Optical Proximity Correction with Hierarchical Bayes Model

Tetsuaki Matsunawa[a], Bei Yu[b] and David Z. Pan[b]

[a] Center for Semiconductor Research & Development, Toshiba Corp., Kawasaki, Japan
[b] ECE Department, Univ. of Texas at Austin, Austin, TX, USA
tetsuaki.matsunawa@toshiba.co.jp, {bei, dpan}@cerc.utexas.edu

ABSTRACT

Optical Proximity Correction (OPC) is one of the most important techniques in today's optical lithography based manufacturing process. Although the most widely used model-based OPC is expected to achieve highly accurate correction, it is also known to be extremely time-consuming. This paper proposes a regression model for OPC using a Hierarchical Bayes Model (HBM). The goal of the regression model is to reduce the number of iterations in model-based OPC. Our approach utilizes a Bayes inference technique to learn the optimal parameters from given data. All parameters are estimated by the Markov Chain Monte Carlo method. Experimental results show that utilizing HBM can achieve a better solution than other conventional models, e.g., linear regression based model, or non-linear regression based model. In addition, our regression results can be fed as the starting point of conventional model based OPC, through which we are able to overcome the runtime bottleneck.

Keywords: Lithography, Optical Proximity Correction (OPC), Hierarchical Bayes Model, Machine Learning, Model-based OPC

1. INTRODUCTION

Although several types of emerging lithography techniques, such as Extremely Ultraviolet Lithography (EUVL),[1] Nano-Imprint Lithography (NIL)[2] and Directed Self-Assembly Lithography (DSAL),[3] are being developed, optical lithography is still widely used in the semiconductor industry in view of its cost. In the optical lithography based manufacturing process, Optical Proximity Correction (OPC) is one of the most important techniques. Fig. 1 shows an overview of the most widely used model-based OPC in which the displacements of fragmented edges in a mask layout are calculated based on a lithography simulation so that the difference between target edge and printed image obtained by lithography simulation is minimized.[4] Although this method is expected to achieve highly accurate correction, it is also known to be extremely time-consuming. To resolve this issue, several fast mask optimization methods, such as inverse lithography, linear regression-based OPC and non-linear regression-based OPC, have been proposed.[5–8]

The inverse lithography technique is expected to obtain a highly accurate prediction model because the ideal mask shape is derived from inverse transformation of the desired resist image on a wafer. However, this method cannot be applied to full-chip calculation owing to huge computational cost.[5] Meanwhile, from the viewpoint of runtime, regression-based methods can be promising candidates to reduce the runtime of model-based OPC because the displacements of fragmented edges in a layout are quickly estimated by using a comparatively simple regression model, which is trained with supervised displacement data calculated by the conventional model-based OPC technique. Since these methods use a straightforward regression technique, they are capable to be applied to full-chip calculation. Conventional regression-based methods, however, have several issues in terms of prediction accuracy.

A linear regression-based OPC method is proposed by Allan Gu et al. and it showed the capability of reducing OPC runtime by using the regression results as a start point of model-based OPC.[6] Also, nonlinear regression-based OPC methods using support vector machine (SVM) or artificial neural network (ANN) are presented[7,8] and showed the possibility of runtime reduction against model-based OPC. However, it is challenging to train a robust regression model owing to the over-fitting problem, whereby a model has poor predictive performance. Furthermore, as device feature sizes continue shrinking, it is increasingly difficult to achieve a highly accurate prediction model owing to the model complexity, since the correction amounts for edge displacements greatly vary in accordance with the surrounding environment of fragmented edges and optical proximity effects (OPEs). In

Optical Microlithography XXVIII, edited by Kafai Lai, Andreas Erdmann, Proc. of SPIE Vol. 9426, 94260X
 · doi: 10.1117/12.2085787

order to overcome these problems that inhibit the practical use of regression-based methods, this paper proposes a new regression-based OPC framework. We develop a hierarchical Bayes model (HBM) to achieve a robust prediction model while preventing the over-fitting issue. Moreover, we propose a new layout representation technique to improve the prediction accuracy.

The remainder of this paper is organized as follows: Section 2 presents the background of this work by referring to conventional regression-based OPC approaches. In Section 3, we give the problem formulation and the overall flow of the proposed method. Section 4 describes methodologies of our framework including the proposed layout feature extraction technique and the HBM training method. Section 5 presents the experimental results, and is followed by the conclusion in Section 6.

