KOSIM: A Knowledge-oriented Derivative-free Subspace Method Based on Inexact Model for Inverse Lithography Problems

Presenter : Min-Feng Hsieh

Advisor : Ting-Chi Wang

Outline

- Introduction
- Preliminaries
- Algorithm
- Experimental Result
- Conclusion

Outline

- Introduction
- Preliminaries
- Algorithm
- Experimental Result
- Conclusion

Introduction

- Optical lithography (光學微影技術)
  - 將曝光光源通過設計過的光罩，光罩上面即具有各種圖案可以阻擋或讓光穿透過去。
  - 若光打到正光阻上，該處會被蝕刻; 若是負光阻的話相反。
  - 是製造集成電路（IC）和其他半導體設備的主要工藝之一
    - 將微細的電子元件、晶體管、電容器等結構準確地印刷在半導體材料上

Introduction

- Due to the tiny scale of circuit device size, the influence of interference and diffraction will distort the image on the wafer very much.
- Lots of works are proposed to resolve this kind of distortion:
  - Optical proximity correction (OPC) 光學接近校正

Introduction

- Optical proximity correction (OPC)
  - Adjust the mask layout such that the output pattern approximates the target.
  - Discrete the mask into the matrix, and use the **pixel-wise imaging function** to characterize the imaging procedure.
  - OPC process is model as an **inverse problem**, and is also called as inverse lithography techniques **(ILT)**.
  - Inverse problem is formulated as the **non-convex optimization problem** with respect to the matrix elements.

Introduction

- To solve ILT:
  - Unconstraint Derivative-free optimization (DFO)
    - To evaluate the efficiency of algorithm, another important index is the **total number of function value evaluations (NF).**
    - Many model-based approached require $O(n^2)\sim O(n^3)$ computational complexity in each iteration.
    - Also need $O(n)$ to evaluate the initialization of the model.

Introduction

- To solve ILT:
  - KOSIM
    - A general subspace method for solving DFO problems in ILT.
    - A novel way in constructing subspaces.
    - Develop a projection technique for computing an inexact gradient.
    - Construct good subspaces while it only evaluates $O(1)$ function values in each iteration.
    - Only produces $O(n)$ computational cost in each iteration.

Outline

- Introduction
- Preliminaries
- Algorithm
- Experimental Result
- Conclusion

Preliminaries

- Most common objective function of the ILT problem is the misfit between the image on wafer and the target pattern:
  - Mask : $U \in \mathbb{R}^{N\times N}, U_{ij} \in (0,1)$ (is discrete into matrix)
  - Image function: $I\colon R^{N\times N} \rightarrow R^{N\times N}$
  - Target pattern: U_0

$$\boxed{\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}}$$

- Optimization problem:
  - $\min_{U\in\mathbb{R}^{N\times N}} ||\mathcal{I}(U) - U_0||_F^2, \quad s.t.\ U_{ij} \in \{0,1\}.$

Preliminaries

- Optimization problem:
  - $\min_{U \in \mathbb{R}^{N\times N}} ||\mathcal{I}(U) - U_0||_F^2, \quad s.t.\ U_{ij} \in \{0, 1\}.$
- Edge placement error (EPE)
  - $EPE := |\mathcal{I}(U) - U_0|\,.$
- *L*2 square error
  - $\left\|\mathcal{U}^{-1}(EPE)\right\|_2^2,$

$$\boxed{\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}}$$

Preliminaries

- However, we will not directly accept an arbitrary solution of the above optimization problem for industry production, because of the irregularity of the corrected mask
- Will do the convolution operation for $N_t \in \mathbb{N}$ times on arbitrary real matrix $U \in \mathbb{R}^{N\times N}$, which the convolution core $\widehat{H}$:

$$\hat{H} := \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}.$$

- Then U will be truncated with the following function:

$$\mathcal{T} := \begin{cases} 1, & x \geq 0.5 \\ 0, & x < 0.5 \end{cases}.$$

Preliminaries

- We define the previous operation (convolution and truncation) as M: $\mathbb{R}^{N\times N} \rightarrow \mathbb{R}^{N\times N}$:

$$\mathcal{M}(U) := \mathcal{T}\left(\left(\hat{H}*\right)^{(N_t)} U\right).$$

- Thus the **modified optimization problem** becomes:

$$\min_{x\in\mathbb{R}^{N^2}} \ ||\mathcal{I}\left(\mathcal{M}\left(\mathcal{U}(x)\right)\right) - U_0||_F^2 .$$

