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

Bor-Yiing Su and Yao-Wen Chang

National Taiwan University

Jiang Hu

Texas A&M University

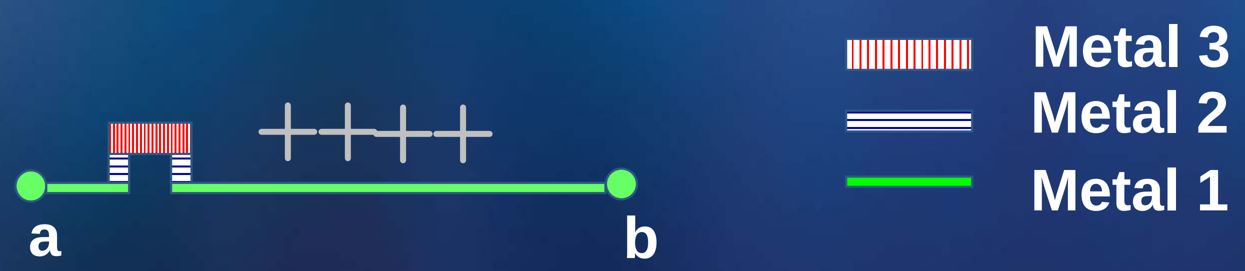
Outline

- Introduction
- Problem Definition
- An Optimal Algorithm for Jumper Insertion
- Complexity Analysis
- Experimental Results
- Conclusions

Antenna Effect

- The process antenna effect is a phenomenon of plasma-induced **gate oxide degradation** caused by **charge accumulation** on conductors.
- During metallization, each time an additional layer of interconnect is added. While the metal line is being manufactured, the **floating interconnect** acts as a temporary **capacitor** to store charges induced from plasma etching.

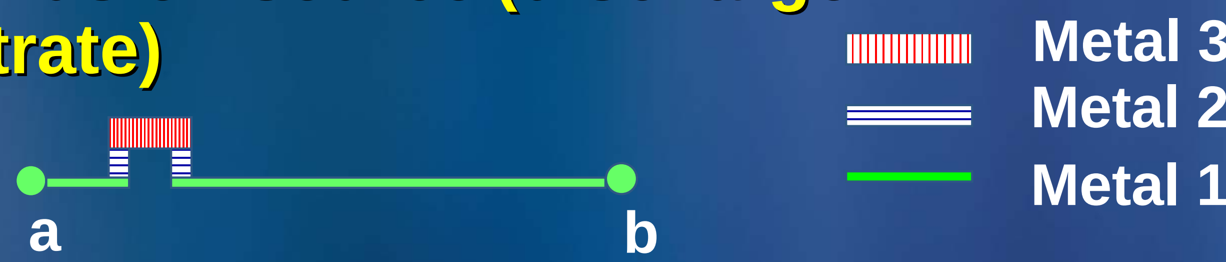
A two-pin net



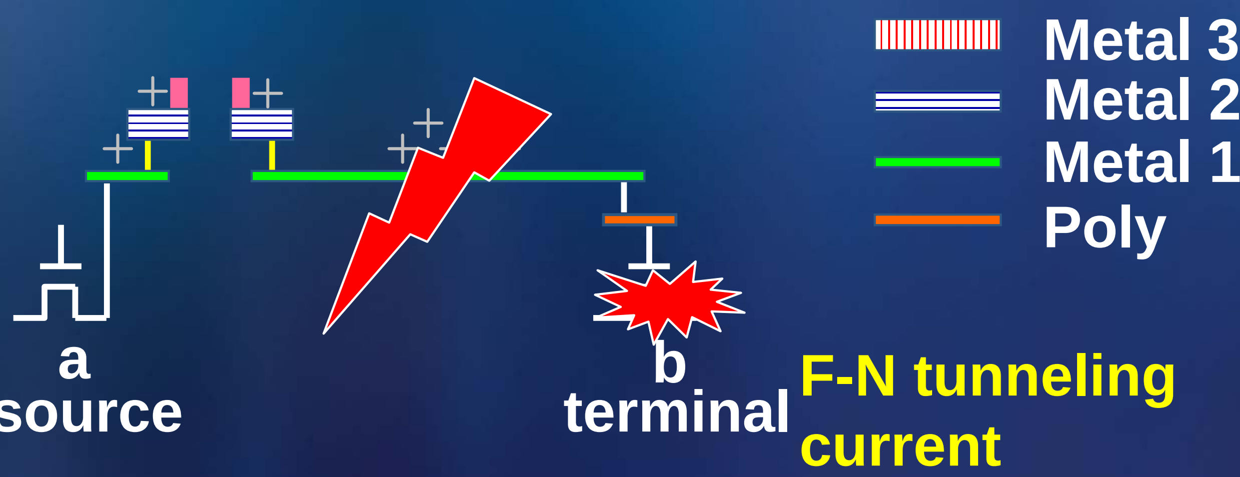
Antenna Effect (cont'd)

- If the charged conductor is connected only to the gate oxide, Fowler-Nordheim (F-N) tunneling current will discharge through the thin oxide and damage the gate.
  - No harm to diffusion source (discharge through substrate)

A two-pin net



Cross section view



Solutions for Antenna Effect

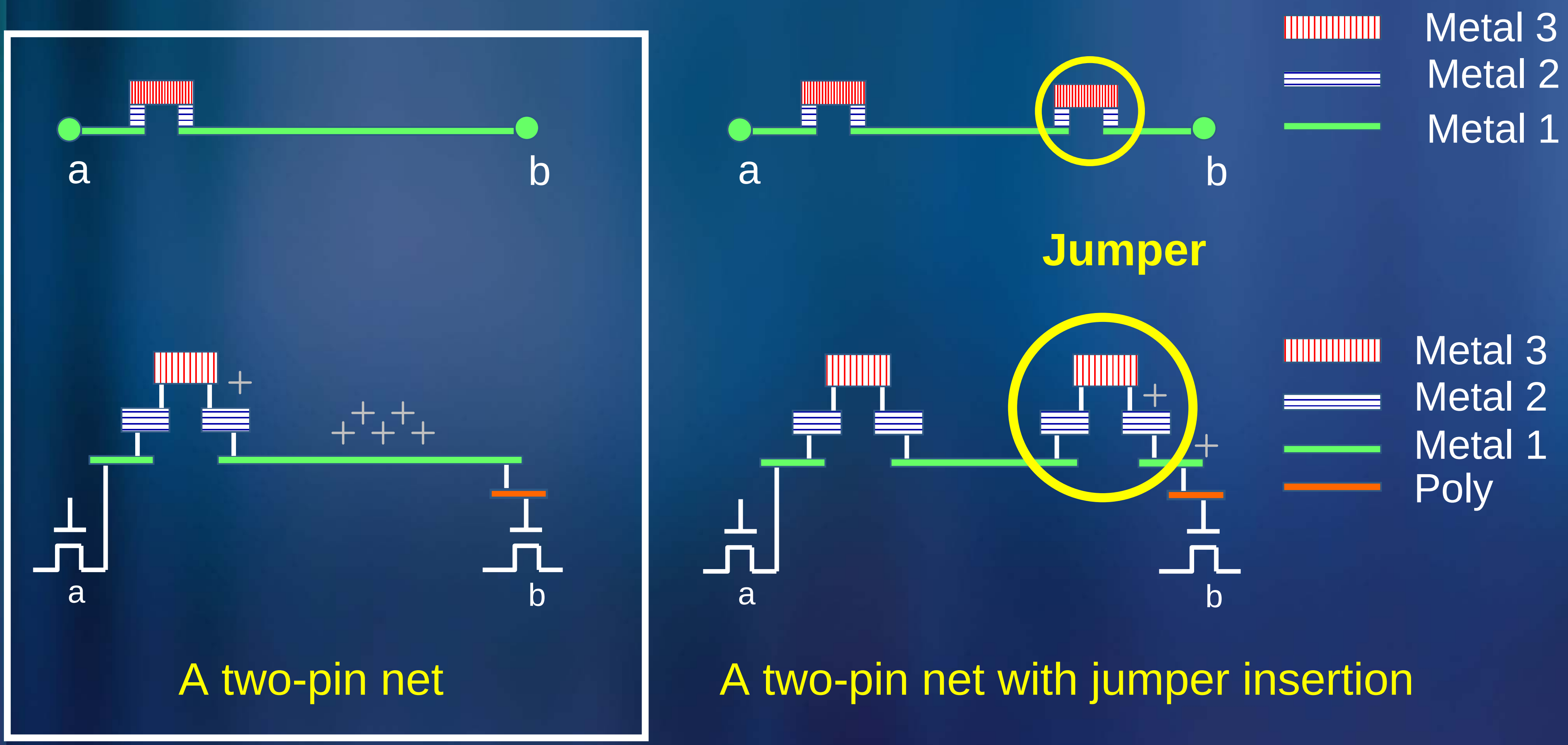
- Jumper Insertion
  - Break the signal wires with antenna violations and route them to the top-most layer
  - Induce vias
- Embedded Protection Diode
  - Add a protection diode on every input port of a cell
  - May consume unnecessary areas
- Diode Insertion
  - Insert "under-the-wire" diodes to fix the violation
  - Need extra silicon space to place the diodes

