Optimal Post-Routing Redundant Via Insertion

Kuang-Yao Lee[†], Cheng-Kok Koh[‡],
Ting-Chi Wang[†], and Kai-Yuan Chao[#]

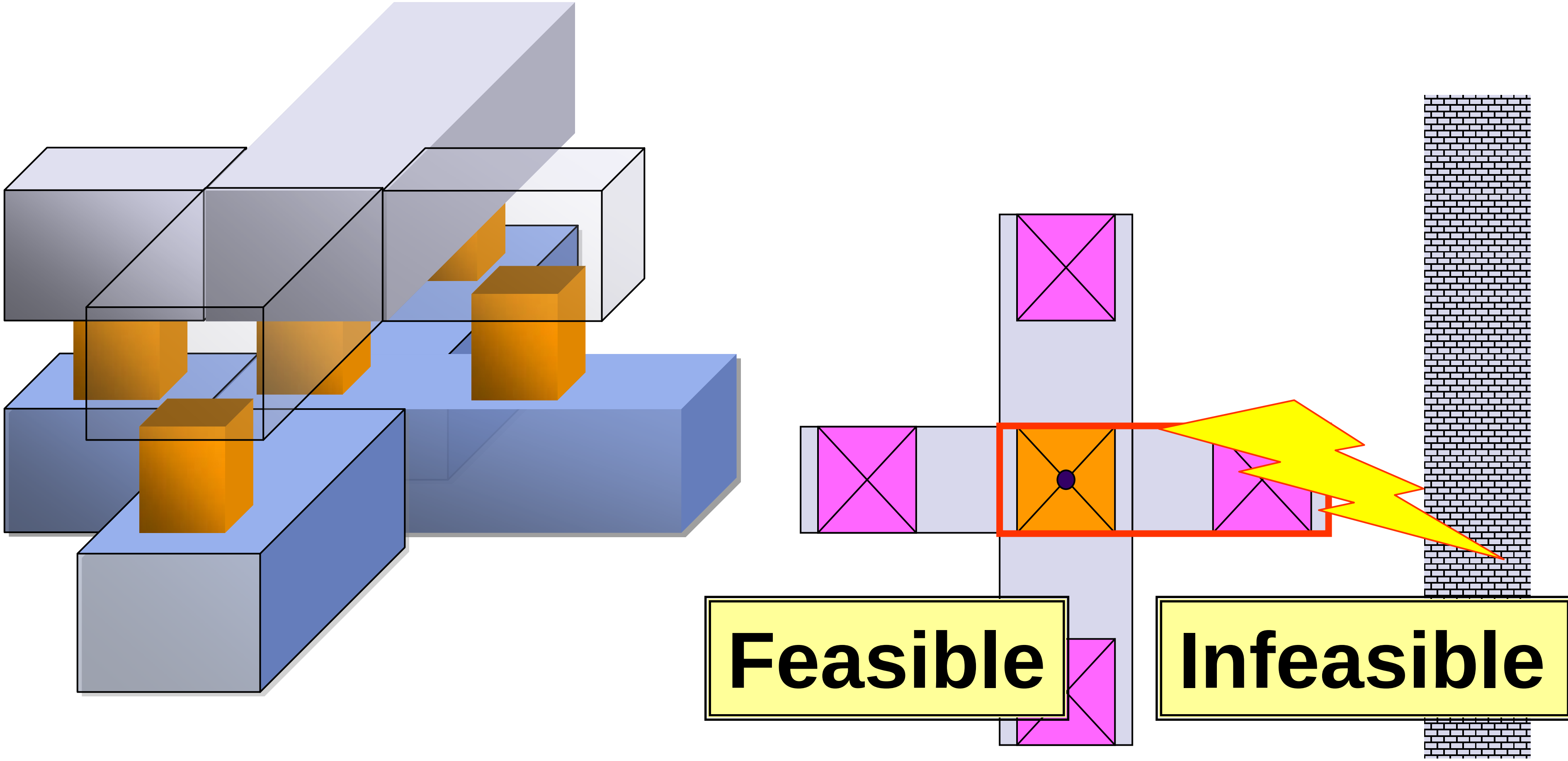
[†]Dept. of CS, National Tsing Hua University
[‡]School of ECE, Purdue University
[#]Intel Corporation

Outline

- Double-cut via insertion (DVI) problem
  - Previous works
  - 0-1 integer linear program (0-1 ILP)
  - Speed-up approaches
- Via density constraint
  - Previous works
  - Extended 0-1 ILP
- Experimental results
- Conclusions

