LRSDP: Low-Rank SDP for Triple Patterning Lithography Layout Decomposition

Yu Zhang[1,2†], Yifan Chen[1†], Zhonglin Xie[1], Hong Xu[2], Zaiwen Wen[1], Yibo Lin[1,3*], Bei Yu[2*]
[1]Peking University, Beijing, China [2]Chinese University of Hong Kong, Hong Kong, China
[3]Institute of Electronic Design Automation, Peking University, Wuxi, China

***Abstract*—Multiple patterning lithography (MPL) has been widely adopted in advanced technology nodes to enhance lithography resolution. As layout decomposition for triple patterning lithography (TPL) and beyond is NP-hard, existing approaches formulate mathematical programming problems and leverage general-purpose solvers such as integer linear programming (ILP) and semidefinite programming (SDP) to trade off quality against runtime. With the aggressive increase in design complexity, existing approaches can no longer scale to solve complicated designs with high solution quality. In this paper, we propose a dedicated low-rank SDP algorithm for MPL decomposition with augmented Lagrangian relaxation and Riemannian optimization. Experimental results demonstrate that our method is 186×, 25×, and 12× faster than the state-of-the-art decomposition approaches with highly competitive solution quality.**

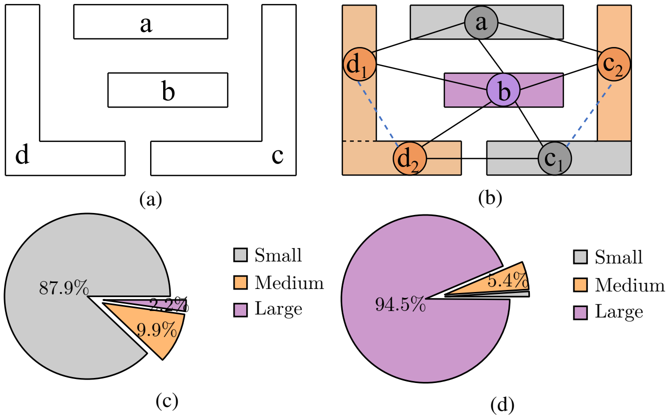
I. Introduction

Multiple patterning lithography (MPL) is widely adopted in advanced technology nodes. It imposes a layout decomposition step in the design flow that layout features close to each other are assigned to different masks to prevent coloring conflicts. Fig. 1 shows an example where a decomposition graph is constructed by regarding features as vertices and connecting features within a certain distance. Different colors denote different masks. If two vertices of a conflict edge are assigned to the same color, then a *conflict* occurs. We can resolve conflicts by assigning different parts of one feature into different colors, which is known as *stitch* insertion. The objective of layout decomposition is to minimize the number of conflicts and stitches, as conflicts lead to printing failure and stitches can cause yield loss [1].

Most existing MPL decomposition studies follow a two-step procedure after constructing the decomposition graph. They first break and simplify the graph into small subgraphs and then apply decomposition algorithms on subgraphs. The decomposition algorithms are based on ILP [2], SDP [3], exact cover [4], or other graph heuristics [5]. Among these methods, ILP can provide exact solutions at the cost of exponential runtime complexity, while SDP and exact cover can provide high-quality approximation with affordable runtime.

However, these methods usually assume the subgraphs after the simplification step are small, i.e., fewer than 100 vertices [2]. When the design complexity increases, the sizes of subgraphs also boost. We observe that 2.2% large graphs ($>$ 1000 vertices) take 94.5% of the runtime, while 87.9% small graphs ($<$ 100 vertices) only take 0.1% of the overall runtime for a state-of-the-art layout decomposer with a general-purpose SDP solver [6], as shown in Fig. 1(c)- 1(d). Therefore, new decomposition algorithms are urgently desired to improve the efficiency for large designs.

To achieve efficient yet high-quality decomposition, we propose a scalable SDP solving algorithm leveraging the low-rank property [7] and exploiting the problem structure. The major contributions of this paper are summarized as follows.



(a) (b) (c) (d)

Fig. 1 (a) Input features. (b) The final decomposed layout with three colors, where the conflict edges are marked as black edges and stitch edges are marked as blue dashed edges. (c) The proportion of small, medium, and large subgraphs after graph simplification. (d) The time ratio spent on solving the TPL decomposition for these subgraphs.

- We propose LRSDP, a scalable low-rank SDP solver for the MPL decomposition problem on large graphs. Specifically, we leverage the decomposition-based augmented Lagrangian method (ALM) to solve the large-scale SDP problem from MPL.
- We propose to exploit the problem structure by restricting the searching space on a smooth manifold (unit sphere). Particularly, the Riemannian gradient descent method with Barzilai–Borwein steps (RGBB) is adopted here to search for the optimal solution with high efficiency.
- Experimental results demonstrate that compared with the state-of-the-art algorithms, including ILP [2], exact cover [4], SDP [3], our LRSDP method can achieve 186×, 25×, and 12× speedup with better solution quality than exact cover and SDP on ISPD2019 contest benchmarks. LRSDP is even scalable to large designs that cannot be solved by the other algorithms.

II. Preliminaries

In this section, we first introduce the program formulation and the SDP relaxation of TPL decomposition. Then, we explain the algorithm for solving general SDP and low-rank property of the decomposition problem.

A. Problem Formulation

Definition 1. (Decomposition Graph): Given a layout represented by a set of polygonal shapes, the decomposition graph (DG) is an undirected graph with a single set of vertices V, and two sets of edges, conflict edges (CE) and stitch edges (SE), respectively. V has one or more vertices for each polygonal shape, and each vertex is associated with a polygonal shape. An edge is in CE if and only if the two polygonal shapes are within minimum coloring distance $\min_s$. An

This work is partially supported by The Research Grants Council of Hong Kong SAR (No. CUHK14209420, CUHK14208021), National Science Fundation of China (No. 62141404), and The 111 Project (No. B18001).

[†]Equal contributors

[*]Corresponding authors

edge is in SE if and only if there is a stitch between the two vertices which are associated with the same polygonal shape.

Problem 1. (Triple Patterning Layout Decomposition): Given a layout which is specified by features in polygonal shapes, a decomposition graph is constructed. The goal of TPL is to assign vertexes in the decomposition graph to three masks (colors) with following cost minimized,

$$\text{cost} = \#\text{conflict} + \alpha\#\text{stitch},$$

where α is set to 0.1 [8].