Solutions for Antenna Effect

- Jumper Insertion
  - Break the signal wires with antenna violations and route them to the highest layer
  - Induce vias
- Embedded Protection Diode
  - Add a protection diode on every input port of a cell
  - May consume unnecessary areas
- Diode Insertion
  - Insert “under-the-wire” diodes to fix the violation
  - Need extra silicon space to place the diodes

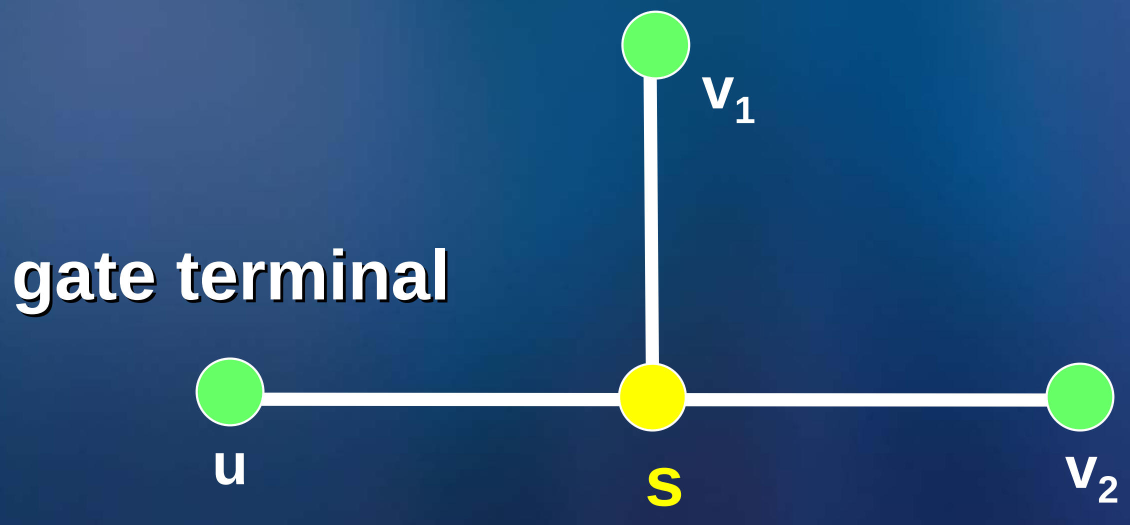
Jumper Insertion

- **Jumper insertion reduces the charge amount for violated nets during manufacturing**
  - **Side effects: Delay and congestion**



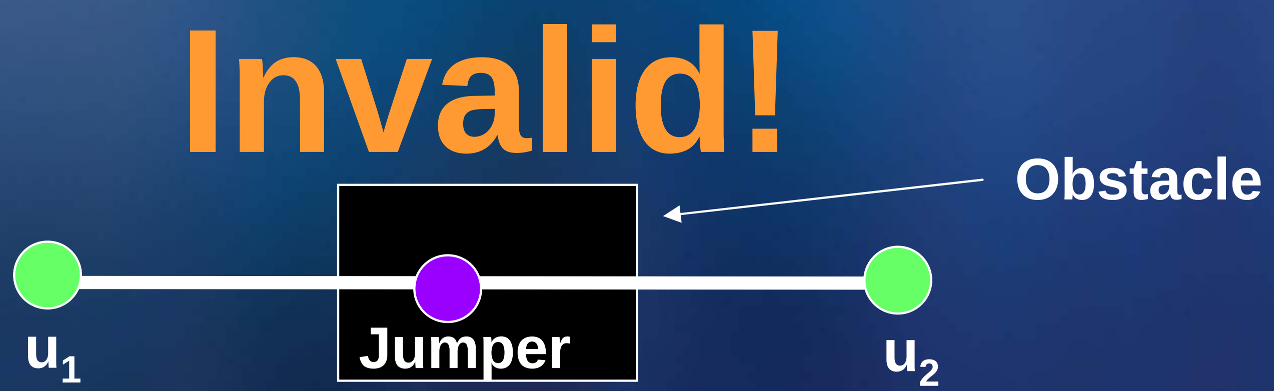
Steiner Tree

- Steiner points are just wire junctions. They cannot help discharge the wire.
- 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 the gate terminal *u*.



Routing Obstacles

- Since a jumper routes a signal wire to the topmost layer, we must consider the routing with **obstacles** in the **active layers**, e.g., pre-routed nets, power/ground nets, clock nets.
  - Active layers: the layers from the current routing layer up to the topmost layer
- Need to avoid obstacles when adding jumpers



Previous Work

- Ho, Chang & Chen, ISPD-04
  - Bottom-up dynamic programming in loglinear time
  - Insert jumpers right beside nodes of a spanning tree
- Wu, Hu & Mahapatra, ISPD-05
  - Insert jumpers at arbitrary positions of edges of a Steiner tree in linear time
  - Optimal only for some special tree topologies
- Su and Chang, DAC-05
  - Insert jumpers at arbitrary positions of edges of a spanning tree in loglinear time
  - Optimal for a spanning tree with arbitrary topologies

Previous Work

Previous works are optimal only for restricted cases & do not consider obstacles!!

	ISPD-04	DAC-05	ISPD-05	This work
Optimal for general routing tree?	N	N	N	Yes
Consider obstacles?	N	N	N	Yes
Consider Steiner trees?	N	N	Yes	Yes
Allow arbitrary insertion positions?	N	Yes	Yes	Yes

Outline

- Introduction to Antenna Effect
- Problem Definition
- An Optimal Algorithm for Jumper Insertion
- Complexity Analysis
- Experimental Results
- Conclusions

Problem Definition

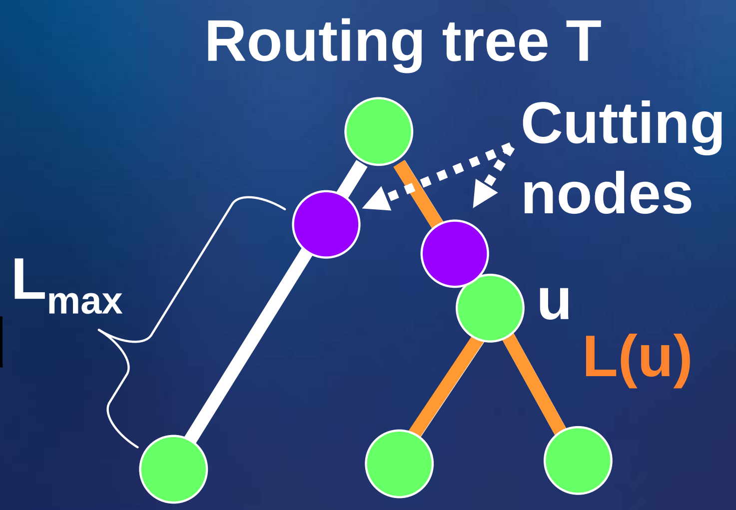
- Formulate the problem of jumper insertion on a routing tree for antenna avoidance/fixing as a **tree-cutting problem**
- $T = (V_G \cup V_N, E)$: a Steiner tree
  - Set V_G of nodes represents all gate terminals
  - Set V_N of nodes represents other nodes
  - Set E of edges denotes the wires connecting the circuit terminals or junctions
- *D:* set of obstacles in the active layers
- Projection of the obstacles in D defines the *forbidden regions* for jumper insertion
- *F:* set of the forbidden regions.
  - $f(u) = 1$ if node u is in a forbidden region $(u \in F)$; $f(u) = 0$, otherwise

Edge Weight Modeling

- The **edge weight** in a Steiner tree models the strength of antenna effect caused by the corresponding wire.
- The edge weight can be
  - the **length** of the wire
  - the **area** of the wire
  - the **perimeter** of the wire
  - the **ratio** of antenna strength to gate size, or
  - any other reasonable models.

