EBL Overlapping Aware Stencil Planning for MCC System

BEI YU, CSE Department, The Chinese University of Hong Kong, Hong Kong
KUN YUAN, Facebook Inc., Menlo Park, CA
JHIH-RONG GAO, Cadence Design Systems, Austin, TX
SHIYAN HU, ECE Department, Michigan Technological University, Houghton, MI
DAVID Z. PAN, ECE Department, The University of Texas at Austin, Austin, TX

Electron beam lithography (EBL) is a promising, maskless solution for the technology beyond 14nm logic nodes. To overcome its throughput limitation, industry has proposed character projection (CP) technique, where some complex shapes (characters) can be printed in one shot. Recently, the traditional EBL system was extended into a multi-column cell (MCC) system to further improve the throughput. In an MCC system, several independent CPs are used to further speed-up the writing process. Because of the area constraint of stencil, the MCC system needs to be packed/planned carefully to take advantage of the characters. In this article, we prove that the overlapping aware stencil planning (OSP) problem is NP-hard. Then we propose E-BLOW, a tool to solve the MCC system OSP problem. E-BLOW involves several novel speedup techniques, such as successive relaxation and dynamic programming. Experimental results show that, compared with previous works, E-BLOW demonstrates better performance for both the conventional EBL system and the MCC system.

CCS Concepts: • **Hardware → VLSI design manufacturing considerations;**

Additional Key Words and Phrases: Electron beam lithography, multi-column cell system, overlapping aware stencil planning

ACM Reference Format:
Bei Yu, Kun Yuan, Jhih-Rong Gao, Shiyan Hu, and David Z. Pan. 2016. EBL overlapping aware stencil planning for MCC system. ACM Trans. Des. Autom. Electron. Syst. 21, 3, Article 43 (May 2016), 24 pages.
DOI: http://dx.doi.org/10.1145/2888394

1. INTRODUCTION

As the minimum feature size continues to scale to sub-22nm, conventional 193nm optical photolithography technology is reaching its printability limit. In the near future, multiple patterning lithography (MPL) has become one of the viable lithography techniques for 22nm and 14nm logic nodes [Kahng et al. 2008; Zhang et al. 2011; Yu et al. 2011; Yu and Pan 2014]. In the longer future, that is, for the logic nodes beyond 14nm, extreme ultra violet (EUV), directed self-assembly (DSA), and electron beam lithography (EBL) are promising candidates as next-generation lithography technologies [Pan et al. 2013]. Currently, both EUV and DSA suffer from some technical barriers: EUV technique is delayed due to tremendous technical issues, such as lack of power sources,

This work is supported in part by NSF grants CCF-0644316 and CCF-1218906, SRC task 2414.001, NSFC grant 61128010, IBM Scholarship, and Chinese University of Hong Kong (CUHK) Direct Grant for Research.
Authors' addresses: B. Yu, CSE Department, The Chinese University of Hong Kong, NT, Hong Kong; email: byu@cse.cuhk.edu.hk; K. Yuan, Facebook Inc., Menlo Park, CA 94025 USA; email: bigfish.yuan@gmail.com; J.-R. Gao, Cadence Design System, TX 78759 USA; email: jrgao@cadence.com; S. Hu, ECE Department, Michigan Technological University, MI 49931 USA; email: shiyan@mtu.edu; D. Z. Pan, ECE Department, The University of Texas at Austin, TX 78712 USA; email: dpan@ece.utexas.edu.

DOI: http://dx.doi.org/10.1145/2888394

resists, and defect-free masks [Arisawa et al. 2010; Zhang et al. 2012]; DSA has mainly the potential to generate contact or via layers [Chang et al. 2010].

The EBL system, on the other hand, has been developed for several decades [Pfeiffer 2009]. Compared with the traditional lithographic methodologies, EBL has several advantages. (1) An electron beam can be easily focused into a nanometer diameter with a charged particle beam, thus can avoid suffering from the diffraction limitation of light. (2) The price of a photomask set is getting unaffordable, especially through the emerging multiple patterning lithography (MPL) techniques. As a maskless technology, EBL can reduce the manufacturing cost. (3) EBL allows a great flexibility for fast turnaround times or late design modifications to correct or adapt a given chip layout. Because of all these advantages, EBL is being used in mask making, small volume integration circuit production, and research to develop the technological nodes ahead of mass production.

