Graph-Based Redundant Via Insertion and Guiding Template Assignment for DSA-MP

Xingquan Li, Bei Yu, *Member, IEEE*, Jiaojiao Ou, Jianli Chen, David Z. Pan, *Fellow, IEEE*, and Wenxing Zhu

***Abstract*— Inserting redundant vias is necessary for improving via yield in circuit designs. Block copolymer directed self-assembly (DSA) is an emerging and promising lithography technology for the manufacture of vias and redundant vias, in which guiding templates are used to enhance the resolution. Considering manufacturability of via layer, multiple patterning (MP) lithography is also needed in advanced designs. In this paper, we study the redundant via insertion and guiding template assignment for DSA with MP problem at the postrouting stage. We propose a graph-methodology-based solution framework. First, by analyzing the structure of guiding templates, we propose a new solution expression by introducing the concept of *multiplet* to discard redundant solutions, and then, honoring the compact solution expression, we construct a conflict graph on the grid model. Second, we formulate the problem with single patterning (SP) as a constrained maximum weight-independent set problem, for which a fast linear interpolation algorithm is introduced to obtain a local optimal solution. To avoid undesirable local optima, we propose an effective initial solution generation method. Our framework is general and is further extended to solve the problem with double patterning (DP) or triple patterning (TP) in a two-stage manner. Experimental results validate the efficiency and effectiveness of our method. Specifically, compared with the state-of-the-art work for the problem with SP, DP, and TP, our method can save 58%, 82%, and 96% runtime, respectively.**

***Index Terms*— Constrained maximum-independent set, directed self-assembly (DSA), multiple patterning (MP), multiplet, redundant via insertion (RVI).**

I. Introduction

IN AN integrated circuit (IC) layout, a via provides the connection between two net segments from adjacent metal layers. A single via may fail partially or completely because of various reasons, such as random defects, cut misalignment, and electromigration or thermal stress [1]. A partial via failure may induce timing problems due to the increase of contact resistance and parasitic capacitance, while a complete via failure will produce a broken net in a circuit [2]. These failures may heavily hinder the functionality and yield of a circuit. Therefore, reducing yield loss due to via failure is one of the most important problems in the IC design flow.

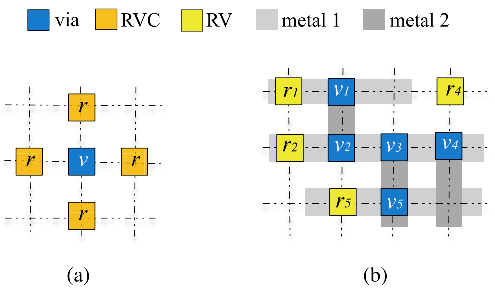


Fig. 1. Example of RVI. (a) Four locations of RVCs. (b) Feasible RVI result.

A promising method for improving via yield and reliability is adding a redundant via adjacent to every via [1], [2], enabling via failure to be tolerated. As shown in Fig. 1(a), four redundant via candidates (RVCs), labeled as r, are next to the via v. The inserted redundant via should not cause any short circuit. That is, it can only be inserted at a free space not occupied by any metal wire, or at a space occupied by only one metal wire that is on the same net as the via [3], [4]. A via is an alive via if it has free spaces for inserting a redundant via, otherwise it is called a dead via [2]. As shown in Fig. 1(b), vias v_1, v_2, v_4, and v_5 are alive vias, while via v_3 is a dead via, since it has no free space for redundant via insertion (RVI). Some works studied the RVI problem at the routing stage [5], [6] or at the postrouting stage [7]–[9]. In fact, to improve yield and reliability, considering the problem is necessary at both the routing and the postrouting stages. In this paper, we consider RVI at the postrouting stage.

The recently proposed block copolymer directed self-assembly (DSA) lithography technology is considered as a promising fit for via layers in the 7-nm technology node and beyond [10]. Many significant improvements have been made on manufacturing, modeling, and simulation of DSA, especially on the graphoepitaxy DSA [11]. The block copolymers form cylinders, and by removing cylinders, the material can be used to fabricate vias. To generate irregularly distributed vias using DSA, guiding templates surrounding vias are required [12]. These guiding templates are manufactured by conventional optical lithography, and thus, the resolution

Manuscript received January 27, 2018; revised May 14, 2018; accepted June 19, 2018. Date of publication July 18, 2018; date of current version October 23, 2018. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61672005 and Grant 11331003 and in part by the Research Grants Council of Hong Kong under Grant CUHK24209017. *(Corresponding author: Wenxing Zhu.)*

X. Li, J. Chen, and W. Zhu are with the Center for Discrete Mathematics and Theoretical Computer Science, Fuzhou University, Fuzhou 350108, China (e-mail: n130320024@fzu.edu.cn; jlchen@fzu.edu.cn; wxzhu@fzu.edu.cn).

B. Yu is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: byu@cse.cuhk.edu.hk).

J. Ou and D. Z. Pan are with the Department of Electronics and Communication Engineering, University of Texas at Austin, Austin, TX 78712 USA (e-mail: jiaojiao@cerc.utexas.edu; dpan@cerc.utexas.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2018.2850044

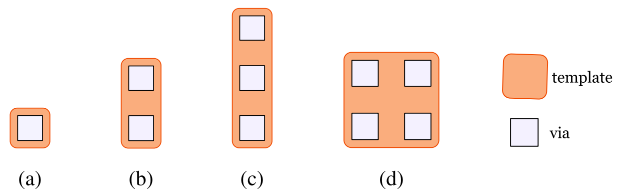


Fig. 2. Four usable types of guiding templates. (a) t_1. (b) t_2. (c) t_3. (d) t_4.

is limited. To improve the resolution, some adjacent vias may be put in a multihole guiding template [11], [13].

To guarantee an acceptable overlay accuracy, only some regular guiding templates with a few holes are available to guide vias. In this paper, the usable four types of guiding templates t_1, t_2, t_3, and t_4 are shown in Fig. 2(a)–(d) as in [14]. Usually, obtaining a desirable guiding template assignment (GTA) for vias is critical in the DSA-based patterning technology [15]. With the feature size decreasing, the density of vias dramatically increases in the via layer correspondingly, and using single patterning (SP) only cannot obtain the required resolution of a guiding template. Hence, multiple patterning (MP) lithography technologies are necessary for higher resolution [16], [17].

Conventionally, the RVI problem and the GTA for DSA problem are handled in two separate stages. Apparently, if the distribution of vias and redundant vias is locally very dense after RVI, assigning vias to regular-shaped DSA guiding templates is very difficult without any design rule violations. Hence, considering the two stages simultaneously is necessary [18]. Fang *et al.* [18] first simultaneously considered the RVI and GTA for DSA problem. Based on all possible GTA candidates (GTACs) (via patterns) for every via or RVC, they proposed two ILP formulations to maximize the manufacture rate (MR) and the insertion rate (IR). In addition, they proposed a graph-based method to obtain a heuristic solution by solving the maximum-independent set problem greedily.

To improve the IR and the MR, Fang and Hong [19] introduced metal wire perturbation to the RVI and GTA for DSA-SP problem. By perturbing some metal wires, it becomes more free for the insertion of redundant vias, but in the cost of increasing the wirelength. Hung *et al.* [20] introduced dummy vias for further improving the IR and the MR. By inserting some dummy vias, more guiding templates can be used for patterning vias. For a dense layout, it may not be easy to obtain a good enough IR and MR under SP, and hence MP may be needed. Ou *et al.* [14] and Shim *et al.* [21] investigated the RVI and GTA for DSA with MP problem and gave respective ILP formulations based on all possible GTACs. The objectives of ILPs in [14] and [21] are different. The ILP in [14] maximizes both the MR and the IR, while the ILP in [21] only maximizes the IR.

In summary, most of the existing works [14], [18]–[21] on the RVI and GTA for DSA problem have the following two crucial issues: 1) the solution space is constructed by finding all possible GTACs, which may be too large, especially for dense layout and 2) to solve the problem, the proposed ILP formulations are directly solved by commercial solvers. Apparently, these ILP solvers may be inefficient in solving the problem with large scale and dense layout.

In this paper, we focus on the RVI and GTA for DSA-MP problem, considering three scenarios: SP, double patterning (DP), and triple patterning (TP). We propose a more effective and efficient solution expression and optimization method for the three problems. Our main contributions are summarized as follows.

1) Unlike GTAC-based solution expression, we express solution by introducing the concept of $multiplet$ to discard redundant solutions. Honoring the compact solution expression, we construct a new ILP formulation, which has less variables and constraints than GTAC-based ILPs.
2) For SP, we construct a new conflict graph based on $multiplet$s and formulate the problem as a constrained maximum weight-independent set (CMWIS) problem. Under the assumption that a redundant via cannot be inserted if its related via is not manufacturable, we prove that the CMWIS problem is equivalent to the initial problem.
3) To make a good tradeoff between solution quality and runtime, we introduce a fast linear interpolation algorithm to solve the MWIS problem, which can obtain a local optimal solution. To avoid an undesirable local optimum, we propose an effective initial solution generation method. Furthermore, we reduce the CMWIS problem to an MWIS problem, such that it can be tackled by the linear interpolation algorithm.
4) For DP/TP, we propose a new solution flow, which is a two-stage method. At the first stage, a contraction graph is constructed, and the contracted vertices are assigned to two or three masks. At the second stage, the solver for the SP is called to achieve a desirable RVI and GTA for every mask.
5) Experimental results show that our algorithm for the problem with SP is faster than the methods in previous works, and our two-stage method for the problem with DP/TP is much faster than the method in [14]. Moreover, the obtained results are better than those of the compared methods.

The rest of this paper is organized as follows. Section II shows the RVI and GTA for DSA-MP problem, and our solution flow. In Section III, we construct a conflict graph on the grid model. In Sections IV and V, we detail our solution methods for the problem with SP and DP/TP, respectively. Experimental comparisons are made in Section VI, followed by the conclusion in Section VII.

II. Preliminaries

In this section, we first introduce the problem handled in this paper, and then we describe our solution flow.

A. Problem Formulation

In this paper, we consider the problem on the grid graph. For a via v_i, the position of an RVC of v_i should satisfy one of the following conditions: 1) no via and metal wire occupy

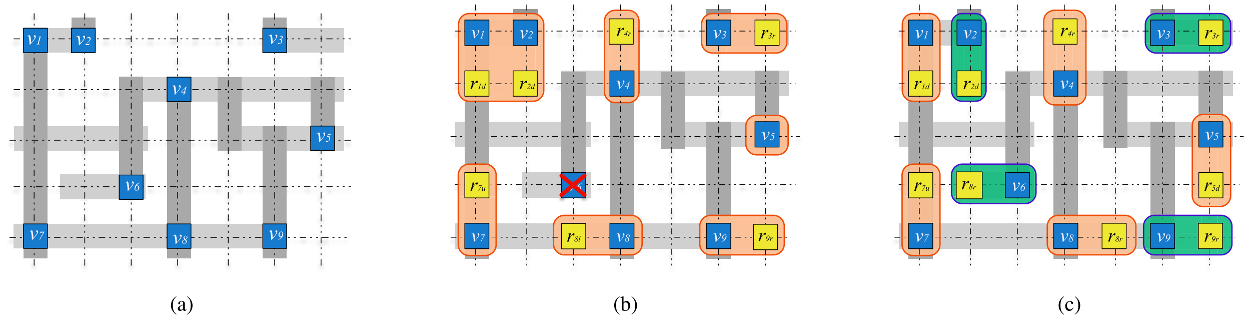


Fig. 3. Example for the RGDS and RGDD problems. (a) Layout with a via layer and two metal layers. (b) Result of the RGDS problem. (c) Result of the RGDD problem.

the position and 2) only one metal wire occupies the position, and the metal wire is on the same net as via v_i. We only need to insert one redundant via for a via. For high resolution and focal depth of guiding templates, the spacing between any two neighboring guiding templates should not be less than the optical resolution limit spacing d_s.