Problem JIROA

- Problem *JIROA* (*Jumper Insertion on a Routing tree with Obstacles for Antenna avoidance/fixing*)
- **Input:** A routing tree $T = (V_G \cup V_N, E)$, an upper bound *Lmax*, and a set *D* of rectangular obstacles
- **Goal:** find the minimum set *C* of cutting nodes
- **Constraint:** $c \neq u$ for any $c \in C$ and $u \in V$, $f(c) = 0$, $\forall c \in C$, such that $L(u) \leq Lmax$, $\forall u \in V_G$.
  - $L(u)$: sum of effective edge weights on node u
  - *Lmax* : antenna upper bound

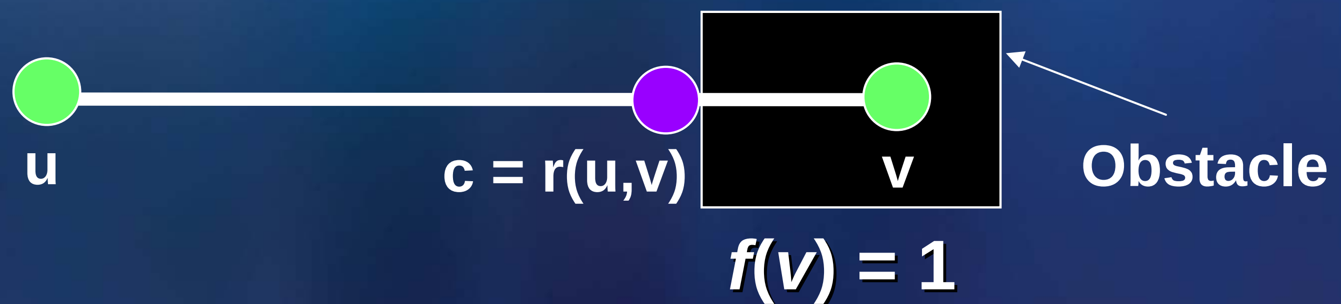


Outline

- Introduction to Antenna Effect
- Problem Definition
- An Optimal Algorithm for Jumper Insertion
- Complexity Analysis
- Experimental Results
- Conclusion

Definitions

- A **subleaf** is a node for which all its children are leaf nodes, and all the children have been processed.
- 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 $l(u, c)$ (weight between nodes u and c) being the **maximum** among every node on edge e.



Algorithm BUJIO

- Algorithm BUJIO (Bottom Up Jumper Insertion with Obstacles) for jumper insertion applies a bottom-up approach to insert cutting nodes on a routing tree.
  - Step 1: Sort the obstacles in *D* by the *x*-axes and then the *y*-axes
  - Step 2: Compute the initial weight for every node
  - Step 3: Makes every leaf node satisfy the antenna rule
  - Step 4: Make every subleaf node satisfy the antenna rule, and then cut the corresponding leaf nodes to turn a subleaf node into a new leaf node

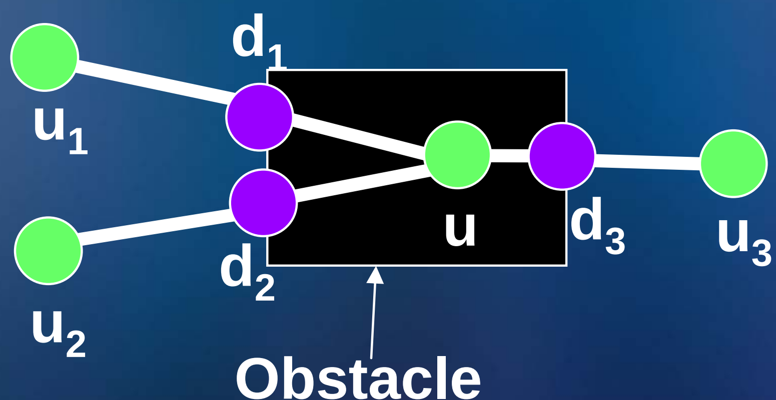
Algorithm BUJIO

- Algorithm BUJIO (Bottom Up Jumper Insertion with Obstacles) for jumper insertion uses a bottom-up approach to insert cutting nodes on a routing tree.
  - Step 1: Sort the obstacles in *D* by the *x*-axes and then the *y*-axes
  - Step 2: Compute the weight of every node
  - Step 3: Let every leaf node satisfy the antenna rule
  - Step 4: Make every subleaf node satisfy the antenna rule, and then cut the corresponding leaf nodes to turn a subleaf node into a new leaf node

Iteratively apply Steps 3 & 4 until all gate terminals satisfy the antenna rule

Steps 1 and 2: Initialization

- Step 1: Sort the obstacles in ***D*** by the ***x*-axes** and then the ***y*-axes**
  - Determine ***f(u)*** for some node u in ***O*(lg *D*)** time
- Step 2: Compute the initial **weight** of every node
  - ***w(u)***: accumulated edge weights on node ***u***

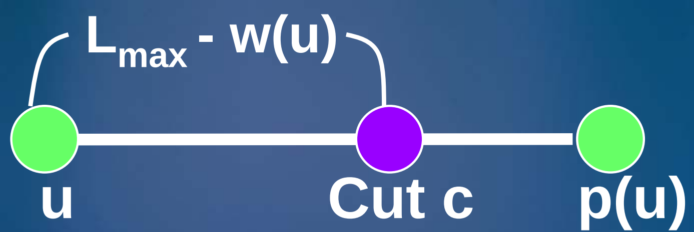


$w(u) = l(u,d_1) + l(u,d_2) + l(u,d_3)$

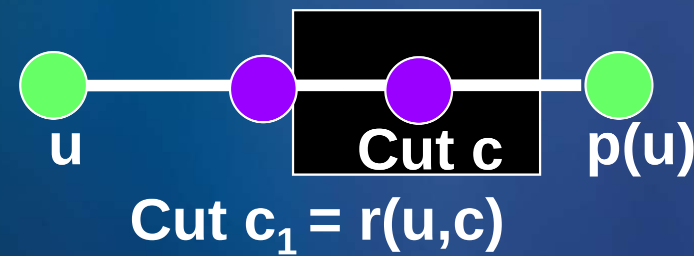
Step 3: Leaf Node Processing

- Step 3: Make every **leaf node** which is a gate terminal satisfy the antenna rule

$l\,(u, p(u)) + w(u) > L_{max}$
$f(c) = 0$



$l\,(u, p(u)) + w(u) > L_{max}$
$f(c) = 1$



Tree node Cutting node

p(u) : parent of u
l(e), l(u,v) : weight of the edge e = (u,v)

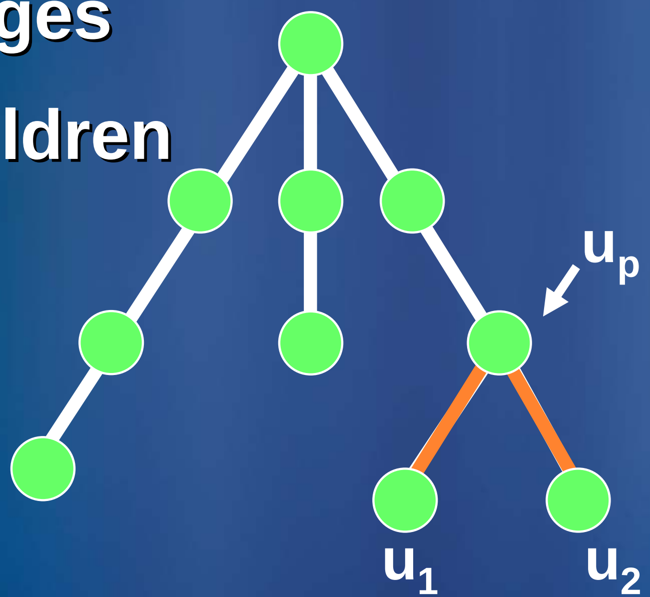
L_{max}: upper bound on antenna
C : cutting set

Step 4: Subleaf Node Processing

- Step 4: Make every **subleaf node** satisfy antenna rule
- ***totallen(u_p)***: total weights of the edges between the node u_p and its children