Double-cut via

- Improve yield and reliability



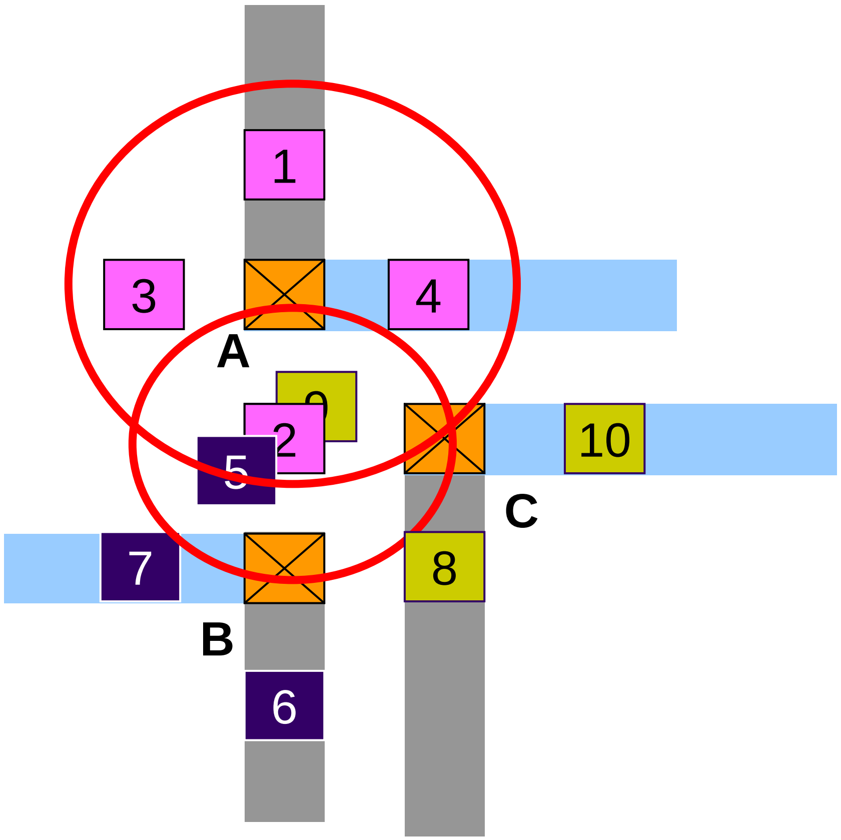
Double-cut via insertion (DVI) problem

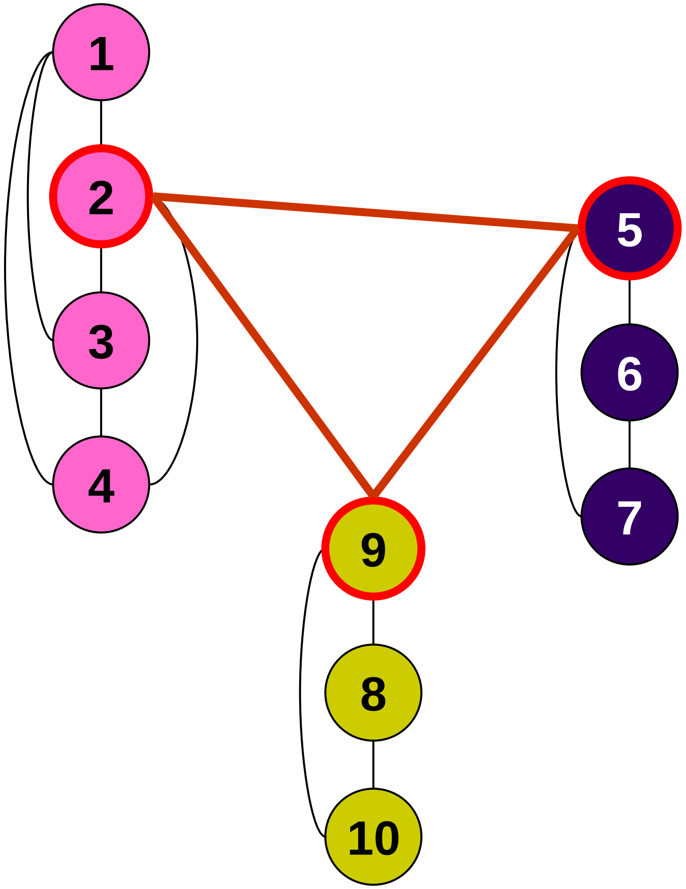
- **Input**
  - A routed design and a set of via-related design rules
- **Goal**
  - To replace as many single vias with double-cut vias as possible
- **Constraints**
  - Do not re-route any net
  - Each single via either remains unchanged or is replaced by a double-cut via
  - After replacement, no design rule is violated

Previous works

- **[Luo et al., ASP-DAC'06]**
  - Allow changes to the routing result
  - Single vias are considered one by one for redundant via insertion
  - Local optimal
- **[Lee et al., ASP-DAC'06]**
  - Consider all single vias simultaneously
  - Formulated as a maximum independent set (MIS) problem
  - NP-C problem
- **[Chen et al., DAC'06]**
  - Extra insertion constraint on stacked vias
  - Formulated as a maximum bipartite matching problem
  - The optimal solution for the grid-based design that involves at most three routing layers

Conflict graph & MIS





[Lee et al., ASP-DAC’06]

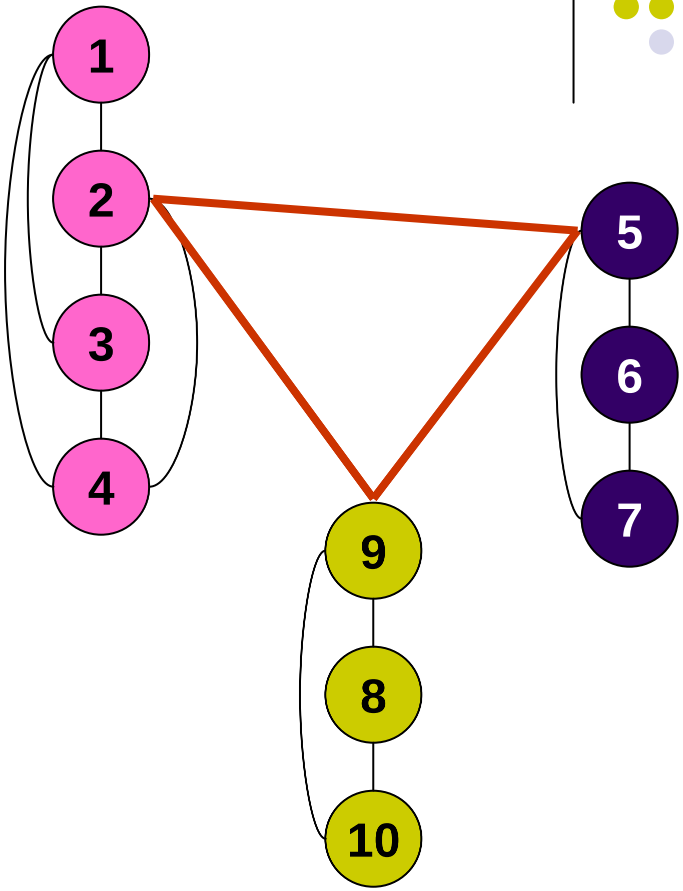
0-1 ILP formulation

Maximize: $$\sum_{1 \le i \le 8} R_i$$

Subject to:

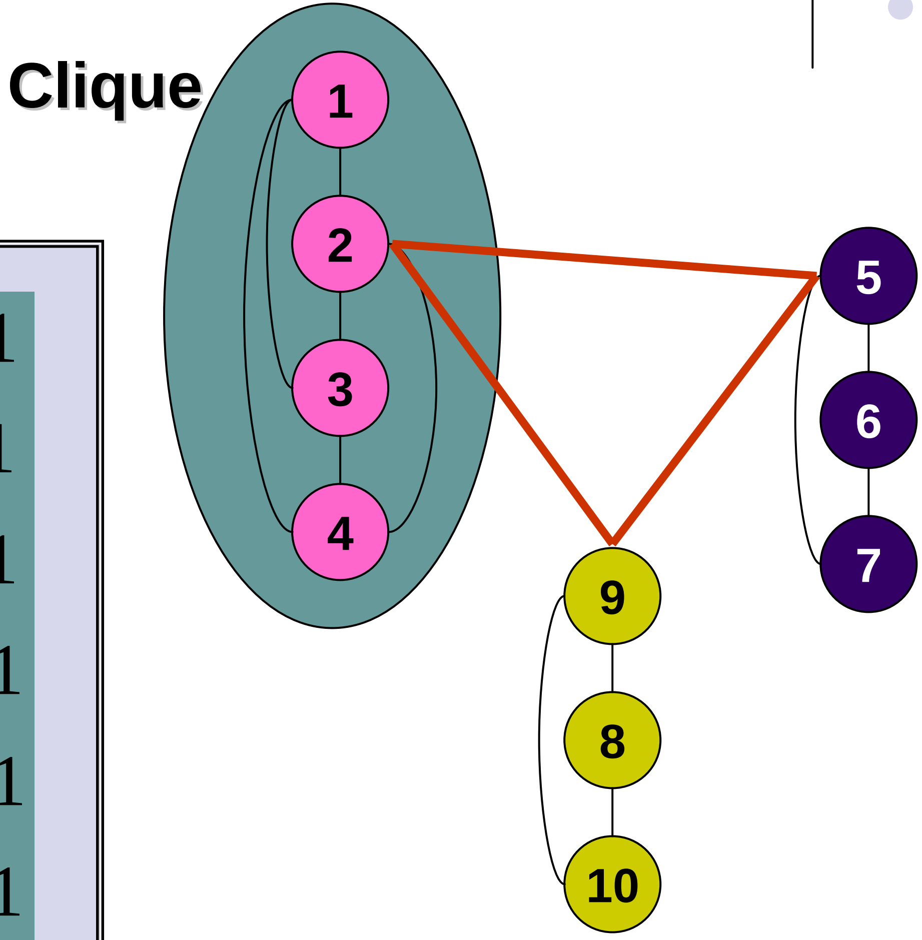
$$R_2 + R_5 \le 1$$
$$R_2 + R_9 \le 1$$
$$R_5 + R_9 \le 1$$

