A Local Optimal Method on DSA Guiding Template Assignment with Redundant/Dummy Via Insertion

Xingquan Li[1], Bei Yu[2], Jianli Chen[3], Wenxing Zhu[3]
[1]School of Mathematics and Statistics, Minnan Normal University
[2]Department of Computer Science and Engineering, The Chinese University of Hong Kong
[3]Center for Discrete Mathematics and Theoretical Computer Science, Fuzhou University
xqli@mnnu.edu.cn;byu@cse.cuhk.edu.hk;jlchen@fzu.edu.cn;wxzhu@fzu.edu.cn

ABSTRACT

As an emerging manufacture technology, block copolymer directed self-assembly (DSA) is promising for via layer fabrication. Meanwhile, redundant via insertion is considered as an essential step for yield improvement. For better reliability and manufacturability, in this paper, we concurrently consider DSA guiding template assignment with redundant via and dummy via insertion at post-routing stage. Firstly, by analyzing the structure property of guiding templates, we propose a building-block based solution expression to discard redundant solutions. Then, honoring the compact solution expression, we construct a conflict graph with dummy via insertion, and then formulate the problem to an integer linear programming (ILP). To make a good trade-off between solution quality and runtime, we relax the ILP to an unconstrained nonlinear programming (UNP). Finally, a line search optimization algorithm is proposed to solve the UNP. Experimental results verify the effectiveness of our new solution expression and the efficiency of our proposed algorithm.

1 INTRODUCTION

As an emerging technology, DSA is considered as the most promising for the via layers [1, 2]. Furthermore, previous work has made many significant improvements on manufacturing, modeling, simulation and graphoepitaxy of DSA [3]. These improvements enable DSA technology to pattern vias. In DSA, block copolymers with right proportion would form cylinders, and the remainder material can be used to fabricate contacts/vias after removing cylinders. To generate irregularly distributed vias using DSA, guiding templates including vias are required [4, 5]. These guiding templates are manufactured by the conventional optical lithography, and thus the resolution is limited by the pattern pitch.

Due to various reasons such as random defects, cut misalignment, electro migration and thermal/mechanical stress [6], a single via may fail partially or completely. Via failure heavily impacts on the functionality and yield of a design [7, 8]. Up to now, redundant via (RV) insertion has been considered as an essential step to reduce via failure, and then improves circuit reliability and yield [9, 10]. The

ASPDAC '19, January 21–24, 2019, Tokyo, Japan
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6007-4/19/01…$15.00
https://doi.org/10.1145/3287624.3288748

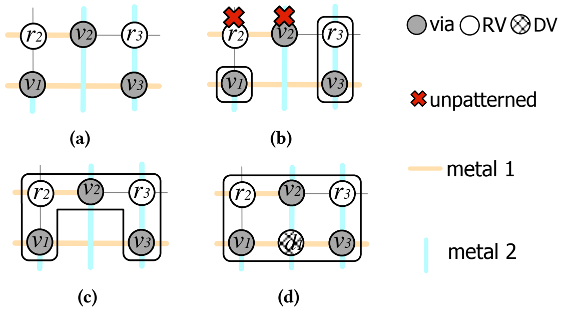

Figure 1: (a) A layout; (b) A redundant via insertion and guiding template assignment; (c) An irregular guiding template; (d) A result with dummy via insertion.

redundant via insertion technique is that a redundant via should be inserted next to every via [11]. In addition, an inserted redundant via should not cause any circuit short, that is, an inserted redundant via should not overlap with any metal wire from other nets of wires. Given a layout with three vias as in Figure 1(a), we can insert redundant vias r_2 and r_3 for vias v_2 and v_3, respectively. Since the surrounding positions of v_1 are illegal, we can not insert any redundant via for v_1.

To improve the resolution, adjacent vias (including redundant vias) may be put into a multi-hole guiding template [12–14]. Naturally, we should assign a guiding template for each via and redundant via. Figure 1(b) shows a guiding template assignment for the layout in Figure 1(a). In Figure 1(b), via v_3 and redundant via r_3 are assigned to a 1×2 hole template, and via v_1 is guided by a single-hole template, while via v_2 and redundant via r_2 cannot be guided and patterned due to the resolution limit. For the five vias and redundant vias in Figure 1(a), a desirable result is that they can be assigned to a same guiding template for patterning as in Figure 1(c). However, this kind of irregular guiding template in Figure 1(c) has a higher probability of generating overlay error. To guarantee the overlay accuracy, only some regular guiding templates are available, the details will be introduced in Section 2.1. If we insert a dummy via (DV) as in Figure 1(d), then the five vias and redundant vias can be guided by a regular 2×3 template.

In the traditional design process, the redundant via insertion and the manufacture of via layers are handled in two independent stages. Fang *et al.* [15] first concurrently considered the redundant via insertion and DSA guiding template assignment problem. For this concurrent consideration, both the number of insertable vias (insertion rate, IR) and the number of manufacturable vias (manufacture rate, MR) are increased.