$$totallen(u_p) = \sum_{i=1}^{k} l(u_i, u_p) + w(u_i)$$

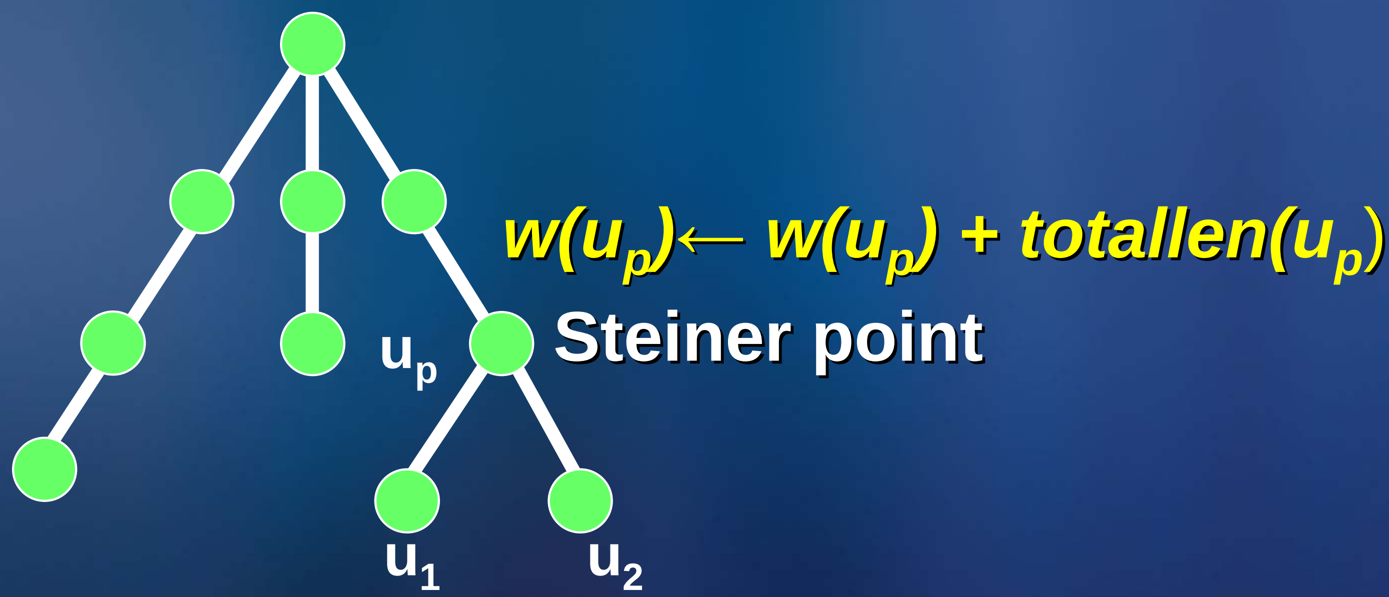
u_p : a subleaf node
u_i : subleaf's children, $\forall 1 \leq i \leq k$



- Classify the subleaf nodes according to *totallen*
  - Case 1: u_p and all its children are not gate terminals
  - Case 2: $totallen(u_p) + w(u_p) \leq L_{max}$
  - Case 3: $totallen(u_p) + w(u_p) > L_{max}$

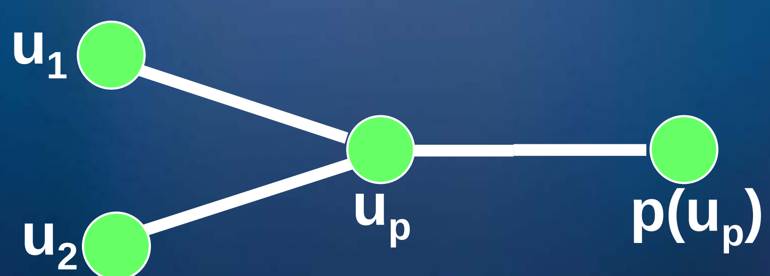
Case 1: u_p and All Its Children Are in V_N

- u_p and all its children are **not gate terminals**, so no antenna violation occurs.
  - $w(u_p) \leftarrow w(u_p) + totallen(u_p)$
  - Cut off u_p's children to turn u_p into a **leaf node**

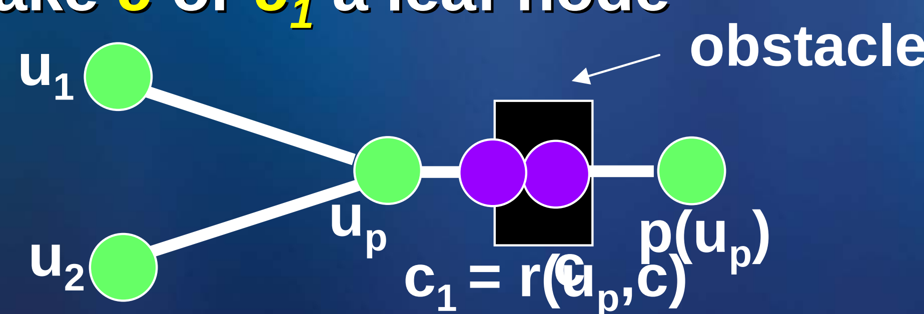


Case 2: $totallen(u_p) + w(u_p) \leq L_{max}$

- If u_p has a parent
  - If $totallen(u_p) + w(u_p) + l(u_p, p(u_p)) \leq L_{max}$
    - If $u_p \in V_N$, assign $w(u_p) \leftarrow w(u_p) + totallen(u_p)$
    - Cut u_p's children from the tree, make u_p a leaf node
  - Else insert the cutting node to make $totallen(u_p) + w(u_p) + l(u_p, c) = L_{max}$
    - If $f(c) = 1$, use $c_1 = r(u_p,c)$ to replace c
    - Reduce the tree to make c or c_1 a leaf node



