Department of Electrical Engineering
National Taiwan University
Introduction to EDA, Spring 2018

March 26, 2018
Iris Hui-Ru Jiang

Programming Assignment #1: Redundant Via Insertion

Due 11:59pm, April 8, 2018

1. Introduction

Redundant via insertion is highly effective in improving chip yield and reliability. In this assignment, we study the problem of double-cut via insertion (DVI) in a post-routing stage, where a single via can have, at most, one redundant via inserted next to it, each inserted redundant via can serve only one via, vias cannot be overlapped, and the goal is to insert as many redundant vias as possible. The DVI problem can be formulated as a 0-1 integer linear program (ILP).

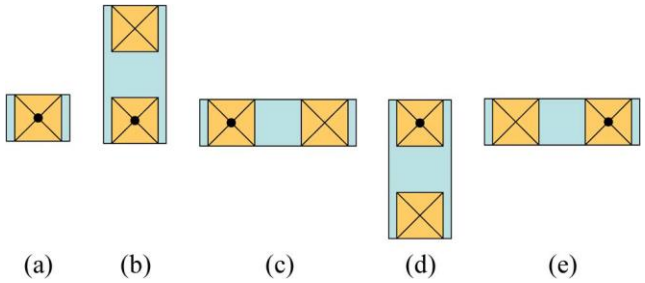

Fig. 1. Double-cut via types. (a) Single via. (b) Type 1. (c) Type 2. (d) Type 3. (e) Type 4.

2. Problem statement

2.1. Brief description

Write a program to generate the 0-1 ILP for inserting redundant vias for a given set of vias.

Given the locations of vias, replace as many single vias with double-cut vias as possible so that each single via either remains unchanged or is replaced by a double-cut via. For each single via, its redundant via can only be inserted at one of its adjacent grids. For simplification, all vias are aligned to grids, and every empty and adjacent grid of a via is available for redundant via insertion.

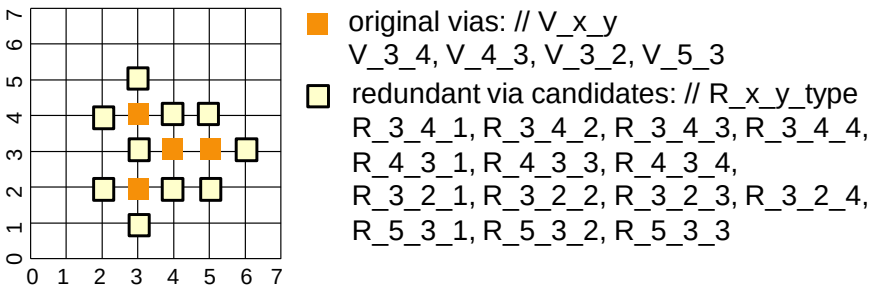

Fig. 2. Example of redundant via insertion.

Fig. 2 shows an instance of redundant via insertion, while Fig. 3 lists the corresponding 0-1 ILP. We create an available redundant via candidate *R_x_y_i* to represent type *i* redundant via for an original via located at (*x*, *y*). An original via has at most 4 redundant via candidates. For example, vias *V_3_4* has 4 candidates, but *V_4_3* has only 3. The objective function (1) tries to maximize the number of inserted vias. Constraints (2)~(5) describes that each single can have at most one redundant via next

to it. Constraints (6)~(8) disallows multiple vias to be inserted at the same location (overlapped vias). Constraint (9) describes that all variables are binary.

$$\max \sum_{x,y,i} R\_x\_y\_i \quad (1)$$
subject to
$$R\_3\_4\_1 + R\_3\_4\_2 + R\_3\_4\_3 + R\_3\_4\_4 \leq 1 \quad (2)$$
$$R\_4\_3\_1 + R\_4\_3\_3 + R\_4\_3\_4 \leq 1 \quad (3)$$
$$R\_3\_2\_1 + R\_3\_2\_2 + R\_3\_2\_3 + R\_3\_2\_4 \leq 1 \quad (4)$$
$$R\_5\_3\_1 + R\_5\_3\_2 + R\_5\_3\_3 \leq 1 \quad (5)$$
$$R\_3\_4\_2 + R\_4\_3\_1 \leq 1 \quad (6)$$
$$R\_3\_4\_3 + R\_4\_3\_4 + R\_3\_2\_1 \leq 1 \quad (7)$$
$$R\_3\_2\_2 + R\_4\_3\_3 \leq 1 \quad (8)$$
$$R\_x\_y\_i \in \{0,1\} \quad (9)$$

Fig. 3. The corresponding 0-1 ILP for the instance given in Fig. 2.