To improve both of the insertion rate and the manufacture rate, the mainstream techniques in previous works [16–19] can be concluded as the following two types: increasing resolution space by multiple patterning [16, 17]; improving the degree of freedom of redundant vias and guiding templates [18, 19]. For the second way, Fang *et al.* [18] investigated the redundant via insertion and DSA guiding template assignment problem with wire bending. By local perturbing some metal wires, it inserts redundant vias at the cost of increasing the wirelength. But in the advanced 1-D metal layer design, wire bending are unwarrantable. To avoid this drawback, Hung *et al.* [19] studied the problem with dummy via insertion, in which some dummy vias are inserted for assisting formation of guiding templates.

After using dummy via insertion, the layout is more free for inserting redundant vias and using multi-hole guiding templates, which achieves a higher insertion rate and manufacture rate. But two crucial challenges are involved: (1) how to express the solution of more effectively; and (2) how to solve the problem with extremely large-scale solution space more efficiently. In [19], the authors generated all guiding template candidates for all the redundant via candidates, dummy via candidates, and immediate neighbor vias. The generated guiding template candidates are utilized to express solution space, which is extremely large. To solve the problem, the authors of [19] proposed an ILP formulation. Unfortunately, solving ILP may be impractical for large scale and dense circuit layouts. It is of importance to derive a more effective and efficient solution space as well as its optimization method. In this paper we are tackling this challenge, and our main contributions are summarized as follows.

- To discard redundant solutions, we introduce a building-block based manner instead of guiding template candidate to express solution. With the help of building-blocks, we model the DSA guiding template assignment with redundant via and dummy via insertion problem to a new ILP formulation based on a conflict graph.
- With a sigmoid function, we relax the ILP to an UNP to make a good trade-off between solution quality and runtime. We develop a line search optimization algorithm to solve the UNP, which is a local optimal algorithm.
- Experimental results verify the efficiency and effectiveness of our solution expression and optimization method. Specifically, our algorithm achieves comparable experimental results with a state-of-the-art work, and saves 92% runtime.

The rest of this paper is organized as follows. In Section 2, we introduce the related concepts and the problem formulation. In Section 3, we discuss the proposed graph model. In Section 4, we detail our ILP formulation and local optimal algorithm for the problem. In Section 5, we list experimental results, followed by conclusion in Section 6.

2 PRELIMINARIES

2.1 Guiding Template Assignment

For the DSA technique, template is used to guide the holes. Since irregular guiding template has a higher probability of generating overlay error, to guarantee the overlay accuracy, we only use some regular guiding templates with few holes. In this paper, we follow [15] designing seven types of guiding templates as shown in

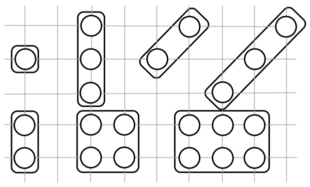

Figure 2: Seven usable types of guiding templates.

Figure 2. In addition, for high resolution and focal depth of guiding templates, the spacing between neighboring guiding templates should not be less than the optical resolution limit spacing d_s. Generally, d_s is set as not less than the distance between a redundant via and its related via, and not less than the hole pitch in a guiding template. We need to decide the assignment of guiding templates such that more vias and redundant vias can be surrounded by guiding templates.

2.2 Dummy Via Insertion

Vias or redundant vias manufactured by DSA technique must be guided by some templates. The most popular manufacture technique for these guiding templates is conventional lithography. For some dense structures in a layout, guiding templates may not be manufactured due to the limitation of optical resolution. In addition, the vias or redundant vias guided by these unmanufactured guiding templates cannot be formed by DSA block copolymer. An effective trick is adding some dummy vias (DV), such that the structure of vias matches a used guiding template. This has been illustrated as in Figure 1(d).

In a circuit, dummy via does not connect to any wire, which is only used for filling the guiding template. The insertion of dummy vias should satisfy the following two conditions: i) the insertion can make up a multi-hole (not less than three holes) guiding template with other vias or redundant vias; ii) it can improve the insertion rate or manufacture rate.

After finding the possible redundant via candidates (RVC), we should find all potential guiding template assignments for every via. In a grid graph, if all grid points covered by a multi-hole (not less than three holes) guiding template are vias or redundant vias, then the guiding template does not need dummy vias; otherwise, every empty grid point needs a dummy via for forming a complete guiding template. If these needed dummy vias on the empty grid points satisfy the above two conditions, then these empty grid points are marked as dummy via candidates (DVC).