1.1. EBL System and Extended MCC System

The conventional EBL system applies a variable-shaped beam (VSB) technique [Fujimura 2010]. In this mode, the entire layout is decomposed into a set of rectangles, each of which is shot into resist by one electron beam. In the printing process of the VSB mode, at first, an electrical gun generates the initial beam, which becomes uniform through a shaping aperture. Then, a second aperture finalizes the target shape with a limited maximum size. Since each pattern needs to be fractured into pieces of rectangles and printed one by one, the VSB mode suffers from serious throughput problems.

One improved technique in the EBL system is called character projection (CP) [Fujimura 2010], where the second aperture is replaced by a *stencil*. Some complex shapes, called *characters*, are prepared on the stencil. The key idea is that if a pattern is pre-designed on the stencil, it can be printed in one electronic shot; otherwise, it needs to be fractured into a set of rectangles and printed one by one through VSB mode. By this way, the CP mode can improve the throughput significantly. In addition, CP exposure has a good, critical dimension control stability compared with VSB [Maruyama et al. 2008]. However, the area constraint of the stencil is the bottleneck. For modern design, due to the numerous distinct circuit patterns, only a limited number of patterns can be employed on the stencil. Those patterns not contained by the stencil are still required to be written by VSB [Manakli et al. 2009]. Thus, one emerging challenge in CP mode is how to pack the characters into the stencil to effectively improve the throughput.

Even with decades of development and the employment of the CP technique, the key limitation of the EBL system has been, and still is, the low throughput. Recently, a multi-column cell (MCC) system was proposed as an extension of the CP technique [Yasuda et al. 2004; Maruyama et al. 2012]. In the MCC system, several independent CPs are used to further speed-up the writing process. Each CP is applied on one section of wafer, and all the CPs can work in parallel to achieve better throughput. In the modern MCC system, there are more than 1,300 CPs [Shoji et al. 2010]. Since one CP is associated with one stencil, there are more than 1,300 stencils in total. The manufacturing of stencils is similar to mask manufacturing. If each stencil is different, then the stencil preparation process could be very time consuming and expensive. Due to the design complexity and cost consideration, different CPs share one stencil design. One example of the MCC printing process is illustrated in Figure 1, where four CPs are bundled to generate an MCC system. In this example, the whole wafer is divided into four regions, w_1, w_2, w_3 and w_4, and each region is printed through one CP. The whole writing time of the MCC system is determined by the maximum one among the four regions. For modern design, because of the numerous distinct circuit patterns, only a limited number of patterns can be employed on a stencil. Since the area constraint of the stencil is the bottleneck, the stencil should be carefully designed/manufactured to contain the most repeated characters.

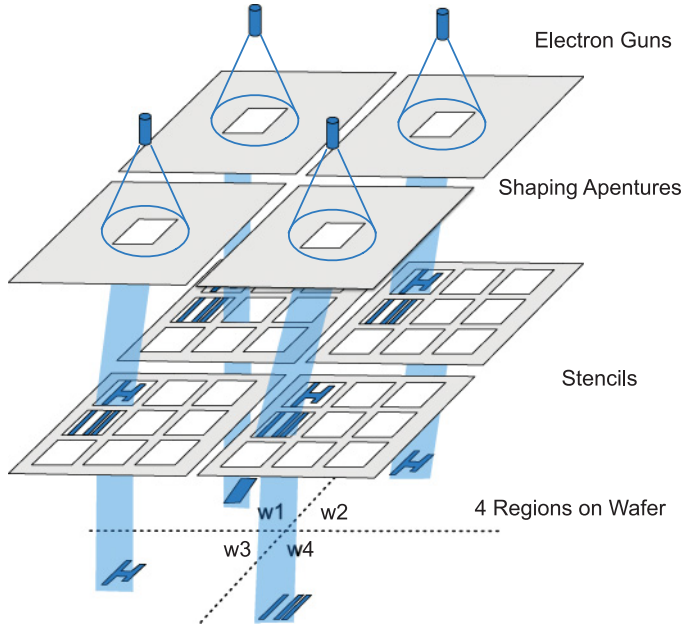

Fig. 1. Printing process of MCC system, where four CPs are bundled.