The problem aims at finding redundant vias for vias, and manufacturing all vias and their redundant vias by the DSA-MP technique. The MR and the IR [14], [18] are introduced to evaluate the effectiveness of previous methods, whose definitions are described as follows.

Definition 1 (Manufacture Rate): The MR is the ratio of the number of manufacturable original vias to the number of vias.

Definition 2 (Insertion Rate): The IR is the ratio of the number of inserted redundant vias to the number of vias.

Since via manufacturability is generally the first consideration for yield, an inserted redundant via should not cause generation of an infeasible via pattern [18]. Hence, we make Assumption 1.

Assumption 1: A redundant via cannot be inserted, if its related via is not manufacturable.

Under Assumption 1, the RVI and GTA for DSA with MP (RGDM) problem is formulated as follows.

Problem 1 (RGDM): Given a postrouting layout with via layers, the usable guiding templates, and M masks, insert redundant vias for vias, assign vias and inserted redundant vias to guiding templates, and assign these guiding templates to M masks, such that: 1) the inserted redundant vias are legal and 2) the spacing between any two neighboring guiding templates should not be less than the optical resolution limit spacing d_s. The objective is maximizing $\mathrm{MR} + \beta \cdot \mathrm{IR}$, where β is a weighting parameter.

Depending on the number of masks, i.e., the value of M, the RGDM problem has different versions, such as RVI and GTA for DSA with DS (RGDS), DP (RGDD), and TP (RGDT) problems. Fig. 3 gives an example of the RGDS and the RGDD problems, respectively.

B. Solution Flow

Our solution flow for the RGDM problem is shown in Fig. 4. There are three parts in the flow, i.e., preprocessing, the RGDS solver, and the RGDD/RGDT solver.

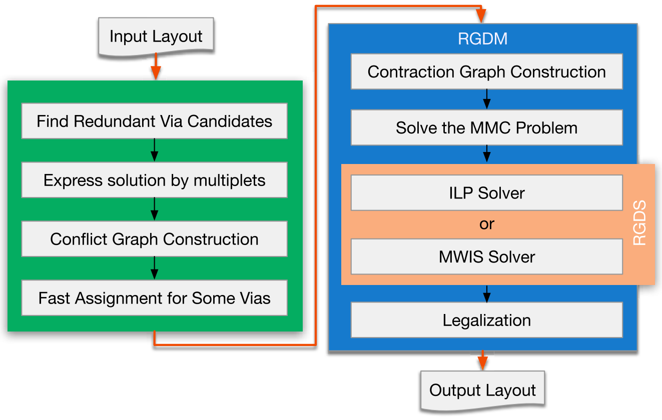

Input Layout
Find Redundant Via Candidates
Express solution by multiplets
Conflict Graph Construction
Fast Assignment for Some Vias
RGDM
Contraction Graph Construction
Solve the MMC Problem
ILP Solver
or
MWIS Solver
RGDS
Legalization
Output Layout

Fig. 4. Our flow of the graph-based method.

In the preprocessing, first, we find all RVCs for every via. Based on these vias and RVCs, we construct some $multiplet$s. Then, we consider every $multiplet$ as a vertex and construct a conflict graph on the grid model. Detailed definitions are described in Section III. Based on the conflict graph, we propose a greedy assignment algorithm for obtaining an optimal assignment of some vias to some guiding templates. After that, we obtain a number of connected components of the conflict graph.

For the RGDS problem, we formulate it as a CMWIS problem, which is an ILP. We have two approaches to solve the problem: 1) use an ILP solver to obtain an optimal solution and 2) use the MWIS solver in Section IV to obtain a good-enough solution, which is a very fast algorithm.

For the RGDD/RGDT problems, we propose a two-stage method. At the first stage, we construct a contraction graph and formulate the problems as the max-M-cut (MMC) problems for obtaining the mask assignment results. At the second stage, we call our RGDS solver to obtain an RVI and GTA for every mask.

III. Conflict Graph Construction on Grid

In this section, first, we find all RVCs for every via. Then, we introduce some $multiplet$s based on these RVCs. At last, we construct a conflict graph, in which vertices are $multiplet$s.

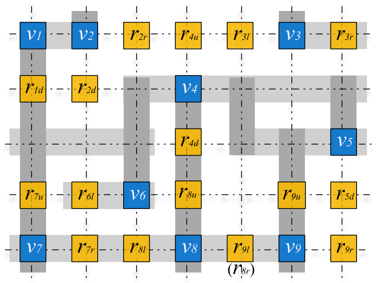


Fig. 5. All RVCs of the layout in Fig. 3(a).

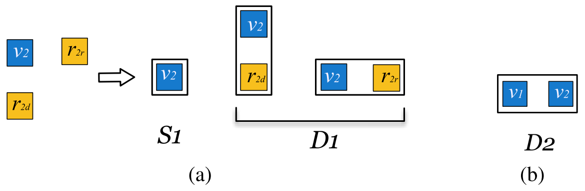


Fig. 6. (a) S_1 and D_1s of via v_4. (b) D_2.

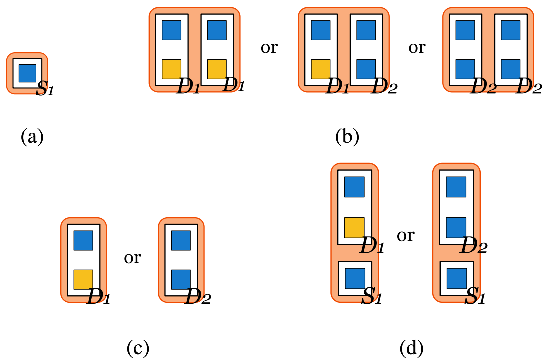


Fig. 7. All possible combinations of multiplets to form four guiding template types. Combinations of multiplets for (a) t_1, (b) t_4, (c) t_2, and (d) t_3.

According to the introduction of RVC in Section II-A, we can easily find all RVCs in time $\mathcal{O}(n)$, where n is the number of vias. All RVCs of the layout in Fig. 3(a) are presented in Fig. 5. For example, r_{1d} in Fig. 5 is the RVC of via v_1, where the index "d" denotes that r_{1d} is under v_1. Moreover, we use the indices "u," "l," and "r" to denote that an RVC is on the top, left, and right of a via, respectively.

According to Assumption 1, we make the following assumption as in [14], which is a necessary condition of Assumption 1 for the RGDS problem, but not for the RGDD/RGDT problems

Assumption 2: A via and its redundant via (if existing) should be assigned to the same guiding template.

After finding all RVCs for every via, we construct a conflict graph for the via layer layout. Suppose that the RVCs of via v_i are r_{iu}, r_{id}, r_{il}, and r_{ir}. We group every RVC with v_i and denote it as $doublet$. Then, we have four $doublet$s: $\{v_i, r_{iu}\}$, $\{v_i, r_{id}\}$, $\{v_i, r_{il}\}$, and $\{v_i, r_{ir}\}$. Via v_i is called a $single$. The four $doublet$s and the $single$ $\{v_i\}$ denote the five possible cases of inserting a redundant via for v_i. If $doublet$ $\{v_i, r_{iu}\}$ is chosen, then r_{iu} is inserted on the upper side of v_i; if $single$ $\{v_i\}$ is chosen, then v_i does not have any RVI. The RVCs, the $doublet$s, and the $single$ of v_2 in Fig. 5 are shown in Fig. 6(a). In addition, we introduce another type of $doublet$, which is composed of two close and aligned vias. As shown in Fig. 6(b), the $doublet$ is composed of vias v_1 and v_2. Let D_1 denote the $doublet$ composed of a via and a redundant via, let D_2 denote the $doublet$ composed of two vias, and let S_1 denote the $single$. Every one of these $doublet$s and $single$ is called collectively a $multiplet$.

We regard the above every $multiplet$ as a vertex in the conflict graph. Based on these vertices, we introduce some edges between them.

Definition 3 (Overlap Edge): If $multiplet$s i and j are overlapped with each other, then there exists an overlap edge e_{ij} between them. Let E_O be the set of overlap edges.

Definition 4 (Conflict Edge): If the distance between two $multiplet$s i and j is within the optical resolution limit spacing, and there does not exist an overlap edge between them, then there exists a conflict edge e_{ij} between i and j. Let E_C be the set of conflict edges.

According to Assumptions 1 and 2, we have some observations: 1) a guiding template may include vias only, but it may not include only redundant vias; 2) the number of redundant vias in a guiding template must not be larger than the number of vias in the guiding template; and 3) for every redundant via in a guiding template, its via must also be in the same guiding template. Then, the usable guiding template types in Fig. 2 are of the following combinations: t_1 includes a via, i.e., an S_1; t_2 includes a via and a redundant via, i.e., a D_1, or includes two vias, i.e., a D_2; t_3 includes two vias and a redundant via, i.e., a D_1 and an S_1, or includes three vias, i.e., a D_2 and an S_1; t_4 includes two vias and two redundant vias, i.e., two D_2s, or includes three vias and a redundant via, i.e., a D_1 and a D_2, or it includes four vias, i.e., two D_2s. All possible combinations of $multiplet$s are shown in Fig. 7. Note that, it seems that the role of D_2 can be replaced by grouping two S_1. However, D_2 is an essential $multiplet$ for handling the number of $multiplet$s in every guiding template. Using D_2, all our usable guiding templates can be constructed by one or two $multiplet$s. This is helpful for reducing the size of solution space.

Definition 5 (Template Edge): For two $multiplet$s i and j, suppose that at least one of them is not S_1. If i and j can be assigned to the same guiding template without any design rule violation, and between i and j there exists a conflict edge $e_{ij} \in E_C$, then e_{ij} is also called a template edge. Let E_T be the set of template edges. Obviously, $E_T \subseteq E_C$.

Definition 6 (Conflict Graph): The conflict graph $CG(V, E)$ is an undirected graph, where vertex $v \in V$ denotes a $multiplet$, $e_{ij} \in E$ is an edge, and $E = (E_C - E_T) \cup E_O$. E_C, E_T, and E_O are the sets of conflict edges, template edges, and overlap edges, respectively.