$$R_1 + R_2 \le 1$$
$$R_1 + R_3 \le 1$$
$$R_1 + R_4 \le 1$$
$$R_2 + R_3 \le 1$$
$$R_2 + R_4 \le 1$$
$$R_3 + R_4 \le 1$$
$$\dots$$



$$R_i \in \{0,1\}$$

Speed-up – Reduction in constraints



$$R_2 + R_5 \le 1$$

$$R_2 + R_9 \le 1$$

$$R_5 + R_9 \le 1$$

$$R_1 + R_2 \le 1$$

$$R_1 + R_3 \le 1$$

$$R_1 + R_4 \le 1$$

$$R_2 + R_3 \le 1$$

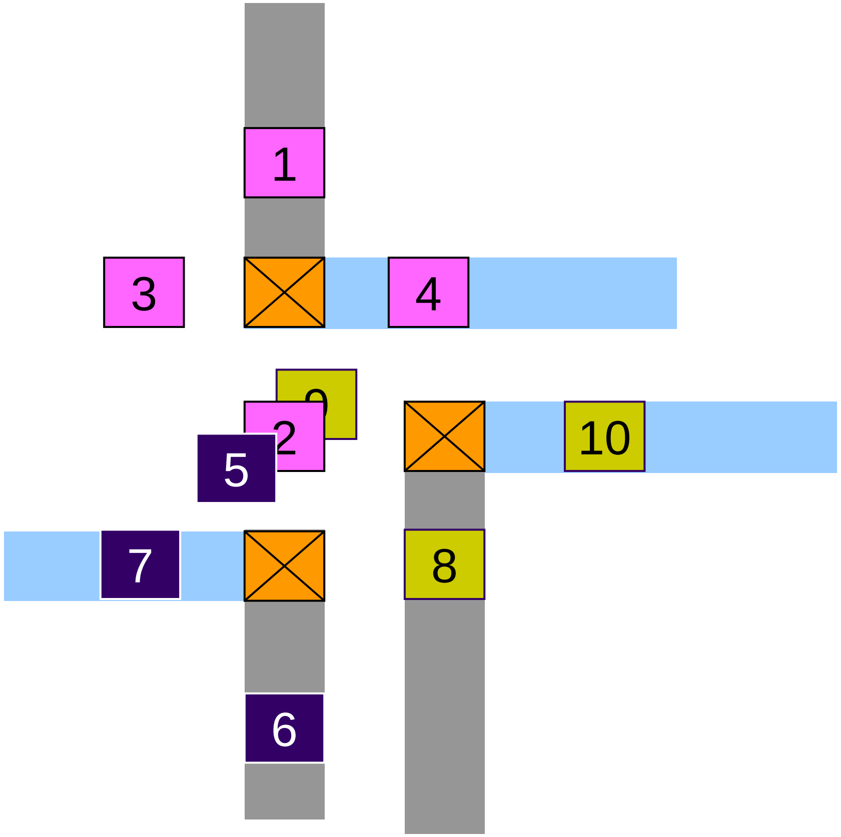
$$R_2 + R_4 \le 1$$

