CPU times for reading the input files. The (Main) column in each algorithm gives the respective CPU times for executing the main body of the algorithm.

TABLE VII
COMPARISONS OF THE CPU TIMES REQUIRED FOR BUJIO, ISPD-05e, DAC-05e, AND ISPD-04e TO FIX THE ANTENNA VIOLATIONS, BASED ON 500 OBSTACLES ON EACH PLANE AND $L_{\max} = 200$ μm

node number	File (s)	BUJIO Main (s)	ISPD-05e Main (s)	DAC-05e Main (s)	ISPD-04e Main (s)
10000	0.125	0.031	0.031	0.031	0.031
20000	0.250	0.078	0.078	0.047	0.063
30000	0.391	0.109	0.125	0.078	0.078
40000	0.531	0.156	0.156	0.093	0.109
50000	0.610	0.203	0.219	0.141	0.156
60000	0.781	0.250	0.265	0.172	0.172
70000	0.891	0.297	0.328	0.204	0.203
80000	1.032	0.359	0.359	0.235	0.235
90000	1.172	0.406	0.422	0.250	0.250
100000	1.250	0.485	0.485	0.281	0.281

As shown in the table, the empirical running time for the four methods are close to linear. In particular, BUJIO requires only 1.25 s to find an optimal solution for a routing tree of 0.1 million nodes. Therefore, BUJIO can handle a test case of a very huge number of nodes in very short time. Fig. 20 shows a layout resulting from BUJIO with 426 jumpers for antenna fixing on a routing tree with 1000 nodes and 500 obstacles on the plane and based on $L_{\max} = 500$ μm. Also, Fig. 21 shows a smaller layout resulting from BUJIO with 197 jumpers for antenna fixing on a routing tree with 200 nodes and 200 obstacles on a 2000×2000 μm plane and based on $L_{\max} = 100$ μm.

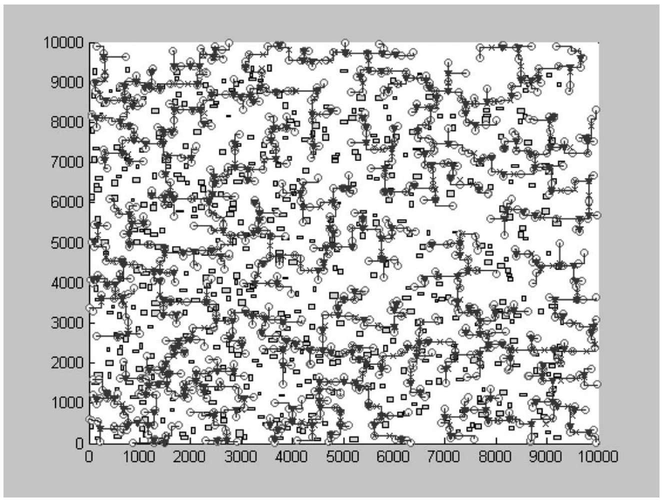

Fig. 20. Layout resulting from the BUJIO algorithm. The rectangles denote the obstacles, the circles denote the nodes of the routing tree, the triangles denote the Steiner points, and the × signs denote the inserted jumpers.

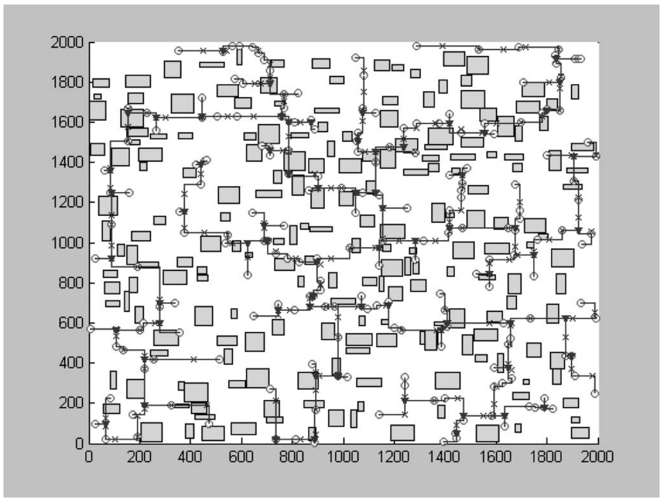

Fig. 21. Layout resulting from the BUJIO algorithm. The rectangles denote the obstacles, the circles denote the nodes of the routing tree, the triangles denote the Steiner points, and the × signs denote the inserted jumpers.

VIII. Concluding Remarks

We have presented an $O((V + D) \lg D)$-time optimal jumper-insertion algorithm 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. This paper 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 postlayout optimizer for antenna violation fixing.

Some potential future work includes timing-aware jumper insertion and integration of jumper insertion and diode insertion for antenna avoidance/fixing. A jumper consists of two vias and might incur significant delay. To quantity the timing affected by jumper insertion, we can incorporate the RC delay of the vias into the interconnect for overall delay computation.