1.2. Previous Works

Many previous works dealt with the design optimization for the EBL system. For VSB mode, Du et al. [2012a], Gao et al. [2014], Ding et al. [2014], and Yang et al. [2015] considered EBL as a complementary lithography technique to print via/cut patterns or additional patterns after multiple patterning mask processes. To avoid stitching line-induced bad patterns for the parallel EBL system, Fang et al. [2013b] and Lin et al. [2016] integrated EBL constraints into early routing stage and placement stage, respectively. Babin et al. [2003] and Fang et al. [2013a] solved the subfield scheduling problem to reduce the critical dimension distortion. Kahng et al. [2006], Ma et al. [2011], Yu et al. [2013a], and Chan et al. [2014] proposed a set of layout/mask fracturing approaches to reduce the VSB shot number. Besides, several works solved the design challenges under CP technique. Before stencil manufacturing, all design steps should consider the character projection, that is, character aware library building [Fujino et al. 2005], technology mapping [Sugihara et al. 2007], and character sizing problem [Sugihara et al. 2006b]. In addition, Du et al. [2012b] and Ikeno et al. [2013a] proposed several character design methods for both via layers and interconnect layers to achieve stencil area-efficiency. After stencil manufacturing, characters are stamped to practical layout patterns to minimize the electron beam shot number, especially for irregular routing or via layout [Minh et al. 2006; Du et al. 2012b]. Ikeno et al. [2013b] proposed a structured routing architecture for high throughput CP mode.

Stencil planning is one of the most challenging problems in CP mode and has earned more and more attention. When blank overlapping is not considered, this problem equals to a character selection problem, which can be solved through an integer linear programming (ILP) formulation to maximize the system throughput [Sugihara et al. 2006a]. When the characters can be overlapped to save more stencil space, the corresponding stencil planning is referred to as *overlapping-aware stencil planning* (OSP).

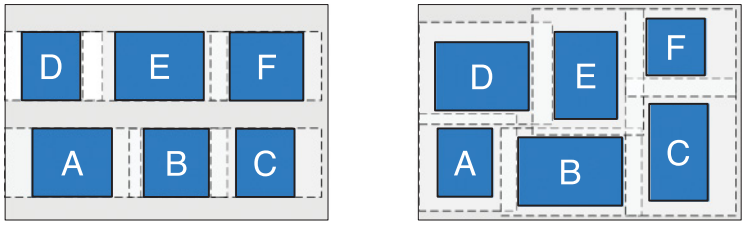

Fig. 2. Two types of OSP problems: (a) one-dimensional problem; (b) two-dimensional problem.

As suggested in Yuan et al. [2012], the OSP problem can be divided into two types: one-dimensional problem and two-dimensional problem.

In the one-dimensional OSP problem, the standard cells with the same height are selected into the stencil. As shown in Figure 2(a), each character implements one standard cell, and the enclosed circuit patterns of all the characters have the same height. Note that here we only show the horizontal blanks, and the vertical blanks are not represented because they are identical. Yuan et al. [2012] proposed a set of heuristics, and the single row reordering was formulated as a minimum cost Hamiltonian path. Kuang and Young [2014a] proposed an integrated framework to solve all the sub-problems effectively: character selection, row distribution, single-row ordering, and inter-row swapping. Guo et al. [2015] proved that single row reordering can be optimally solved in polynomial time. In addition, Chu and Mak [2014] and Mak and Chu [2014] assumed that the pattern position in each character can be shifted and integrated the character re-design into the OSP problem.

In the two-dimensional OSP problem, the blank spaces of characters are non-uniform along both horizontal and vertical directions. By this way, the stencil can contain both complex via patterns and regular wires (see Figure 2(b) for an example). Yuan et al. [2012] solved the problem through a modified floor-planning engine, while Yu et al. [2013b] further sped-up the engine through a clustering technique. Kuang and Young [2014b] proposed a set of fast and effective deterministic methods to solve this problem.

1.3. Our Contributions

In this article, we focus on the one-dimensional OSP problem. We present a tool, **E-BLOW**, to effectively solve this problem. Since only the one-dimensional problem is studied, for convenience, we use the term OSP to refer to the one-dimensional OSP in the rest of this article.

Compared with the conventional EBL system, the MCC system introduces two main challenges in the OSP problem. (1) The objective is new: in the MCC system, the wafer is divided into several regions and each region is written by one CP. Therefore, the new OSP should minimize the maximal writing time of all regions. However, in the conventional EBL system the objective is simply to minimize the wafer writing time. (2) The stencil for an MCC system can contain more than 4,000 characters, and previous methodologies for the EBL system may suffer from runtime penalty. No existing stencil planning work has been done toward the MCC system. In this work we propose E-BLOW, a tool to overcome both challenges. Our main contributions are summarized as follows.

—We provide the proof that even a simplified version of the OSP problem is NP-hard.
—We develop an ILP formulation to co-optimize character selection and placement on the stencil. To the best of our knowledge, this is the first mathematical formulation for this problem.