2.3 Problem Statement

The problem aims at inserting a redundant via for every via, and manufacturing all vias and their redundant vias by the DSA technique with the help of dummy via insertion. The redundant via insertion rate and the manufacture rate are considered as evaluation indicators in [15, 16]. The redundant via insertion and DSA guiding template assignment with dummy via insertion (RDD) problem is formulated as follows:

Problem 1 (RDD). Given a post-routing via layers layout, the usable types of guiding templates, and the optical resolution limit spacing d_s, insert a redundant via for every via, assign guiding templates for vias, redundant vias and dummy vias, such that: i)

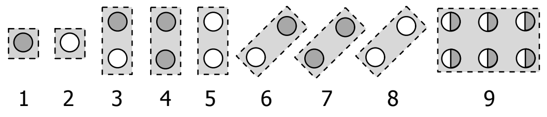

Figure 3: Building-blocks.

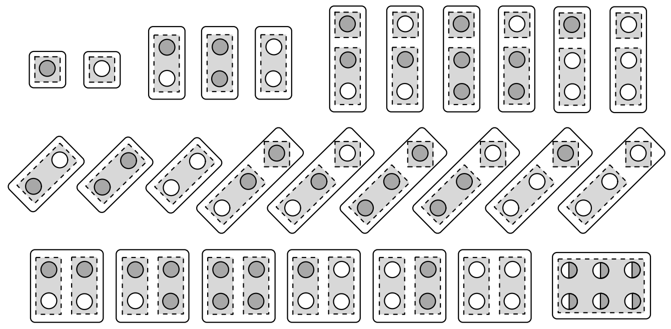

Figure 4: All possible combinations of *bblock*s to form the seven types of guiding templates.

the inserted redundant vias are legal; ii) the spacing between neighboring guiding templates should not be less than d_s. The objective is maximizing $MR + \beta \cdot IR$, where β is a weighting parameter.

3 CONFLICT GRAPH CONSTRUCTION

After finding all possible redundant via candidates and dummy via candidates, previous work directly checks all the possible guiding template assignments. However, detecting all possible guiding template assignment would spend much calculating time. In addition, solving RDD problem with extremely large number of guiding template assignments is also time-consuming.

To obtain a compact solution expression, we introduce a concept of building-block (*bblock*). A *bblock* is composed of some vias and some candidates, and *bblock*s can be used to compose various types of guiding templates. Nine types of *bblock*s are shown in Figure 3, where *bblock*_1 includes a via; *bblock*_2 includes a redundant via; *bblock*_3 includes a via and a redundant via; *bblock*_4 includes two vias; *bblock*_5 includes two redundant vias; *bblock*_6 includes a via and a redundant via in diagonal; *bblock*_7 includes two vias in diagonal; *bblock*_8 includes two redundant vias in diagonal; and *bblock*_9 includes six vias or redundant vias, which can be covered by a six-hole guiding template. These types of *bblock*s compose a dictionary.

Then the seven types of usable guiding templates in Figure 2 can be formed by grouping some *bblock*s in the dictionary, as shown in Figure 4. Here we only list the vertical cases and skip the horizontal cases due to similarity.

A dummy via candidate (DVC) must belong to a guiding template. At the DVC finding stage, we can easily find out which guiding template a DVC belongs to. Given a result of finding redundant via candidates as in Figure 5(a), we can identify all possible *bblock*s as in Figure 5(b), and these *bblock*s are regarded as vertices in the conflict graph. Based on these vertices, we construct a conflict graph [17], as shown in Figure 5(c). There are three types of edges in conflict graph: 1) Overlap edges E_O: if *bblocks* i and j are overlap, then $e_{ij} \in E_O$; 2) Conflict edges E_C: if the distance between *bblocks* i and j is not larger than d_s and $e_{ij} \notin E_O$, then $e_{ij} \in E_C$; 3) Template edges E_T: if *bblocks* i and j with $e_{ij} \in E_C$ can be assigned simultaneously to a guiding template without any design error, and at least one of i and j is not S_1, then $e_{ij} \in E_T$. Obviously, $E_T \subseteq E_C$. Then, a conflict graph $CG(V, E)$ is constructed, where $v \in V$ is *bblock* and $e_{ij} \in E$ is an edge and $E = (E_C - E_T) \cup E_O$.

In Figure 5(c), the black edges are the overlap edges, the red edges are the conflict edges, and the green dotted edges are the template edges. From Figure 5(a), we know *bblocks* a and d are overlapped with each other at v_1, hence there is an overlap edge between them as in Figure 5(c). The distance between *bblocks* a and e is not larger than d_s, hence there is a conflict edge between them. Since *bblocks* a and c can be assigned to a 2×2 hole guiding template as in Figure 5(c), there is a template edge between a and c.

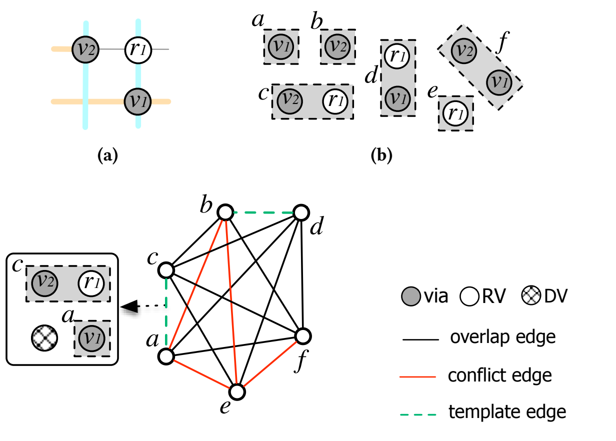

Figure 5: (a) A layout; (b) All *bblock*s of the layout in Figure 5(a); (c) Conflict graph.