- Where the matrization operator:

$$\mathcal{U} : \mathbb{R}^{N^2} \rightarrow \mathbb{R}^{N\times N}, \ x \mapsto (x_{Ni+j})_{ij}.$$

Preliminaries

$$U : \mathbb{R}^{N^2} \to \mathbb{R}^{N \times N}, \quad x \mapsto \to \begin{bmatrix} x_1 & x_2 & \cdots & x_N \\ x_{N+1} & x_{N+2} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{(N-1)N+1} & x_{(N-1)N+2} & \cdots & x_{N^2} \end{bmatrix}$$

Outline

- Introduction
- Preliminaries
- Algorithm
- Experimental Result
- Conclusion

Algorithm - knowledge-oriented inexact gradient

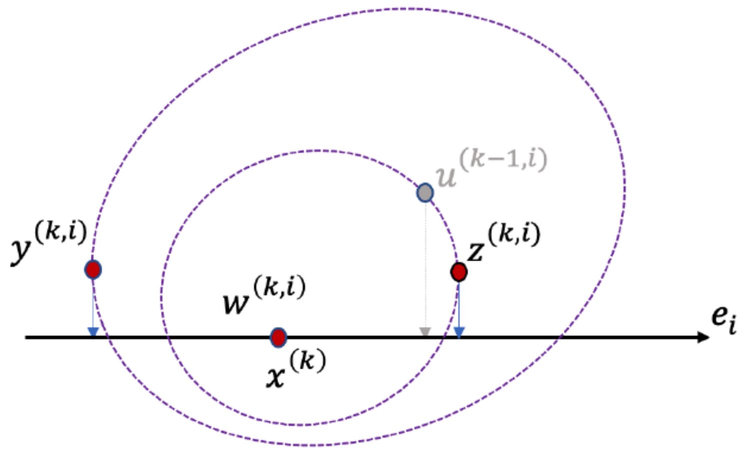
- To compute the inexact gradient $g^{(k)}$:
  1. To determine the m dimensional subspace $S_g^{(k)}$ (m << n, where m = $O(1)$).
  2. Approximately calculate the projection $\mathcal{P}_{S_g^{(k)}} \nabla f(x^{(k)})$.
     - Invokes a simple finite difference with step length $\rho(k)$, which is adaptively chosen.
- When prior information is available, i.e., a prior generator D is inputted, we set $D\left(x^{(k)}\right) \in S_g^{(k)}$.

Algorithm - Separable quadratic inexact model

- Lower computational complexity compare with quadratic-model-based approaches.
- $m^{(k)}(x) = \sum_{i=1}^{n} m_i^{(k)}(x_i) - (n-1) f\left(x^{(k)}\right)$
  - $m_i^{(k)}(u)$ is a 1D quadratic model approximating $f\left(x^{(k)} + \left(u - x_i^{(k)}\right) e_i\right)$, $i = 1, \ldots, n$.
  - Use sampling points off the line $x^{(k)} + span(e_i)$, so that historical sampling points can be utilized.
  - Assume three sampling points for the ith model $m_i^{(k)}$ consist $\mathcal{Y}_i^{(k)} := \{y^{(k,i)}, z^{(k,i)}, w^{(k,i)}\}$, then the interpolation follows:

$$m_i^{(k)}(u) := \sum_{cyc} f\left(y^{(k,i)}\right)\left(u - z_i^{(k,i)}\right)\left(u - w_i^{(k,i)}\right) \Big/ \left[\left(y_i^{(k,i)} - z_i^{(k,i)}\right)\left(y_i^{(k,i)} - w_i^{(k,i)}\right)\right].$$