B. SDP-Based Layout Decomposition

In TPL decomposition, there are three possible colors. We set a unit vector $\boldsymbol{v}_i$ for every vertex i $(i = 1, \cdots, n)$. Naturally, if e_{ij} is a conflict edge, we want vertices $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ to be far apart. If e_{ij} is a stitch edge, we hope vertices $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ to be the same. Bearing this in mind, we associate all the vertices with three different unit vectors: $(1, 0), (-\frac{1}{2}, \frac{\sqrt{3}}{2}), (-\frac{1}{2}, -\frac{\sqrt{3}}{2})$, so we have the following property:

$$\boldsymbol{v}_i^\top \boldsymbol{v}_j = \begin{cases} 1, & \boldsymbol{v}_i = \boldsymbol{v}_j, \\ -\frac{1}{2}, & \boldsymbol{v}_i \neq \boldsymbol{v}_j. \end{cases}$$

Based on the above property, we can formulate the triple patterning layout decomposition as the following vector program [3]:

$$\min \quad \frac{2}{3} \sum_{e_{ij} \in \text{CE}} (\boldsymbol{v}_i^\top \boldsymbol{v}_j + \frac{1}{2}) + \frac{2\alpha}{3} \sum_{e_{ij} \in \text{SE}} (1 - \boldsymbol{v}_i^\top \boldsymbol{v}_j), \tag{1}$$

$$\text{s.t.} \quad \boldsymbol{v}_i \in \{(1, 0), (-\frac{1}{2}, \frac{\sqrt{3}}{2}), (-\frac{1}{2}, -\frac{\sqrt{3}}{2})\}, \tag{1a}$$

where the left part is the cost of all conflicts and the right part represents the cost of all stitches.

The vector program (1) is NP-hard due to the discrete constraint (1a). The problem can be further relaxed as the following semidefinite program [3]:

$$\min_{\boldsymbol{X} \in \mathbb{R}^{n \times n}} \quad \langle \boldsymbol{C}, \boldsymbol{X} \rangle \tag{2}$$

$$\text{s.t.} \quad x_{ii} = 1, \quad \forall i \in V \tag{2a}$$

$$x_{ij} \geq -\frac{1}{2}, \quad \forall e_{ij} \in \text{CE} \tag{2b}$$

$$\boldsymbol{X} \geq 0, \tag{2c}$$

where $\langle \boldsymbol{C}, \boldsymbol{X} \rangle = tr(\boldsymbol{C}^\top \boldsymbol{X})$ represents the Euclidean inner product between two matrices $\boldsymbol{C}$ and $\boldsymbol{X}$, i.e., $\sum_i \sum_j c_{ij} x_{ij}$. Here x_{ij} is the entry at the i-th row and the j-the column of matrix $\boldsymbol{X}$, and c_{ij} is the entry at the i-th row and the j-the column of matrix $\boldsymbol{C}$:

$$c_{ij} = \begin{cases} 1, & \forall e_{ij} \in \text{CE}, \\ -\alpha, & \forall e_{ij} \in \text{SE}, \\ 0, & \text{otherwise}. \end{cases}$$

Without discrete constraint (1a), program (2) is not NP-hard now and can be solved in polynomial time [3]. What's more, constraint (2c) indicates that the symmetric matrix $\boldsymbol{X}$ should be positive semidefinite. Therefore, we can always find a matrix $\boldsymbol{R}$ such that $\boldsymbol{X} = \boldsymbol{R}^\top \boldsymbol{R}$.

C. Interior-Point Method to Solve SDP

Interior-point methods are widely-adopted to solve SDP [6], [8]. The basic idea of primal-dual interior-point methods is to find a feasibly optimal point that satisfies the KKT conditions of the semidefinite programs, along with Newton's method to solve the subproblem at each iteration [9]. In practice, these methods are capable of solving small- to medium-sized problems precisely, but not scalable due to their inherent high demand for storage and computation. Particularly, the time complexity of the interior-point methods is cubic in the number of variables, i.e., $O(n^3)$ [9], thereby prompting research for alternative approaches.

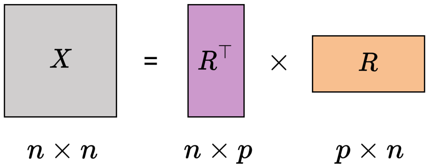


Fig. 2 Low-rank decomposition

D. Low-Rank Property of TPL Decomposition

It has been recognized that the solution to the SDP generated by the relaxation of combinatorial optimization tasks is often low-rank. Hence, a class of low-rank decomposition-based methods has been adopted in dealing with such SDP problems [7], [10]. Here we illustrate this property with a simple TPL decomposition problem in Fig. 1(b). After solving the SDP in (2), we can get a matrix $\boldsymbol{X}$ as:

$$\boldsymbol{X} = \begin{pmatrix} 1 & -0.49 & 0.21 & -0.5 & -0.5 & 0.21 \\ & 1 & -0.5 & -0.5 & -0.5 & -0.5 \\ & & 1 & 0.43 & 0.15 & -0.5 \\ & & & 1 & 0.95 & 0.15 \\ & \cdots & & & 1 & 0.43 \\ & & & & & 1 \end{pmatrix}.$$

Observe that $\boldsymbol{X}$ is essentially a rank-deficient semidefinite matrix, so we can further decompose it as $\boldsymbol{X} = \boldsymbol{R}^\top \boldsymbol{R}$, where $\boldsymbol{R} \in \mathbb{R}^{p \times n} (p < n)$. Then we have:

$$\boldsymbol{R} = \begin{pmatrix} 0.26 & 0.72 & -0.39 & -0.95 & -0.95 & -0.39 \\ -0.96 & 0.70 & -0.32 & 0.26 & 0.26 & -0.32 \\ 0 & 0 & -0.87 & -0.16 & 0.16 & 0.87 \end{pmatrix}.$$

The low-rank factorization process is illustrated in Fig. 2. In particular, the benefits of the low-rank factorization are three-fold:

- The semidefinite constraint (2c) can be naturally omitted as it is implied by the factorization;
- This low-rank factorization leads to many fewer variables in the problem of interest, as X is explicitly represented by the matrix $\boldsymbol{R}$ of dimension $p \times n$ (typically with $p \ll n$), thereby decreasing the computational complexity;
- Although Formula (2) becomes nonlinear and nonconvex after low-rank factorization, the global optimality of the solution to this factorized surrogate can still be ensured by simply choosing $p \geq \sqrt{2m}$, where m is the number of constraints [7].

With these benefits, we thereby perform low-rank factorization to our SDP program (2), which will be discussed in detail in Section III-A.