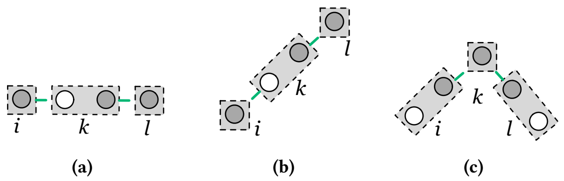

Figure 6: Three kinds of conflict structures.

4 PROPOSED ALGORITHMS

In this section, we first formulate RDD problem into an ILP. And then, we relax the ILP into an UNP, and propose a line search based optimization algorithm to solve the UNP.

4.1 ILP Formulation

In conflict graph, if two *bblocks* i and j are overlapped with each other, i.e., $e_{ij} \in E_O$, then only one of the them can be patterned. In addition, if two *bblocks* are within the optical resolution limit spacing d_s, i.e., $e_{ij} \in E_C$, then only one of them can be patterned due to limitation of resolution, unless the two *bblocks* are assigned into the same guiding template, i.e., $e_{ij} \notin E_T$.

If two *bblocks* i and j are connected by a template edge, then they may be assigned to the same guiding template, but not necessarily. Specially, for the structure shown in Figure 6(a), *bblocks* i and l

are connected to k by two template edges, but i, k and l cannot be simultaneously assigned to a same guiding template, since we do not have a guiding template with four holes aligned in a line (same as the structures in Figure 6(b) and Figure 6(c)). We call these unordered triplets (i, k, l) as conflict structures, which are defined as follow.

Definition 1 (Conflict Structure)**.** The conflict structure is a structure composed of three *bblock*s i, k and l, in which e_{ik} and e_{kl} are template edges and there does not exist any edge between i and l.

We denote CS as the set of conflict structures. In addition, different *bblock*s include different vias and redundant vias. The objective of RDD problem is intend to maximize $MR + \beta \cdot IR$, i.e., maximizing the weighted sum of the number of manufacturable vias and the number of inserted redundant vias. Suppose the weights of a via and a redundant via is 1 and β, respectively. We jointly consider MR and IR by assigning weight w_i to every *bblock* i as

$$w_i = N_v + \beta \cdot N_r, \tag{1}$$

where N_v and N_r are the numbers of included vias and redundant vias by *bblock* i, respectively. Let W be the set of weights, then the conflict graph $CG(V, E)$ is weighted and written as $CG(V, E, W)$.

Thus, we formulate RDD problem as following ILP:

$$\begin{aligned} \max_{\mathbf{x}} \quad & \sum_{i \in V} w_i x_i & & (2) \\ \text{s.t.} \quad & x_i + x_j \le 1, & \forall e_{ij} \in E; \quad & (2a) \\ & x_i + x_k + x_l \le 2, & \forall (i,k,l) \in CS; \quad & (2b) \\ & x_i \in \{0, 1\}, & \forall i \in V. \quad & (2c) \end{aligned}$$

In above ILP, Constraint (2a) indicates that, if there exists $e_{ij} \in E_O$ or $e_{ij} \in E_C - E_T$ between vertices i and j, then at most one of them can be patterned; Constraint (2b) ensures that, if i, k and l compose an conflict structure, then at most two of them can be patterned.

4.2 A Local Optimal Algorithm

It is time consuming to solve the ILP by commercial solver for a large scale layout. In this subsection, we develop a fast algorithm to obtain a local optimal solution of RDD problem.

Firstly, the ILP formulation of Problem (2) is equivalent to:

$$\begin{aligned} \max_{\mathbf{x}} \quad & \sum_{i \in V} w_i x_i & & (3) \\ \text{s.t.} \quad & x_i x_j = 0, & \forall e_{ij} \in E; \quad & (3a) \\ & x_i x_k x_l = 0, & \forall (i,k,l) \in CS; \quad & (3b) \\ & x_i \in \{0, 1\}, & \forall i \in V, \quad & (3c) \end{aligned}$$

where $\mathbf{w} = (w_1, w_2, \cdots, w_n)^\top \in \mathbb{R}^n$, $\mathbf{x} = (x_1, x_2, \cdots, x_n)^\top \in \{0, 1\}^n$, $n = |V|$. Since Constraints (3a) and (3b) are equality, they can be directly incorporated in the objective function. That is, Problem (3) can be further rewritten as following integer nonlinear programming:

$$\begin{aligned} \max_{\mathbf{x}} \quad & \sum_{i \in V} \{w_i x_i \prod_{\substack{j \in V \\ e_{ij} \in E}} (1 - x_j) \prod_{\substack{k,l \in V \\ (i,k,l) \in CS}} (1 - x_k x_l)\} & (4) \\ \text{s.t.} \quad & x_i \in \{0, 1\}, \ \forall i \in V. & (4a) \end{aligned}$$