The conflict graph of vias v_1, v_2, v_3, and v_4 of the layout in Fig. 3(a) is constructed, as shown in Fig. 8 [for clearness, we omit the conflict graph of other vias in Fig. 3(a)]. In Fig. 8, nodes a, c, g, and l are S_1, nodes b, d, e, h, i, j, and k are D_1, and node f is D_2. The red lines are the conflict edges, the black edges are the overlap edges, and the green dotted edges are the template edges. The $multiplet$s corresponding to vertices a, b, c, d, and f are shown in the box. In Fig. 8, vertices a and b are overlapped with each other at v_1, and thus

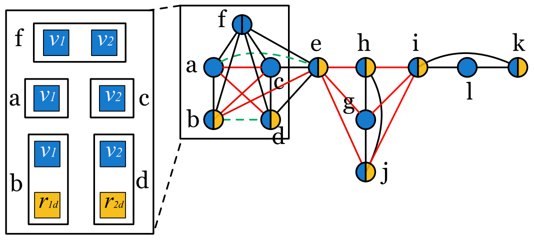


Fig. 8. Conflict graph CG for vias v_1, v_2, v_3, and v_4 of the layout in Fig. 3(a).

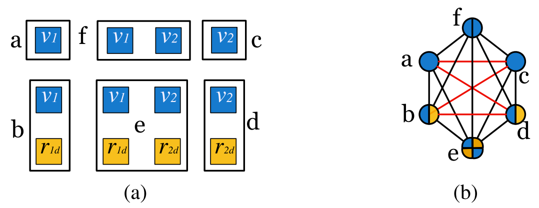


Fig. 9. Graph constructed based on GTACs for vias v_1 and v_2 of the layout in Fig. 3(a). (a) GTACs. (b) Conflict graph based on GTACs.

there is an overlap edge between them. The distance between vertices b and c is within the optical resolution limit spacing (one grid), and thus a conflict edge is generated between them. Since vertices b and d can be assigned to a quadruple guiding template as in Fig. 3(b), there is a template edge between b and d.

It must be remarked that introducing $multiplet$s may reduce the size of a conflict graph. In the previous works [14], [18], [21], the constructed graphs are based on all the possible GTACs instead of $multiplet$s. In a layout, the number of GTACs is greater than the number of $multiplet$s, so does the sizes of the corresponding graphs. For example, all the possible GTACs of vias v_1 and v_2 in Fig. 3(a) are shown in Fig. 9(a), while the graph based on these GTACs is shown in Fig. 9(b). The graph in Fig. 9(b) is larger and denser than the partial conflict graph in the box of Fig. 8. Especially, when the number of vias in a structure is larger, using $multiplet$ would benefit more.

Before solving the RGDM problem, we introduce a fast GTA and RVI for some vias. On the conflict graph $CG(V, E)$, if a vertex is D_1, and there is no conflict edge incident to it, then the via and the redundant via in the D_1 are assigned to a guiding template, and the adjacent vertices of the vertex connected by overlap edges are removed from the conflict graph. After that, we calculate the connected components, and the RGDM problem is considered on the connected components one by one.

IV. Algorithms for RGDS Problem

In this section, we describe the technical details of solving the RGDS problem. First, we formulate the RGDS problem as a CMWIS problem and give the corresponding ILP formulation. Then, we prove the equivalence between the RGDS problem and the CMWIS problem, through which the RGDS problem can be solved by the ILP solver. In addition, for fast solving the CMWIS problem, we introduce a fast algorithm for the MWIS problem and reduce the CMWIS problem to the MWIS problem.

A. Constrained Maximum Weight-Independent Set Problem

In the conflict graph of the RGDS problem, if between two $multiplet$s i and j there is an $e_{ij} \in E_O$, then only one of the $multiplet$s can be chosen. Furthermore, for the RGDS problem, if two $multiplet$s i and j are within the optical resolution limit spacing, i.e., $e_{ij} \in E_C$, then only one of them can be patterned, unless the two $multiplet$s are assigned to the same guiding template, i.e., $e_{ij} \notin E_T$. Hence, the RGDS problem is similar to the independent set problem.

The objective of the RGDS problem is maximizing the weighted sum of MR and IR, i.e., maximizing the weighted sum of the number of manufacturable vias and the number of inserted redundant vias. We jointly consider MR and IR by assigning weight to every vertex in CG as

$$w_i = \begin{cases} 1+\beta, & \text{if } i \text{ is a } D_1 \\ 2, & \text{if } i \text{ is a } D_2 \\ 1, & \text{if } i \text{ is an } S_1 \end{cases} \tag{1}$$

where w_i is the weight of vertex i and β is a parameter, which is used to balance MR and IR. Let W be the set of weights, and then the conflict graph $CG(V, E)$ is weighted and is written as $CG(V, E, W)$.

Then, we formulate the RGDS problem as the MWIS problem

$$\max_{x} \sum_{i \in V} w_i x_i \tag{2}$$

$$\text{s.t. } x_i + x_j \le 1 \quad \forall e_{ij} \in E \tag{2a}$$

$$x_i \in \{0, 1\} \quad \forall i \in V. \tag{2b}$$

In the problem, constraint (2a) means that if between vertices i and j there exists $e_{ij} \in E_O$ or $e_{ij} \in E_C - E_T$, then at most one of them can be patterned, i.e., $x_i = 1$, $x_j = 0$, or $x_i = 0$, $x_j = 1$, or $x_i = 0$, $x_j = 0$.

It must be remarked that the MWIS problem is not equivalent to the RGDS problem. Some special structures in a layout make the equivalence not hold. These structures have a feature that every two of several close $multiplet$s can be assigned to the same guiding template, but all of these $multiplet$s cannot be included in the same guiding template. We present two examples in Fig. 10. In Fig. 10(a), i, j, and k are three $multiplet$s which are D_1, e_{ij} and e_{jk} are template edges, and there is no edge between i and k. For this structure, i and j can be assigned to a t_4 guiding template, and so does j and k, but all of i, j, and k cannot be assigned to the same guiding template. However, by solving the MWIS problem on the structure in Fig. 10(a), we get an optimal solution $x_i = 1$, $x_j = 1$, and $x_k = 1$, which means i, j, and k can be patterned simultaneously. However, one of the three $multiplet$s cannot be patterned, since this is the RGDS problem and all of the three $multiplet$s cannot be assigned to the same guiding template. For the structure in Fig. 10(a), this situation still holds. We call the structures in Fig. 10 as incompatibility structure (INC), which can be defined as follows.

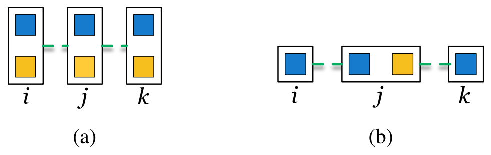


Fig. 10. Two kinds of incompatibility structures.

Definition 7 (INC): For the RGDS problem, the incompatibility structure is a structure composed of three *multiplet*s i, j, and k, in which e_{ij} and e_{jk} are template edges and there does not exist any edge between i and k.

It must be noted that the definition of INC only includes three *multiplet*s, since any one of the four usable guiding templates includes at most two *multiplet*s, as shown in Fig. 7. If there exists another shape of guiding template, we still can define a corresponding INC.

In order to exclude solutions of the MWIS problem, which are infeasible for the RGDS problem due to INC, we add the following constraint to the MWIS problem:

$$x_i + x_j + x_k \leq 2, \quad \text{if } i, j \text{ and } k \text{ compose an INC.} \tag{3}$$

Then, we call the new problem as a CMWIS problem.

Now, we show the relationship between the CMWIS problem and the RGDS problem. Before that, we show the relationship between Assumptions 1 and 2 for the RGDS problem.

Lemma 1: For the RGDS problem, Assumption 2 "a via and its redundant via (if existing) should be assigned to the same guiding template," is a necessary condition of Assumption 1 "a redundant via cannot be inserted if its related via is not manufacturable."

Proof: Under Assumption 1, suppose Assumption 2 does not hold. Then, an inserted redundant via is not in the same guiding template as its via. Obviously, there exists a conflict edge between the via and its redundant via due to the spacing rule. However, for the RGDS problem, only a mask can be used to pattern guiding templates. Thus, the via cannot be patterned, which contradicts Assumption 1. Hence, Assumption 2 holds. ■

Theorem 1: The CMWIS problem is equivalent to the RGDS problem.

Correctness of Theorem 1 is explained as follows. In the CMWIS problem, all possible RVCs and GTAs (under Assumption 1) are considered. After obtaining a solution of the CMWIS problem, we only need to assign the vertices with $x_i = 1$ and connected by template edges to a guiding template, and then we obtain a solution of the RGDS problem. Forced by constraints (2a) and (3), the obtained GTAs must be legal. Furthermore, the objective of the RGDS problem is maximizing MR $+\beta\cdot$IR, which is equivalent to maximizing $N_M + \beta \cdot N_I$, where N_M is the number of patterned vias and N_I is the number of inserted redundant vias. According to the definition of *multiplet*s and the weighting rule of *multiplet*, we have $\sum_{i\in V} w_i x_i = N_M + \beta \cdot N_I$.

Compared with the via layer layout, the conflict graph of the CMWIS problem has much more vertices and has too many edges introduced for constraints. Hence, solving the CMWIS problem on a large dense graph is time-consuming. However, after fast assignment of some vias to some guiding templates, the conflict graph is divided into a number of connected components. Thus, it is possible to solve the CMWIS problem by an ILP solver.

B. Fast Algorithm for the MWIS Problem

To solve the CMWIS problem faster, we introduce a fast algorithm for the MWIS problem. Then, we reduce the CMWIS problem to the MWIS problem in Section IV-C. Many fast algorithms have been proposed to fast obtain good-enough solutions. Among them, we introduce the algorithm in [22] and improve it for our usage.

The MWIS problem can be reformulated as an integer quadratic program. Detailed derivation is as follows. The ILP formulation (2) of the MWIS problem is equivalent to

$$\max_{\mathbf{x}} \ \mathbf{w}^\top \mathbf{x} \tag{4}$$
$$\text{s.t.} \ \ x_i x_j = 0 \quad \forall e_{ij} \in E \tag{4a}$$
$$x_i \in \{0, 1\} \quad \forall i \in V \tag{4b}$$

where $\mathbf{w} = (w_1, w_2, \ldots, w_n)^\top \in \mathbb{R}^n$, $x = (x_1, x_2, \ldots, x_n)^\top \in \{0, 1\}^n$, $n = |V|$. Let $\mathbf{A} = (A_{ij})$ be the adjacency matrix with $A_{ij} = 1$ if $e_{ij} \in E$ and $A_{ij} = 0$ if $e_{ij} \notin E$. Then, the IP formulation (4) can be rewritten as

$$\max_{\mathbf{x}} \ \mathbf{w}^\top \mathbf{x} \tag{5}$$
$$\text{s.t.} \ \mathbf{x}^\top \mathbf{A} \mathbf{x} = \mathbf{0} \tag{5a}$$
$$x_i \in \{0, 1\} \quad \forall i \in V. \tag{5b}$$

Putting constraint (5a) into the objective function, we get

$$\max_{\mathbf{x}} \ f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} - \frac{1}{2}\alpha \mathbf{x}^\top \mathbf{A} \mathbf{x} \tag{6}$$
$$\text{s.t.} \quad x_i \in \{0, 1\} \quad \forall i \in V \tag{6a}$$

where $\alpha > 0$ is a regularization parameter. Since the adjacency matrix $\mathbf{A}$ may not be positive semidefinite, $f(\mathbf{x})$ may not be a concave function.