Algorithm - Detail



A. Initializing and Updating the sampling points

- $y^{(0,i)} = z^{(0,i)} = w^{(0,i)} := x^{(0)} + (i-2)r\mathbf{1}, \quad i = 1,2,3$
  - r is an arbitrary parameter where $r > 0$.
  - $\mathbf{1} = [1,1,1,\ldots,1]^T$.
- After an iteration, we will get $x^{(k+1)}$, and update one of the sampling points $u^{(k,i)}$ in $\mathcal{Y}_i^{(k)}$
  - $\mathcal{Y}_i^{(k)}$ will be updated to $\mathcal{Y}_i^{(k+1)}$
  - $u^{(k,i)} := \min_{u \in \mathcal{Y}_i^{(k)}} \dfrac{\left|u_i - x_i^{(k+1)}\right|}{f(u) - f\left(x^{(k+1)}\right)}, \; i = 1, \ldots, n,$

Algorithm - Detail

B. Solving the model

- The direction of the model
  - $$d^{(k)} := \min_{d \in \Omega^{(k)}} m^{(k)}\left(x^{(k)} + d\right).$$
- Box region $\Omega^{(k)}$
  - $$\Omega^{(k)} := \prod_{i=1}^{n} \left[-r_i^{(k)}, r_i^{(k)}\right],$$

$$\boxed{r_i^{(k)} := \max\left\{\left|y_i^{(k,i)} - x_i^{(k)}\right|, \left|z_i^{(k,i)} - x_i^{(k)}\right|, \left|w_i^{(k,i)} - x_i^{(k)}\right|\right\},\ i = 1, \ldots, n.}$$

Algorithm - Detail

C. Solving the subproblem

- Construct a complete three-dimensional quadratic model $q(s)$ approximating $f(x^{(k)} + B^{(k)}s)$ by interpolation using 10 sampling points $s^{(k,1)}$ ... $s^{(k,10)}$.
- $B(k) \in \mathbb{R}^{N\times 3}$ stands for the basis matrix of $S^{(k)}$ after a Gram-Schmidt procedure.
- Then we solve $\min_{s\in\mathbb{R}^3}[q(s) + \lambda \parallel s \parallel^2]$ and obtain $s_q^{(k)}$.
  - $\lambda \geq 0\ satisfies\ \nabla^2 m\left(x^{(k)}\right) + \lambda I\ > 0.$
- $$s^{(k)} := \arg\min \left\{ f\left(x^{(k)}\right), f\left(x^{(k)} + B^{(k)} s^{(k,1)}\right), \ldots, f\left(x^{(k)} + B^{(k)} s^{(k,10)}\right), f\left(x^{(k)} + B^{(k)} s_q^{(k)}\right) \right\}.$$

Algorithm - Detail

D. Updating the finite-difference step length

- For updating the finite difference step length $\rho^{(k)}$, we choose a scaling factor $\sigma^{(k)} > 0$ satisfying $0 < \sigma_l \leq \sigma^{(k)} \leq \sigma_u$.
  - $\rho^{(k+1)} := \max\left\{\frac{1}{2}\rho^{(k)}, \sqrt{\left[f\left(x^{(k)}\right) - f\left(x^{(k+1)}\right)\right] / \sigma^{(k)}}\right\}$
  - $\sigma^{(k)} := \max\left\{10^{-6}, \min\left\{10^{6}, \left\|\nabla^2 m^{(k)}\right\|_2\right\}\right\}$

Algorithm - Detail

Step	Complexity	NF
Initializing the model	$\mathcal{O}(n)$	2
Updating the model	$\mathcal{O}(n)$	0
Computing the inexact gradient	$\mathcal{O}(n)$	m
Solving the subproblem	$\mathcal{O}(n)$	10

Table 1: Computational complexity and NF of each step in Algorithm 1

Algorithm 1: Knowledge-Oriented Subspace Method based on Inexact Model (KOSIM)

Input: objective function $f : \mathbb{R}^n \to \mathbb{R}$, prior generator $\mathcal{D} : \mathbb{R}^n \to \mathbb{R}^n$.

Output: $x^{(k)}, f\left(x^{(k)}\right)$.

Choose an initial guess $x^{(0)} \in \mathbb{R}^n$, and set $k := 0$.