$w(u_p) \leftarrow w(u_p) + totallen(u_p)$

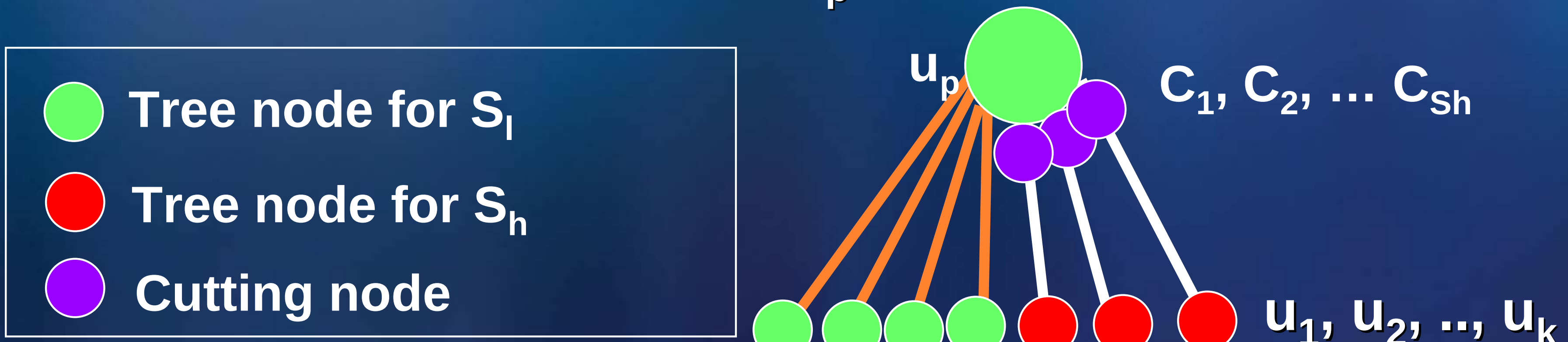


$totallen(u_p) + w(u_p) + l(u_p, c) = L_{max}$

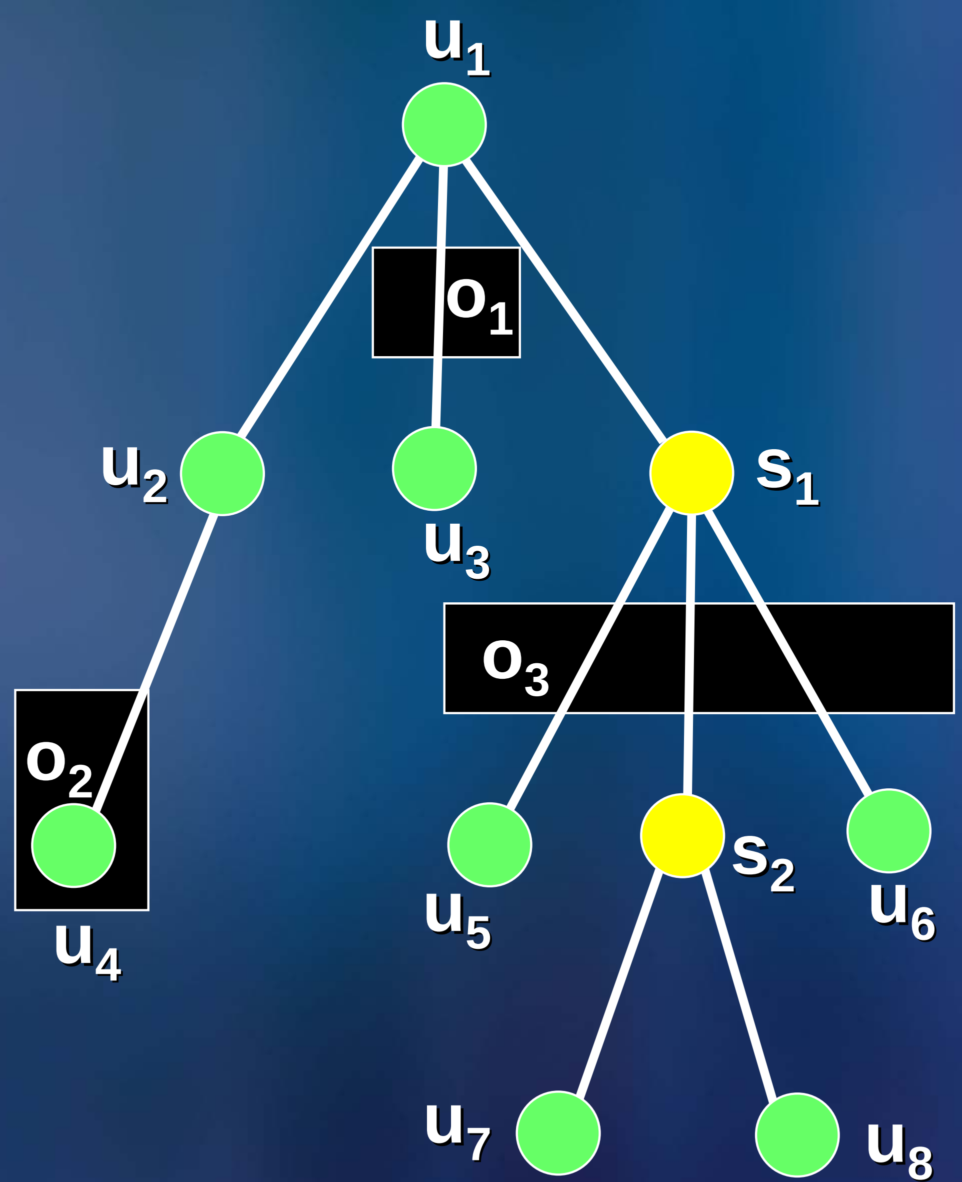
Tree node Cutting node

Case 3: $totallen(u_p) + w(u_p) > L_{max}$

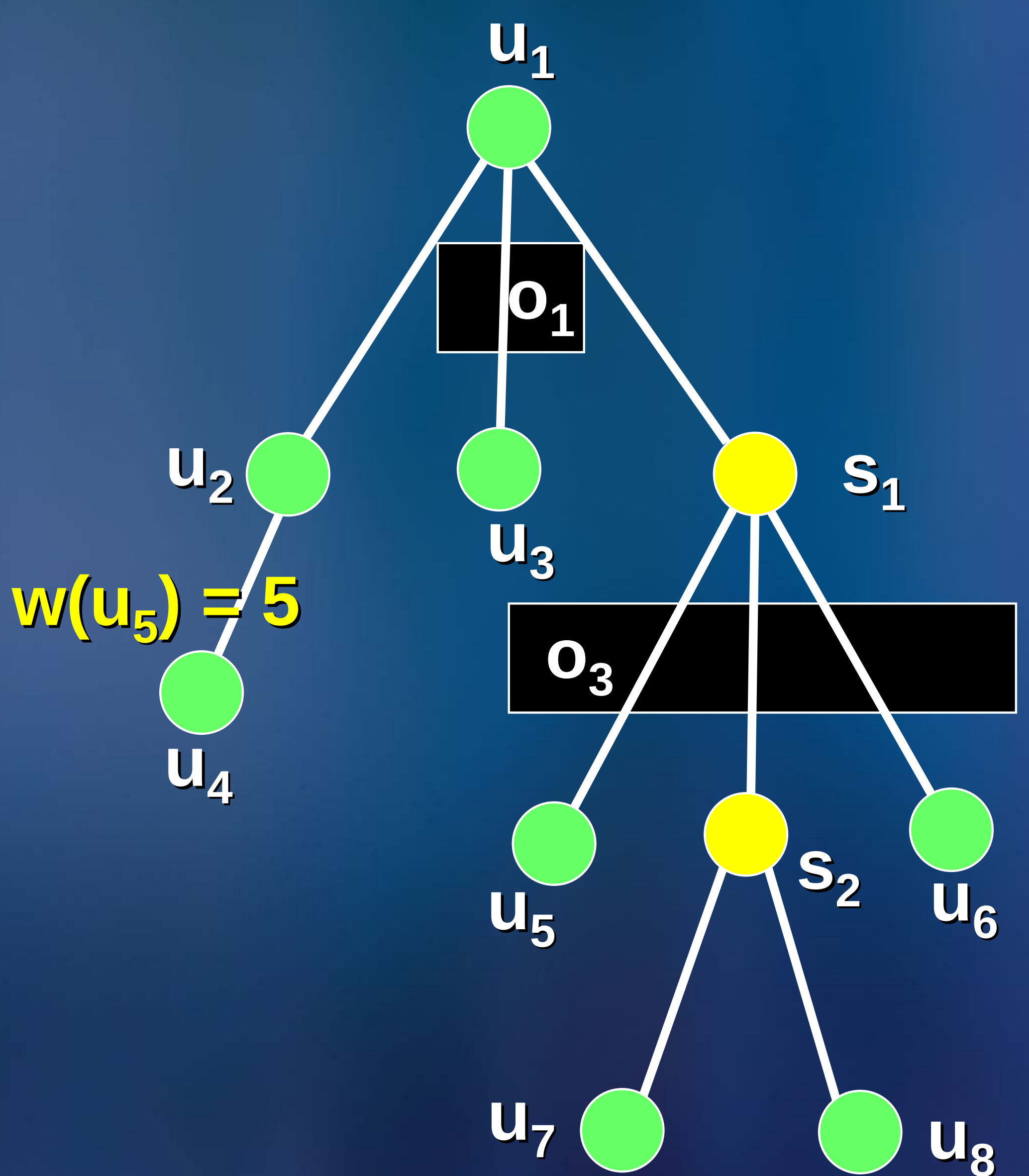
- Case 3: $totallen(u_p) + w(u_p) > L_{max}$
  - Step 1: $S = \cup_{i=1}^{k}\{l(e(u_i,u_p)) + w(u_i)\}$
  - Step 2: Find $S = S_l \cup S_h$, $S_l \cap S_h = \varphi$, such that
    - For any $a \in S_l$ and $b \in S_h$, $a \le b$
    - $\sum_{s \in S_l} s \le L_{max} - w(u_p)$
    - For any $b \in S_h$, $\sum_{s \in S_l} s + b > L_{max} - w(u_p)$
  - Step 3: Add cutting nodes $c_1, \ldots, c_{Sh}$ on every $s \in S_h$
  - Step 4: Use Case 2 to cut u_p into a leaf node