In (4), $e_{ij} \in E$ and $(i, k, l) \in CS$ are used to describe the relationship among vertices. Let $\mathbf{B} = (B_{ij}) \in \{0, 1\}^{n \times n}$ be the adjacency matrix of the conflict graph CG. If $e_{ij} \in E$, then $B_{ij} = 1$ and $(1 - x_j)^{B_{ij}} = (1 - x_j)$; and if $e_{ij} \notin E$, then $B_{ij} = 0$ and $(1 - x_j)^{B_{ij}} = 1$. Moreover, we use $\mathbf{C} = (C_{ikl}) \in \{0, 1\}^{n \times n \times n}$ to represent the conflict structures in layout. If $(i, k, l) \in CS$, then $C_{ikl} = 1$ and $(1 - x_k x_l)^{C_{ikl}} = (1 - x_k x_l)$, otherwise $C_{ikl} = 0$ and then $(1 - x_k x_l)^{C_{ikl}} = 1$. The objective of (4) can be more conveniently written using adjacent matrix and tensor of CG as Problem (5).

$$\begin{aligned} \max_{\mathbf{x}} \quad & \sum_{i \in V} \{w_i x_i \prod_{j \in V} (1 - x_j)^{B_{ij}} \prod_{k,l \in V} (1 - x_k x_l)^{C_{ikl}}\} & (5) \\ \text{s.t.} \quad & x_i \in \{0, 1\}, \ \forall i \in V. & (5a) \end{aligned}$$

Problem (5) is equivalent to Problem (2), and still falls to the catogory of discrete formulation. To design a more efficient solution, we further relax this problem into a continuous domain. First, we introduce an auxiliary vector $\mathbf{y} = (y_i) \in \mathbb{R}^n$, and approximate the constraint $x_i \in \{0, 1\}$, $\forall i \in V$ with the sigmoid function

$$x_i \approx \sigma(y_i) = (1 + e^{-\gamma y_i})^{-1}. \tag{6}$$

where γ is set to 8 in this paper for a sharper sigmoid function. Then Problem (5) is approximated as

$$\begin{aligned} & \max_{\mathbf{y}} \ f(\mathbf{y}) = & (7) \\ & \sum_{i \in V} \{w_i \sigma(y_i) \prod_{j \in V} (1 - \sigma(y_j))^{B_{ij}} \prod_{k,l \in V} (1 - \sigma(y_k)\sigma(y_l))^{C_{ikl}}\}. & \end{aligned}$$

If we obtain a solution $\mathbf{y}^*$ of Problem (7), then the final solution $\mathbf{x}^*$ is obtained by rounding the sigmoid function value $\sigma(y_i^*)$ to the nearest integer, $\forall i \in V$. Problem (7) is an UNP. Let

$$g_i(\mathbf{y}) = \sigma(y_i) \prod_{j \in V} (1 - \sigma(y_j))^{B_{ij}} \prod_{k,l \in V} (1 - \sigma(y_k)\sigma(y_l))^{C_{ikl}} \tag{8}$$

and $g_i = g_i(\mathbf{y})$, then the objective of Problem (7) is

$$f(\mathbf{y}) = \sum_{i \in V} w_i g_i.$$

We aim at finding a maximal solution $\mathbf{y}^* \in \mathbb{R}^n$ of Problem (7). At each iteration t, the solution is updated by the following gradient direction of $f(\mathbf{y})$:

$$\mathbf{y}^{(t+1)} = \mathbf{y}^{(t)} + \alpha \nabla f(\mathbf{y}^{(t)}), \tag{9}$$

where α is the step length, which is obtained by the Wolfe-Powell inexact line search method in [20]. Besides, $[\nabla f(\mathbf{y}^{(t)})]_i = \partial f(\mathbf{y}^{(t)}) / \partial y_i$ is calculated by

$$\begin{aligned} [\nabla f(\mathbf{y}^{(t)})]_i = & \gamma w_i g_i^{(t)} \{(1 - \sigma(y_i^{(t)})) - \sum_j B_{ij} \sigma(y_j^{(t)}) & (10) \\ & - \sum_k \sum_l C_{ikl} \frac{\sigma(y_k^{(t)})(1 - \sigma(y_k^{(t)}))\sigma(y_l^{(t)})}{1 - \sigma(y_k^{(t)})\sigma(y_l^{(t)})}\}, & \end{aligned}$$

where $g_i^{(t)} = g_i(\mathbf{y}^{(t)})$. It can be shown that first order dynamic in (9) increases $f(\mathbf{y}^{(t)})$ at every iteration t, and will converge to a local optimum.