Let $\hat{\mathbf{x}}$ and $\mathbf{x}^* \in \{0, 1\}^n$ be a candidate solution and a solution of problem (6), respectively, and let $\mathbf{y} \in [0, 1]^n$ be a point in the continuous domain. Based on problem (6), the MWIS algorithm visits a sequence of continuous points $\{\mathbf{y}^{(t)}\}$ in iterations $t = 0, 1, 2, \ldots$, where $\mathbf{y}^{(t)} \in [0, 1]^n$, and finds discrete candidate solutions $\hat{\mathbf{x}}^{(t)}$ in the respective neighborhoods of $\mathbf{y}^{(t)}$, until convergence. Let $h(\mathbf{y}, \mathbf{y}^{(t)})$ be the first-order Taylor series approximation of $f(\mathbf{y})$ at $\mathbf{y}^{(t)}$, i.e.,

$$\begin{aligned} h(\mathbf{y}, \mathbf{y}^{(t)}) &= f(\mathbf{y}^{(t)}) + (\mathbf{y} - \mathbf{y}^{(t)})^\top (\mathbf{w} - \alpha \mathbf{A} \mathbf{y}^{(t)}) \\ &= \mathbf{y}^\top (\mathbf{w} - \alpha \mathbf{A} \mathbf{y}^{(t)}) + \text{constant} \end{aligned} \tag{7}$$

where "constant" does not depend on $\mathbf{y}$. Since the approximation $h(\mathbf{y}, \mathbf{y}^{(t)})$ is linear in $\mathbf{y}$ and simpler than $f(\mathbf{y})$, we can easily obtain a discrete maximizer $\hat{\mathbf{x}}^{(t)} = \text{argmax}_{\mathbf{y}\in\{0,1\}^n} h(\mathbf{y}, \mathbf{y}^{(t)})$ by assigning

$$\hat{x}_i^{(t)} = \begin{cases} 1, & \text{if } (\mathbf{w} - \alpha \mathbf{A} \mathbf{y}^{(t)})_i > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

Then, $\mathbf{y}^{(t)}$ is mainly updated by a linear interpolation of $\mathbf{y}^{(t)}$ and $\hat{\mathbf{x}}^{(t)}$, i.e., $\mathbf{y}^{(t+1)} = \mathbf{y}^{(t)} + \eta(\hat{\mathbf{x}}^{(t)} - \mathbf{y}^{(t)})$, where η

Algorithm 1 Initial Solution Generation

```
Input: A connected component of $CG(V, E, W)$;
Output: Initial solution $\hat{\mathbf{x}}^{(0)}$ of the MWIS problem;
1: repeat
2:     Compute $w_s(k)$ by (10), $\forall k \in V$;
3:     $\hat{x}_i^{(0)} \leftarrow 1$, where $i$ = argmin$_{k\in V} w_s(k)$;
4:     for every $j$ in $V$ with $e_{ji} \in E$ do
5:         $\hat{x}_j^{(0)} \leftarrow 0$, and $V \leftarrow V - \{j\}$;
6:     end for
7:     $V \leftarrow V - \{i\}$;
8: until $V = \emptyset$
```

is an interpolation parameter with $\eta \in [0, 1]$ and satisfying $df(\mathbf{y}^{(t+1)})/d\eta \geq 0$. This ensures that the sequence $\{f(\mathbf{y}^{(t)})\}$ is nondeceasing. Brendel and Todorovic [22] provided a closed-form solution of η as

$$\eta = \min\left\{\max\left\{\frac{(\mathbf{w} - \alpha \mathbf{A}\mathbf{y}^{(t)})^\top(\hat{\mathbf{x}}^{(t)} - \mathbf{y}^{(t)})}{\alpha(\hat{\mathbf{x}}^{(t)} - \mathbf{y}^{(t)})^\top \mathbf{A}(\hat{\mathbf{x}}^{(t)} - \mathbf{y}^{(t)})}, 0\right\}, 1\right\} \quad (9)$$

and proved that the sequence $\{\mathbf{y}^{(t)}\}$ converges to a local optimal solution of problem (2).

However, from (8), we can see that the solution quality of $\hat{\mathbf{x}}^{(t)}$ is highly dependent on $\mathbf{y}^{(0)}$. If we choose a bad $\mathbf{y}^{(0)}$, then the obtained local optimal value $f(\mathbf{x}^*)$ may be far away from the global optimal value. Hence, in order to obtain a better solution, we must have a good initial solution $\hat{\mathbf{x}}$ for this method.

We propose a greedy-based algorithm to obtain a good enough initial solution $\hat{\mathbf{x}}^{(0)}$ as in Algorithm 1. In line 2 of the algorithm, the selection weight $w_s(k)$ is computed by

$$w_s(k) = \begin{cases} d_c(k) - d_t(k), & \text{if } k \text{ is } D_1 \\ N + d_c(k) - d_t(k), & \text{otherwise} \end{cases} \quad (10)$$

where $d_c(k)$ is the number of conflict edges incident to vertex k and $d_t(k)$ is the number of template edges incident to vertex k. The selection weight $w_s(k)$ is used to evaluate the degree of conflict of vertex k. N is a number larger than all d_c. This setting indicates that we always prefer D_1s over other *multiplet*s. Since the maximal degree of vertices is a constant for the conflict graph, the runtime complexity of Algorithm 1 is $\mathcal{O}(|V|)$. Based on the previous analysis, a fast algorithm for the MWIS problem is outlined in Algorithm 2.

Lines 16–23 in Algorithm 2 are used to obtain a final feasible solution $\mathbf{x}^*$ of the MWIS problem. The complexity per iteration of Algorithm 2 is only $\mathcal{O}(|E|)$, and empirically, the iteration between lines 3 and 15 converges in only a few number of iterations [22]. Moreover, according to our experiments, even Algorithm 2 can obtain optimal solutions of the RGDS problem on many connected components. Fig. 11 shows an optimal solution of the RGDS problem on a partial graph (i.e., for vias v_1, v_2,...,v_4 only) of the conflict graph shown in Fig. 8, which is obtained by Algorithm 2. The corresponding result is shown in Fig. 3(b).

Algorithm 2 MWIS Algorithm

```
Input: A connected component of $CG(V, E, W)$,
       convergence threshold $\delta$, and $\alpha \leftarrow 2$.
Output: Solution $\mathbf{x}^*$ of the MWIS problem.
1: Initialize $t \leftarrow 0$, and $\mathbf{y}^{(0)} \in [0, 1]^n$, $\mathbf{y}^{(0)} \neq \mathbf{0}$;
2: Generate $\hat{\mathbf{x}}^{(0)}$ by Algorithm 1, and $\mathbf{x}^* \leftarrow \hat{\mathbf{x}}^{(0)}$;
3: repeat
4:     if $f(\hat{\mathbf{x}}^{(t)}) \geq f(\mathbf{y}^{(t)})$ then
5:         $\mathbf{y}^{(t+1)} \leftarrow \hat{\mathbf{x}}^{(t)}$;
6:     else
7:         Obtain $\eta$ by (9);
8:         $\mathbf{y}^{(t+1)} \leftarrow \mathbf{y}^{(t)} + \eta(\hat{\mathbf{x}}^{(t)} - \mathbf{y}^{(t)})$;
9:     end if
10:    Obtain $\hat{\mathbf{x}}^{(t+1)}$ by (8);
11:    if $f(\hat{\mathbf{x}}^{(t+1)}) \geq f(\mathbf{x}^*)$ then
12:        $\mathbf{x}^* \leftarrow \hat{\mathbf{x}}^{(t+1)}$;
13:    end if
14:    $t \leftarrow t + 1$;
15: until $||\mathbf{y}^{(t+1)} - \mathbf{y}^{(t)}|| < \delta$;
16: $V^1 \leftarrow \{i : x_i^* = 1, i = 1, 2, \cdots, n\}$;
17: repeat
18:    $x_{i_0}^* \leftarrow 1$, where $i_0$ = argmax$_{i\in V^1} w_i$;
19:    for every $j$ in $V^1$ with $e_{ji_0} \in E$ do
20:        $x_j^* \leftarrow 0$, and $V^1 \leftarrow V^1 - \{j\}$;
21:    end for
22:    $V^1 \leftarrow V^1 - \{i_0\}$;
23: until $V^1 = \emptyset$
```

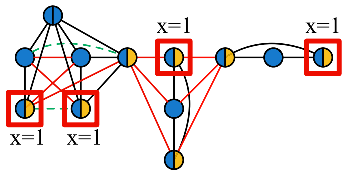


Fig. 11. Optimal solution of the MWIS problem on the conflict graph in Fig. 8.

C. Reduction of CMWIS to MWIS

According to the analyses in Section IV-A, the CMWIS problem is composed of the MWIS problem and constraint (3). According to our statistics, most of the connected components obtained after the fast assignment for some vias stage do not have incompatibility structure. Hence, constraint (3) is not active on these connected components, and we only need to solve the MWIS problem using Algorithm 2 instead of solving the CMWIS problem.

However, on the connected components with incompatibility structure, we reduce the CMWIS problem to the MWIS problem by restricting constraint (3) to constraint (2a), which is by deleting a template edge in every INC from the conflict graph CG. Note that, the deleted template edge e_{ij} of an INC appears in E as a conflict edge due to $E = E_C - E_T$, and i and j should satisfy constraint (2a). Consequently, combining constraint (2a) and the other vertex in the INC implies that constraint (3) holds. Naturally, we use the following two template edge deletion criteria: first, we delete template edges as

Algorithm 3 Incompatibility Structure Elimination

```
Input: A connected component ($CC\_CG$) of $CG$ with INC;
Output: A $CC\_CG$ of $CG$ without INC;
1: Construct the incompatibility graph $IG$ of $CC\_CG$;
2: Find all connected components ($CC\_IGs$) of $IG$;
3: for every $CC\_IG$ do
4:     Extract all $multiplets$ in $CC\_IG$;
5:     Construct the template graph $TG$;
6:     Solve maximum weight matching on $TG$;
7:     Mark all un-matched template edges, and delete them
       from $CC\_CG$;
8: end for
```

few as possible; second, we prior delete those template edges which connect two *multiplets* with low probability being assigned to the same guiding template. The incompatibility structure eliminating algorithm is outlined in Algorithm 3.

Definition 8 (Incompatibility Graph): The incompatibility graph $IG(I, E_I)$ is an undirected graph, where node $I_i \in I$ is an INC. An incompatibility edge $e_{ij} \in E_I$ exists between INCs I_i and I_j if they share a template edge in the conflict graph CG.