III. Algorithms

In this section, we will present our basic algorithms, which take full advantage of the specific problem structure and thus reduce the numerical difficulty in solving the TPL decomposition problem. First, we will reformulate the TPL decomposition problem with low-rank factorization and recast the Euclidean optimization problem to a Riemannian optimization problem on a smooth manifold. Then, we will introduce a decomposed augmented lagrangian method (ALM) for TPL decomposition, followed by a Riemannian gradient descent method with Barzilai-Borwein steps (RGBB) to solve the subproblem at each iteration. Finally, we will give a convergence analysis for the decomposed ALM.

A. Low-Rank Factorization

Canonical semidefinite programs involve optimizing a matrix-valued variable $\boldsymbol{X}$ (a symmetric $n \times n$ matrix), where the number of variables grows quadratically, so it quickly becomes unaffordable for SDP solvers employing exact methods to deal with large-scale layouts. Motivated by the low-rank property of solutions to such SDP problems, we factorize $\boldsymbol{X}$ as $\boldsymbol{X} = \boldsymbol{R}^\top \boldsymbol{R}$, where $\boldsymbol{R} \in \mathbb{R}^{p \times n}$, then the original semidefinite programs in (2) becomes:

$$\min_{\boldsymbol{R} \in \mathbb{R}^{p \times n}} \quad \langle \boldsymbol{C}, \boldsymbol{R}^\top \boldsymbol{R} \rangle \tag{3}$$

$$\text{s.t.} \quad \|\boldsymbol{r_i}\|_2 = 1, \quad \forall i \in V \tag{3a}$$

$$\boldsymbol{r_i}^\top \boldsymbol{r_j} \geq -\frac{1}{2}, \quad \forall e_{ij} \in \text{CE}, \tag{3b}$$

where $\boldsymbol{r_i}$ is the i-th column vector of $\boldsymbol{R}$ and $\|\cdot\|_2$ denotes the $L2$-norm of vectors.

B. Smooth Semidefinite Programs

In order to fully utilize the TPL decomposition problem structure, we further restrict the optimization of variable $\boldsymbol{R}$ from $\mathbb{R}^{p \times n}$ to a smooth manifold. Specifically, the constraint (3a) can be naturally satisfied by forcing $\boldsymbol{R}$ in a smooth manifold: $\mathcal{M} = \{\boldsymbol{R} \in \mathbb{R}^{p \times n} | \boldsymbol{R} = [\boldsymbol{r_1}, \cdots, \boldsymbol{r_n}], \|\boldsymbol{r_i}\|_2 = 1\}$, which is exactly a unit sphere (see $\mathcal{M}$ in Fig. 3(b)). In other words, we consider the constraint (3a) as a manifold and optimize the variable of interest on this manifold. Compared to Euclidean space optimization, the reasons for constraining the optimization of $\boldsymbol{R}$ on a smooth manifold $\mathcal{M}$ are summarized as follows:

- The satisfiability of constraint Equation (3a) is naturally guaranteed as the solutions are always on the smooth manifold $\mathcal{M}$;
- The structure information of a smooth manifold can be fully utilized to reduce searching space as we essentially search for the optimal solution on a unit sphere, thereby solving large-scale TPL decomposition problems efficiently.

Now, the only difficult constraint in Formula (3) is the inequality constraint (3b). Here we introduce an auxiliary variable $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ to remove the inequality constraint, so the factorized SDP is reformulated as:

$$\min_{\boldsymbol{R} \in \mathcal{M}} \quad \langle \boldsymbol{C}, \boldsymbol{R}^\top \boldsymbol{R} \rangle + h(\boldsymbol{W}) \tag{4}$$

$$\text{s.t.} \quad \boldsymbol{P} \odot \boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{W}, \tag{4a}$$

where $\odot$ denotes element-wise product, h is a characteristic function, and $\boldsymbol{P}$ encodes the information of conflict edges:

$$h(\boldsymbol{W}) = \begin{cases} 0, & w_{ij} \geq -\frac{1}{2}, \forall e_{ij} \in \text{CE}, \\ +\infty, & \text{otherwise}, \end{cases}$$

$$p_{ij} = \begin{cases} 1, & \forall e_{ij} \in \text{CE}, \\ 0, & \forall e_{ij} \notin \text{CE}. \end{cases}$$

Here w_{ij} is the entry at the i-th row and the j-th column of matrix $\boldsymbol{W}$, and p_{ij} is the entry at the i-th row and the j-th column of matrix $\boldsymbol{P}$.

Algorithm 1: The LRSDP algorithm

```
Input: Initialize start point $\boldsymbol{R}^0 \in \mathcal{M}$, ALM step size $\alpha$, $\rho > 1$,
       penalty parameter $\sigma_0 > 0$;
Set $k = 0$, $\boldsymbol{y}^k = 0$;
while not yet converged do
    Obtain $\boldsymbol{R}^{k+1}$ by solving (6);
    Update $\boldsymbol{W}^{k+1}$ via Equation (7);
    Update Lagrangian multipliers $\boldsymbol{y}^{k+1}$ based on
      Equation (6a), and $\sigma_{k+1}$ by Equation (6b) ;
    Set $k = k + 1$.
end
```

C. Augmented Lagrangian Method

After transforming the inequality constraints (3b) to equality constraints (4a), the augmented Lagrangian method (ALM) is adopted here to solve the nonlinear and non-convex programming problem for TPL decomposition. The basic idea behind ALM is the idea of penalization, i.e., the method optimizes an augmented objective which includes an additional term that penalizes infeasible points. In particular, the augmented Lagrangian function associated with Formula (4) is denoted by:

$$\begin{aligned} L_\sigma(\boldsymbol{R}, \boldsymbol{W}, \boldsymbol{y}, \sigma) = \langle \boldsymbol{C}, \boldsymbol{R}^\top \boldsymbol{R} \rangle + h(\boldsymbol{W}) - \\ \langle \boldsymbol{y}, \boldsymbol{P} \odot \boldsymbol{R}^\top \boldsymbol{R} - \boldsymbol{W} \rangle + \frac{\sigma}{2} \|\boldsymbol{P} \odot \boldsymbol{R}^\top \boldsymbol{R} - \boldsymbol{W}\|_F^2. \end{aligned} \tag{5}$$