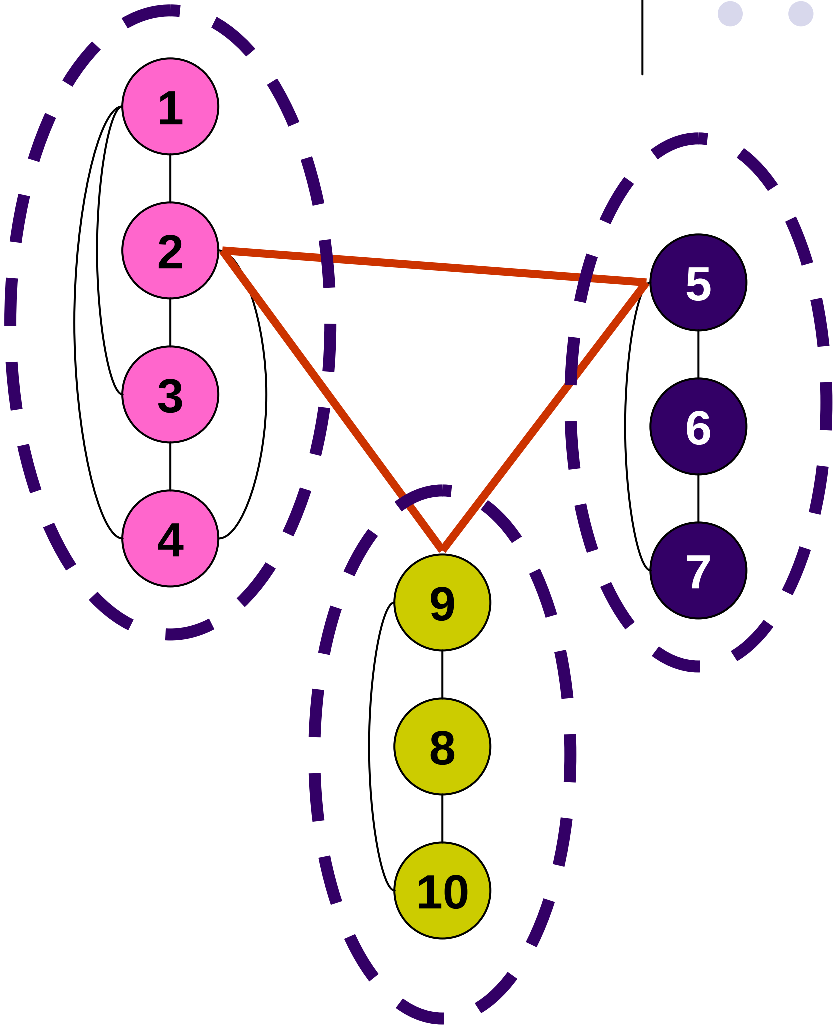
$$R_3 + R_4 \le 1$$

…

$$R_1 + R_2 + R_3 + R_4 \le 1$$

Speed-up –
Reduction in constraints

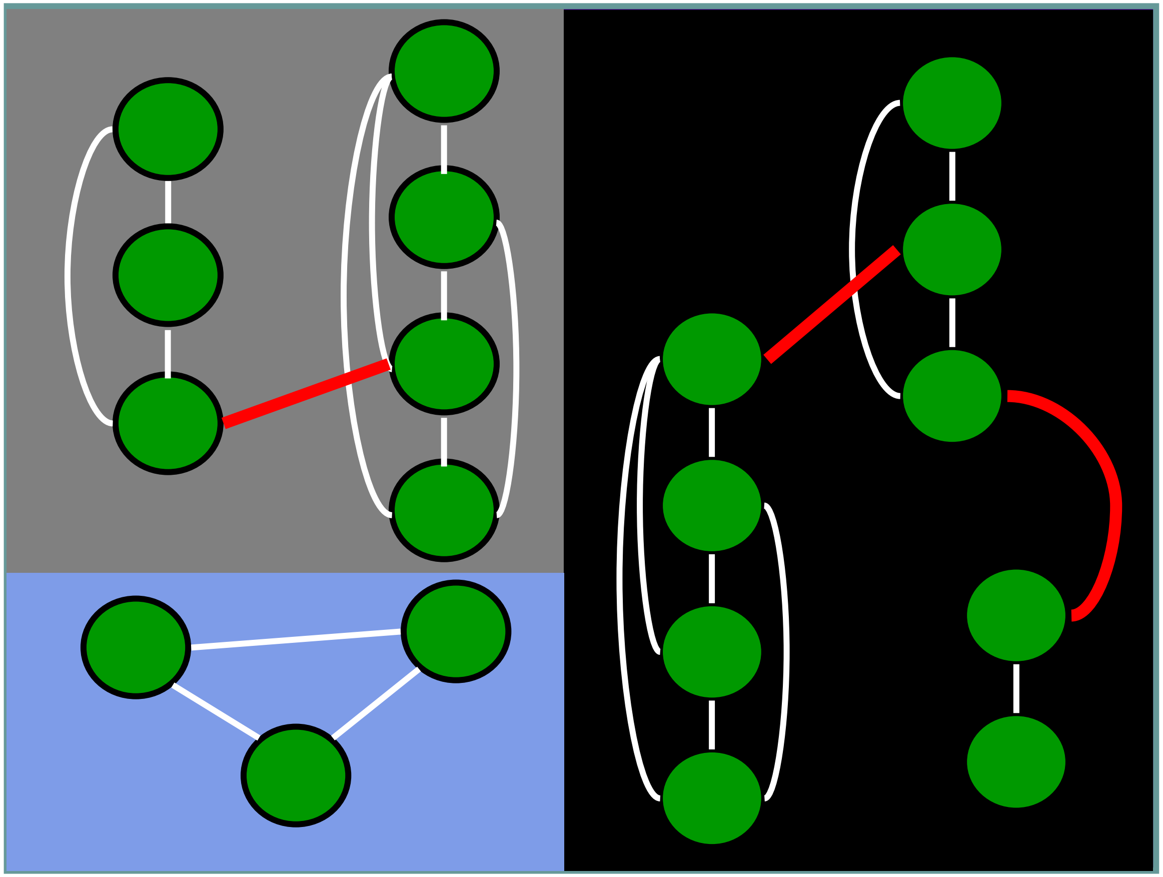




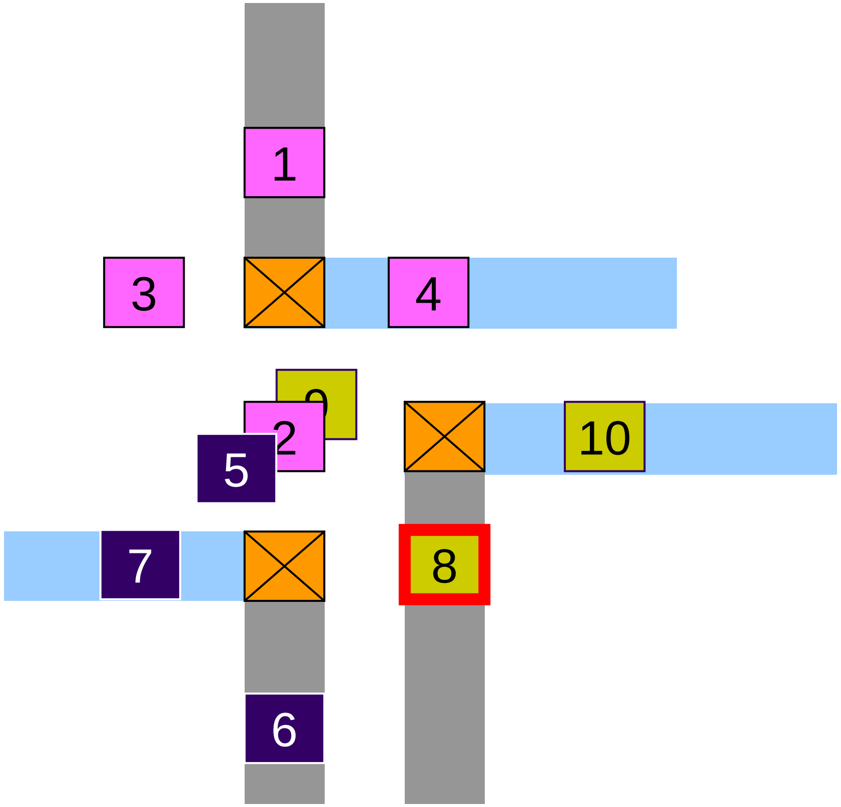
- # of constraints: 15 → 6

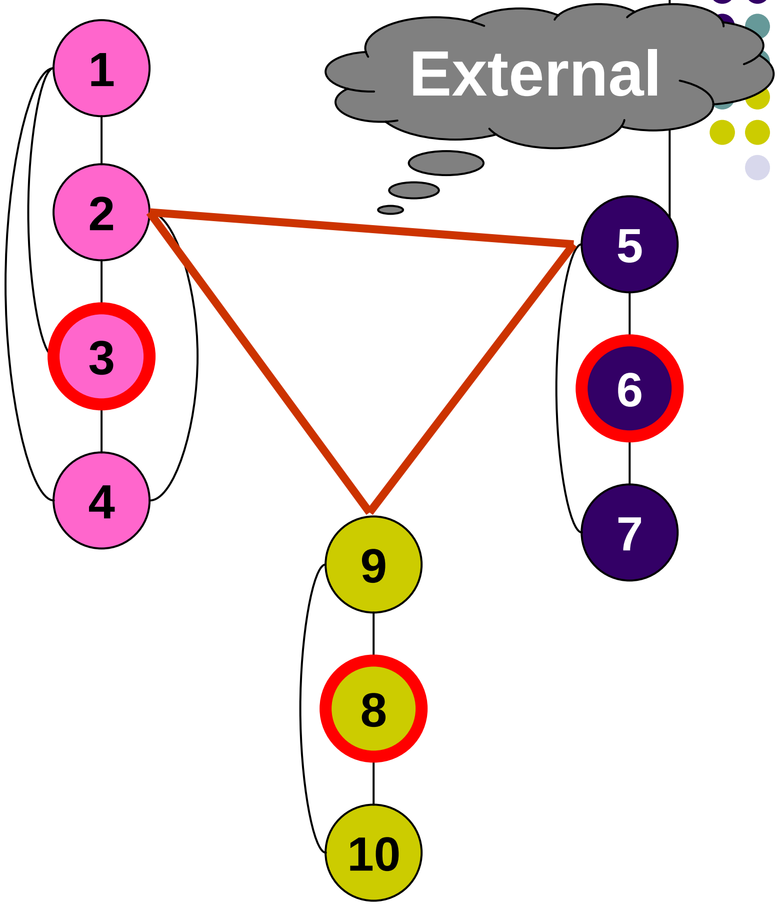
Speed-up – Connected components

- Divide into several smaller 0-1 ILP problems
  - Compute connected components (by DFS)