Jumpers and diodes use different resources—jumpers use routing resources while diodes consume silicon areas. Therefore, it might not be easy/accurate to model their tradeoff. For example, how can we correlate silicon area (mainly for logic) and routing area (mainly for interconnections) accurately? Nevertheless, they can be considered together to achieve the best fixing rate of the antenna violations. If there is not sufficient silicon area for diode insertion, jumpers can be used to fix antenna violation as long as there is available routing resource, and vice versa. Therefore, the jumper insertion and diode insertion problems are usually considered separately by formulating the routing and silicon resources as constraints. If we want to integrate the two techniques, however, one way to do so is to correlate the costs (e.g., delay) of vias and diodes (their extension wires) and optimize the combined cost of jumper and diode insertion.

References

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

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. New York: McGraw-Hill, 2001.

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

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

[5] S. Kundu and J. Misra, "A linear tree partitioning algorithm," *SIAM J. Comput.*, vol. 6, no. 1, pp. 151–154, Mar. 1977.

[6] H. Shin, C.-C. King, and C. Hu, "Thin oxide damage by plasma etching and ashing process," in *Proc. IRPS*, Mar./Apr. 1992, pp. 37–41.

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

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

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

[10] P. H. Chen, "Beat the competition: A knowledge based design process addressing the antenna effect and cell placement," *IEEE Circuits Devices Mag.*, vol. 20, no. 3, pp. 18–27, Jun. 2004.

Bor-Yiing Su received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 2005.

He is currently with the SpringSoft Company in Taiwan. He plans to conduct graduate study in the Department of Electrical Engineering at the topmost universities in America and to obtain a Ph.D. degree in the area of physical design. His current research interest is placement-related topics. He is now working on large-scale congestion driven placement.

Mr. Su was the recipient of the Presidential Award from National Taiwan University for four semesters during his college years.

Yao-Wen Chang (S'94–M'96) received the B.S. degree from National Taiwan University, Taipei, Taiwan, R.O.C., in 1988, and the M.S. and Ph.D. degrees from the University of Texas at Austin in 1993 and 1996, respectively, all in computer science.

He is a Professor of the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, National Taiwan University. He is currently also a Visiting Professor at Waseda University, Kitakyushu, Japan. He was with IBM T. J. Watson Research Center, Yorktown Heights, NY, in summer 1994. From 1996 to 2001, he was on the faculty of National Chiao Tung University, Hsinchu, Taiwan. His current research interests lie in VLSI physical design, design for manufacturing, and FPGA. He has been working closely with industry on projects in these areas.

Dr. Chang received an award at the 2006 ACM ISPD Placement Contest, Best Paper Award at ICCD-1995, and eight Best Paper Nominations from DAC-2007, ISPD-2007, DAC-2005, 2004 ACM TODAES, ASP-DAC-2003, ICCAD-2002, ICCD-2001, and DAC-2000. He has received many awards for research performance, such as the 2005 and 2006 First-Class Principal Investigator Awards and the 2004 Mr. Wu Ta You Memorial Award from the National Science Council of Taiwan, the 2004 MXIC Young Chair Professorship from the MXIC Corporation, and for excellent teaching from National Taiwan University and National Chiao Tung University. He is an Editor of the *Journal of Computer and Information Science*. He currently serves on the ACM/SIGDA Physical Design Technical Committee and the technical program committees of a few important conferences on VLSI design automation, including ASP-DAC (topic chair), DAC, DATE, FPT, GLSVLSI, ICCAD, ICCD, ISPD, SOCC, and VLSI-DAT. He is currently the Chair of the Design Automation and Test (DAT) Consortium of the Ministry of Education, Taiwan, a member of the Board of Governors of the Taiwan IC Design Society, and a member of the IEEE Circuits and Systems Society, ACM, and ACM/SIGDA.

Jiang Hu (S'98–M'01) received the B.S. degree in optical engineering from Zhejiang University, Hangzhou, China, in 1990, the M.S. degree in physics from the University of Minnesota, Duluth, in 1997, and the Ph.D. degree in electrical engineering from the University of Minnesota, Minneapolis, in 2001.

He was with IBM Electronics Design Automation from January 2001 to June 2002. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Texas A&M University, College Station. His research interest is computer-aided design for very large scale integration circuits and systems, in particular, interconnect optimization, clock network synthesis, energy efficiency design, and design for manufacturability.

Dr. Hu was the recipient of the Best Paper Award at the Association for Computing Machinery (ACM)/IEEE Design Automation Conference in 2001 and an IBM First Plateau Invention Achievement Award in 2003. He is an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS. He has served on the technical program committees of the Design Automation Conference (DAC), International Conference on Computer-Aided Design (ICCAD), International Symposium on Physical Design (ISPD), IEEE/ACM Design, Automation and Test in Europe (DATE), International Conference on Computer Design (ICCD), and International Symposium on Circuits And Systems (ISCAS).