Initialize $m^{(0)}$.

while *certain termination criterion is not satisfied* **do**

- Determine random subspace $\mathcal{S}_g^{(k)} \subset \mathbb{R}^n$ satisfying $\mathcal{D}\left(x^{(k)}\right) \in \mathcal{S}_g^{(k)}$ (if $\mathcal{D} \neq$ None).
- Approximately compute $g^{(k)} := \mathcal{P}_{\mathcal{S}_g^{(k)}} \nabla f\left(x^{(k)}\right)$.
- Compute a solution $d^{(k)}$ of $m^{(k)}$.
- Define subspace $\mathcal{S}^{(k)} := \operatorname{span}\left(g^{(k)}, d^{(k)}, s^{(k-1)}\right)$.
- Inexactly solve $\min_{s \in \mathcal{S}^{(k)}} f\left(x^{(k)} + s\right)$ and obtain $s^{(k)}$.
- Update $x^{(k+1)} := x^{(k)} + s^{(k)}$.
- Update $m^{(k)}$ to $m^{(k+1)}$.
- Set $k := k + 1$.

Outline

- Introduction
- Preliminaries
- Algorithm
- Experimental Result
- Conclusion

Experimental Result - Gradient

- All codes are implemented in C, and packed as a Python interface.
- All the tests of KOSIM were performed on a Lenovo ST8810 cluster.

n	$\|\|Ax - b\|\|/\|\|b\|\|$	iteration number		walltime		speed-up ratio
		CG	KOSIM	CG	KOSIM	
100	9.67e-04	13307	464	2.11s	0.14s	14.45×
200	5.36e-02	40000	765	7.36s	0.43s	16.97×
400	7.50e-04	66537	1349	17.30s	1.64s	10.51×
800	1.38e-03	103658	2093	114.96s	7.38s	15.55×