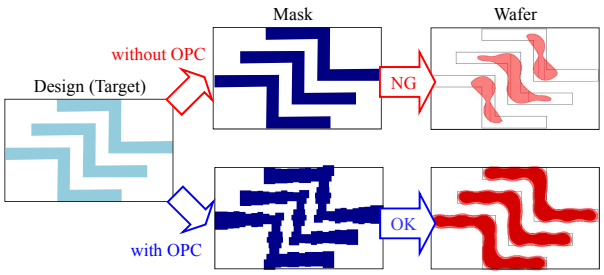


Figure 1. Optical Proximity Correction.

2. PRELIMINARIES

Several regression-based OPC methods have been proposed and shown to be effective for reducing the runtime of model-based OPC. However, as layout features become complicated in the future technology nodes, it is difficult to learn a highly accurate prediction model using straightforward linear regression technique. Specifically, in the linear regression-based OPC,[6] the following simple linear model involves a linear combination of the input layout feature vectors.

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \cdots + w_D x_D \tag{1}$$

where D is the number of dimensions of feature vectors $\mathbf{x}$ described as $\mathbf{x} = (x_1, \ldots, x_D)^{\mathrm{T}}$, $\mathbf{w} = (w_0, \ldots, w_D)^{\mathrm{T}}$, and w_0 is the bias parameter. The details of the feature are described in Section 4. The coefficients of this model $\mathbf{w}$ can be written by using the following normal equations in which $\mathbf{w}$ are derived so that the difference between a predicted edge displacement and a supervised data is minimized.

$$\mathbf{w} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} \tag{2}$$

where $\mathbf{X}$ is the design matrix, whose elements are given by $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^{\mathrm{T}}$, N is the total number of supervised data samples, and $\mathbf{y} = (y_1, \ldots, y_N)^{\mathrm{T}}$ are the target vectors corresponding to the edge displacements obtained by model-based OPC. Although the least square method is applicable to uncomplicated regression problems, it includes the following two disadvantages: (1) Over-fitting: Model learning with limited samples or non-representative samples causes over-fitting or over-generalization that lowers predictive performance of the regression model. (2) Limited applications: It is difficult to apply the linear regression method to complex nonlinear phenomena because the algorithm includes a linear assumption in which all model parameters are linearly correlated with input feature vectors.

To model complex nonlinear problems, several nonlinear regression techniques based on artificial neural network (ANN) or support vector regression (SVR) have been proposed.[7,8] These related works showed that it is possible to model nonlinear problems through mechanisms of interconnected neurons[7] or kernel methods.[8] However, the over-fitting issue is still open since a sufficient amount of supervised data is not necessarily given in all cases. Furthermore, practical applications of the regression model are also restricted even with nonlinear algorithms because there are relatively few adjustable model parameters that can improve predictive performance.

On the one hand, OPEs from adjacent patterns contribute to one of the factors that increases difficulty of model training. This is because the tendency of displacement of mask edges differs among different types of edges.[6] Fig. 2 illustrates examples of edge types including normal edge, convex edge, concave edge, or line end edge. The difference of edge types leads to different displacement amounts. In the linear regression-based OPC,[6] an accurate regression model is realized by separately learning a regression model per edge. Since the separate model requires a larger number of supervised data, this approach also faces the over-fitting issue. A new efficient regression algorithm is required to realize flexible modeling for complex phenomena consisting of a small number of data.

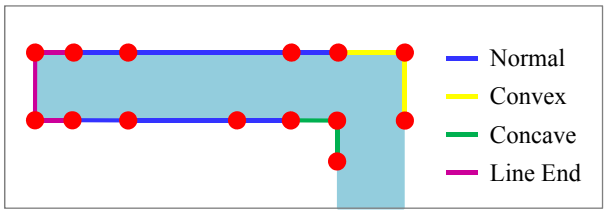


Figure 2. Examples of edge types.

3. PROBLEM FORMULATION AND OVERALL FLOW

3.1 Problem Formulation

To evaluate the performance of regression-based OPC, we define the RMSE as follows:

Definition 1 (Root Mean Square Error: RMSE).

$$RMSE = \sqrt{\frac{\sum_i^N (y_i - \hat{y}_i)^2}{N}} \tag{3}$$

where N is the total number of supervised data samples, y_i is the fragment movement on i^{th} edge determined by model-based OPC and $\hat{y}$ is the predicted fragment movement.

We give the problem formulation of regression-based OPC as follows:

Problem 1 (Regression-based OPC). *Given layout data including the displacement amount of all fragments, a regression model is calibrated to predict displacements of unknown fragments in a verification layout. The goal of the regression-based OPC is to minimize the RMSE.*

3.2 Overall Flow

Our regression-based OPC method consists of two phases, a learning phase and a testing phase as shown in Fig. 3. In the beginning of the learning phase, a training layout data is given.