Definition 9 (Template Graph): The template graph TG(T, E_T, W_T) is an induced subgraph of conflict graph CG by template edges, where node $T_i \in T$ is a *multiplet* in an INC. An edge $e_{ij} \in E_T$ exists between nodes T_i and T_j if $e_{ij} \in E_T$ in CG. $w^t_{ij} \in W^T$ is the weight of template edge e_{ij}, which is set as (11).

For line 1 of Algorithm 3, the detailed definition of incompatibility graph IG is described in Definition 8. An example for a connected component CC_IG of IG is shown in Fig. 12(b), which is composed of two INCs $I_i(i, j, k)$ and $I_j(j, k, l)$, where I_i is composed of *multiplets* i, j, and k, and I_j is composed of *multiplets* j, k, and l. Template edge e^t_{jk} is in both of I_i and I_j. Thus, there is an incompatibility edge between I_i and I_j, and I_i and I_j are in the same connected component of IG. In line 5 of Algorithm 3, template graph TG is defined as in Definition 9, and the template graph of Fig. 12(b) is shown in Fig. 12(c). It must be noted that, in a TG, the degree of every *multiplet* is 1 or 2 (at least two *multiplets* are with degree 1). The detailed explanation is stated as follows. The three *multiplets* in every INC should be in a line, and any two connected INCs in a connected component of IG should share a template edge. As a result, all *multiplets* in the TG constructed by line 5 in Algorithm 3 are in a line. Thus, the TG is a bipartite graph.

In line 5 of Algorithm 3, the weight of a template edge e^t_{ij} is set as

$$w^t_{ij} = \frac{1}{\sqrt{n^t_{i-j} + n^t_{j-i}}} \tag{11}$$

where n^t_{i-j} is the number of *multiplets* which satisfies that: 1) these *multiplets* are connected to i but not to j by conflict edges in the conflict graph CG and 2) if several *multiplets* satisfy 1), but they are connected to each other by overlap edges in the conflict graph CG, then these *multiplets* are counted

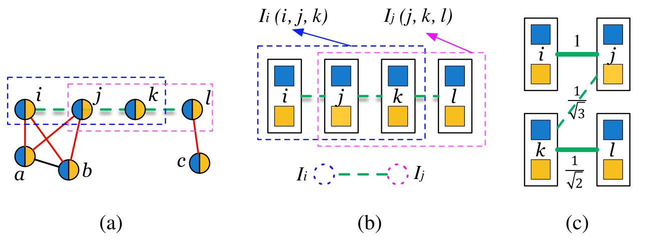


Fig. 12. (a) CG with seven *multiplets*. (b) IG with two INCs $I_i(i, j, k)$ and $I_j(j, k, l)$. (c) TG and its maximum weight matching result.

as one *multiplet*. n^t_{j-i} has the similar meaning, and at least one of n^t_{i-j} and n^t_{j-i} is not less than 1. w^t_{ij} in (11) is used to reflect the similarity of connected *multiplets*. If i and j share more connected *multiplets*, then the value of w^t_{ij} is larger. And in this case, *multiplets* i and j are more likely assigned to the same guiding template. In Fig. 12(a), *multiplets* i, a, and b are connected to j but not to k by conflict edges, and *multiplets* a and b are connected to each other by an overlap edge, and hence $n^t_{j-k} = 2$. Correspondingly, the weights of all template edges are calculated by (11), as shown in Fig. 12(c).

In order to obtain the best template edge deletion, we obtain the maximum weight matching on TG by solving a maximum flow problem. Generally, the runtime complexity of solving a maximum flow problem is $\mathcal{O}(|V|^3)$. Since the degrees of vertices in TG are 1 or 2 and the capacity of every edge in TG is 1, our TG is a unit capacity simple network. Thus, solving the maximum flow problem on our TG can be finished in $\mathcal{O}(|T|^{(1/2)} \cdot |E_T|)$. In addition, since the size of a TG is very small, we can fast obtain a maximum weight matching result and further delete those unmatched template edges from the conflict graph CG. For example, for the TG in Fig. 12(c), we can obtain a maximum weight matching (i, j) and (k, l), and then we delete the unmatched template edge (j, k) from the conflict graph. Thus, the two INCs are eliminated.

V. Algorithms for RGDD/RGDT Problems

In this section, we consider RGDD or RGDT. The RGDD and the RGDT problems can also be formulated as ILPs and then may be solved by the ILP solver. However, the work in [14] indicates that the ILP formulations have too many variables and constraints and are very hard to solve, especially on large and dense layouts. We solve the RGDD and RGDT problems in two stages. First, vias are assigned to M masks such that the vias in every mask can be easily inserted redundant vias and patterned. Second, the RGDS solver is called to assign vias and inserted redundant vias to templates for every mask.

A. Mask Assignment for RGDD/RGDT

At the first stage of our two-stage assignment method, we assign the vias on via layer to M masks, such that the vias in every mask can be easily inserted redundant vias and patterned. In order to achieve the assignment, we construct a contraction graph CoG(C, E^c, W^c).

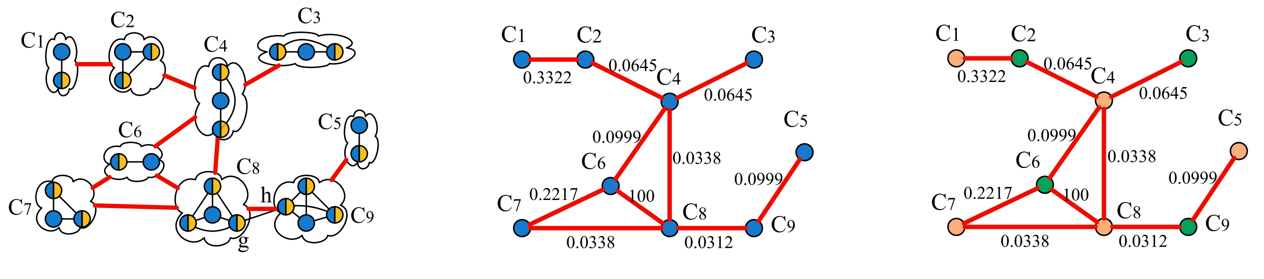


Fig. 13. (a) Process of contraction graph construction. (b) Contraction graph of the conflict graph in Fig. 8. (c) Result of the max-2-cut problem on the contraction graph in (b).

Definition 10 (Contraction Graph): The contraction graph CoG(C, E^c, W^c) is an undirected graph, where node $C_i \in C$ is composed of all S_1 and D_1s of via i. A contraction edge $e^c_{ij} \in E^c$ exists between nodes C_i and C_j if there exists a pair of *multiplets* p and $q \in V$ of CG with $e_{pq} \in E_C$, where p is an S_1 or a D_1 in node C_i, and q is an S_1 or a D_1 in node C_j. $W^c_{ij} \in W^c$ is the weight of contraction edge e^c_{ij}.

The weight $W^c_{ij} \in W^c$ of the contraction edge e^c_{ij} indicates the degree of conflict between C_i and C_j, and it is calculated by

$$W^c_{ij} = \frac{1}{\overline{W}_{ij} - \widetilde{W}_{ij} + 0.01} \tag{12}$$

where

$$\overline{W}_{ij} = \sum_{p \in C_i, q \in C_j} \widetilde{w}_{pq} \tag{13}$$

$$\widetilde{W}_{ij} = \sum_{\substack{p \in C_i, q \in C_j \\ e_{pq} \in E_C}} \widetilde{w}_{pq} \tag{14}$$

and

$$\widetilde{w}_{pq} = \begin{cases} 2 + 2\beta, & \text{if both of } p \text{ and } q \text{ are } D_1 \\ 2 + \beta, & \text{if only one of } p \text{ and } q \text{ is } D_1 \\ 2, & \text{if both of } p \text{ and } q \text{ are } S_1. \end{cases} \tag{15}$$

$\overline{W}_{ij}$ is the weight sum of all pairs of *multiples* p and q in C_i and C_j. $\widetilde{W}_{ij}$ is the weight sum of *multiplets* $p \in C_i$ and $q \in C_j$ between which there exists a conflict edge e_{pq}. Furthermore, $\overline{W}_{ij} - \widetilde{W}_{ij}$ can be seen as the degree of freedom between C_i and C_j. The weight W^c_{ij} of the contraction edge e^c_{ij} is calculated by (12), which indicates the degree of conflict between C_i and C_j, where 0.01 in the denominator is used to avoid the denominator being 0.

The contraction graph CoG of the layout in Fig. 3(a) is constructed in Fig. 13(b), where the red bold lines are contraction edges. Detailed process of contraction is shown in Fig. 13(a), where every cloud is a node. If β is set as 0.5, the weights are calculated as in Fig. 13(b). In Fig. 13(b), $W^c_{48} = 0.0338$ is the degree of conflict between nodes C_4 and C_8, which is smaller than the other edge weights. This shows that we can still insert easily redundant vias and pattern vias for vias v_4 and v_8, even though they are assigned to the same mask, while $W^c_{68} = 100$ is the degree of conflict between C_6 and C_8, which is much larger than the others. In fact, if nodes C_6 and C_8 are assigned to the same mask, then at least one of the vias v_6 and v_8 cannot be patterned due to conflicts.

After constructing the contraction graph CoG(C, E^c, W^c), we perform the first stage for the RGDD and RGDT problems. That is, the nodes in CoG are assigned to M masks ($M = 2$ or 3), which is an MMC problem on CoG. The MMC problem is an NP-hard problem. Fortunately, the number of nodes and the number of edges in CoG are much less than those of the CG. Hence, solving the MMC problem on the contraction graph by the ILP solver is possible. Before that, some graph reduction techniques are used to speed up the solver.

In this paper, we utilize three graph reduction techniques, which can keep optimality of the MMC problem: 1) vertex with degree less than M removal; 2) bridge edge removal; and 3) connected component calculation. All the three techniques have been widely used to reduce the size of a graph for a series of partition problems [23]–[25]. Here, we skip the details.