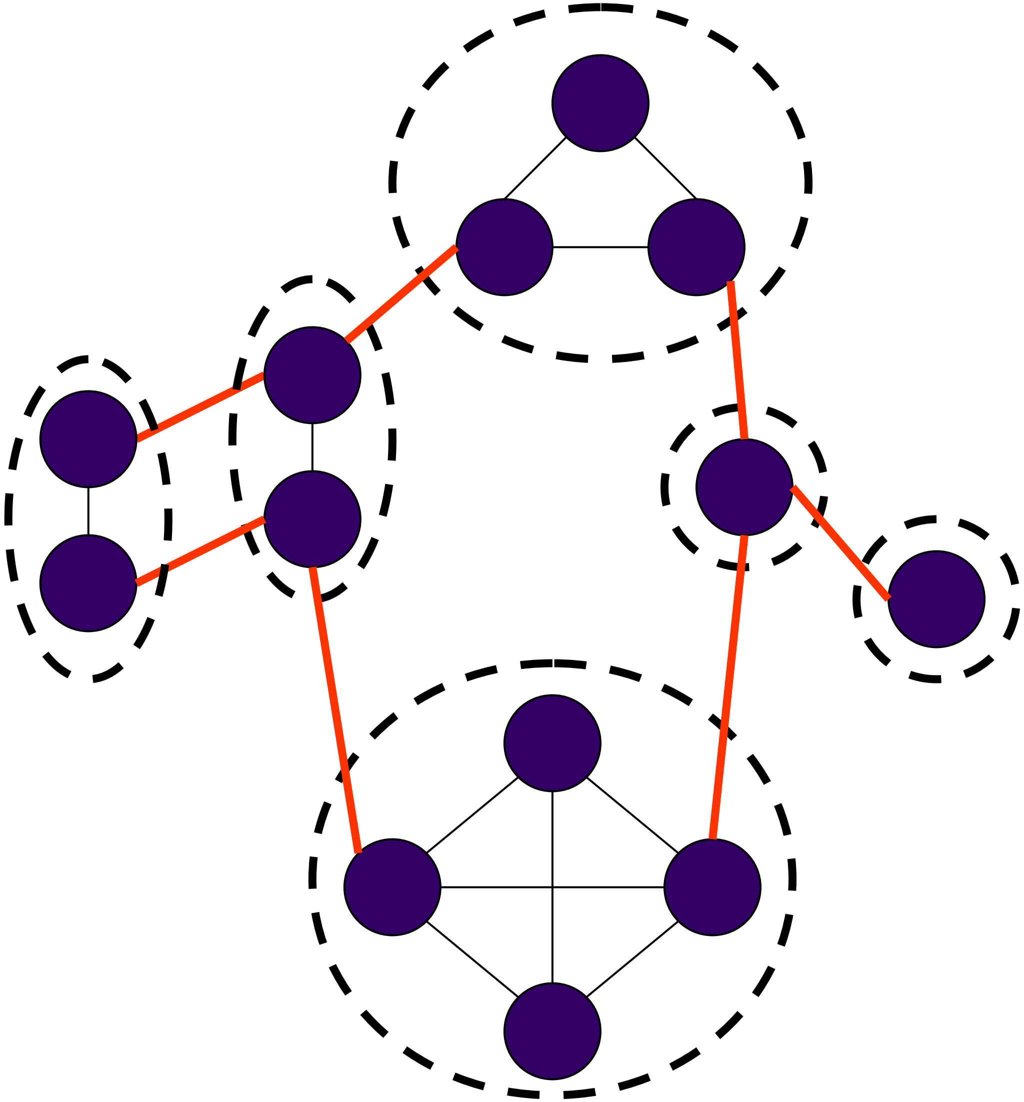
Speed-up – Pre-selection



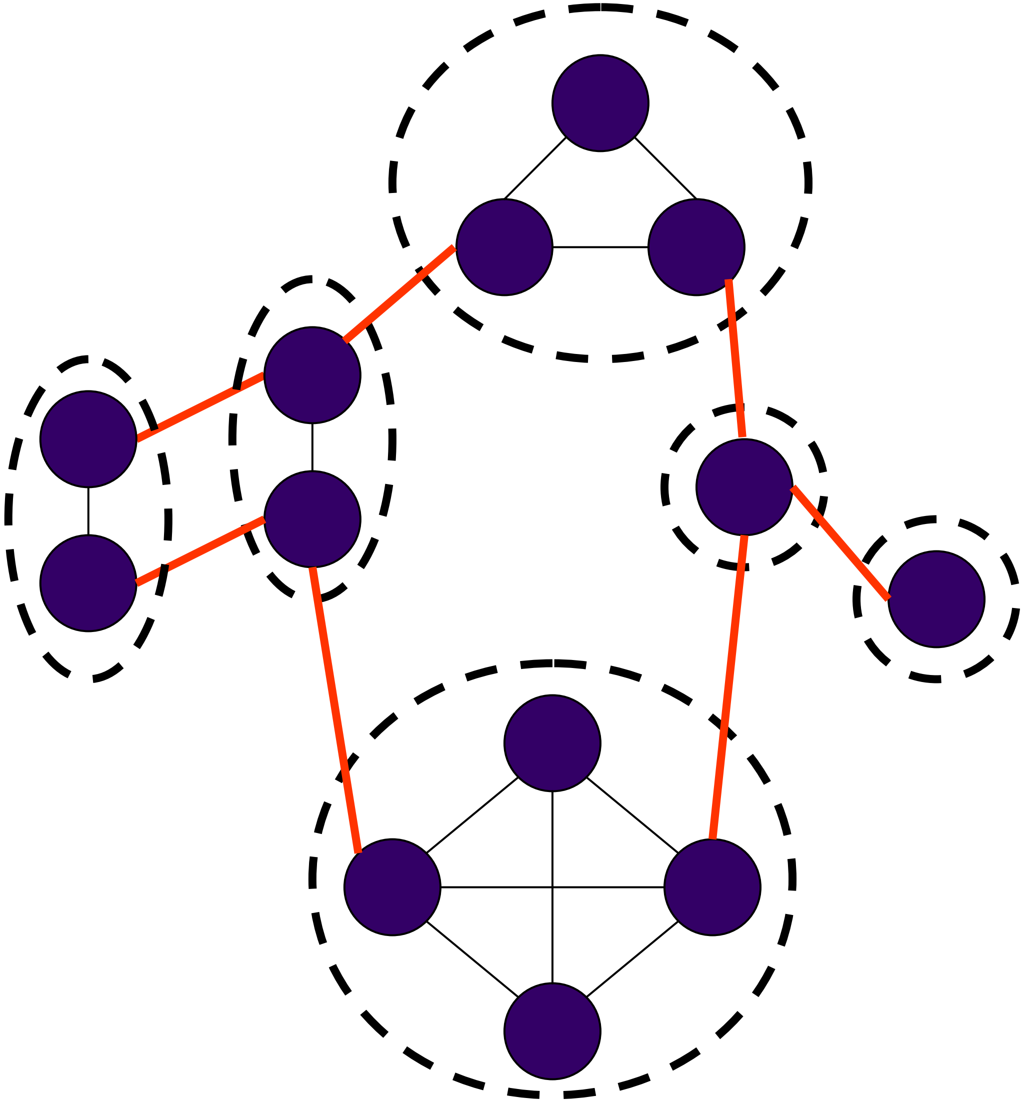


- The size of given conflict graph can be reduced
- Could even solve the whole DVI problem directly
- By a BFS-like approach we can efficiently pre-select the maximum # of vertices