Then based on a design rule check (DRC) all the edges of the training layout are fragmented into different types, such as normal edge, convex edge, concave edge, and line end edge (see Fig. 2). After recognizing these different types, the displacement amount of each fragment is computed by using the conventional model-based OPC technique as a supervised data. Meanwhile, layout features in each fragmented edge are extracted as described in Section 4. Finally, a prediction model is trained using the supervised displacement data obtained by model-based OPC and the extracted layout features.

In the testing phase, a verification layout data is used as an input. Based on the DRC in the learning phase, all edges of the layout are fragmented and edge types are recognized as well. After layout feature extraction, the edge displacements in the verification layout are predicted by the model trained in the learning phase.

4. METHODOLOGIES

In this section, we first discuss a layout feature extraction method for OPC regression, and then a model training approach including a concept of hierarchical Bayes model (HBM) and a Markov Chain Monte Carlo technique is presented.

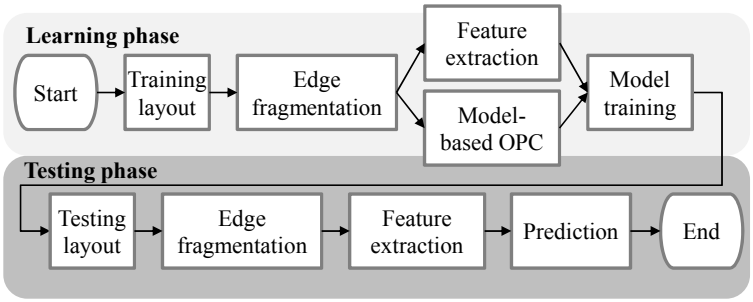
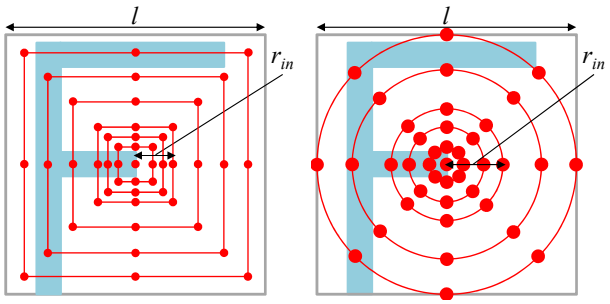


Figure 3. Overview of the proposed method.

4.1 Feature Extraction

In a prediction model, the input layout is difficult to be directly handled, owing to its high-dimensional space. Therefore, the geometrical information of each fragmented edge is encoded into a feature vector. For instance, assuming there are 0.6 μm^2 areas on 1 nm grid to consider edge displacement of a pattern, the number of dimensions of the pattern becomes 600×600. In this kind of high-dimensional space, $\mathbb{R}^{360000}$, it is difficult to prepare a sufficient number of supervised samples and solve the regression problem within a practical time.

Layout feature extraction is a very important procedure in the regression-based OPC because the prediction model performance is heavily determined by the types of layout features. Recently, concentric square sampling (CSS) has been proposed in the linear regression-based OPC[6] as an efficient feature for OPC modeling and showed reasonable prediction accuracy. In this paper we propose a new layout feature extraction method, concentric circle area sampling (CCAS), for further improvement of OPC regression. In the following, we first present an overview of the CCS and then our proposed CCAS is introduced.



(a) CSS. (b) CCAS.

Figure 4. Layout features.

4.1.1 Concentric Square Sampling (CSS)

CSS is proposed in the linear regression-based method[6] to train a linear model for OPC regression. A feature vector $\mathbf{x}$ contains subsampled pixel values on concentric squares of layout patterns. Fig. 4(a) indicates the basic concept of CSS of "F" shaped test pattern. Parameters of a feature consist of the total size of the encoding area l and the sampling density controlling parameter r_{in}. The radius of the concentric square is $0, 4, 8, \cdots, r_{in}, r_{in}+8, r_{in}+16, \cdots, l/2$ pixels, respectively. The total number of dimensions will be 257 if the parameters l and r_{in} are 0.4 μm and 60 nm, respectively. It can be expected to achieve a high generalization capability because the feature can correctly express a positional relationship to layout patterns. Although CSS can reduce the number of dimensions compared to exploring all pixel values in the layout patterns, prediction model training might remain difficult because the number of dimensions is still high.