Here $\boldsymbol{y} \in \mathbb{R}^{n \times n}$, and $\sigma > 0$ is a parameter for ALM. In contrast to the canonical Lagrangian method, the augmented Lagrangian function differs merely in the penalization term involving σ. This term measures the infeasibility of $\boldsymbol{R}$ with respect to constraint (4a) and is scaled by the penalty parameter σ. Then, based on Equation (5), the k-th iteration of the ALM is given as follows:

$$(\boldsymbol{R}^{k+1}, \boldsymbol{W}^{k+1}) = \underset{\boldsymbol{R} \in \mathcal{M}, \boldsymbol{W} \in \mathbb{R}^{n \times n}}{\text{argmin}} \quad L_{\sigma_k}(\boldsymbol{R}, \boldsymbol{W}, \boldsymbol{y}^k, \sigma^k), \tag{6}$$

$$\boldsymbol{y}^{k+1} = \boldsymbol{y}^k - \alpha \sigma_k (\boldsymbol{P} \odot (\boldsymbol{R}^{k+1})^\top \boldsymbol{R}^{k+1} - \boldsymbol{W}^{k+1}), \tag{6a}$$

$$\sigma_{k+1} = \begin{cases} \sigma_k, & u_{k+1} < u_k, \\ \rho \sigma_k, & \text{otherwise}, \end{cases} \tag{6b}$$

where α is the ALM step size, $\rho > 1$, and $u_k = \|\boldsymbol{P} \odot (\boldsymbol{R}^k)^\top \boldsymbol{R}^k - \boldsymbol{W}^k\|_F^2$ implies the infeasibility of $\boldsymbol{R}^k$. The rationale here for the update rule of σ_{k+1} is that if $u_{k+1} < u_k$, we know that the Lagrangian framework moves towards reducing the infeasibility so that there is no need to increase the penalty parameter. If, on the other hand, $u_{k+1} \geq u_k$, then (6b) scales σ_k by ρ to force the update of Lagrangian multipliers to the target level.

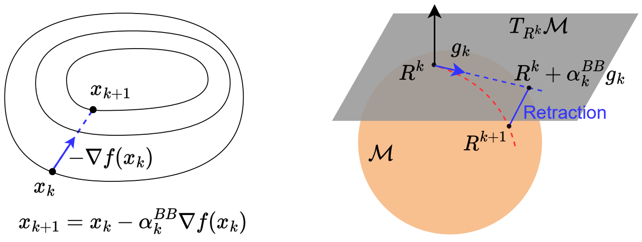
Based on the above update functions, the major challenge is to determine optimal $(\boldsymbol{R}^{k+1}, \boldsymbol{W}^{k+1})$ from Equation (6) as $\boldsymbol{y}^{k+1}$ and σ_{k+1} can be explicitly updated by Equation (6a) and Equation (6b), respectively. To begin with, considering minimizing the function $L_{\sigma_k}(\boldsymbol{R}, \boldsymbol{W}, \boldsymbol{y}^k, \sigma^k)$ with respect to a fixed $\boldsymbol{R}$, the optimal solution of $\boldsymbol{W}$ follows:

$$\boldsymbol{W} = \Pi_S(\boldsymbol{P} \odot \boldsymbol{R}^\top \boldsymbol{R} - \boldsymbol{y}^k / \sigma_k), \tag{7}$$

where $\Pi_S(\boldsymbol{A})$ is the projection of $\boldsymbol{A}$ on set $S = \{\boldsymbol{W} \in \mathbb{R}^{n \times n} | w_{ij} \geq -\frac{1}{2}, \forall e_{ij} \in \text{CE}\}$. Therefore, given $\boldsymbol{R}$, we can always find the optimal $\boldsymbol{W}$ by Equation (7). Now, the only problem is to obtain the optimal solution $\boldsymbol{R}$ of $L_{\sigma_k}(\boldsymbol{R}, \boldsymbol{W}, \boldsymbol{y}^k, \sigma^k)$, which plays a critical role in the overall efficiency of the LRSDP algorithm. Here we apply a Riemannian gradient descent method with Barzilai-Borwein steps (RGBB) to optimize $\boldsymbol{R}$, which will be discussed with details in Section III-D. Given an approximate solution $\boldsymbol{R}^{k+1}$, the overall algorithm is summarized in Algorithm 1.

D. Riemannian Optimization Method with Barzilai–Borwein Steps

We now discuss how we obtain the optimal solution $\boldsymbol{R}$ via Riemannian optimization. In particular, the main computational work of ALM lies in solving the optimization subproblem:



(a) (b)

Fig. 3 (a) The gradient descent method with BB steps. (b) The Riemannian optimization method with BB steps.

$\operatorname*{argmin}_{\boldsymbol{R}\in\mathcal{M}} L_{\sigma_k}(\boldsymbol{R}, \boldsymbol{W}, \boldsymbol{y}^k, \sigma^k)$. First, as $\boldsymbol{W}$ can be explicitly represented by $\boldsymbol{R}$, we plug the optimal solution of $\boldsymbol{W}$ (7) into this subproblem:

$$\begin{aligned}\Phi_k(\boldsymbol{R}) :=& \inf_{\boldsymbol{W}\in\mathbb{R}^{n\times n}} L_{\sigma_k}(\boldsymbol{R}, \boldsymbol{W}, \boldsymbol{y}^k, \sigma^k) \\ =& \langle \boldsymbol{C}, \boldsymbol{R}^\top \boldsymbol{R}\rangle + h(\Pi_S(\boldsymbol{P}\odot \boldsymbol{R}^\top \boldsymbol{R} - \boldsymbol{y}^k/\sigma_k)) \\ & - \frac{\|\boldsymbol{y}^k\|_2^2}{2\sigma_k} + \frac{\sigma_k}{2}\|\boldsymbol{P}\odot \boldsymbol{R}^\top \boldsymbol{R} - \boldsymbol{y}^k/\sigma_k \\ & - \Pi_S(\boldsymbol{P}\odot \boldsymbol{R}^\top \boldsymbol{R} - \boldsymbol{y}^k/\sigma_k)\|_F^2.\end{aligned} \tag{8}$$

Then, the optimal $\boldsymbol{R}^{k+1}$ can be computed as: $\boldsymbol{R}^{k+1} = \operatorname*{argmin}_{\boldsymbol{R}\in\mathcal{M}}\Phi_k(\boldsymbol{R})$. In order to further simplify Equation (8), we leverage orthogonal decomposition that correlates the projection $\Pi_S(\boldsymbol{A})$ with its orthogonal complement $\Pi_{S^\perp}(\boldsymbol{A})$, i.e., $\boldsymbol{A} = \Pi_S(\boldsymbol{A}) + \Pi_{S^\perp}(\boldsymbol{A})$. In particular, denote $T(\boldsymbol{R}) = \Pi_{S^\perp}(\boldsymbol{P}\odot \boldsymbol{R}^\top \boldsymbol{R} - \boldsymbol{y}^k/\sigma_k)$, we have:

$$T(\boldsymbol{R}) = \boldsymbol{P}\odot \boldsymbol{R}^\top \boldsymbol{R} - \boldsymbol{y}^k/\sigma_k - \Pi_S(\boldsymbol{P}\odot \boldsymbol{R}^\top \boldsymbol{R} - \boldsymbol{y}^k/\sigma_k) \tag{9}$$