Overall approach

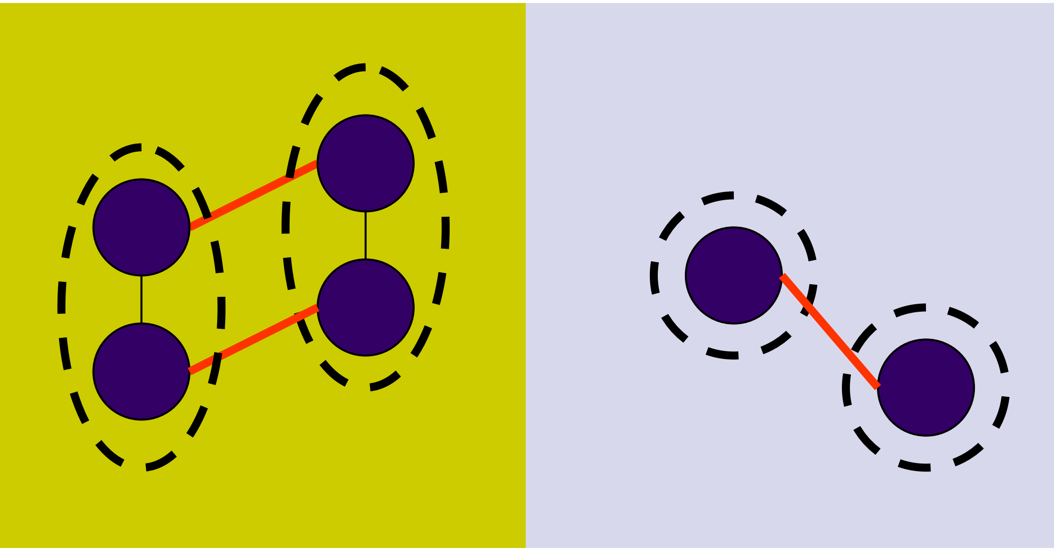


Overall approach



1. Pre-selection

Overall approach



1.Pre-selection

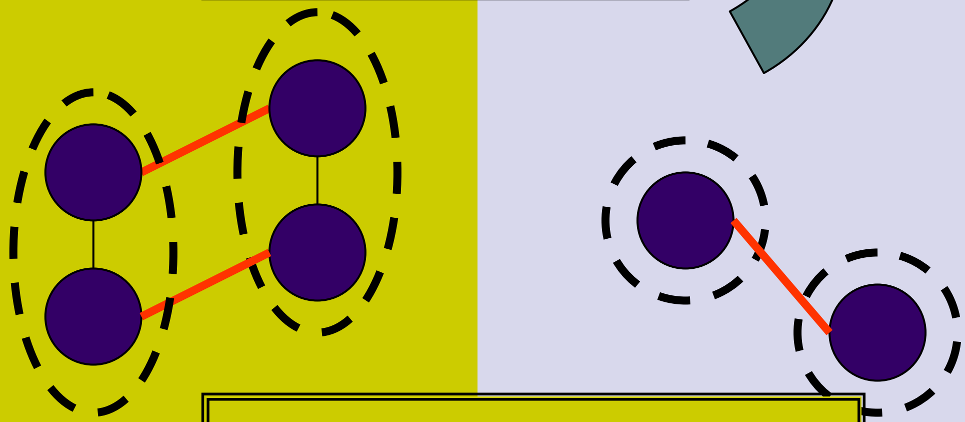
2.Connected Components

Overall approach

Max $\sum_{1 \le i \le 2} R_i$

Subject to:

$R_1 + R_2 \le 1$



Max $\sum_{1 \le i \le 4} R_i$

Subject to:

$R_1 + R_2 \le 1 \quad R_1 + R_3 \le 1$

$R_3 + R_4 \le 1 \quad R_2 + R_4 \le 1$

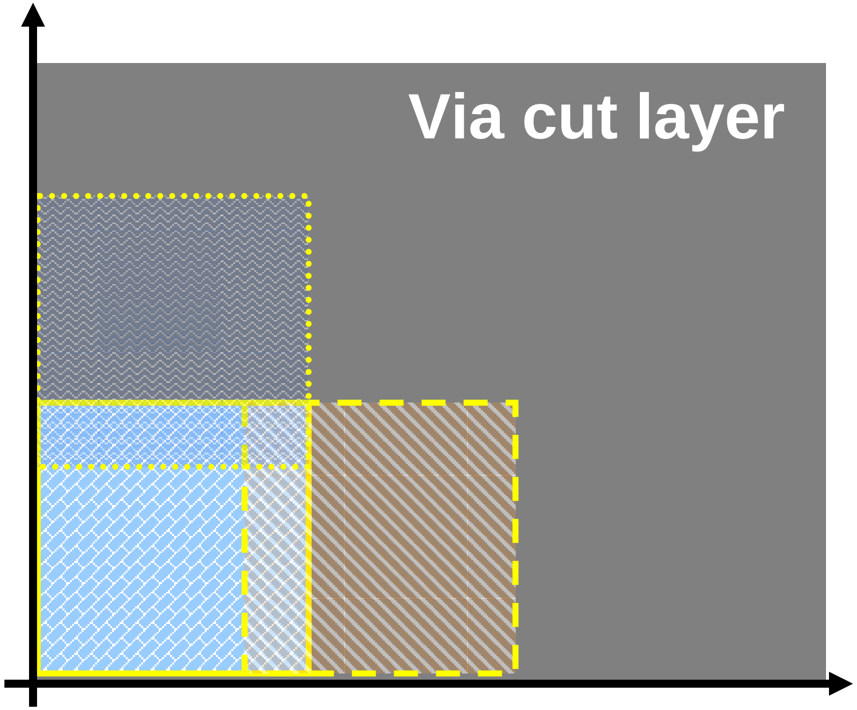
1. Pre-selection
2. Connected Components
3. Reduced 0-1 ILP

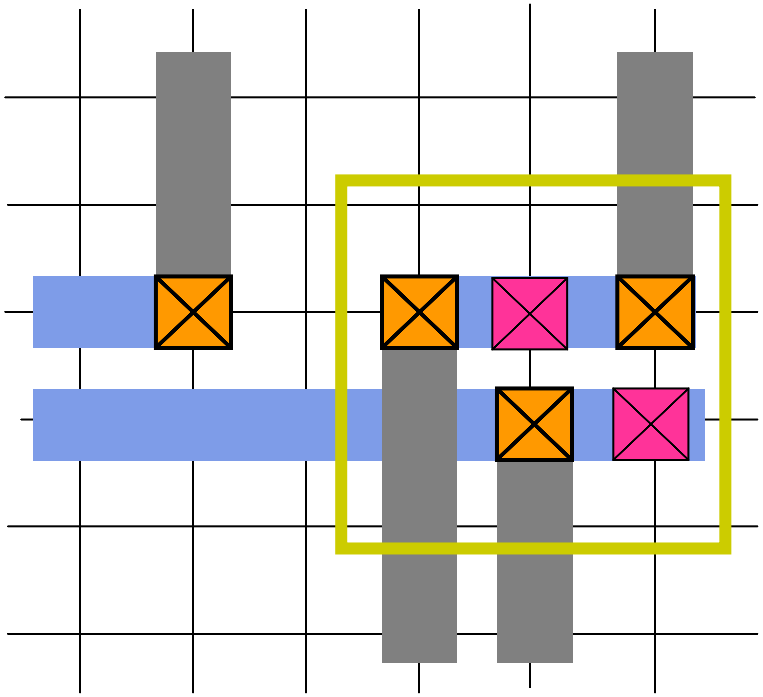
Outline

- Double-cut via insertion (DVI) problem
  - Previous works
  - 0-1 integer linear program (0-1 ILP)
  - Speed-up approaches
- Via density constraint
  - Previous works
  - Extended 0-1 ILP
- Experimental results
- Conclusions