For the RGDD and RGDT problems, the corresponding max-2-cut problem and max-3-cut problem on CoG(C, E^c, W^c) are presented in ILP formulations. In the previous works [26]–[28], various ILP formulations for the MMC problem have been proposed. In this paper, the max-2-cut problem for RGDD is formulated as

$$\min_{z,c} \sum_{e^c_{ij} \in E^c} W^c_{ij} c_{ij} \tag{16}$$

$$\text{s.t. } z_i + z_j \geq 1 - c_{ij} \quad \forall e^c_{ij} \in E^c \tag{16a}$$

$$z_i + z_j \leq 1 + c_{ij} \quad \forall e^c_{ij} \in E^c \tag{16b}$$

$$z_i, c_{ij} \in \{0, 1\} \quad \forall C_i \in C. \tag{16c}$$

The max-3-cut problem for RGDT is formulated as

$$\min_{z,c} \sum_{e^c_{ij} \in E^c} W^c_{ij} c_{ij} \tag{17}$$

$$\text{s.t. } z_{1i} + z_{2i} \geq 1 \quad \forall C_i \in C \tag{17a}$$

$$z_{2i} - z_{1i} + z_{2j} - z_{1j} \leq 1 + c_{ij} \quad \forall e^c_{ij} \in E^c \tag{17b}$$

$$z_{1i} - z_{2i} + z_{1j} - z_{2j} \leq 1 + c_{ij} \quad \forall e^c_{ij} \in E^c \tag{17c}$$

$$z_{1i} + z_{2i} + z_{1j} + z_{2j} \leq 3 + c_{ij} \quad \forall e^c_{ij} \in E^c \tag{17d}$$

$$z_{1i}, z_{2i}, c_{ij} \in \{0, 1\} \quad \forall C_i \in C. \tag{17e}$$

TABLE I
COMPARISON WITH THE ISPD 2016 WORK [14] ON THE RGDS PROBLEM

Circuits	#V	SP_ILP [14]			Imp_ILP [14]			Our_MWIS			Our_ILP		
		MR(%)	IR(%)	CPU(s)	MR(%)	IR(%)	CPU(s)	MR(%)	IR(%)	CPU(s)	MR(%)	IR(%)	CPU(s)
efc	4983	76.33	75.15	0.85	80.59	76.09	2.36	80.35	75.49	0.17	80.59	76.09	1.48
ecc	5523	80.1	78.68	0.84	84.03	79.28	2.74	83.97	78.59	0.18	84.03	79.28	1.57
ffu	7026	78.49	76.77	1.17	82.23	77.32	3.35	81.69	76.39	0.25	82.23	77.32	1.90
alu	7046	74.65	72.79	1.33	81.64	75.21	3.62	81.3	75.59	0.24	81.64	75.21	2.13
byp	28847	75.14	70.21	6.31	75.24	69.1	12.55	74.81	68.35	0.76	75.24	69.1	6.20
mul	62988	70.23	68.59	30.98	75.21	69.89	38.94	74.53	68.69	2.04	75.21	69.89	11.33
Avg.	19402	75.82	73.70	6.91	79.82	74.48	10.59	79.45	73.85	0.61	79.82	74.48	4.45
Ratio		0.95	0.99	1.55	1.00	1.00	2.38	1.00	0.99	0.14	1.00	1.00	1.00

Since the size of every connected component of CoG is small, we directly use the ILP solver to solve problems (16) and (17). A max-2-cut result of CoG in Fig. 13(b) is shown in Fig. 13(c). The objective value is $0.0338+0.0338 = 0.0676$.

B. Solving the RGDS Problem on Every Mask and Legalization

In this section, we explain the second stage for the RGDD and the RGDT problems. After assigning nodes to M masks ($M = 2$ or 3), we need to consider the RGDS problem on every mask. First, we reconstruct the conflict graph $\mathrm{CG}_k(V_k, E_k)$ for every mask $k = 1, 2, \ldots, M$. Then, we solve the RGDS problem on every $\mathrm{CG}_k(V_k, E_k)$ using the methods in Section IV-B. Here, the MWIS solver is chosen as the RGDS solver.

Since our two-stage method deals with GTA and RVI on masks one by one, it may cause some illegal RVIs. For example, for vertices g and h in Fig. 13(a), if we deal with node C_8 by the RGDS solver, we may have $x_g = 1$, and when we deal with node C_9, we may also have $x_h = 1$. But there is an overlap edge between g and h due to the overlap between $multiplet$s g and h, which indicates that only one of g and h can be patterned. In order to handle this issue, we propose a trick as follows.

After GTA and RVI on a mask, the corresponding positions of vias and inserted redundant vias are marked as occupied. Then, we refind all RVCs on the remaining unoccupied positions for the other masks and reconstruct conflict graphs.

It must be remarked that, for the RGDD and RGDT problems, Lemma 1 may not hold. This means a via and its redundant via (if existing) may be assigned to different guiding templates, and Assumption 2 does not hold. Furthermore, under Assumption 2, we may miss the optimal GTAs for the RGDD and RGDT problems. However, our proposed method can be extended to consider the scenario without Assumption 2 by introducing some extra $multiplet$s. But, such consideration would greatly increase the size of solution space, and such consideration cannot always improve the solution quality. Hence, we still consider the RGDD and the RGDT problems under Assumption 2 by using the methods in Section III. In addition, our two-stage method also may lose the optimal solution. Fortunately, for the RGDD and the RGDT problems, after assigning all vias to M masks, the layout in every mask is sparse, and the GTA under Assumption 2 may be good enough. Our experimental results verify this statement.

VI. EXPERIMENTAL RESULTS

Our methods for GTA and RVI for DSA-MP were programmed in C++ and run on a personal computer with 2.7-GHz CPU, 8-GB memory and the Unix operating system. We tested our methods on the benchmarks based on the OpenSPARC T1 design [29] provided by Ou *et al.* [14], and on the MCNC benchmarks and the industry Faraday benchmarks, provided by Fang *et al.* [18]. Since the usable guiding templates in [14] and [18] are different, we designed two different experiments for fair comparisons. In both experiments, the Branch and Bound solver in the software package CPLEX [30] was chosen as our ILP solver.

A. First Experiment

We implemented our methods for the RGDS, RGDD, and RGDT problems on the OpenSPARC T1 benchmarks to compare with the methods in [14].[1] As [14], layouts of all benchmarks were transformed to grid models. The grid size was set as one metal pitch. The distance between a via and its redundant via was set as one metal pitch, and the optical resolution limit spacing d_s of adjacent guiding templates was set as two metal pitches. The available guiding template types were t_1, t_2, t_3, and t_4 as in Fig. 2. The experimental results of RGDS, RGDD, and RGDT are listed in Tables I–III, respectively, where the last two rows of every table are the average results and the ratios based on some corresponding average results.

In Table I, four solvers for the RGDS problem are compared. The data in the column "SP_ILP [14]" are the results cited directly from [14]. The column "Our_ILP" lists our results, which were obtained by solving the ILP of the CMWIS problem using the ILP solver. The results in "Our_MWIS" were obtained by using Algorithm 2. From Table I, we can see that the results obtained by our ILP are slightly better than that by the ILP in [14]. The difference is mainly generated by more RVCs in our experiments. For fair, we add another comparison by testing the ILP formulation in [14] on vias and our RVCs, and list the obtained results into the column "Imp_ILP [14]."

[1]The results of [14] are directly cited for comparisons. Their platform was a computer with Core i7 3.4-GHz CPU and 32-GB memory, and CBC was used as the ILP and LP solvers.

TABLE II
COMPARISON WITH THE ISPD 2016 WORK [14] ON THE RGDD PROBLEM

Circuits	DP_AP [14]			Our_DP		
	MR(%)	IR(%)	CPU(s)	MR(%)	IR(%)	CPU(s)
efc	98.63	96.84	3.36	98.58	97.09	2.56
ecc	98.55	97.44	3.80	99.19	98.17	2.28
ffu	98.83	97.35	4.99	98.52	97.44	3.12
alu	98.14	96.39	5.40	97.87	97.01	3.51
byp	97.29	91.35	41.41	97.34	92.11	15.85
mul	96.39	94.91	417.76	97.65	95.21	55.07
Avg.	97.97	95.71	79.45	98.19	96.17	13.73
Ratio	1.00	1.00	5.79	1.00	1.00	1.00

TABLE III
COMPARISON WITH THE ISPD 2016 WORK [14] ON THE RGDT PROBLEM

Circuits	TP_AP [14]			Our_TP		
	MR(%)	IR(%)	CPU(s)	MR(%)	IR(%)	CPU(s)
efc	99.53	97.89	7.38	99.94	99.04	1.72
ecc	99.67	98.62	7.93	99.93	99.60	1.41
ffu	99.45	97.93	12.02	99.93	99.19	1.83
alu	99.16	97.44	11.55	99.88	98.95	2.11
byp	98.98	92.88	136.69	99.83	95.42	10.10
mul	98.31	96.85	1613.23	99.78	98.41	47.12
Avg.	99.18	96.94	298.13	99.88	98.44	10.72
Ratio	0.99	0.98	27.82	1.00	1.00	1.00

In Table I, "#V" is the number of vias and "CPU(s)" is the runtime in seconds. "MR(%)" and "IR(%)" are, respectively, the MR and the redundant via IR, which are calculated by

$$\mathrm{MR} = \frac{\#\mathrm{MV}}{\#\mathrm{V}} \times 100\%, \quad \mathrm{IR} = \frac{\#\mathrm{RV}}{\#\mathrm{V}} \times 100\%$$

where #MV is the number of manufactured vias (excluding redundant vias), and #RV is the number of inserted redundant vias. MR and IR are the primary indicators for comparison. In this section, weight parameter β was set a value the same as in [14] for our algorithms[2] for balancing MR and IR.

Compared with the ILP formulation in [14], our ILP formulation can be solved faster and can obtain better results. From the row "Ratio" in Table I, it can be seen that the improvements of "Our_ILP" over "SP_ILP [14]" on the average MR and IR are 5% and 1%, respectively. And the average runtime of "SP_ILP [14]" is 1.55$\times$ more than that of "Our_ILP." Although both of the results in columns "Our_ILP" and "Imp_ILP [14]" are equal, the average runtime of "Imp_ILP [14]" is 2.38$\times$ more than that of "Our_ILP." These comparisons show that our *multiplet*-based solution expression is effective and our ILP formulation is better than that in [14] for the RGDS problem. Furthermore, comparing "Our_MWIS" with "Our_ILP," it can be found that "Our-ILP" achieves a little better average MR and IR (both of two improvements are 1%) than those of "Our_MWIS," but the average runtime is about 7$\times$ more than that of "Our_MWIS." This shows that our MWIS algorithm is fast and effective. Actually, for most of the connected components, our MWIS solver also can achieve an optimum.

Table II lists the comparison results of two solvers on the RGDD problem. "DP_AP [14]" is the approximation algorithm proposed by Ou *et al.* [14], which is based on a linear program relaxation and is much faster than the ILP-based method. "Our_DP" is our two-stage method for the RGDD problem. Both of "Our_DP" and "DP_AP [14]" achieve almost the same results. However, the average runtime of "DP_AP [14]" is 5.79$\times$ more than that of "Our_DP," which shows that "Our_DP" is a very fast method for the RGDD problem. The great runtime improvement is due to our two-stage process, in which we divide a large and dense conflict graph into many small size connected components.

In Table III, "TP_AP [14]" is the approximate algorithm proposed in [14]. "Our_TP" is our two-stage method for the RGDT problem. Comparing "Our_TP" with "TP_AP [14]," "Our_TP" obtains greater average MR and IR with the improvements 1% and 2%, respectively. Moreover, the average runtime of "TP_AP [14]" is 27.82$\times$ more than that of "Our_TP," which shows that "Our_TP" is a fast algorithm for the RGDT problem. Furthermore, the average runtime of "TP_AP [14]" in Table III is much slower than that of "DP_AP [14]" in Table II, since the numbers of variables and constraints in "TP_AP" are much more than those in "DP_AP." On the contrary, the average runtime of "Our_TP" in Table III is a little less than that of "Our_DP" in Table II. The main reason is that the conflict graph is divided into more connected components in "Our_TP" in the first stage of our two-stage process.