4.1.2 Concentric Circle Area Sampling (CCAS)

This feature is first presented in this paper and represents pattern information that affects propagation of diffracted light from a mask pattern. Fig. 4(b) indicates the basic concept of CCAS of "F" shaped test pattern. Parameters of a feature consist of the total size of the encoding area l and the sampling density controlling parameter r_{in}. Although at first glance the basic concept is almost the same as that of CSS, CCAS can be expected to achieve a better generalization capability than CSS because the subsampled pixel values in CCAS correspond to important physical phenomena, in which diffracted light from a mask pattern is propagated concentrically. This feature also can improve the expressive capability of a layout by sampling not only pixel values but by summation of pixels around the sampling area. In Section 5 we will further analyze the advantages of this feature, and compare it with the conventional CSS.

4.2 OPC Modeling

The principal difference between our approach and the conventional linear regression model is that our model is not restricted by linear correlation of model parameters. This indicates that it is expected to realize flexible modeling even of a complicated phenomenon having a large variation in input data or including unknown variables that cannot be measured. In this subsection, we present the concept of the proposed OPC regression, followed by the solution of the parameter estimation technique.

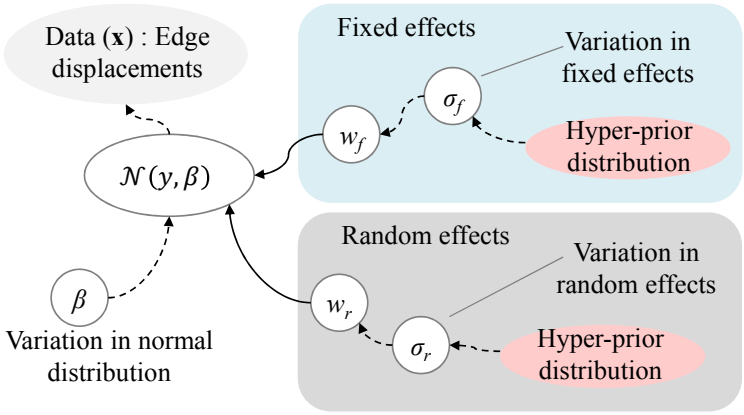


Figure 5. Concept of the proposed HBM.

4.2.1 Hierarchical Bayes Model (HBM)

As mentioned in Section 2, the edge displacements vary depending on edge types, such as normal edge, convex edge, concave edge, and line end edge. To simulate the trend of edge displacement, our proposed model considers the regression-based OPC problem with a generalized linear mixed model (GLMM) regarding the edge types as random effects that can express an effect of variance of supervised data. Besides, the GLMM is modeled by a Bayesian approach to deal with many random effects. Fig. 5 shows the basic concept of our proposed model where data, edge displacement, follows the Normal distribution of center y and variance σ_y. The model parameters $w_{fj}, \forall j \in D$ correspond to fixed effects that are the common effects for all edges. $w_{rj}, \forall j \in D$ indicate random effects that are identification effects assigned to edge types. σ_f and σ_r are the variances of fixed effects and random effects, respectively. Although the Bayes inference technique requires some prior information, it is not provided in many practical cases. We therefore propose a regression model using non-informative prior distribution for unknown variables. Finally, our proposed model can be written as follows:

$$p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(y(\mathbf{x}, \boldsymbol{\theta}), \sigma_y) \tag{4}$$

$$y(\mathbf{x}, \boldsymbol{\theta}) = w_{f_0} + w_{r_0} + \sum_{j=1}^{D} (w_{f_j} + w_{r_j}) x_j \tag{5}$$

where $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$ is the probability of edge displacements $\mathbf{y}$ given feature vectors $\mathbf{x}$ and model parameters $\boldsymbol{\theta}$, N is the Normal distribution of center y and variance σ_y, and D is the total number of dimensions. The prior distributions are defined as follows:

$$w_{f_i} \sim \mathcal{N}(0, \sigma_f) \tag{6}$$

$$w_{r_i} \sim \mathcal{N}(0, \sigma_r) \tag{7}$$

$$\sigma_y \sim \mathcal{U}(0, 10^4) \tag{8}$$

where $\mathcal{N}(0, \sigma_f)$ is the Normal distribution of center 0 and variance σ, and $\mathcal{U}(0, 10^4)$ is the uniform distribution in the interval $(0, 10^4)$. Because our model assumes there is no prior knowledge about the variances of prior distributions, the model utilizes hierarchical prior distributions as non-informative hyper-priors to consider every possibility of unknown parameters.