Table 2: Comparison of KOSIM with gradient information and CG

Experimental Result - DFO

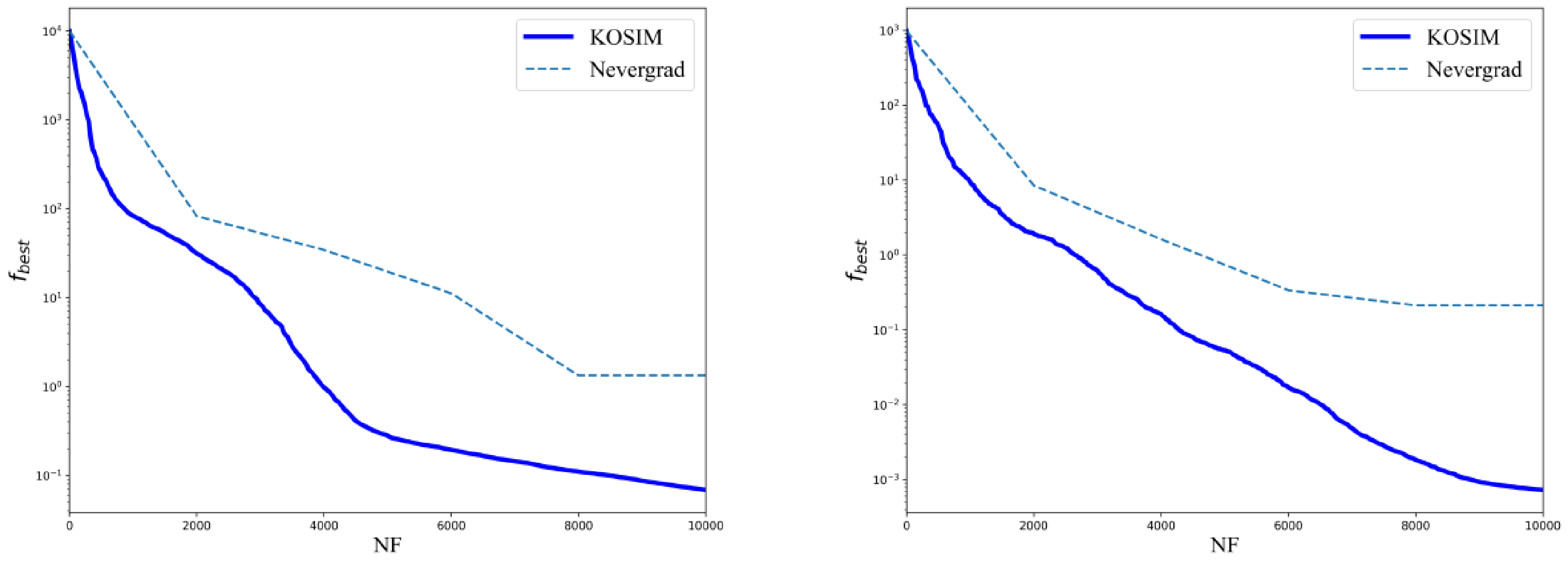


Figure 2: Comparing KOSIM and Nevergrad on problem constructed by CHROSEN (left) and SINQUAD (right) (n = 10000)

Experimental Result - DFO

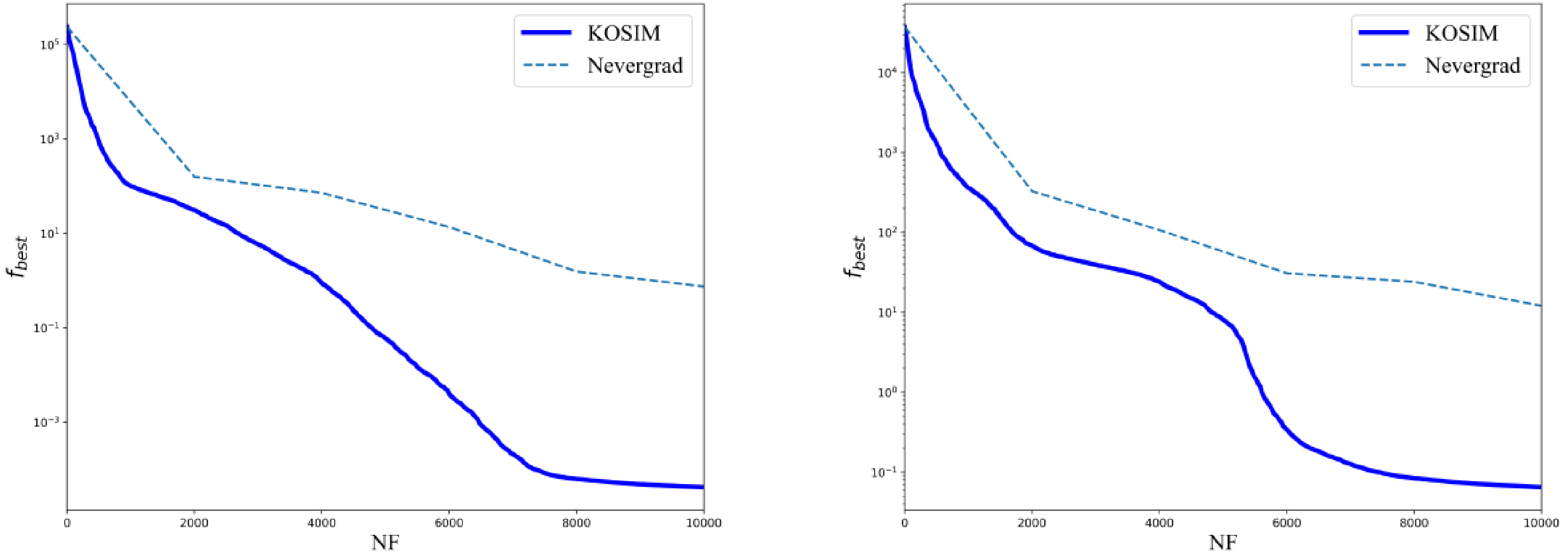


Figure 3: Comparing KOSIM and Nevergrad on problem constructed by CHROSEN (left) and SINQUAD (right) ($n = 50000$)

Experimental Result – Solve ILT

- $N_t = 50$.