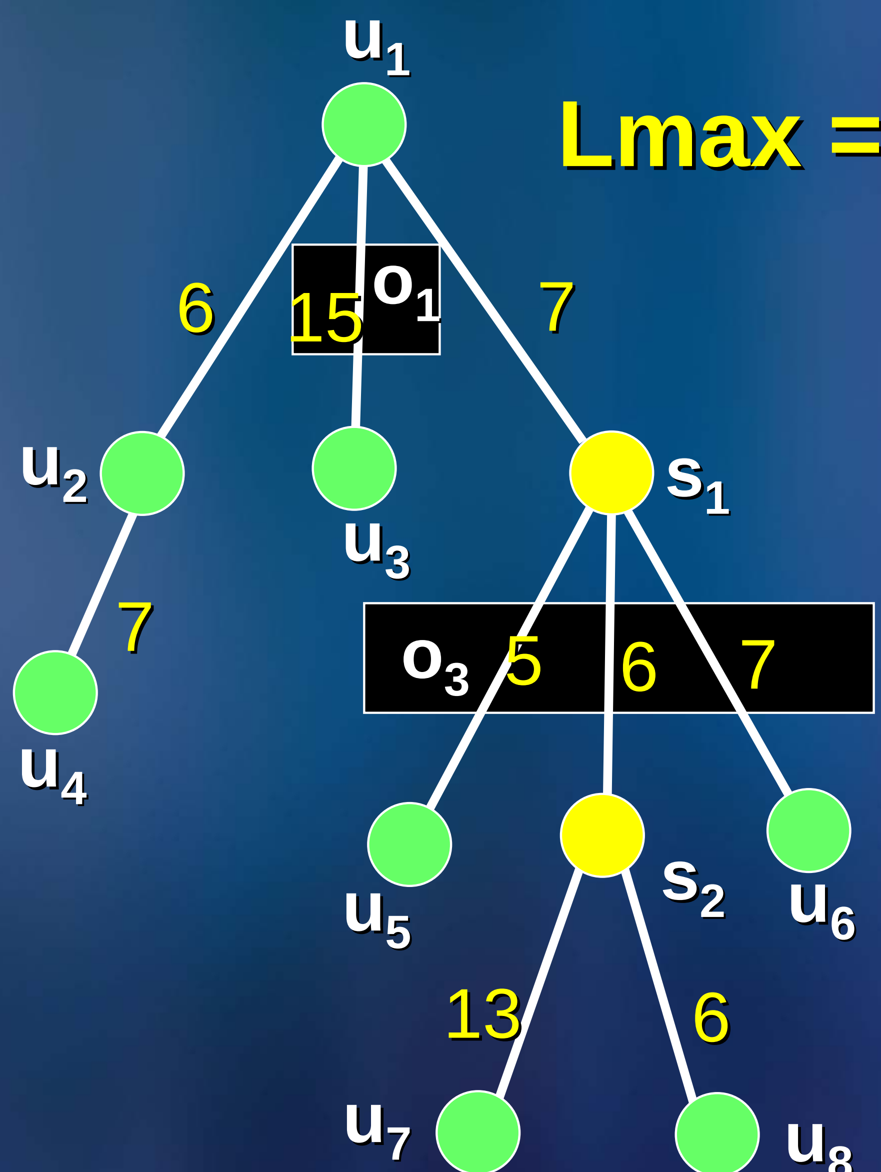
An Example



An Example

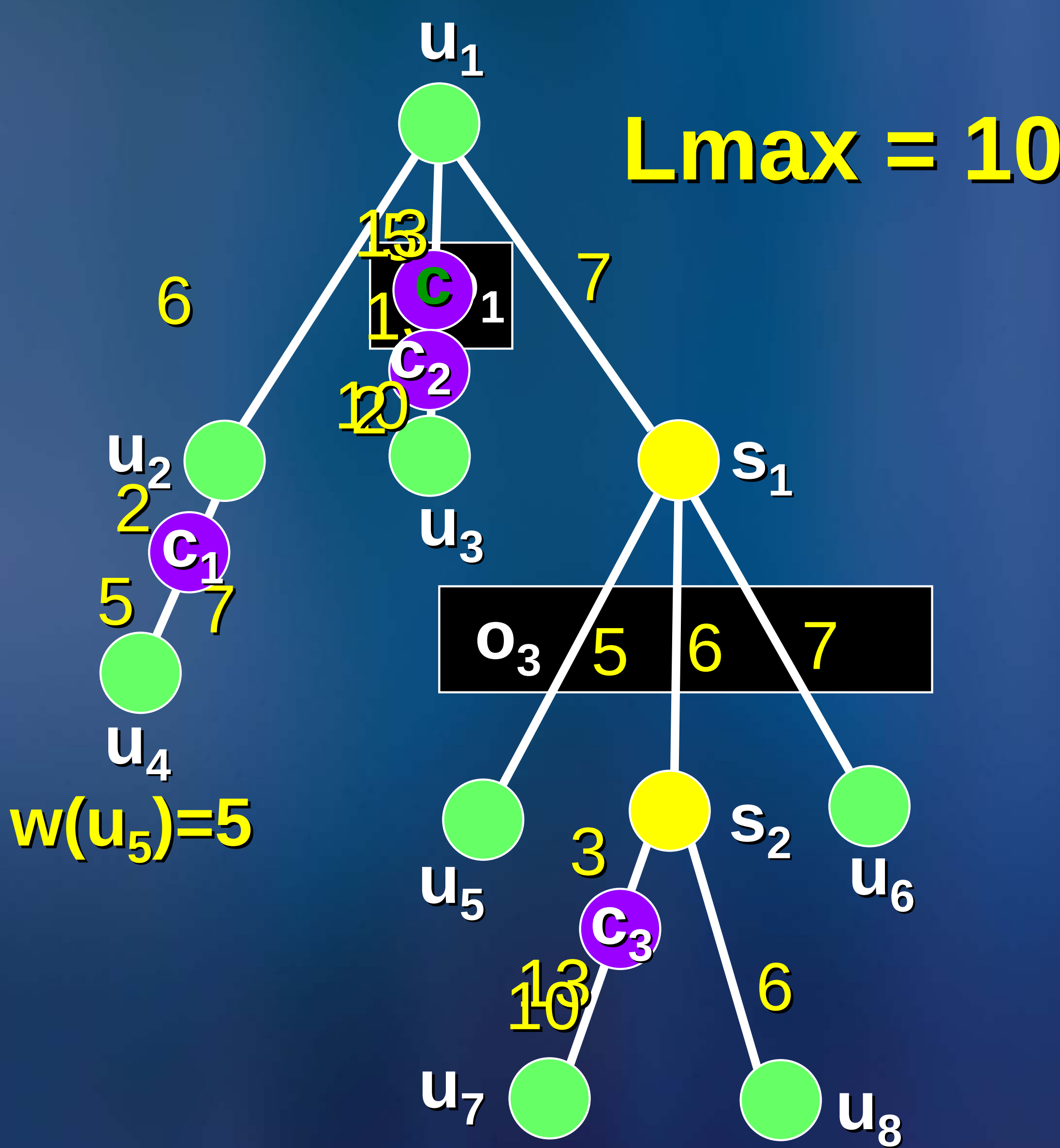


An Example

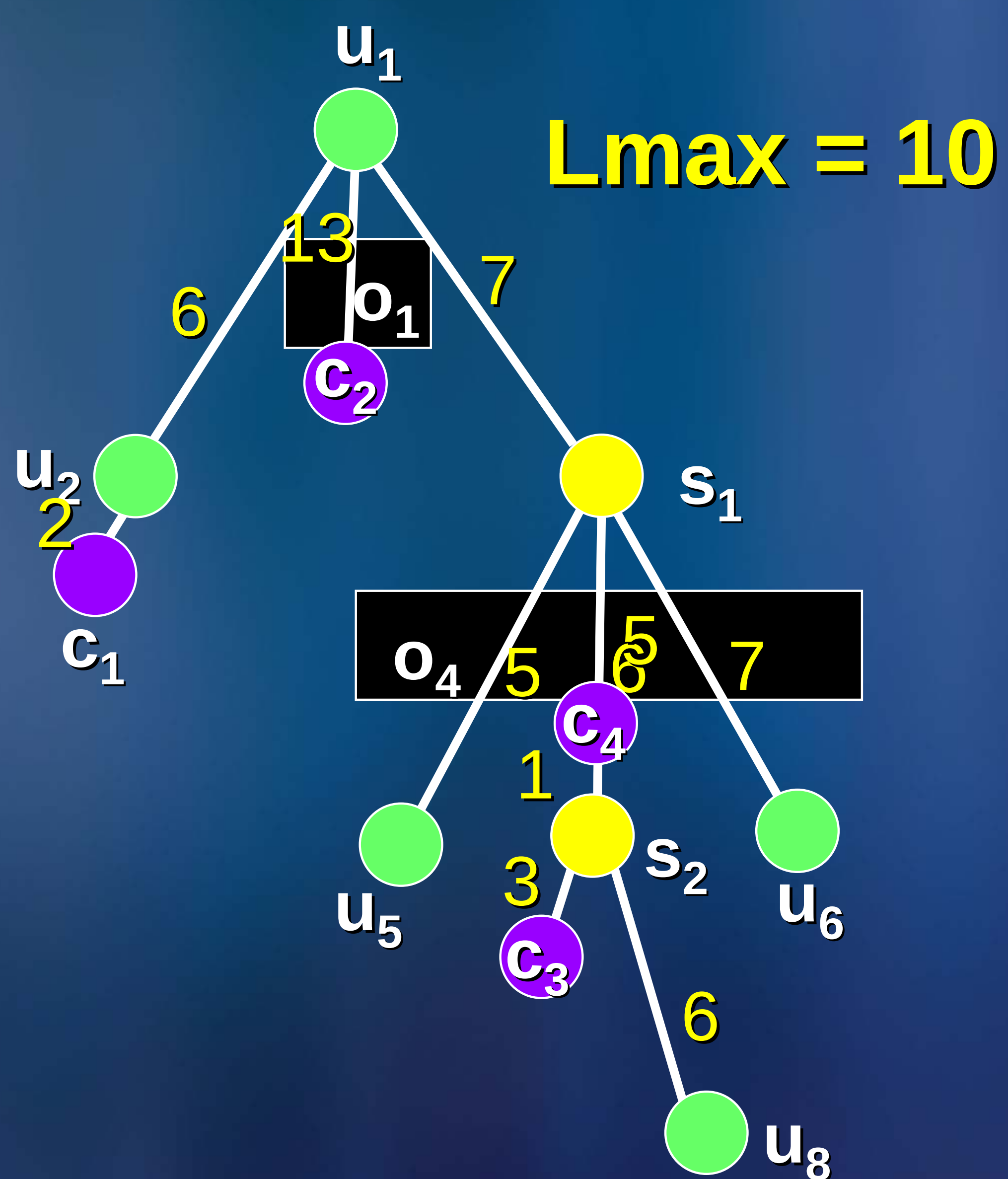


Lmax = 10

An Example

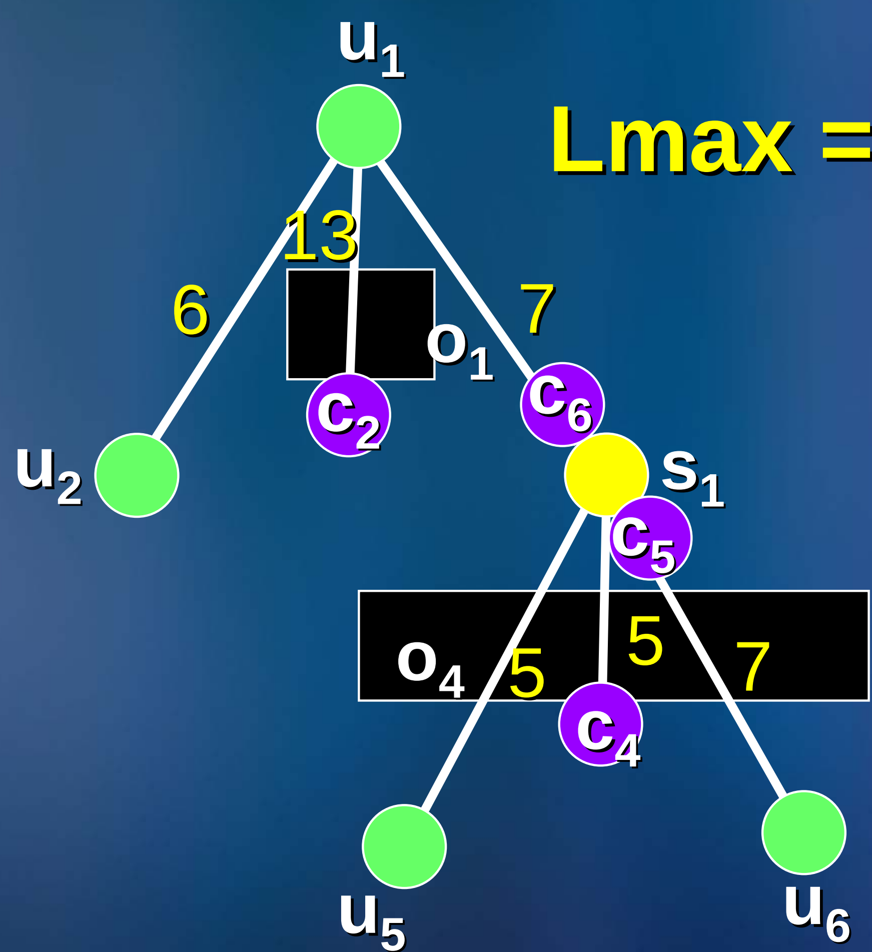


An Example

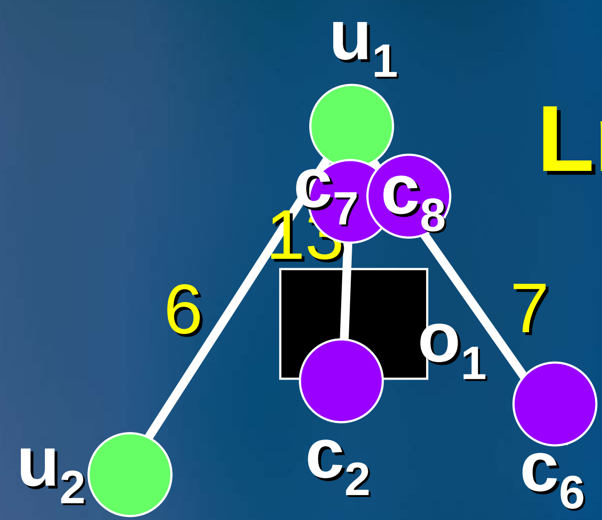


An Example

Lmax = 10

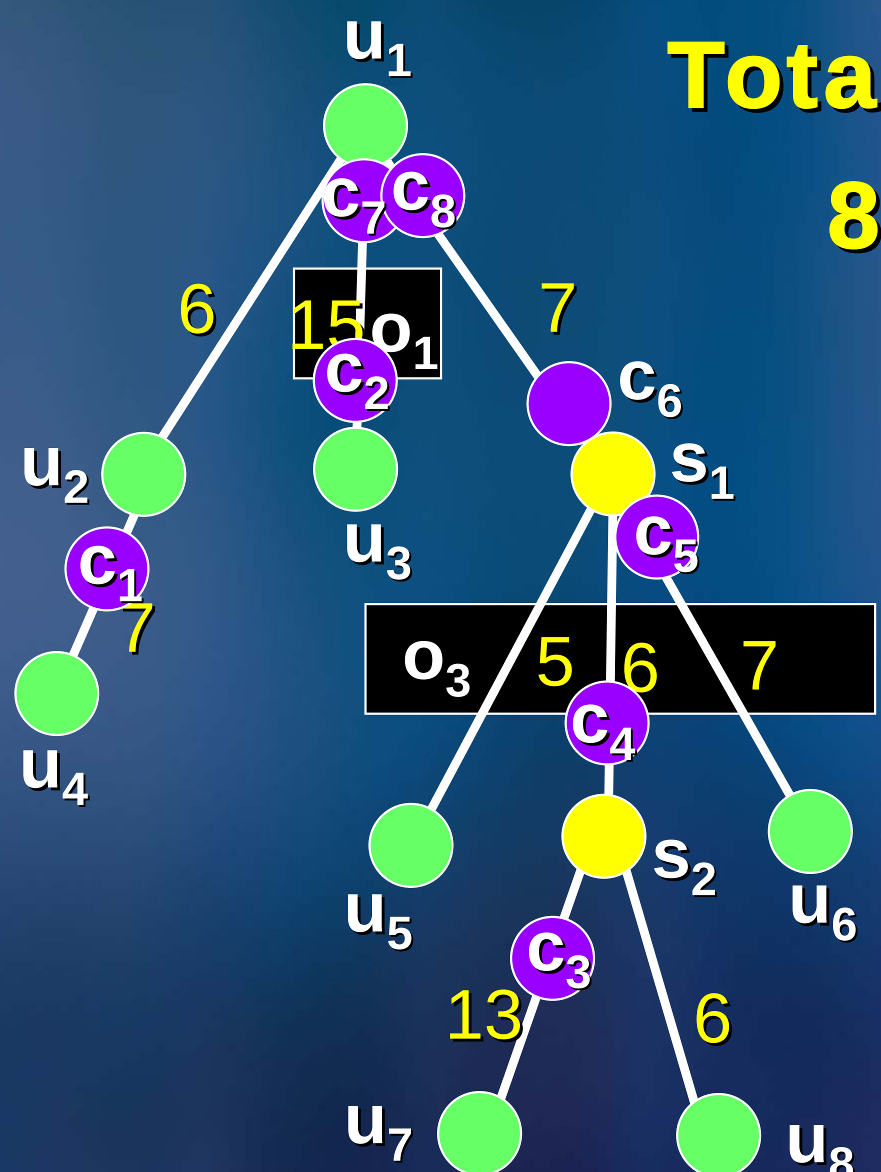


An Example



Lmax = 10

An Example



Totally we insert 8 jumpers on this Steiner tree!

Outline

- Introduction to Antenna Effect
- Problem Definition
- An Optimal Algorithm for Jumper Insertion
- Complexity Analysis
- Experimental Results
- Conclusions

Time & Space Complexity

- The tree-cutting problem JIROA exhibits the properties of optimal substructures and greedy choices.
- Algorithm BUJIO optimally solves the JIROA problem in O((V + D) lg D) time using O(V) space
  - V: # of tree nodes
  - D: # of obstacles

Outline

- Introduction to Antenna Effect
- Problem Definition
- An Optimal Algorithm for Jumper Insertion
- Complexity Analysis
- Experimental Results
- Conclusions

Experimental Settings

- Implemented BUJIO in C++ on a 2.4 GHz Intel Pentium PC with 256 MB memory in Windows XP
- Generated gate terminals on 1 cm x 1 cm planes with a number of nodes and obstacles
- Compared with ISPD-04, ISPD-05, DAC-05 works, extending their works to handle obstacles using the same obstacle processing in this paper
- Our BUJIO and the ISPD-05 algorithms work on Steiner trees while the ISPD-04 and DAC-05 ones on spanning trees.