However, since the objective function of Problem (7) is highly non-linear and non-concave, the above iteration is highly dependent on the initial solution $\mathbf{y}^{(0)}$ and may converge to an undesirable local optimum. Hence, in order to obtain a better solution, the iteration would be better starting from a good initial solution $\mathbf{y}^{(0)}$. We propose an $\mathcal{O}(|V|)$ complexity algorithm to obtain a desirable initial solution, as detailed in Algorithm 1.

Algorithm 1 Initial Solution Generation

```
Input: A connected component of $CG(V, E, W)$;
Output: Initial solution $\mathbf{x}^{(0)}$ of ILP (2);
 1: repeat
 2:     $S \leftarrow \{k \mid k \in \text{argmin}_{l \in V} w_n(l)\}$;
 3:     repeat
 4:         $\forall k \in S$, compute $w_s(k)$;                ▷ Equation (11)
 5:         $x_i^{(0)} \leftarrow 1$, where $i = \text{argmin}_{k \in S} w_s(k)$;
 6:         for every $j$ in $V$ with $e_{ji} \in E$ do
 7:             $x_j^{(0)} \leftarrow 0$, and $V \leftarrow V - \{j\}$;
 8:             if $j \in S$, $S \leftarrow S - \{j\}$;
 9:         end for
10:         $S \leftarrow S - \{i\}$, and $V \leftarrow V - \{i\}$;
11:     until $S = \emptyset$
12: until $V = \emptyset$
```

In line 2 of Algorithm 1, $w_n(l)$ is the number of vias and redundant vias covered by *bblock* l. In line 4, the selection weight $w_s(k)$ of *bblock* k is calculated by

$$w_s(k) = d_c(k) - d_t(k), \tag{11}$$

where $d_c(k)$ is the number of conflict edges incident to *bblock* k, and $d_t(k)$ is the number of template edges incident to *bblock* k.

After obtaining the desirable initial solution, we present our optimization method for the ILP (2) in Algorithm 2, where the objective value is increased at every iteration, and is converged to a maximal solution. Experimentally, Algorithm 2 only takes a few iterations before achieving the convergence condition. Furthermore, we know from (10) that every iteration of Algorithm 2 can be finished in $\mathcal{O}(\max\{|V| \cdot |E|, |V| \cdot ||\mathbf{C}||_0\})$, where $||\mathbf{C}||_0$ is the number of nonzero elements in tensor $\mathbf{C}$.

Algorithm 2 UNP Solver

```
Input: A connected component of $CG(V, E, W)$, convergence
    threshold $\delta = 10^{-4}$;
Output: Solution $\mathbf{x}^*$ of ILP (2);
 1: Initialize $t \leftarrow 0$;
 2: Generate $\mathbf{x}^{(0)}$;                                ▷ Algorithm 1
 3: If $x_i^{(0)} = 1$, let $y_i^{(0)} \leftarrow 1$; otherwise, let $y_i^{(0)} \leftarrow -1$;
 4: repeat
 5:     $\forall i \in V$, compute $g_i^{(t)}$;                   ▷ Equation (8)
 6:     Obtain $\nabla f(\mathbf{y}^{(t)})$;                       ▷ Equation (10)
 7:     $\alpha \leftarrow \texttt{LineSearch}(\mathbf{y}^{(t)})$;
 8:     $\mathbf{y}^{(t+1)} \leftarrow \mathbf{y}^{(t)} + \alpha \nabla f(\mathbf{y}^{(t)})$;
 9:     $t \leftarrow t + 1$;
10: until $||\nabla f(\mathbf{y}^{(t)})|| < \delta$
11: Get $x_i^*$ by rounding $\sigma(y_i^{(t)})$ to the nearest integer, $\forall i \in V$.
```

We can achieve a local optimal result by performing Algorithm 2. In this paper, we skip the detailed proof due to space limitation. In addition, as will be verified in experiments, if Algorithm 2 starts from a desirable initial solution $\mathbf{x}^{(0)}$ by Algorithm 1, it likely returns a near global optimal result.

5 EXPERIMENTAL RESULTS

Our proposed algorithms are implemented in C++ and run on a personal computer with 2.7GHz CPU, 8GB memory and Unix operating system. We test our method on MCNC benchmarks and an industry Faraday benchmarks, provided by Fang *et al.* [15]. As in [18], layouts of all benchmarks are transformed to grid models, where a grid size is one metal pitch. In the experiments, the distance between a via and its redundant via is set to one metal pitch, and the optical resolution limit spacing d_s of adjacent guiding templates is set to one metal pitch too. The user-defined parameter β is set to 1.