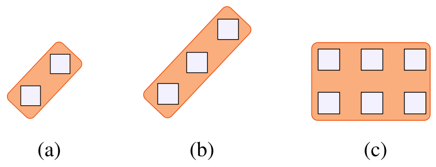


Fig. 14. Three more usable types of guiding templates. (a) t_5. (b) t_6. (c) t_7.

It must be noted that some vias and redundant vias cannot be fabricated even for TP. If the number of unfabricated vias and redundant vias is small, complementary electron beam lithography is a promising technique for further manufacturing the vias and redundant vias, which is low cost and high resolution [31]. Otherwise, if the remained unmanufacturable vias or redundant vias are numerous, then more masks might be considered.

B. Second Experiment

In the second experiment, we further compare our methods with the method in [18] on the MCNC and the industry Faraday benchmarks for the RGDS problem. The layouts of all benchmarks were also transformed to grid models, where a grid size is one metal pitch, and the optical resolution limit spacing d_s of adjacent guiding templates is set as one metal pitch too. And for fairness, the weight parameter β is set the value as in [18].

For this comparison, three more guiding template types t_5, t_6, and t_7 are used besides the types t_1, t_2, t_3, and t_4 in Fig. 2. The three guiding template types are shown in Fig. 14. In order to form the guiding templates t_5 and t_6, correspondingly, we introduce a new *multiplet* D_3, as shown in Fig. 15(a). Under Assumption 2, the possible combinations

[2]In [14], β was set as 250, while in [18], β was set as 0.01.

TABLE IV
COMPARISON WITH THE ILP IN TCAD 2017 WORK [18] ON THE RGDS PROBLEM

Circuits	#V	ILP [18]			Imp_ILP [18]			Our_ILP		
		MR(%)	IR(%)	CPU(s)	MR(%)	IR(%)	CPU(s)	MR(%)	IR(%)	CPU(s)
struct	12551	99.86	99.42	30.00	99.67	99.52	58.21	99.67	99.52	12.96
primary1	8764	99.80	99.21	28.00	99.68	99.63	43.56	99.68	99.63	9.02
primary2	32684	99.54	98.97	72.00	99.67	99.46	96.51	99.67	99.46	37.01
s5378	8649	81.10	61.78	11.00	84.09	68.87	13.55	84.09	68.87	2.80
s9234	6874	80.07	59.54	9.00	82.80	67.03	8.62	82.80	67.03	2.19
s13207	18780	84.36	66.93	23.00	84.22	70.99	29.98	84.23	70.99	5.73
s15850	22694	82.70	64.41	28.00	83.99	69.17	36.83	83.99	69.18	7.37
s38417	54225	84.01	65.71	65.00	83.45	70.70	82.10	83.45	70.70	19.47
s38584	74155	81.53	63.01	88.00	82.90	69.20	118.67	82.90	69.20	25.69
dma	34697	97.85	95.29	55.00	97.22	95.88	69.53	97.22	95.88	15.27
dsp1	30317	99.05	97.57	53.00	98.42	97.66	66.71	98.42	97.66	16.40
dsp2	31301	98.51	96.68	52.00	98.26	97.54	63.57	98.26	97.54	16.70
risc1	43858	98.78	96.93	75.00	98.18	96.93	112.82	98.18	96.93	23.73
risc2	44385	98.80	96.91	77.00	98.16	96.87	112.65	98.16	96.87	23.85
Avg.	30281	91.86	83.03	47.57	92.19	85.68	65.24	92.19	85.68	15.59
Ratio		1.00	0.97	3.05	1.00	1.00	4.19	1.00	1.00	1.00

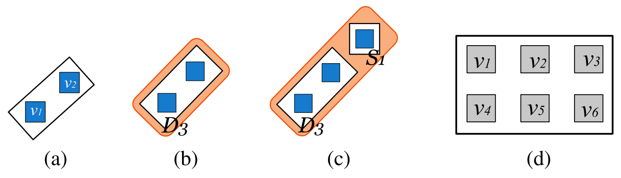


Fig. 15. (a) D_3. (b) and (c) All possible combinations of $multiplet$s to form guiding templates t_5 and t_6. (d) Sextet.

of $multiplet$s to form the guiding templates t_5 and t_6 are listed in Fig. 15(b) and (c), respectively. Then, the template edge between $multiplet$s S_1 and D_3 can be calculated. There exist some incompatibility structures for t_6 template, which are similar for t_3 in Fig. 10(b). To exclude these incompatibility structures, constraint (3) should be added into the MWIS problem (2).

In order to handle the template type t_7 with six holes, we introduce a new $multiplet$ called $sextet$, as shown in Fig. 15(d), where v_i, $i \in \{1, 2, \dots, 6\}$, can be a via or a redundant via. We use the detection windows in [18] to identify all $sextet$s, and construct a new conflict graph by adding the $multiplets$ D_3 and $sextet$ and the related edges. Then, we solve the RGDS problem by the ILP solver and the MWIS solver on the new conflict graph, respectively.

In Table IV, we list the results of three solvers "ILP [18],"[3] "Imp_ILP [18]," and "Our_ILP" on the RGDS problem, where the last two rows of the table are the average results and the ratios based on the average results of "Our_ILP." "Imp_ILP [18]" is the same as "Imp_ILP [14]" in Table I, in which the results were obtained by solving the ILP in [18] on vias and our RVCs. The results in column "Our_ILP" were obtained by using CPLEX [30] to solve our ILP formulation.

Comparing "ILP [18]" with "Our_ILP," we improve the average IR by 3%. Furthermore, the average runtime of "ILP [18]" is about 3.05× more than that of "Our_ILP." Both of the results in columns "Imp_ILP [18]" and "Our_ILP" are

[3]The results of ILP [18] are directly cited from the paper, in which the platform was Core i7 3.5 GHz with 72-GB memory, and CPLEX was used as the ILP solver.

TABLE V
COMPARISON WITH THE GRAPH METHOD IN TCAD 2017 WORK [18] ON THE RGDS PROBLEM

Circuits	Graph [18]			Our_MWIS		
	MR(%)	IR(%)	CPU(s)	MR(%)	IR(%)	CPU(s)
struct	99.86	99.06	2.60	99.62	99.37	8.63
primary1	99.80	98.71	3.20	99.54	99.24	7.89
primary2	99.55	97.73	8.00	99.48	98.98	30.19
s5378	81.11	56.89	0.30	83.72	67.88	0.37
s9234	80.07	55.34	0.30	81.80	67.47	0.29
s13207	84.35	62.35	0.80	83.13	70.34	0.87
s15850	82.71	59.65	0.90	83.56	69.27	1.26
s38417	84.00	61.08	2.40	83.31	70.71	4.75
s38584	81.52	58.23	3.10	82.75	68.48	7.83
dma	97.86	92.89	3.10	97.37	93.75	1.91
dsp1	99.06	96.14	3.70	98.58	95.99	2.25
dsp2	98.50	95.20	3.40	98.40	95.81	2.27
risc1	98.77	95.18	6.10	98.42	95.51	4.41
risc2	98.81	95.09	5.80	98.44	95.29	4.68
Avg.	91.85	80.25	3.12	92.01	84.86	5.54
Ratio	1.00	0.95	0.56	1.00	1.00	1.00

the same, but the average runtime of "Imp_ILP [18]" is about 4.19× more than that of "Our_ILP." These verify that our ILP formulation is better than that in [18] for RGDS problem, which has less variables and constraints.

The results in "Graph [18]" and "Our_MWIS" of Table V were obtained by the graph method in [18] and our MWIS solver, respectively. The average runtime of "Graph [18]" is half of "Our_MWIS," but the average IR of "Graph [18]" is 5% less than that of "Our_MWIS" and the average MR of "Graph [18]" is a little less than of that of "Our_MWIS."

To demonstrate the scalability of our ILP formulation and the proposed fast MWIS algorithm, we further tested them on nine much larger benchmarks in [18]. The experimental results are listed in Table VI. Compared with "ILP [18]," "Our_ILP" achieves 2.68× shorter runtime. Compared with the graph method in [18], our ILP-based method improves the average IR by 10%. On these larger test cases, our MWIS-based algorithm is still fast and effective. It takes only 0.28× average runtime of our ILP-based method with only 1% MR loss and 2% IR loss.

TABLE VI
COMPARISON WITH THE TCAD 2017 WORK [18] ON THE RGDS PROBLEM

Circuits	#V	ILP [18]			Graph [18]			Our_MWIS			Our_ILP		
		MR(%)	IR(%)	CPU(s)	MR(%)	IR(%)	CPU(s)	MR(%)	IR(%)	CPU(s)	MR(%)	IR(%)	CPU(s)
mgc_des_perf_1	736470	94.04	76.89	962.00	94.05	72.37	52.00	94.23	82.15	139.37	95.19	83.74	422.39
mgc_des_perf_a	765166	95.98	76.41	928.00	95.98	72.58	60.00	96.21	79.70	112.42	96.69	81.18	393.14
mgc_des_perf_b	720412	96.68	83.35	951.00	96.69	79.75	61.00	95.64	85.55	149.39	96.76	86.88	376.71
mgc_fft_1	238324	94.79	78.53	317.00	94.78	73.97	17.00	93.79	83.72	14.65	95.03	85.09	91.99
mgc_fft_2	255324	95.64	81.70	336.00	95.65	77.42	18.00	94.49	85.34	15.62	95.82	86.61	102.24
mgc_fft_a	234441	96.45	84.11	328.00	96.46	79.93	24.00	94.91	86.81	15.32	96.29	88.09	93.67
mgc_fft_b	247866	95.36	79.66	334.00	95.36	74.99	24.00	94.61	84.06	15.80	95.73	85.47	104.44
mgc_pci_bridge32_a	198376	96.80	82.88	258.00	96.79	79.10	14.50	95.76	85.80	11.32	96.74	87.15	85.00
mgc_pci_bridge32_b	172483	97.55	88.74	273.00	97.54	86.06	20.10	95.53	90.04	14.43	97.06	91.22	78.16
Avg.	396540	95.92	81.36	520.78	95.92	77.35	32.29	95.02	84.80	54.26	96.15	86.16	194.19
Ratio		1.00	0.94	2.68	1.00	0.90	0.17	0.99	0.98	0.28	1.00	1.00	1.00

VII. CONCLUSION

In this paper, we have considered the RVI and GTA for the DSA with MP problem, including RGDS, RGDD, and RGDT. For the RGDS problem, we constructed a new ILP formulation basing on our conflict graph. The vertices in the conflict graph are *multiplets* instead of guiding template assignments (GTACs), which can greatly reduce the size of the conflict graph. To fast solve the ILP, a local optimal MWIS solver was introduced to obtain a local optimal result. By a two-stage procedure, our solution flow for the RGDS problem is extended to address the RGDD and RGDT problems. Experimental results validate the efficiency and effectiveness of our ILP formulation and algorithms. It must be remarked that our framework can be extended to handle the problem with different template costs, which are tackled in many existing works on DSA-MP.