2.2. Input/output specification

The binary should be named as rvi. The program should be invoked like this:

```
./rvi [test*.in] [test*.lp]
```

- Input

The input file describes the locations of vias of a routed design.

The first line describes the total number N of vias. In the following N lines, each line indicates the *x*-coordinate and *y*-coordinate of one via. These vias are aligned to grids. The sample input for Fig. 2 is as follows.

Sample input

```
4
3 4
4 3
3 2
5 3
```

- Output

The program generates the corresponding ILP.

Sample output

The ILP for the given example should be:

```
Maximize
    R_3_4_1 + R_3_4_2 + R_3_4_3 + R_3_4_4 +
    R_4_3_1 + R_4_3_3 + R_4_3_4 +
    R_3_2_1 + R_3_2_2 + R_3_2_3 + R_3_2_4 +
    R_5_3_1 + R_5_3_2 + R_5_3_3
Subject To
    R_3_4: R_3_4_1 + R_3_4_2 + R_3_4_3 + R_3_4_4 <= 1
```

```
    R_4_3: R_4_3_1 + R_4_3_3 + R_4_3_4 <= 1
    R_3_2: R_3_2_1 + R_3_2_2 + R_3_2_3 + R_3_2_4 <= 1
    R_5_3: R_5_3_1 + R_5_3_2 + R_5_3_3 <= 1
    C1: R_3_4_2 + R_4_3_1 <= 1
    C2: R_3_4_3 + R_4_3_4 + R_3_2_1 <= 1
    C3: R_3_2_2 + R_4_3_3 <= 1
Bounds
Binary
    R_3_4_1 R_3_4_2 R_3_4_3 R_3_4_4
    R_4_3_1 R_4_3_3 R_4_3_4
    R_3_2_1 R_3_2_2 R_3_2_3 R_3_2_4
    R_5_3_1 R_5_3_2 R_5_3_3
End
```

The generated ILP will be solved by Gurobi, and the ILP solution will be checked based on its optimality and correctness. Fig. 4 shows the log of solving the above ILP by Gurobi. The maximum number of inserted vias is 4 for the instance given in Fig. 2 (see Solution count; Best objective).

```
Academic license - for non-commercial use only

Gurobi Optimizer version 7.5.2 build v7.5.2rc1 (linux64)
Copyright (c) 2017, Gurobi Optimization, Inc.

Read LP format model from file example.lp
Reading time = 0.00 seconds
: 7 rows, 14 columns, 20 nonzeros
Optimize a model with 7 rows, 14 columns and 20 nonzeros
Variable types: 0 continuous, 14 integer (14 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]
Found heuristic solution: objective 4.0000000
Presolve removed 7 rows and 14 columns
Presolve time: 0.00s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.00 seconds
Thread count was 1 (of 24 available processors)

Solution count 1: 4

Optimal solution found (tolerance 1.00e-04)
Best objective 4.000000000000e+00, best bound 4.000000000000e+00, gap 0.0000%
```

Fig. 4. The snapshot of log generated by Gurobi.

3. Evaluation

Your program should be able to handle 1M vias. Each case is individually evaluated by the correctness (80%) and optimality (20%). For example, if there are maximum 10 inserted vias, while you report 8, and 5 are correct vias, you will get 5/8*80 points for correctness and 5/10*20 points for

optimality. If the binary takes more than **2 minutes** to generate ILP, this case will be graded fail even if the results are correct.

4. References

[1] K.-Y. Lee, C.-K. Koh, T.-C. Wang and K.-Y. Chao, "Optimal Post-Routing Redundant Via Insertion," in *Proc. ACM ISPD*, 2008. http://www.ispd.cc/slides/ispd2008-files/S6-1.pdf

[2] Gurobi optimizer v 7.5.2. http://www.gurobi.com/

5. Submission

- ATTENTION: Plagiarism MUST be avoided. Every submission will be tested by a plagiarism catcher, running on all submissions from this class. If plagiarism is discovered (similarity ≥ 40%), all students involved will receive only partial credit. Specifically, if the score received is x and there are n students involved, then the score for each student is x/n.
- Compress the source code, binary code, and a readme file to a zip file named as: StudentID_pa1.zip (e.g., 0015001_pa1.zip) Please briefly describe how to compile and execute your code in readme.
- Submit the zip file to ceiba