To evaluate the performance of the proposed local optimal algorithm in Section 4.2, we compare the obtained results with the ILP results in TCAD'17 [15] and ASPDAC'17 [19]. The experimental comparisons are reported in Table 1. Columns "TCAD'17 [15]" and "ASPDAC'17 [19]" are the results in [15] and [19], respectively. column "Extended TVLSI'18 [17]" is obtained by extending the method in [17] to consider dummy via insertion. The results in "Ours" are obtained by solving Algorithm 2 in Section 4.2. Moreover, in this table, column "#V" lists the numbers of vias, and column "CPU(s)" is the runtime in second. "MR(%)" and "IR(%)" are, respectively, the manufacture rate and the redundant via insertion rate.

$$\text{MR} = \frac{\#\text{MV}}{\#\text{V}} \times 100\%, \qquad \text{IR} = \frac{\#\text{RV}}{\#\text{V}} \times 100\%.$$

In the above equations, #MV is the number of manufacturable vias (excluding redundant vias), #RV is the number of inserted redundant vias.

The difference between the results in "Our" and in "TCAD'17 [15]" is that our work considers dummy via insertion but TCAD'17 does not. As shown from row "Ratio", the proposed Algorithm 2 improves MR and IR up to 6% and 7%, respectively. These improvements mainly result from the help of dummy via insertion. Naturally, considering dummy via insertion will extremely increase the size of solution space, which leads to more challenge for solving. In spite of this, our Algorithm 2 achieves 3.99× less runtime than the ILP in TCAD'17 [15].

Both of the methods in ASPDAC'17 [19] and our Algorithm 2 consider dummy via insertion as a complementary technique for improving MR and IR. From the comparison in TABLE 1, our Algorithm 2 achieves almost the same results as the ILP in [19]. This shows our proposed local optimal algorithm can achieve near-optimal result. It must be noted that, the average runtime of the ILP in [19] is 13.32× slower than our Algorithm 2. The improvement in runtime owes to our compact solution expression, which greatly reduces the solution space.

Compared with the results in column "Extended TVLSI'18 [17]", "Ours" improves MR and IR up to 3% and 2%, respectively. These comparisons show that our fast algorithm is very effective and efficient.

Table 1: Comparison of Four Methods for RDD Problem.

Benchmarks	#V	TCAD'17 [15]			ASPDAC'17 [19]			Extended TVLSI'18 [17]			Ours		
		MR(%)	IR(%)	CPU(s)	MR(%)	IR(%)	CPU(s)	MR(%)	IR(%)	CPU(s)	MR(%)	IR(%)	CPU(s)
struct	12551	99.86	99.42	30.00	99.96	99.79	40.47	99.80	99.43	11.42	99.95	99.69	10.78
primary1	8764	99.80	99.21	28.00	99.92	99.33	29.14	99.79	99.42	11.97	99.80	99.45	12.75
primary2	32684	99.54	98.97	72.00	99.92	99.49	111.22	99.67	98.73	36.02	99.70	99.11	37.15
s5378	8649	81.10	61.78	11.00	96.41	75.27	34.41	90.32	71.89	0.58	95.42	75.09	0.55
s9234	6874	80.07	59.54	9.00	96.21	74.51	38.63	89.80	71.93	0.49	94.66	75.12	0.50
s13207	18780	84.35	66.93	23.00	97.13	79.28	99.96	91.27	75.57	2.27	96.19	79.01	1.85
s15850	22694	82.70	64.41	28.00	96.63	77.35	121.94	90.75	74.36	2.94	95.61	77.58	2.73
s38417	54225	84.00	65.71	65.00	96.83	78.13	320.35	91.01	75.58	15.27	95.92	78.59	13.06
s38584	74155	81.53	63.01	88.00	96.37	76.83	416.11	90.12	73.87	30.32	95.18	77.12	24.38
dma	34697	97.85	95.29	55.00	99.61	97.49	208.75	98.81	96.63	6.65	99.06	97.22	6.44
dsp1	30317	99.05	97.57	53.00	99.74	98.45	176.42	99.33	97.93	10.77	99.45	98.25	10.52
dsp2	31301	98.50	96.68	52.00	99.76	98.74	179.37	99.16	97.79	8.96	99.35	98.16	8.55
risc1	43858	98.77	96.93	75.00	99.70	98.04	216.61	99.20	97.54	18.80	99.34	97.90	18.92
risc2	44385	98.79	96.91	77.00	99.70	97.98	229.22	99.18	97.47	18.46	99.32	97.89	18.69
Avg.	30281	91.85	83.03	47.57	98.42	89.33	158.76	95.59	87.72	12.49	97.78	89.30	11.92
Ratio		0.94	0.93	3.99	1.00	1.00	13.32	0.97	0.98	1.04	1.00	1.00	1.00

6 CONCLUSION

In this paper, we have concurrently considered dummy via insertion and redundant via insertion for DSA guiding template assignment problem. Thanks to building-blocks, the vertices number in conflict graph can be effectively reduced. On the conflict graph, we model the problem as an ILP formulation, and relax it to an unconstrained nonlinear programming. We develop a line search optimization algorithm to obtain a local optimal solution. Since the algorithm is highly dependent on the initial solution, we further design an efficient initial solution generation instead of random guess. Experimental results demonstrate the effectiveness of our solution expression and the proposed algorithm.