Experiment #1: #Jumpers for Random Cases

- Random cases (10,000 nodes, 500 obstacles)
- Quality rating: BUJIO > ISPD-05 > DAC-05 > ISPD-04
- Runtimes are about the same for all methods (0.15 sec)

L_{max} (um)	BUJIO	ISPD-05		ISPD-04		DAC-05	
	#J	#J	%More	#J	%More	#J	%More
220	1533	1537	+0.3%	2359	+53.9%	1806	+17.8%
230	1341	1348	+0.5%	2020	+50.6%	1607	+19.8%
240	1144	1149	+0.4%	1742	+52.3%	1378	+20.5%
250	961	966	+0.5%	1501	+56.2%	1210	+26.0%
260	823	827	+0.5%	1279	+55.4%	1050	+27.6%
270	706	708	+0.3%	1109	+57.1%	919	+30.2%
280	599	601	+0.2%	928	+55.0%	779	+30.0%
290	517	518	+0.2%	767	+48.4%	649	+25.5%
300	434	435	+0.2%	645	+48.6%	562	+29.5%

+0.3% +52% +25%

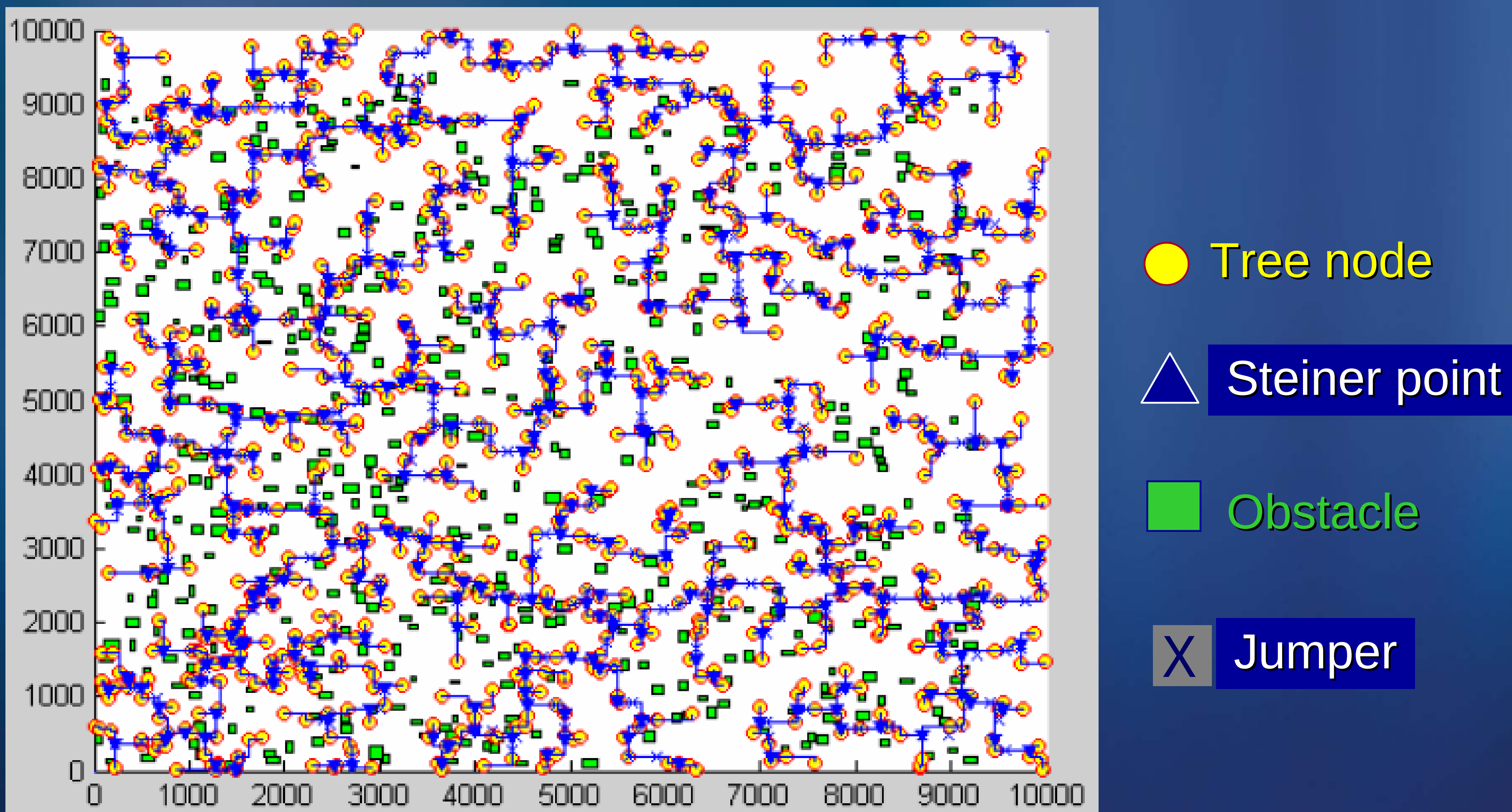
Experiment #2: #Jumpers for Synthetic Cases

- Synthetic cases: make gate terminals adjacent to each other
- Quality rating: BUJIO > DAC-05 > ISPD-05 > ISPD-04

L_{max} (um)	BUJIO	ISPD-05		ISPD-04		DAC-05	
	#J	#J	%More	#J	%More	#J	%More
220	4272	5455	+27.6%	5943	+39.1%	4557	+6.6%
230	4157	5311	+27.8%	5762	+38.6%	4421	+6.4%
240	4056	5174	+27.6%	5604	+38.2%	4305	+6.1%
250	3972	5061	+27.4%	5455	+37.3%	4187	+5.4%
260	3889	4951	+27.3%	5299	+36.3%	4076	+4.8%
270	3814	4844	+27.0%	5157	+35.2%	3983	+4.4%
280	3728	4738	+27.1%	5028	+34.9%	3895	+4.5%
290	3657	4637	+26.8%	4915	+34.4%	3809	+4.2%
300	3605	4541	+26.0%	4796	+33.0%	3728	+3.4%
			+27%		+36%		+5%

Resulting Layout

- 1000 nodes, 500 obstacles, 426 jumpers, L_{max} = 500 um



Outline

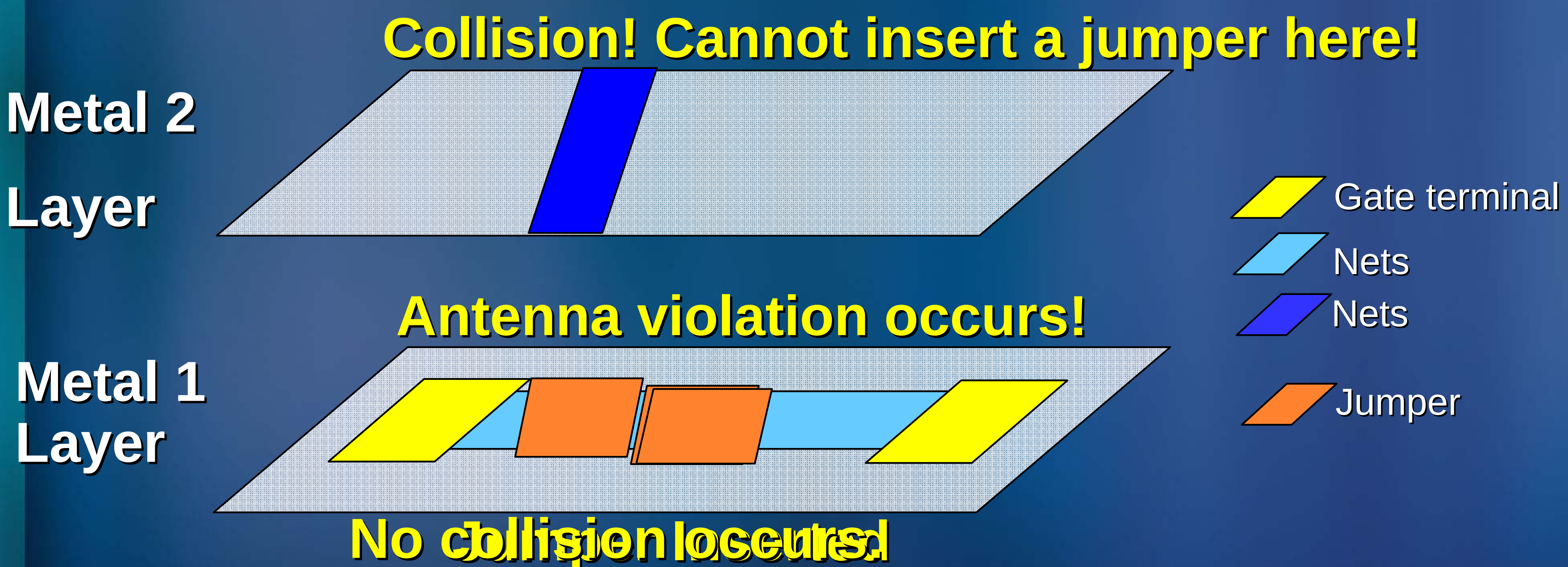
- Introduction to Antenna Effect
- Problem Definition
- An Optimal Algorithm for Jumper Insertion
- Complexity Analysis
- Experimental Results
- Conclusions

Conclusions

- We have presented **the first optimal** algorithm to solve the JIROA problem in **O((V+D) lg D) time** and **O(V) space**, where V is the number of vertices and D is the number of obstacles.
- Experimental results have shown that our algorithm leads to more robust and much better solutions than the previous works.
- Our work can be applied to any routing trees, and thus readily be incorporated into a router for antenna avoidance or a post-layout optimizer for antenna fixing.

Thank You!!

Obstacle Example



We can insert a jumper here.