Ignoring the constant term, the minimization of $\boldsymbol{R}$ in Equation (8) can be rewritten as:

$$\begin{aligned}\min_{\boldsymbol{R}\in\mathcal{M}} \Phi_k(\boldsymbol{R}) =& \langle \boldsymbol{C}, \boldsymbol{R}^\top \boldsymbol{R}\rangle + h(\boldsymbol{P}\odot \boldsymbol{R}^\top \boldsymbol{R} - \boldsymbol{y}^k/\sigma_k - T(\boldsymbol{R})) \\ & + \frac{\sigma_k}{2}\|T(\boldsymbol{R})\|_F^2.\end{aligned} \tag{10}$$

The subproblem in Equation (10) is virtually an unconstrained Riemannian manifold optimization problem. Observing that $\Phi_k(\boldsymbol{R})$ is continuously differentiable but may not be twice continuously differential, we thus apply a Riemannian gradient descent method with Barzilai–Borwein steps (RGBB) [11] to solve Equation (10) to a high-precision. The key idea of RGBB method lies in the explicit use of first-order information (gradient) of the objective function on one side, and, on the other side, in the implicit use of second-order information embedded in the step length through a rough approximation of the Hessian of the objective function. This is crucial in the solution of layout decomposition problems where computing the Hessian represents a heavy burden, owing to the large problem dimension.

Traditionally, given the problem $\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x})$, where f is a smooth cost function in the Euclidean case, the simplest gradient-type method is the Newton method based on the steepest descent direction:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k), \tag{11}$$

where step length $\alpha_k = \boldsymbol{H}^{-1}$ ($\boldsymbol{H}$ is the Hessian matrix of $f(\boldsymbol{x}_k)$). However, this ideal step length is usually unnecessarily expensive to compute for a general nonlinear cost function f, such as $\Phi_k(\boldsymbol{R})$ in our problem. Therefore, a more practical strategy is to identify a step length that achieves an adequate reduction in f with minimal cost.

The Barzilai–Borwein (BB) method [11] provides an alternative strategy for the clever choice of step length. Although it does not guarantee the steepest decrease of the objective function at each step, it yields impressive good practical performance. The basic idea of the BB method is to approximate the computationally expensive Hessian by solving, for $k \geq 1$, the least-square problem

$$\min_t \|\boldsymbol{s}_k t - \boldsymbol{y}_k\|_2, \tag{12}$$

with $\boldsymbol{s}_k := \boldsymbol{x}_k - \boldsymbol{x}_{k-1}$ and $\boldsymbol{y}_k := \nabla f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}_{k-1})$. Obviously, Equation (12) has the unique solution $t = \frac{\boldsymbol{s}_k^\top \boldsymbol{y}_k}{\boldsymbol{s}_k^\top \boldsymbol{s}_k}$, which inexactly approximates the Hessian matrix of $f(\boldsymbol{x}_k)$. When $\boldsymbol{s}_k^\top \boldsymbol{y}_k > 0$, the BB step-length is

$$\alpha_k^{BB} = \frac{\boldsymbol{s}_k^\top \boldsymbol{s}_k}{\boldsymbol{s}_k^\top \boldsymbol{y}_k}. \tag{13}$$

Then the gradient descent in Equation (11) becomes $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k^{BB}\nabla f(\boldsymbol{x}_k)$. In essence, the BB method is a Quasi-Newton approach where the second-order information (Hessian matrix) is implicitly embedded in the step length α_{k+1}^{BB} through a cheap approximation in Equation (13). The gradient descent with BB method in Euclidean space is illustrated in Fig. 3(a).

As in the Euclidean case, the idea of the RGBB method in our problem is to approximate the Riemannian Hessian of $\Phi_k(\boldsymbol{R})$ at a certain point. In particular, at the k-th step, the Hessian is a linear map from $T_{\boldsymbol{R_{k-1}}}\mathcal{M}$ to $T_{\boldsymbol{R_k}}\mathcal{M}$, where $T_{\boldsymbol{R_k}}\mathcal{M}$ is the tangent space at point $\boldsymbol{R_k}$. Similarly, we would like to use BB step size to approximate the Hessian in Riemannian optimization.

To begin with, instead of the subtraction $\boldsymbol{x}_k - \boldsymbol{x}_{k-1}$, we now consider the vector $(-\alpha_{k-1}\boldsymbol{g_{k-1}})$ belonging to $T_{\boldsymbol{R_{k-1}}}\mathcal{M}$ and transport it to $T_{\boldsymbol{R_k}}\mathcal{M}$, yielding

$$\boldsymbol{s}_k := -\alpha_{k-1}T_{\boldsymbol{R_{k-1}}\to\boldsymbol{R_k}}(\boldsymbol{g_{k-1}}). \tag{14}$$

Then, to obtain $\boldsymbol{y}_k$, we need to subtract two gradients lying in two different tangent spaces. To be coherent with the manifold structure and to work on tangent space $T_{\boldsymbol{R_k}}\mathcal{M}$, this difference should be made after $\boldsymbol{g_{k-1}}$ is transported to $T_{\boldsymbol{R_k}}\mathcal{M}$ so that

$$\boldsymbol{y}_k := \boldsymbol{g_k} - T_{\boldsymbol{R_{k-1}}\to\boldsymbol{R_k}}(\boldsymbol{g_{k-1}}). \tag{15}$$

Subsequently, the least-squares approximation with respect to $T_{\boldsymbol{R_k}}\mathcal{M}$ yields $t = \frac{\langle \boldsymbol{s}_k, \boldsymbol{y}_k\rangle_{\boldsymbol{R_k}}}{\langle \boldsymbol{s}_k, \boldsymbol{s}_k\rangle_{\boldsymbol{R_k}}}$. Therefore, the Riemannian BB step length has the form:

$$\alpha_k^{BB} = \frac{\langle \boldsymbol{s}_k, \boldsymbol{s}_k\rangle_{\boldsymbol{R_k}}}{\langle \boldsymbol{s}_k, \boldsymbol{y}_k\rangle_{\boldsymbol{R_k}}}, \tag{16}$$

provided that $\langle \boldsymbol{s}_k, \boldsymbol{y}_k\rangle_{\boldsymbol{R_k}} > 0$. With BB step size, we can now update $\boldsymbol{R}_{k+1}$ with the retraction from $\boldsymbol{R_k} + \alpha_k^{BB}\boldsymbol{g}_k$ to the smooth manifold $\mathcal{M}$, i.e., $\boldsymbol{R}_{k+1} = C_{\boldsymbol{R_k}}(\alpha_k^{BB}\boldsymbol{g}_k)$. The RGBB at k-th step is illustrated in Fig. 3(b), from which we can see that the main difference between Euclidean and Riemannian optimization is the retraction process. In practice, we implement a nonmonotone line search strategy [12] to guarantee the global convergence to stationary points. Now, the whole LRSDP framework is shown in Fig. 4.