ACKNOWLEDGMENTS

The authors would like to thank Dr. S.-Y. Fang for providing binaries and benchmarks. The authors would also like to thank the editor and anonymous reviewers, whose insightful comments and suggestions helped for improving the quality of the manuscript greatly.

REFERENCES

[1] Y. Zorian, D. Gizopoulos, C. Vandenberg, and P. Magarshack, "Guest editors' introduction: Design for yield and reliability," *IEEE Design Test Comput.*, vol. 21, no. 3, pp. 177–182, May 2004.

[2] L.-T. Wang, K.-T. Cheng, and Y.-W. Chang, *Electronic Design Automation: Synthesis, Verification, and Test*. San Mateo, CA, USA: Morgan Kaufmann, 2009.

[3] G. Xu, L.-D. Huang, D. Z. Pan, and M. D. F. Wong, "Redundant-via enhanced maze routing for yield improvement," in *Proc. Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2005, pp. 1148–1151.

[4] H. Y. Chen, M. F. Chiang, Y. W. Chang, L. Chen, and B. Han, "Full-chip routing considering double-via insertion," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 5, pp. 844–857, May 2008.

[5] K.-L. Lin and S.-Y. Fang, "Guiding template-aware routing considering redundant via insertion for directed self-assembly," in *Proc. ACM/IEEE Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2017, pp. 170–175.

[6] 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. ACM Int. Symp. Phys. Design (ISPD)*, Mar. 2017, pp. 91–98.

[7] K.-Y. Lee and T.-C. Wang, "Post-routing redundant via insertion for yield/reliability improvement," in *Proc. Asia South Pacific Conf. Design Autom.*, Jan. 2006, pp. 303–308.

[8] K. Y. Lee, C. K. Koh, T. C. Wang, and K. Y. Chao, "Fast and optimal redundant via insertion," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 12, pp. 2197–2208, Dec. 2008.

[9] 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. 16th Asia South Pacific Design Autom. Conf.*, 2011, pp. 633–638.

[10] H. Yi, X.-Y. Bao, R. Tiberio, and H.-S. P. Wong, "Design strategy of small topographical guiding templates for sub-15nm integrated circuits contact hole patterns using block copolymer directed self assembly," *Proc. SPIE*, vol. 8680, p. 868010, Mar. 2013.

[11] Y. Ma *et al.*, "Directed self-assembly (DSA) grapho-epitaxy template generation with immersion lithography," *Proc. SPIE*, vol. 9423, p. 942306, May 2015.

[12] A. Latypov *et al.*, "Simulations of spatial DSA morphology, DSA-aware assist features and block copolymer-homopolymer blends," *Proc. SPIE*, vol. 9049, p. 904908, Mar. 2014.

[13] Z. Xiao *et al.*, "Directed self-assembly (DSA) template pattern verification," in *Proc. 51st Annu. Design Autom. Conf.*, 2014, pp. 1–6.

[14] J. Ou, B. Yu, and D. Z. Pan, "Concurrent guiding template assignment and redundant via insertion for DSA-MP hybrid lithography," in *Proc. Int. Symp. Phys. Design*, 2016, pp. 39–46.

[15] Z. Xiao, C.-X. Lin, M. D. F. Wong, and H. Zhang, "Contact layer decomposition to enable DSA with multi-patterning technique for standard cell based layout," in *Proc. 21st Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2016, pp. 95–102.

[16] Y. Badr, A. Torres, and P. Gupta, "Mask assignment and synthesis of DSA-MP hybrid lithography for sub-7nm contacts/vias," in *Proc. 52nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2015, pp. 1–6.

[17] J. Kuang, J.-J. Ye, and F.-Y. Young, "Simultaneous template optimization and mask assignment for DSA with multiple patterning," in *Proc. 21st Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2016, pp. 75–82.

[18] S.-Y. Fang, Y.-X. Hong, and Y.-Z. Lu, "Simultaneous guiding template optimization and redundant via insertion for directed self-assembly," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 1, pp. 156–169, Jan. 2017.

[19] S.-Y. Fang and Y.-X. Hong, "Design optimization considering guiding template feasibility and redundant via insertion for directed self-assembly," in *Proc. IEEE Asia Pacific Conf. Circuits Syst. (APCCAS)*, Oct. 2016, pp. 526–529.

[20] C.-Y. Hung, P.-Y. Chou, and W.-K. Mak, "Optimizing DSA-MP decomposition and redundant via insertion with dummy vias," in *Proc. 22nd Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2017, pp. 378–383.

[21] S. Shim, W. Chung, and Y. Shin, "Redundant via insertion for multiple-patterning directed-self-assembly lithography," in *Proc. 53rd Annu. Design Autom. Conf.*, 2016, pp. 410–417.

[22] W. Brendel and S. Todorovic, "Segmentation as maximum-weight independent set," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 307–315.

[23] B. Yu, K. Yuan, B. Zhang, D. Ding, and D. Z. Pan, "Layout decomposition for triple patterning lithography," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 3, pp. 433–446, Mar. 2015.

[24] S.-Y. Fang, Y.-W. Chang, and W.-Y. Chen, "A novel layout decomposition algorithm for triple patterning lithography," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 3, pp. 397–408, Mar. 2014.

[25] J. Kuang and E. F. Y. Young, "An efficient layout decomposition approach for triple patterning lithography," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, May 2013, p. 69.

[26] A. Frieze and M. Jerrum, "Improved approximation algorithms for MAX *k*-CUT and MAX BISECTION," *Algorithmica*, vol. 18, no. 1, pp. 67–81, 1997.

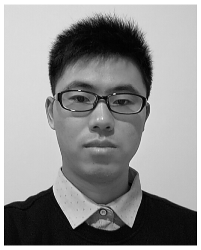
[27] G. Jäger and A. Srivastav, "Improved approximation algorithms for maximum graph partitioning problems," *J. Combinat. Optim.*, vol. 10, no. 2, pp. 133–167, 2005.

[28] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. New York, NY, USA: Springer-Verlag, 2003.

[29] *OpenSPARC T1*. Accessed: Dec. 1, 2016. [Online]. Available: www.oracle.com/technetwork/systems/opensparc

[30] *CPLEX*. Accessed: Sep. 1, 2016. [Online]. Available: www.ibm.com/jm-en/marketplace/ibm-ilog-cplex

[31] D. K. Lam, "Maskless electron-beam lithography for trusted microchip production," Multibeam Corp., Santa Clara, CA, USA, Tech. Rep. 1, 2016.



Xingquan Li received the B.Sc. degree in applied mathematics from the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China, in 2013, where he is currently working toward the Ph.D. degree at the Center for Discrete Mathematics and Theoretical Computer Science.

His current research interests include VLSI physical design and design for manufacturability and optimization theory and algorithms.

Mr. Li was a recipient of the First Place in the ICCAD 2017 Contest.

Bei Yu (S'11–M'14) received the Ph.D. degree from The University of Texas at Austin, Austin, TX, USA, in 2014.

He is currently an Assistant Professor at the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

Dr. Yu received five best paper awards from *Integration, the VLSI Journal* in 2018, the International Symposium on Physical Design 2017, the SPIE Advanced Lithography Conference 2016, the International Conference on Computer Aided Design 2013, and the Asia and South Pacific Design Automation Conference 2012, and four ICCAD/ISPD contest awards. He has served on the editorial boards of *Integration, the VLSI Journal* and the *IET Cyber-Physical Systems: Theory & Applications*.

Jiaojiao Ou received the M.S. degree in microelectronics from Peking University, Beijing, China, in 2013. She is currently working toward the Ph.D. degree at the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA.

Her current research interests include physical design and design for manufacturability with emerging nanolithography.

Jianli Chen received the B.Sc. degree in information and computing sciences, the M.Sc. degree in computer application technology, and the Ph.D. degree in applied mathematics from Fuzhou University, Fuzhou, China, in 2007, 2009, and 2012, respectively.

Since 2012, he has been with Fuzhou University, where he has been an Associate Professor with the Center for Discrete Mathematics and Theoretical Computer Science. His current research interests include VLSI physical design, design for manufacturability, and optimization theory and algorithms.

Dr. Chen received the Best Paper Award at the Design Automation Conference 2017 and the First Place in the ICCAD 2017 Contest.

David Z. Pan (S'97–M'00–SM'06–F'14) received the B.S. degree from Peking University, Beijing, China, and the M.S. and Ph.D. degrees from the University of California at Los Angeles (UCLA), Los Angeles, CA, USA.

From 2000 to 2003, he was a Research Staff Member with the IBM T. J. Watson Research Center, Yorktown Heights, NY, USA. He is currently an Engineering Foundation Endowed Professor with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX, USA. He has authored over 200 papers in refereed journals and conferences. He holds eight U.S. patents. His current research interests include cross-layer nanometer IC design for manufacturability/reliability, new frontiers of physical design, and CAD for emerging technologies, such as 3-D-IC, bio, and nanophotonics.

Dr. Pan has received a number of awards, including the SRC 2013 Technical Excellence Award, DAC Top 10 Author in Fifth Decade, the DAC Prolific Author Award, the ASPDAC Frequently Cited Author Award, and 11 Best Paper Awards, several international CAD contest awards, the Communications of the ACM Research Highlights Award in 2014, the ACM/SIGDA Outstanding New Faculty Award in 2005, the NSF CAREER Award in 2007, the SRC Inventor Recognition Award three times, the IBM Faculty Award four times, the UCLA Engineering Distinguished Young Alumnus Award in 2009, and the UT Austin RAISE Faculty Excellence Award in 2014. He has served as the Program/General Chair of ISPD, the TPC Subcommittee Chair for DAC, ICCAD, ASPDAC, ISLPED, and ICCD, the Tutorial Chair for DAC 2014, and the Workshop Chair for ICCAD 2015, among others. He has served as a Senior Associate Editor for the *ACM Transactions on Design Automation of Electronic Systems* and an Associate Editor for the IEEE TRANSACTIONS ON COMPUTER AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II, *Science China Information Sciences*, and the *Journal of Computer Science and Technology*.

Wenxing Zhu received the B.Sc. degree in applied mathematics, and the M.Sc. and Ph.D. degrees in operations research and cybernetics from Shanghai University, Shanghai, China, in 1989, 1992, and 1996, respectively.

Since 1996, he has been with Fuzhou University, Fuzhou, China, where he has been a Professor with the Center for Discrete Mathematics and Theoretical Computer Science since 2004. He has authored over 65 papers in refereed journals, including the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-C, the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE TRANSACTIONS ON ENERGY CONVERSION, the *INFORMS Journal on Computing*, and the *SIAM Journal on Discrete Mathematics*. His current research interests include optimization theory and algorithms, algorithms for VLSI design automation, and algorithms for sparse optimization.

Dr. Zhu received the Best Paper Award at the Design Automation Conference 2017 and the First Place in the ICCAD 2017 Contest.