$$\sigma_f \sim \mathcal{U}(0, 10^4) \tag{9}$$

$$\sigma_r \sim \mathcal{U}(0, 10^4) \tag{10}$$

The non-informative hyper-priors that follow the uniform distribution in the interval $(0, 10^4)$ indicate that the variance can take any values within the range of 0 and 10^4. Finally, the posterior distribution is derived as follows:

$$p(\boldsymbol{\theta}|\mathbf{y}) = \prod_{i=1}^{N} \prod_{j=0}^{D} p(y_i|\theta_j) p(w_{f_j}|\sigma_f) p(w_{r_j}|\sigma_r) p(\sigma_y) p(\sigma_f) p(\sigma_r) \tag{11}$$

where N is the total number of samples, and θ include all model parameters, such as fixed effects, random effects, variance of y, and variances of fixed effects and random effects. Because the posterior distribution includes hierarchical prior distributions, this model is called the hierarchical Bayes model (HBM). It is difficult to solve all parameters using the likelihood estimation method owing to the complexity of integral computation. Therefore, we estimate all parameters using a sampling technique.

4.2.2 Markov Chain Monte Carlo (MCMC)

MCMC is a method of estimating model variables by sampling parameters from a posterior distribution, or a parameter distribution proportional to the likelihood. Generally, it is difficult to apply likelihood estimation method, which is a method of estimating model variables that maximizes the likelihood function, to a complicated model such as equation (11) because the method must compute the likelihood in the combination of all model parameters. In contrast, it can be expected that appropriate parameters can be obtained by MCMC since the estimation performance is less susceptible to the model complexity.[9]

We briefly describe the flow of MCMC as follows. In this algorithm, a target distribution $\pi(\boldsymbol{\theta})$, a distribution of model parameter $\boldsymbol{\theta}$, can be obtained from a probability distribution $q(\boldsymbol{\theta}')$ called the proposal distribution. The model parameters are estimated by the following steps: suppose the target distribution $\pi(\boldsymbol{\theta}|\mathbf{x})$ of parameter $\boldsymbol{\theta}$ given data $\mathbf{x}$, where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$ and m is the total number of parameters. We first give the initial values of $\boldsymbol{\theta}$ as $\boldsymbol{\theta}^{(0)} = (\theta_1^{(0)}, \ldots, \theta_m^{(0)})$, where $\boldsymbol{\theta}^{(t)}$ indicates a random variable at a time point t. Then, we repeat the following steps for $t = 0, 1, \ldots, k$:

1. Give the initial variables $\boldsymbol{\theta}^{(0)} = (\theta_1^{(0)}, \ldots, \theta_m^{(0)})$.
2. Repeat sampling for $t = 0, 1, \ldots, k$.
   (a) Generate $\boldsymbol{\theta}'$ from $q(\boldsymbol{\theta}', \boldsymbol{\theta}^{(0)}|\mathbf{x})$.
   (b) Generate u from $\mathcal{U}(0, 1)$ and select $\boldsymbol{\theta}^{(t+1)}$ based on the following equation:

$$\boldsymbol{\theta}^{(t+1)} = \begin{cases} \boldsymbol{\theta}', & \text{if } u \leq \alpha(\boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}|\mathbf{x}) \\ \boldsymbol{\theta}^{(t)}, & \text{otherwise} \end{cases} \tag{12}$$

where $\mathcal{U}(0, 1)$ is the uniform distribution in the interval $(0, 1)$, α is the selection rate defined by:

$$\alpha(\boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}|\mathbf{x}) = \min\left\{1, \frac{\pi(\boldsymbol{\theta}'|\mathbf{x}) q(\boldsymbol{\theta}', \boldsymbol{\theta}^{(t)}|\mathbf{x})}{\pi(\boldsymbol{\theta}^{(t)}|\mathbf{x}) q(\boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}'|\mathbf{x})}\right\} \tag{13}$$

This is known as the Metropolis-Hastings (MH) algorithm.[9] Since MCMC requires many iterated calculation to sample stable parameters, evaluating convergence of MCMC is important for measuring the performance of estimated samples. To monitor the convergence of MCMC, this paper uses $\hat{R}$ index, which is proposed by Gelman and Rubin.[10] This can be computed based on the variance in the Markov chains and defined as follows:

$$\hat{R} = \sqrt{\frac{\widehat{\text{var}}(\psi|y)}{W}} \tag{14}$$

$$\widehat{\text{var}}(\psi|y) = \frac{k-1}{k}W + \frac{1}{k}B \tag{15}$$

where k is the number of iterations, B is the between-chain variance and W is the within-chain variance defined by:

$$B = \frac{k}{m-1}\sum_{j=1}^{m}(\bar{\psi}_j - \bar{\psi})^2 \tag{16}$$

$$W = \frac{1}{m}\sum_{j=1}^{m}\left[\frac{1}{k-1}\sum_{i=1}^{k}(\bar{\psi}_{ij} - \bar{\psi}_j)^2\right] \tag{17}$$

where m is the number of chains and ψ is the sampled parameter. It is showed that $\hat{R}$ can evaluate the convergence of the sampling results in actual problems.[11] Empirically, if every $\hat{R}$ is smaller than 1.1, it can be concluded that the sampling results are fully-converged.[11]