E. Convergence Analysis of ALM

In this section, we consider the convergence of ALM in solving the factorized semidefinite program in (4). In nonlinear programming, the optimum $(\boldsymbol{R}, \boldsymbol{W})$ necessarily satisfies the KKT conditions. Specif-

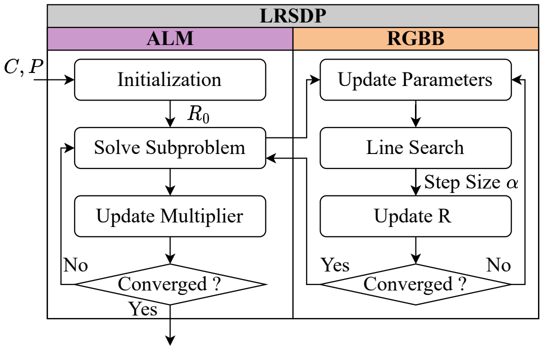


Fig. 4 The algorithm flow of LRSDP.

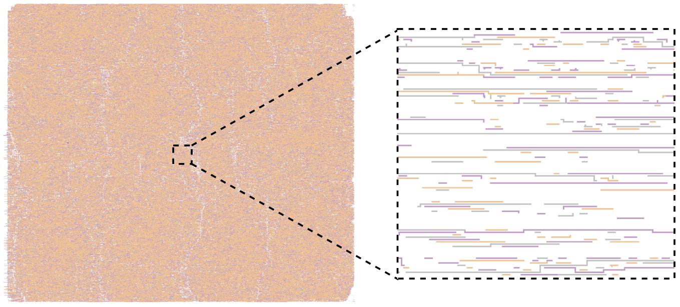


Fig. 5 Decomposition results of `test6_100` in ISPD'19 benchmarks.

ically, we say $(\boldsymbol{R}, \boldsymbol{W})$ satisfies the KKT conditions if there exist Lagrange multipliers $\boldsymbol{y} \in \mathbb{R}^{n\times n}$ such that

$$\begin{aligned} &\boldsymbol{P} \odot \boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{W}, \\ &\quad 0 \in \partial h(\boldsymbol{W}), \\ &\quad 0 \in 2\boldsymbol{R}(\nabla f(\boldsymbol{R}^\top \boldsymbol{R}) - \boldsymbol{P}^* \boldsymbol{y}) + N_{\boldsymbol{R}}\mathcal{M}, \end{aligned} \tag{17}$$

where $f(\boldsymbol{R}^\top \boldsymbol{R}) = \langle \boldsymbol{C}, \boldsymbol{R}^\top \boldsymbol{R}\rangle$ and $N_{\boldsymbol{R}}\mathcal{M}$ represents the normal cone of $\mathcal{M}$ at R. In order to discuss the framework convergence, we use the following two stopping criteria in solving the subproblem in (10):

$$\|\text{grad}\Phi_k(\boldsymbol{R}^{k+1})\|_F \le \epsilon_k, \tag{18}$$

$$\Phi_k(\boldsymbol{R}^{k+1}) - \inf_{\boldsymbol{R}\in\mathcal{M}} \Phi_k(\boldsymbol{R}) \le \epsilon_k. \tag{19}$$

First, under the condition (18), we have the following conclusion for identifying the KKT condition satisfaction.

Theorem 1. *Assume that the sequence* $\{\boldsymbol{R}^k, \boldsymbol{W}^k\}$ *obtained by RGBB satisfies the condition* (18) *and let* $\boldsymbol{R}^*$ *and* $\boldsymbol{W}^*$ *be limit points of* $\{\boldsymbol{R}^k\}$ *and* $\{\boldsymbol{W}^k\}$*. Suppose* $lim_{k\to\infty}\epsilon_k = 0$ *and* $\sigma_{k+1} = \sigma_k$*. Then,* $(\boldsymbol{R}^*, \boldsymbol{W}^*)$ *satisfies the KKT conditions* (17) *for the optimization problem.*

Then, with condition (19), we establish the global convergence of the augmented Lagrangian method as:

Theorem 2. *Let* $\boldsymbol{X}^*$ *and* $\boldsymbol{W}^*$ *be limit point of* $\{\boldsymbol{X}^k\}$ *and* $\{\boldsymbol{W}^k\}$*. Assume that* $\{\epsilon_k\}$ *is bounded. Then, we have* $h(\boldsymbol{W}^*) < \infty$*. For any* $\boldsymbol{X} \in \mathcal{D}$ *and* $\boldsymbol{W} \in \mathbb{R}^{n\times n}$ *satisfying* $h(\boldsymbol{W}) < \infty$*, we would have*

$$\|\boldsymbol{P}\odot\boldsymbol{X} - \boldsymbol{W}\|_F^2 \le \|\boldsymbol{P}\odot\boldsymbol{X}^* - \boldsymbol{W}^*\|_F^2, \tag{20}$$

Here $\mathcal{D} = \{\boldsymbol{X} | x_{ii} = 1, \boldsymbol{X} \ge 0\}$ *in our TPL decomposition problem. Moreover, if we further assume that* $lim_{k\to\infty}\epsilon_k = 0$ *and* $\sigma_{k+1} = \sigma_k$ *as in Theorem 1. Then,* $(\boldsymbol{X}^*, \boldsymbol{W}^*)$ *is a global minimizer of (5).*

For any $\boldsymbol{R} \in \mathcal{M}$ and $W \in \mathbb{R}^{n\times n}$, (20) holds according to Theorem 5.1 in [13]. Similarly, the global convergence of ALM follows from Theorem 5.2 in [13].