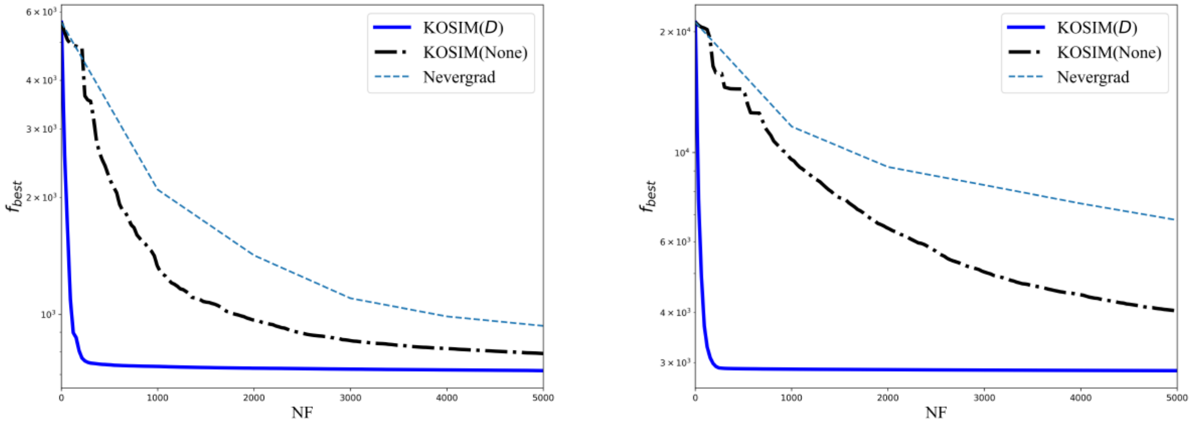
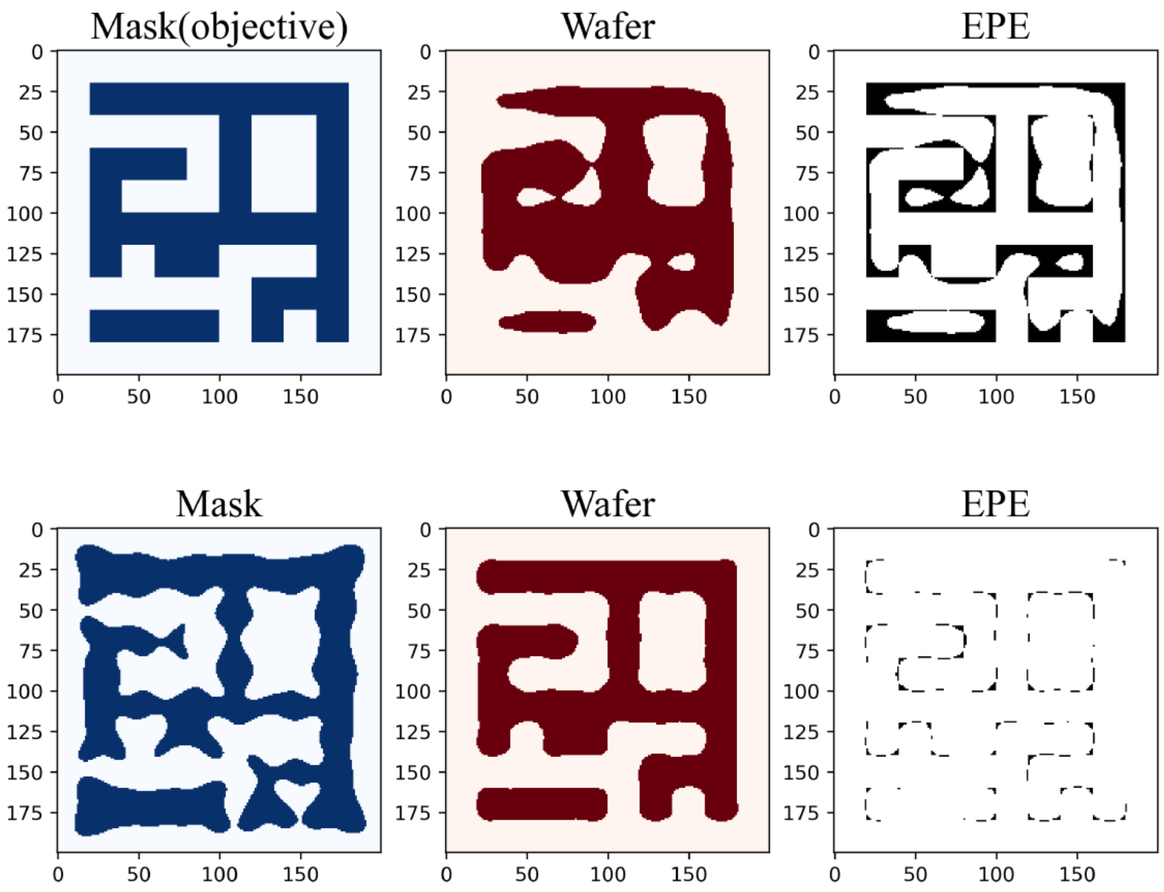