MCMC can estimate reasonable model parameters even in high-dimensional parameter space by constructing a Markov chain. Furthermore, model parameters are estimated without a local solution because of a stochastic process even if the likelihood functions include multi peak distributions. Therefore, it can be expected that the parameter distributions nearly equal to the true distributions are obtained by iterative sampling.

5. EXPERIMENTS AND RESULTS

The proposed methodologies are implemented in C++ and Python on a Linux machine with eight 3.4GHz CPUs and 32GB memory. Calibre[12] is used to perform lithography simulation with wavelength $\lambda = 193$ nm and $NA = 1.35$. Two 32 nm node industrial chips in metal routing layer, a layout A and a layout B, are applied. The areas of the layouts A and B are 9291.37 μm^2 and 11702.20 μm^2, respectively. We conducted four experiments related to feature selection, model training, comparison with linear/nonlinear regression, and comparison of model-based OPC, respectively.

5.1 Feature Selection

To evaluate the effectiveness of the proposed layout feature, we compare the predictive performance between the CSS and the CCAS based on the RMSE values in a testing layout when the feature parameters are changed. In this comparison, the regression model is fixed to the linear regression algorithm to compare with [6]. Also, the model is trained with a part of supervised data including all edge types and consists of 5000 samples that are randomly extracted from the training layout A. The RMSEs (nm) on the testing layout are computed by using 5000 samples extracted from the layout B. Table 1 shows the RMSEs from the models trained by the feature parameters within the following ranges: r_{in} is set to be 50 to 500 (nm) and $l = 1.0$ (μm), and the RMSEs that are calculated by using the following feature parameters: l is set to the range of 0.5 to 1.4 (μm) and $r_{in} = 150$ (nm). From the table, the model trained by the proposed CCAS shows a better predictive performance than the model trained by the CSS features. Therefore, we employ the parameters $l = 1.0$ μm and $r_{in} = 150$ nm, which demonstrate the best prediction accuracy, for the experiments described below.

Table 1. Comparison of CSS and CCAS.

r_{in} (nm)	CSS	CCAS
50	4.145	3.952
100	4.039	3.997
150	3.989	3.933
200	3.991	3.954
250	4.014	3.934
300	4.011	3.937
350	4.025	3.942
400	4.016	3.956
450	4.021	3.980
500	4.022	3.997

l (μm)	CSS	CCAS
0.5	4.105	4.068
0.6	4.527	3.991
0.7	4.343	4.053
0.8	4.610	3.962
0.9	4.262	4.029
1.0	3.989	3.933
1.1	4.590	3.963
1.2	4.449	4.045
1.3	4.603	3.994
1.4	4.274	4.048

5.2 Sampling Results

We train the proposed HBM for OPC regression equation (11). As mentioned in Section 4, HBM includes many parameters. In our experiments, the layout pattern is sampled 2.5 nm per 1 pixel resulting in 400×400 pixel binary map to fit the grid size of the optical model. Under this condition, since the total number of dimensions is 257, there are 1293 parameters in total, e.g. 258 fixed effects including bias parameter w_{f_0}, 1032 (258×4) random effects including bias parameters w_{r_0}, 2 hierarchical hyper-priors σ_f, σ_r, and 1 variance in normal distribution σ_y. All parameters are estimated by using the MCMC algorithm described in Section 4 and the MCMC parameters, where the number of iterations is 5000, the number of chains is 4, the burn-in number is half of the iterations, and the amount of thinning is 10. Since it is difficult to introduce all estimation results, we only show the results of hierarchical parameters peculiar to HBM, σ_f and σ_r, in Fig. 6. In this figure, some of the sampling results are shown on the left, and the estimated probability density functions of sampled parameters are indicated on the right.