IV. Experimental Results

We implement LRSDP in C++, using Eigen [14] library as the backend and Spectra library [15] as the eigenvector solver. Then LRSDP is integrated into the OpenMPL framework [16] as an optional coloring solver. The experiments are conducted on the ISPD'19 benchmarks under the same problem setting as OpenMPL [16]. Each selected layer `n` on test `m` is represented by `testm_n` in the following tables. For example, `test1_100` represents layer `100` on test `1` of ISPD2019. As OpenMPL uses one thread at solver level, to fairly compare the performance and efficiency of different solvers, we use one thread in OpenMPL for evaluation. All experiments are tested on a Linux machine with two 20-core Intel Xeon Gold 6230 CPUs @ 2.10GHz and 500 GB memory.

In this section, we demonstrate the effectiveness of LRSDP by comparing our method with three state-of-the-art TPL decomposition methods, including ILP [17], exact cover (EC) [16], and SDP [18] employing interior-point methods, all of which are available coloring algorithms in OpenMPL. Specifically, Gurobi is adopted as the ILP solver, and CSDP is utilized as the SDP solver. Our method leverages LRSDP for the decomposed subgraphs with > 24 vertices and uses CSDP for the rest, since CSDP is sufficiently fast on these tiny subgraphs. To be fair, both CSDP and LRSDP use double-precision. Meanwhile, to solve within a reasonable time, the three algorithms, including ILP, CSDP, and ours, which support setting up a time limit, are terminated when the runtime on a single subgraph exceeds one hour. We also set the total time limit for each test case to 16 hours. Note that `test4_102` and `test10_102` are not included in our experiments as they are so large that would cause a segmentation fault in OpenMPL before being sent to the coloring solver. All the other layers in ISPD'19 are included in our experiments, and they are divided into two tables based on whether they can be solved by all four decomposers or not. Fig. 5 shows the decomposition result for the case `test6_100` of ISPD'19 benchmarks.

First, TABLE I compares the results of the test cases that all four decomposers can solve. Among two SDP-based approaches, our method is 12.48× faster than CSDP on average with 5% lower cost. Notably, in large cases, our method can achieve up to 80.67× speed up with even better solution quality (see `test3_100` in TABLE I). Besides, our approach can achieve 25.80× acceleration and 29% cost reduction compared to the search-based EC algorithm. The ILP outperforms other algorithms for cost yet suffers the most in efficiency. More specifically, our method is 186.62× faster than ILP and only increases about 11% cost, which makes a better trade-off between performance and efficiency.

Second, TABLE II presents the results of cases in which some algorithms crash ('Failed' in table) or exceed the time limit ('TLE' in table). From this table, we observe that our method is able to deal with fairly large cases within the time limit, whereas CSDP is prone to fail on these large layouts. What's more, the runtime of interior-point-based CSDP increases dramatically with the rising of problem size, demonstrating the stability and scalability of decomposition-based LRSDP. The EC algorithm is unable to obtain a feasible solution after limiting the runtime of a single subgraph, while its total runtime is also unacceptable in large-scale layouts. ILP may obtain a feasible solution by constraining the running time, yet the solution quality will decrease significantly with limited runtime. Taking `test4_101` as an example, our approach can achieve 46% lower cost than ILP. Our method costs more time here on `test4_101` because SDP typically

TABLE I Experiments on different decomposition algorithms. The cases can be solved by all the 4 decomposers.

	Vertices			ILP				EC				CSDP				Ours			
test case	Total	Mean	Max	conflict	stitch	cost	time/s	conflict	stitch	cost	time/s	conflict	stitch	cost	time/s	conflict	stitch	cost	time/s
test1 100	8073	25	171	241	299	270.9	88.9	364	266	390.6	17.3	269	287	297.7	4.5	262	285	290.5	2.8
test1 101	4398	61	834	78	138	91.8	3739.1	156	129	168.9	104.6	94	134	107.4	34.1	98	141	112.1	4.8
test1 102	109	16	46	1	1	1.1	2.2	1	2	1.2	0.0	1	1	1.1	0.1	1	1	1.1	0.1
test2 100	253454	34	1068	5046	8934	5939.4	22120.4	8996	8626	9858.6	1910.8	6439	8179	7256.9	330.1	6456	8202	7276.2	101.2
test2 102	13021	42	2375	213	502	263.2	12243.6	565	313	596.3	3794.0	479	475	526.5	579.8	297	486	345.6	28.6
test3 100	21064	92	7060	680	757	755.7	24566.2	1271	213	1292.3	10213.0	1058	1109	1168.9	13577.3	911	733	984.3	168.3
test3 101	8682	71	2858	130	270	157.0	10422.4	343	141	357.1	3806.8	196	276	223.6	854.4	194	266	220.6	30.7
test3 102	76	13	26	2	1	2.1	0.1	1	1	1.1	0.0	2	1	2.1	0.0	2	1	2.1	0.0
test5 100	9187	19	781	354	330	387.0	5523.0	495	329	527.9	90.9	396	329	428.9	43.9	402	321	434.1	6.8
test5 101	12515	20	246	467	232	490.2	113.1	601	300	631.0	29.3	527	228	549.8	9.9	496	229	518.9	3.8
test5 102	8265	51	3295	197	174	214.4	7225.2	379	66	385.6	707.8	262	151	277.1	1526.5	238	144	252.4	40.8
test6 102	26540	28	978	115	482	163.2	451.1	296	567	352.7	99.6	144	477	191.7	65.0	150	479	197.9	10.8
test7 100	287412	18	2678	8424	9740	9398.0	36696.6	10585	10145	11599.5	2401.7	9020	9509	9970.9	2936.4	9089	9490	10038.0	698.6
test8 100	95194	8	78	5683	4606	6143.6	158.2	5785	4586	6243.6	50.9	5750	4547	6204.7	47.2	5752	4549	6206.9	38.0
test8 101	553934	25	4897	6199	13139	7512.9	52660.6	10780	14896	12269.6	14176.5	7275	12741	8549.1	7466.4	7235	12840	8519.0	820.7
test9 100	144539	8	71	8739	6969	9435.9	249.3	8966	6884	9654.4	80.3	8842	6880	9530.0	73.1	8841	6879	9528.9	60.3
test10 100	211030	10	362	9775	9580	10733.0	409.3	10197	9406	11137.6	195.2	9963	9457	10908.7	115.9	9964	9457	10909.7	94.7
average ratio	–	–	–	0.87	1.03	0.89	186.62	1.33	0.97	1.29	25.80	1.05	1.03	1.05	12.48	1.00	1.00	1.00	1.00

TABLE II Experiments on different decomposition algorithms. Some algorithms crash ('Failed') or exceed the time limit ('TLE').