Via density rules

- Minimum via density rule
- Maximum via density rule





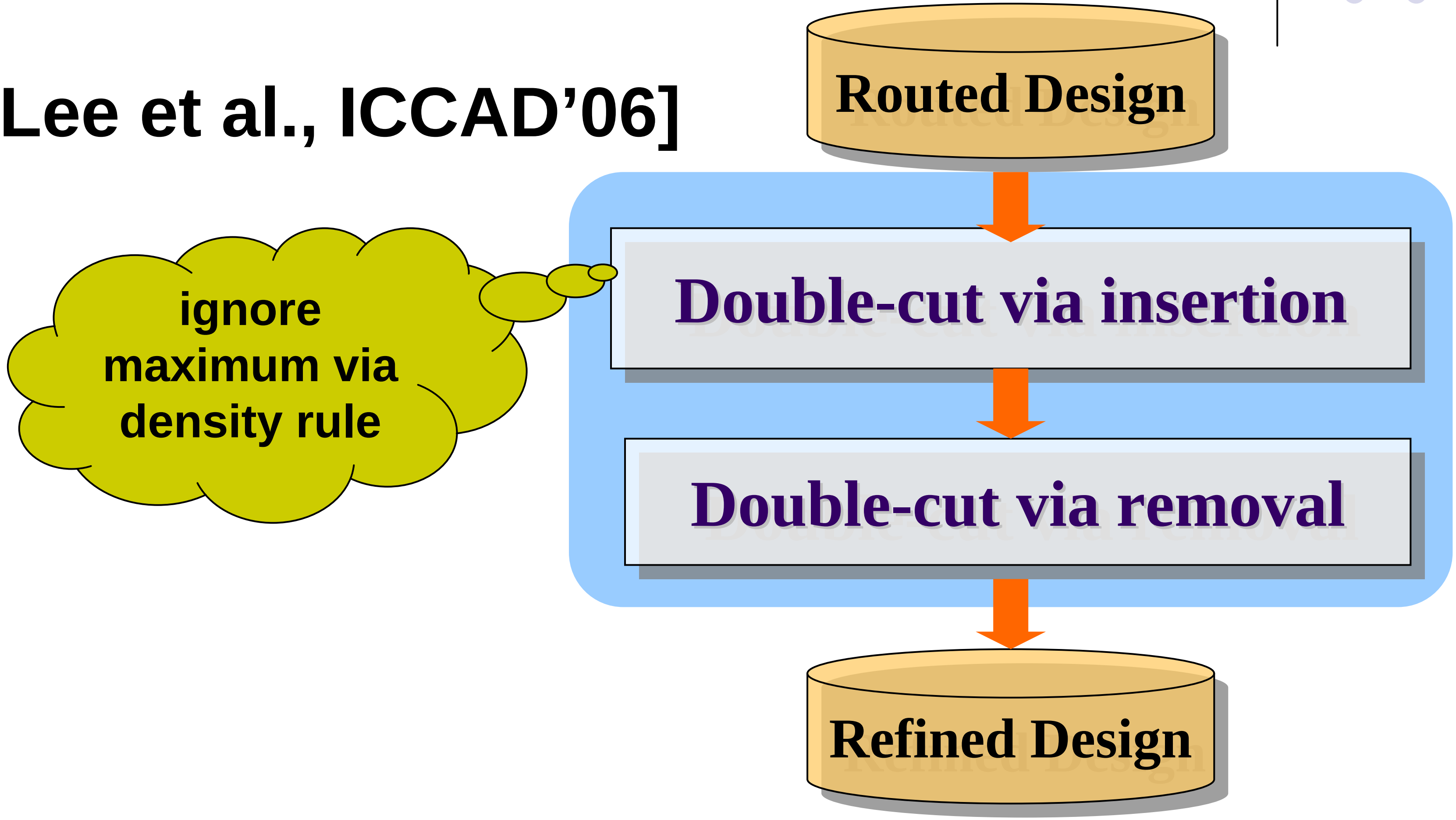
[Lee et al., ICCAD’06]

Double-cut via insertion with via density constraint (DVI/VD)

- **Input**
  - A routed design (satisfying via density rules) and a set of via-related design rules
- **Goal**
  - To replace as many single vias with double-cut vias as possible
- **Constraints**
  - Do not re-route any net
  - Each single via either remains unchanged or is replaced by a double-cut via
  - After replacement, no design rule is violated

Previous work – Two stage approach

- [Lee et al., ICCAD'06]

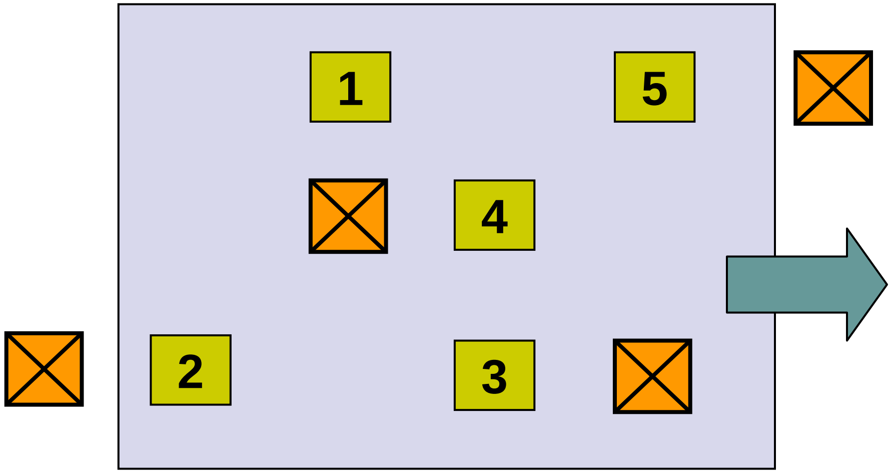


- Cannot guarantee the optimality

Modifications – Density constraint

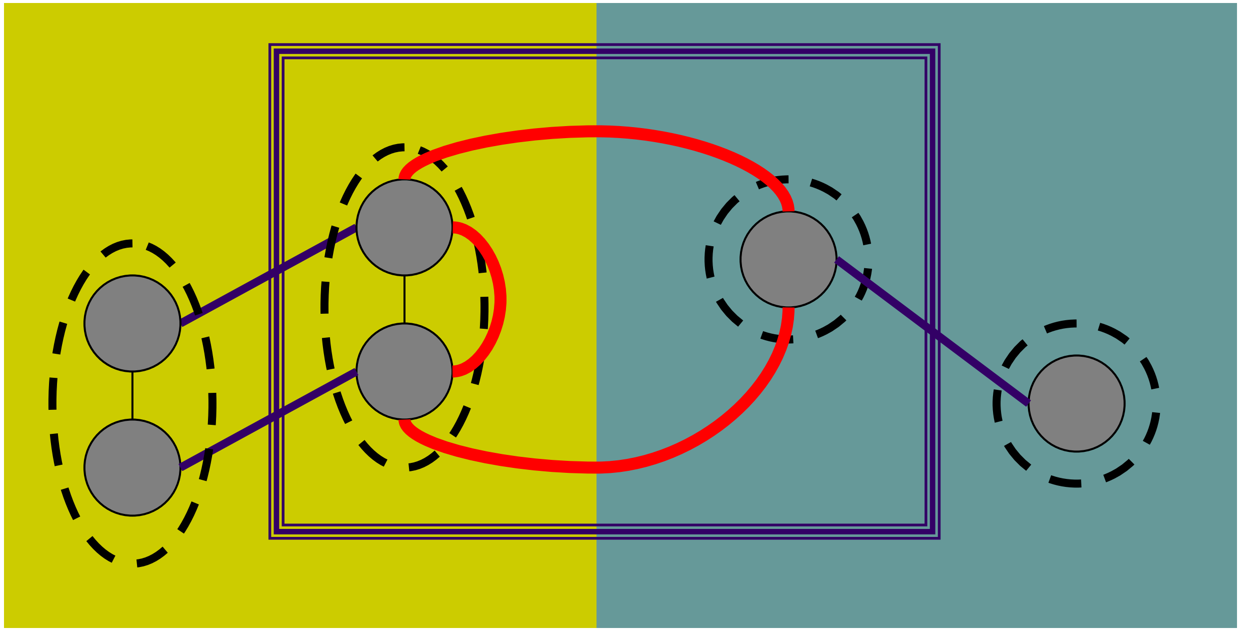
- Possible violating region
  - After double-cut via replacement, the region may violate the maximum via density rule
  - (FDV_1, FDV_2, … FDV_K) and 1 < **U** < K