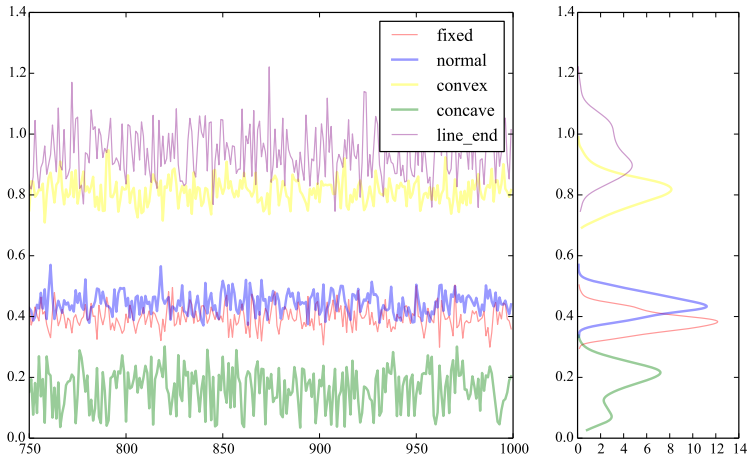


Figure 6. Sampling results of hidden variables. $\hat{R}$ values of σ_f, $\sigma_r(normal)$, $\sigma_r(convex)$, $\sigma_r(concave)$ and $\sigma_r(line\text{-}end)$ are 1.09, 1.01, 0.99, 0.98 and 1.00, respectively.

Although the sampling results of random effects are similar, they are far from being the same. The result shows that there is a clear difference among edge types and these hidden relationships can be estimated by HBM without any prior knowledge. Furthermore, the figure indicates that the parameters are well converged because all $\hat{R}$ are less than 1.1.

5.3 Comparison with Linear Regression and Nonlinear Regression

To evaluate the performance of HBM, we compare with the linear and the nonlinear regression method, respectively. For this comparison, a linear regression algorithm and a support vector regression (SVR) algorithm as a nonlinear regression are used. In both algorithms, the samples including all edges and the separate samples in each type of edge of the layout A are used as supervised data to compare the model performance because

the separate regression model showed a better predictive performance than a unifying model that is trained by using all samples.[6] Table 2 indicates the predictive RMSEs for the samples extracted from the layout B by each method.

Table 2. Comparison of different algorithms.

RMSE on Training Layout					
edge	LR(all)	LR(sep)	SVR(all)	SVR(sep)	HBM
all	3.575	3.154	0.518	0.506	3.179
normal	2.852	2.696	0.534	0.534	2.703
convex	5.109	4.460	0.505	0.429	4.466
concave	5.166	3.667	0.150	0.150	3.860
lineend	4.666	3.428	0.511	0.509	3.581

RMSE on Testing Layout					
edge	LR(all)	LR(sep)	SVR(all)	SVR(sep)	HBM
all	3.701	3.664	8.875	7.424	3.492
normal	2.888	2.778	6.765	6.361	2.747
convex	5.566	5.166	10.419	9.744	5.114
concave	5.172	4.670	6.506	6.497	4.380
lineend	4.697	6.262	20.088	11.252	4.980

From Table 2 we can see that the SVR model, nonlinear regression, has relatively better RMSE for training data set compared to other algorithms. However, the table also shows that it is difficult to achieve high prediction accuracy for unknown data with the nonlinear model, and this indicates a typical over-fitting issue. Also, Table 2 shows that our proposed HBM achieves the best prediction accuracy.

Table 3. Comparison with linear regression model.

# training samples	LR(sep)	HBM
All (33127)	3.664	3.492
Middle (5000)	6.467	3.713
Low (1000)	12.113	5.244

We further compare the difference of prediction accuracy between HBM and the linear regression model while changing the number of training data sets. Table 3 shows that prediction accuracy drastically deteriorates if the number of training data is low in the linear regression model. In contrast, HBM can prevent degradation of prediction accuracy even with small numbers of training data. This is because HBM learns hidden variables corresponding to variance components included in given data. From the above results, by using HBM, it is possible to achieve a robust OPC regression model that has good predictive performance even when the number of samples is relatively small.

5.4 Comparison with Model-based OPC

In order to confirm the effectiveness of HBM compared to conventional model-based OPC, we further measure correction accuracy based on an edge displacement error (EPE) distribution. Fig. 7 indicates the result of comparison of EPE distributions between model-based OPC and HBM. In the figure, the blue line, the red line and the dot show the EPEs at 2nd, 6th and 10th iteration in model-based OPC, respectively. The green line illustrates the EPE distribution in the predicted results by HBM without any model-based iteration. Also, Table 4 lists the comparison of variance, std. dev., mean and median between model-based OPC and our regression model. From the table we can see that the regression results by HBM can achieve almost the same variance as the result of 8th iteration in model-based OPC. This indicates that the number of iterations can be reduced to two by using the regression results as an initial input of model-based OPC. Furthermore, since the regression model achieves the best mean and median EPEs, further improvement of prediction accuracy can be expected by adding several hidden variables to HBM, tuning MCMC parameters or adjusting model-based OPC recipe.