	Vertices			ILP				EC				CSDP				Ours			
test case	Total	Mean	Max	conflict	stitch	cost	time/s	conflict	stitch	cost	time/s	conflict	stitch	cost	time/s	conflict	stitch	cost	time/s
test2 101	165137	90	3505	TLE	TLE	TLE	TLE	TLE	TLE	TLE	TLE	4553	5026	5055.6	12327.0	3837	5124	4349.4	489.5
test4 100	203283	63	20521	TLE	TLE	TLE	TLE	TLE	TLE	TLE	TLE	Failed	Failed	Failed	Failed	16377	10559	17432.9	18357.1
test4 101	231944	76	57176	18012	6250	18637.0	30439.1	TLE	TLE	TLE	TLE	TLE	TLE	TLE	TLE	12041	7238	12764.8	41421.6
test6 100	632812	28	309	14954	23427	17296.7	11318.6	Failed	Failed	Failed	Failed	17657	22134	19870.4	407.9	17596	22215	19817.5	213.8
test6 101	399298	96	25155	TLE	TLE	TLE	TLE	TLE	TLE	TLE	TLE	Failed	Failed	Failed	Failed	8851	12238	10074.8	7469.2
test7 101	762019	57	31521	TLE	TLE	TLE	TLE	TLE	TLE	TLE	TLE	Failed	Failed	Failed	Failed	13831	18247	15655.7	23700.9
test7 102	314479	92	9473	TLE	TLE	TLE	TLE	TLE	TLE	TLE	TLE	TLE	TLE	TLE	TLE	6480	6192	7099.2	1880.9
test8 102	568566	66	94828	TLE	TLE	TLE	TLE	TLE	TLE	TLE	TLE	Failed	Failed	Failed	Failed	97885	8937	98778.7	16016.8
test9 101	911524	25	12887	TLE	TLE	TLE	TLE	18008	24630	20471.0	40595.0	24475	21418	26616.8	11375.8	12031	21909	14221.9	2471.8
test9 102	903364	56	49695	TLE	TLE	TLE	TLE	TLE	TLE	TLE	TLE	Failed	Failed	Failed	Failed	10015	17668	11781.8	33270.6
test10 101	1304220	38	23389	TLE	TLE	TLE	TLE	TLE	TLE	TLE	TLE	Failed	Failed	Failed	Failed	18480	28608	21340.8	9748.4

requires a rounding procedure to recover the solution to the original problem. Besides, on `test6_100`, our approach is 53× faster than ILP. In addition, our algorithm can further be adjusted from double-precision to single-precision, which can bring about a 2× speed-up in large cases. Generally, the experimental results demonstrate the robustness and effectiveness of our method on large test cases that other state-of-the-art methods cannot solve.

V. Conclusion

In this paper, we propose a scalable low-rank SDP solver for the large-scale TPL decomposition problem. Specifically, the augmented Lagrangian method (ALM) is leveraged here to solve the nonlinear and nonconvex semidefinite program after low-rank decomposition. To fully utilize the problem structure, we restrict the optimization of variables on a smooth manifold and adopt a Riemannian gradient descent method with Barzilai–Borwein steps (RGBB) to solve the subproblem with high efficiency. Experimental results show that our methods are very effective on dense layouts. Specifically, compared with state-of-the-art TPL decomposition algorithms, our proposed LRSDP achieves 186×, 25×, and 12× speed-up with decent solution quality on ISPD'19 benchmarks. In general, the proposed low-rank SDP solver is robust and effective in solving large-scale TPL decomposition problems. We expect to see more optimization-based research on physical design flow as our algorithm provides a new research direction for solving quadratic programs efficiently.

References

[1] B. Yu, K. Yuan, B. Zhang, D. Ding, and D. Z. Pan, "Layout decomposition for triple patterning lithography," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2011, pp. 1–8.

[2] K. Yuan, J.-S. Yang, and D. Pan, "Double patterning layout decomposition for simultaneous conflict and stitch minimization," in *ACM International Symposium on Physical Design (ISPD)*, 2009, pp. 107–114.

[3] B. Yu, K. Yuan, D. Ding, and D. Z. Pan, "Layout decomposition for triple patterning lithography," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 34, no. 3, pp. 433–446, March 2015.

[4] H.-Y. Chang and I. H.-R. Jiang, "Multiple patterning layout decomposition considering complex coloring rules," in *ACM/IEEE Design Automation Conference (DAC)*, 2016, pp. 1–6.

[5] J. Kuang and E. F. Young, "An efficient layout decomposition approach for triple patterning lithography," in *ACM/IEEE Design Automation Conference (DAC)*, 2013, pp. 1–6.

[6] B. Borchers, "CSDP, AC library for semidefinite programming," *Optimization methods and Software*, vol. 11, no. 1-4, pp. 613–623, 1999.

[7] N. Boumal, V. Voroninski, and A. Bandeira, "The non-convex burer-monteiro approach works on smooth semidefinite programs," *Conference on Neural Information Processing Systems (NIPS)*, vol. 29, 2016.

[8] B. Yu and D. Z. Pan, "Layout decomposition for quadruple patterning lithography and beyond," in *ACM/IEEE Design Automation Conference (DAC)*, 2014, pp. 53:1–53:6.

[9] S. Wright, J. Nocedal *et al.*, "Numerical optimization," *Springer Science*, vol. 35, no. 67-68, p. 7, 1999.

[10] Y. Wang, K. Deng, H. Liu, and Z. Wen, "A decomposition augmented lagrangian method for low-rank semidefinite programming," *arXiv preprint arXiv:2109.11707*, 2021.

[11] B. Iannazzo and M. Porcelli, "The riemannian barzilai–borwein method with nonmonotone line search and the matrix geometric mean computation," *IMA Journal of Numerical Analysis*, vol. 38, no. 1, pp. 495–517, 2018.

[12] H. Zhang and W. W. Hager, "A nonmonotone line search technique and its application to unconstrained optimization," *SIAM Journal on Optimization (SIOPT)*, vol. 14, no. 4, pp. 1043–1056, 2004.

[13] E. G. Birgin and J. M. Martínez, *Practical augmented Lagrangian methods for constrained optimization*. SIAM, 2014.

[14] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," 2010.

[15] "Spectra." [Online]. Available: https://spectralib.org/

[16] W. Li, Y. Ma, Q. Sun, L. Zhang, Y. Lin, I. H.-R. Jiang, B. Yu, and D. Z. Pan, "Openmpl: An open-source layout decomposer," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 40, no. 11, pp. 2331–2344, 2020.

[17] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2022.

[18] B. Borchers, "Csdp, a c library for semidefinite programming," *Optimization Methods and Software*, vol. 11, no. 1-4, pp. 613–623, 1999.