Figure 4: This figure shows the comparison of KOSIM with and without prior and Nevergrad on two ILT problems with mask size 200×200 and 400×400. Best function values within 5000 function evaluations are reported.

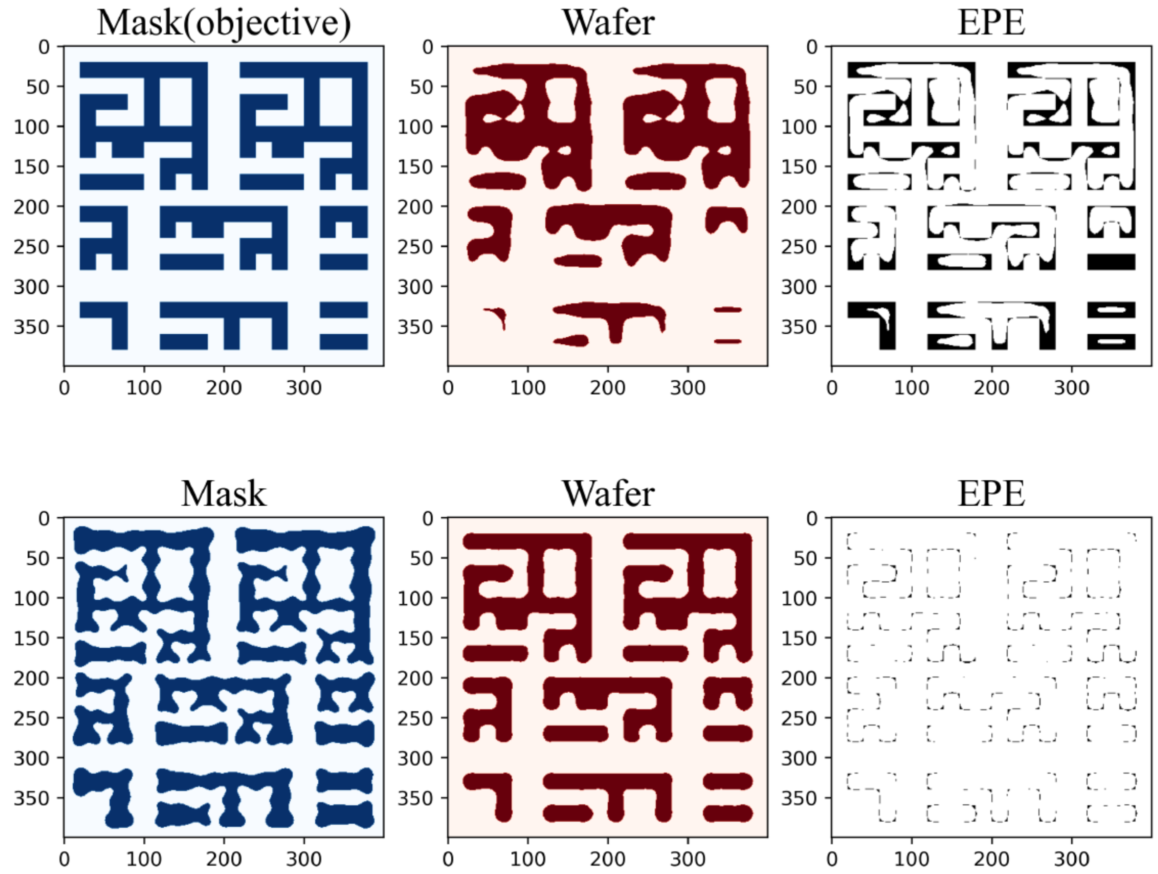
Experimental Result – Solve ILT

- Mask size = 200 × 200



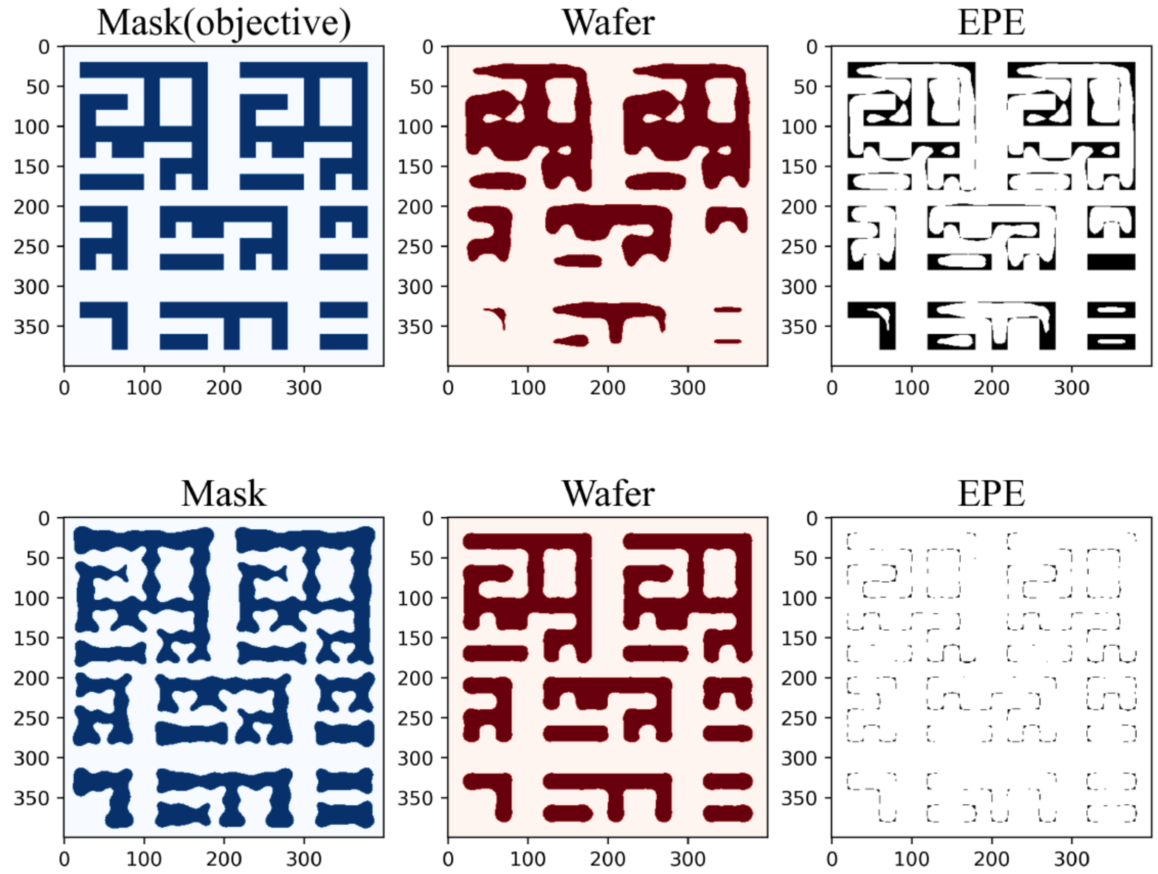
Experimental Result – Solve ILT

- Mask size = 400 × 400



Experimental Result – Solve ILT

- Mask size = 1000 × 1000



Outline

- Introduction
- Preliminaries
- Algorithm
- Experimental Result
- Conclusion

Conclusion

- Several techniques are developed for KOSIM:
  - Knowledge-oriented inexact gradients.
  - Maintaining inexact models.
  - Constructing subspaces.
  - Solving the subproblems.
- KOSIM evaluates only O(1) function values and has only O(n) computational complexity per iteration.

Thank you for listening!