Table 4. Comparison with model-based OPC.

	MB(i2)	MB(i4)	MB(i6)	MB(i8)	MB(i10)	HBM
variance	71.094	52.208	35.086	13.131	9.768	14.514
std. dev.	8.432	7.226	5.923	3.624	3.125	3.810
mean	-3.036	-2.214	-1.521	-0.507	0.216	-0.196
median	-0.292	-0.329	-0.250	-0.158	0.039	0.000

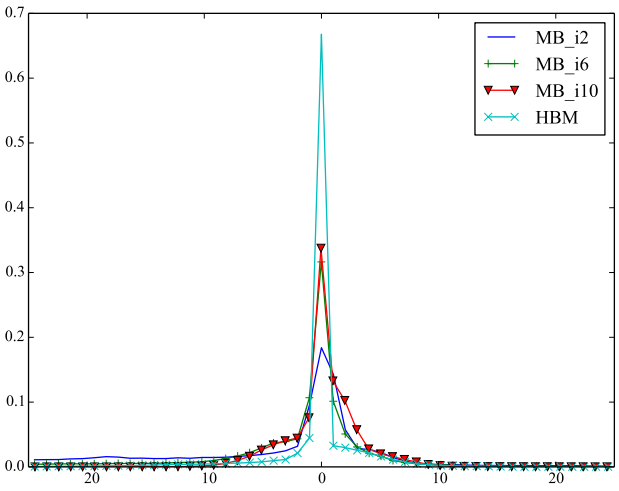


Figure 7. EPE distributions.

6. CONCLUSIONS

This paper proposes a new regression-based OPC technique based on a hierarchical Bayes model (HBM). By applying HBM, flexible modeling is realized without being restricted by linearity of model parameters or the number of supervised samples, which can improve predictive performance of a regression model with a small number of data. Markov Chain Monte Carlo is able to estimate all model parameters including unknown variables that correspond to variance components included in given data. The experimental results show that our method can reduce the number of model-based OPC iterations to two by using the regression results as an initial input of model-based OPC, and thus it promises to dramatically reduce the cost of process development.

REFERENCES

[1] Bakshi, V., "EUV lithography," vol. 178, Spie Press Bellingham (2009).

[2] Higashiki, T., Nakasugi, T., and Yoneda, I., "Nanoimprint lithography for semiconductor devices and future patterning innovation," in *Proc. of SPIE*, 797003–797003 (2011).

[3] Seino, Y., Yonemitsu, H., Sato, H., Kanno, M., Kato, H., Kobayashi, K., Kawanishi, A., Azuma, T., Muramatsu, M., Nagahara, S., et al., "Contact hole shrink process using directed self-assembly," in *Proc. of SPIE*, 83230Y–83230Y (2012).

[4] Miyama, S., Yamamoto, K., and Koyama, K., "Large-area optical proximity correction with a combination of rule-based and simulation-based methods," *Japanese journal of applied physics* **35**(12B), 6370–6373 (1996).

[5] Jia, N. and Lam, E. Y., "Machine learning for inverse lithography: using stochastic gradient descent for robust photomask synthesis," *Journal of Optics* **12**(4), 045601 (2010).

[6] Gu, A. and Zakhor, A., "Optical proximity correction with linear regression," *IEEE Transactions on Semiconductor Manufacturing* **21**(2), 263–271 (2008).

[7] Luo, R., "Optical proximity correction using a multilayer perceptron neural network," *Journal of Optics* **15**(7), 075708 (2013).

[8] Luo, K.-s., Shi, Z., Yan, X.-l., and Geng, Z., "SVM based layout retargeting for fast and regularized inverse lithography," *Journal of Zhejiang University SCIENCE C* **15**(5), 390–400 (2014).

[9] Gilks, W. R., "Markov chain monte carlo," Wiley Online Library (2005).

[10] Gelman, A. and Rubin, D. B., "Inference from iterative simulation using multiple sequences," *Statistical science* , 457–472 (1992).

[11] Brooks, S. P. and Gelman, A., "General methods for monitoring convergence of iterative simulations," *Journal of computational and graphical statistics* **7**(4), 434–455 (1998).

[12] "Calibre." `http://www.mentor.com/products`.