ACKNOWLEDGEMENTS

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61672005 and Grant 11331003 and the Research Grants Council of Hong Kong SAR (Project No. CUHK24209017).

REFERENCES

[1] H. Yi, X.-Y. Bao, J. Zhang, R. Tiberio, J. Conway, L. Chang, S. Mitra, H.-S P. Wong, and W. M. Tong "Contact-hole patterning for random logic circuits using block copolymer directed self-assembly", In *Proc. of SPIE*, vol. 8323, 2012.

[2] J. Ou, B. Yu, J.-R. Gao, and D. Z. Pan "Directed self-assembly cut mask assignment for unidirectional design" *J. Micro/Nanolith. MEMS MOEMS* vol. 14, no. 3, 031211-1–031211-8, 2015.

[3] H.-S. P. Wong, C. Bencher, H. Yi, X.-Y. Bao, and L.-W. Chang, "Block copolymer directed self-assembly enables sublithographic patterning for device fabrication", In *Proc. of SPIE*, vol. 8323, 2012.

[4] A. Latypov, T. H. Coskun, G. Garner, M. Preil, G. Schmid, J. Xu, and Y. Zou, "Simulations of spatial DSA morphology, DSA-aware assist features and block copolymer-homopolymer blends", In *Proc. of SPIE*, vol. 9049, San Jose, CA, USA, 2014.

[5] S. Shim, W. Chung, and Y. Shin, "Redundant via insertion for multiple-patterning directed-self-assembly lithography", In *Proc. of DAC*, 2016, pp. 410–417.

[6] Y. Zorian, D. Gizopoulos, C. Vandenberg and P. Magarshack, "Guest editors introduction: design for yield and reliability", *IEEE TCAD*, vol. 21, no. 12, pp. 2197–2208, 2004.

[7] G. Xu, L. D. Huang, D. Z. Pan, and M. D. F. Wong, "Redundant-via enhanced maze routing for yield improvement", In *Proc. of ASPDAC*, 2005, pp. 1148–1151.

[8] J. Ou, B. Yu, X. Xu, J. Mitra, Y. Lin, and D. Z. Pan, "DSAR: DSA aware routing with simultaneous DSA guiding pattern and double patterning assignment?, In *Proc. of ISPD*, 2017, pp. 91–98.

[9] H.-Y. Chen, M.-F. Chiang, Y.-W. Chang, L. Chen, and B. Han, "Full-chip routing considering double-via insertion", *IEEE TCAD*, vol. 27, no. 5, pp. 844–857, 2008.

[10] K.-Y. Lee, C.-K. Koh, T.-C. Wang, and K.-Y. Chao, "Fast and optimal redundant via insertion", *IEEE TCAD*, vol. 27, no. 12, pp. 2197–2208, 2008.

[11] S.-T. Lin, K.-Y. Lee, T.-C. Wang, C.-K. Koh, and K.-Y. Chao, "Simultaneous redundant via insertion and line end extension for yield optimization", In *Proc. of ASPDAC*, pp. 633–638, 2011.

[12] Y. Badr, A. Torres, and P. Gupta, "Mask assignment and synthesis of DSA-MP hybrid lithography for sub-7nm contacts/vias", In *Proc. of DAC*, 2015, pp. 1–6.

[13] Z. Xiao, Y. Du, H. Tian, M. D. F. Wong, H. Yi, H.-S. P. Wong, and H. Zhang, "Directed self-assembly (DSA) template pattern verification", In *Proc. of DAC*, 2014, pp. 1–6.

[14] J. Kuang, J.-J. Ye, and F.-Y. Young, "Simultaneous template optimization and mask assignment for DSA with multiple patterning", In *Proc. of ASPDAC*, 2016, pp. 75–82.

[15] S.-Y. Fang, Y.-X. Hong, and Y.-Z. Lu, "Simultaneous guiding template optimization and redundant via insertion for directed self-assembly", *IEEE TCAD*, vol. 36, no. 1, pp. 156-169, 2017.

[16] J. Ou, B. Yu, and D. Z. Pan, "Concurrent guiding template assignment and redundant via insertion for DSA-MP hybrid lithography", In *Proc. of ISPD*, 2016, pp. 39-46.

[17] X. Li, B. Yu, J. Ou, J. Chen, D. Z. Pan and W. Zhu, "Graph based redundant via insertion and guiding template assignment for DSA-MP", *IEEE TVLSI*, DOI: 10.1109/TVLSI.2018.2850044, 2018.

[18] S.-Y. Fang, and Y.-X. Hong, "Design optimization considering guiding template feasibility and redundant via insertion for directed self-assembly", In *Proc. of APCCAS*, 2016.

[19] C.-Y. Hung, P.-Y. Chou, and W.-K. Mak, "Optimizing DSA-MP decomposition and redundant via insertion with dummy vias", In *Proc. of ASPDAC*, 2017.

[20] J. Nocedal and S. J. Wright, "Numerical Optimization, 2nd Edition", Springer, 2006.