MAX = 5 → U=3



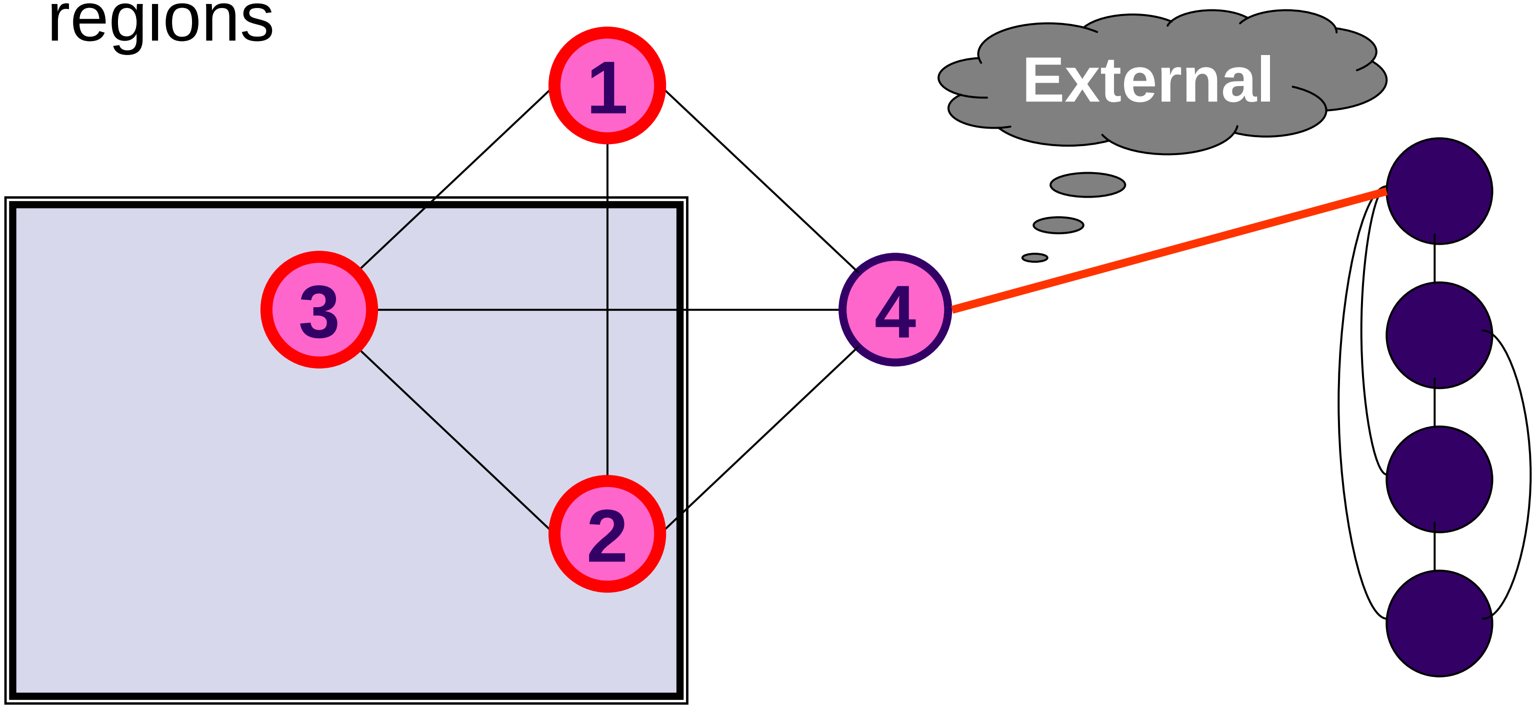
$$\boxed{R_1 + R_2 + R_3 + R_4 + R_5 \leq 3}$$

Modifications – Connected components



Modifications – Pre-selection

- Vertex candidate for pre-selection
  - Has no external edges
  - Does not involved in any possible violating regions



Outline

- Double-cut via insertion (DVI) problem
  - Previous works
  - 0-1 integer linear program (0-1 ILP)
  - Speed-up approaches
- Via density constraint
  - Previous works
  - Extended 0-1 ILP
- Experimental results
- Conclusions

Experimental setup

- Linux based machine with 2.4GHz processor and 2GB memory
- Adopted CPLEX as our 0-1 ILP solver

Circuit	CU(%)	#Nets	#I/Os	#Vias	#A-Vias	#M-Layers
C1	98	4309	20	24594	17522	5
C2	70	5252	211	41157	28591	5
C3	70	18157	85	127059	91727	5
C4	95	17692	415	151912	102347	5
C5	70	44720	99	357386	255301	5

Performance of speed-up approaches

- DVI/VD

Circuit	WO/ acceleration	With acceleration	Speed-up
C1	**7**	**3**	**2.3X**
C2	**12**	**4**	**3.0X**
C3	**158**	**5**	**31X**
C4	**202**	**48**	**4.2X**
C5	—	**3**	—

DVI results

Circuit	Tool*	MIS-DVI*		0-1 ILP	
	#RVI	#RVI	T(s)	#RVI	T(s)
C1	14402	17461	5	+0	3
C2	25918	28507	11	+0	3
C3	80827	91461	86	+0	4
C4	91574	101765	86	+1	5
C5	225142	325429	104	+1	3
Normalized			15.7		1

*[Lee et al., ASP-DAC'06]

DVI/VD results

Circuit	Two-Stage*		0-1 ILP	
	#RVI	T(s)	#RVI	T(s)
C1	17074	6	+175	3
C2	27881	12	+183	4
C3	89960	88	+522	5
C4	91134	100	+2106	48
C5	252056	106	+712	3
Normalized		12.0		1

***[Lee et al., ICCAD’06]**

Outline

- Double-cut via insertion (DVI) problem
  - Previous works
  - 0-1 integer linear program (0-1 ILP)
  - Speed-up approaches
- Via density constraint
  - Previous works
  - Extended 0-1 ILP
- Experimental results
- Conclusions

Conclusions

- Studied the double-cut via insertion problems with/without via density constraints
- Formulated the DVI problems as a set of 0-1 ILP problems
- Exploited the properties of the DVI problems for reducing the problem size and the number of constraints
- Up to 35X faster than existing methods