Table I. Notations Used in This Article

Term	Meaning	Term	Meaning
W	width of each stencil row	K	number of regions
T_k	system writing time on region k	T	total writing time of all regions
m	number of rows	n	number of characters
w	width of each character	C	set of characters $C = \{c_1, \ldots, c_n\}$
a_i	character c_i's writing time in VSB mode	x_i	x-position of character c_i
sl_i	left blank of character c_i	sr_i	right blank of character c_i
s_i	$\lceil (sl_i + sr_i)/2 \rceil$	b_{ij}	0-1 variable, $b_{ij} = 1$, if c_i is on row j
t_{ik}	repeating times of character c_i in region k	o_{ij}	horizontal overlap between c_i and c_j
		p_{ij}	0-1 variable, $p_{ij} = 0$, if c_i is left of c_j

—We proposes a simplified formulation to achieve a good tradeoff in terms of performance and runtime.
—We present a successive relaxation algorithm to search for a near-optimal solution.

The rest of this article is organized as follows. Section 2 provides the problem formulation. Section 3 proves the problem is NP-hard. Section 4 presents algorithmic details of the proposed framework, E-BLOW. Section 5 reports experimental results, followed by the conclusion in Section 6.

2. PROBLEM FORMULATION

In this section, we give the problem formulation. For convenience, Table I lists the notations used in this article. Note that in this article, we denote $[n]$ as a set of integers $\{1, 2, \ldots, n\}$.

In an MCC system with K CPs, the whole wafer is divided into K regions, and each region is written by one particular CP. We assume cell extraction [Manakli et al. 2009] has been resolved first. In other words, a set of n character $C = \{c_1, \ldots, c_n\}$ has already been given to the MCC system. For each character $c_i \in C$, its width is w_i. Meanwhile, the writing time through VSB mode and CP mode are a_i and 1, respectively. The stencil is divided into m rows, and the width of each row is W. For each row j, b_{ij} indicates whether character c_i is selected on row j:

$$b_{ij} = \begin{cases} 1, & \text{candidate } c_i \text{ is selected on row } j \\ 0, & \text{otherwise.} \end{cases}$$

Different regions have different layout patterns, thus, the throughputs would be also different. For region k ($k \in [K]$), character $c_i \in C$ repeats t_{ik} times. If c_i is prepared on stencil, the total writing time of character c_i on region r_c is $t_{ik} \cdot 1$. Otherwise, c_i should be printed through VSB, and its writing time would be $t_{ik} \cdot a_i$. Therefore, for region k ($k \in [K]$), the writing time T_k is as follows:

$$T_k = \sum_{i \in [n]} t_{ik} \cdot a_i - \sum_{i \in [n]} \sum_{j \in [m]} b_{ij} \cdot t_{ik} \cdot (a_i - 1), \tag{1}$$

where the first term is the writing time using VSB mode, while the second term is the writing time improvement through CP mode. Therefore, the total writing time of the MCC system is formulated as follows:

$$T = \max_{k \in [K]} \{T_k\}. \tag{2}$$

Based on the above notations, we define the MCC system OSP problem (MOSP) as follows.

Problem 1 (*MCC System OSP*). In an MCC system, a stencil is given with m row, and each row is with width W. A set of character C is also given. We select a subset of characters in C and place them on the stencil. The objective is to minimize the MCC system writing time T expressed by Equation (2), while the placement of all characters is bounded by the outline of the stencil.

Note that when region number K is 1, the MOSP problem equals to the conventional OSP problem. Our proposed framework can effectively solve both MOSP and OSP problems.

3. PROOF OF NP-HARDNESS

In this section, we prove that both OSP and MOSP problems are NP-hard. Guo et al. [2015] provided a polynomial time algorithm to optimally solve the single row ordering problem, which is a sub-problem of OSP. Mak and Chu [2014] proved that when an additional constraint, character re-design, is integrated, the extended OSP problem is NP-hard. Although a sub-problem of OSP is in P [Guo et al. 2015], while an extension of OSP is NP-hard [Mak and Chu 2014], the complexity of OSP is still an open question. This article is the first work proving the complexity of the OSP problem. In addition, we will show that even a simpler version of the OSP problem, where there is only one row, is NP-hard, as well.

To facilitate the proof, we first define a Bounded Subset Sum (BSS) problem as follows.

Problem 2 (*Bounded Subset Sum*). Given a list of n numbers $\{x_1, \ldots, x_n\}$ and a number s, where $\forall i \in [n]\ 2 \cdot x_i > x_{\max}(\triangleq \max_{i\in[n]}|x_i|)$, decide if there is a subset of the numbers that sums up to s.

For example, given three numbers $1{,}100, 1{,}200, 1{,}413$, and $T = 2{,}300$, we can find a subset $\{1{,}100, 1{,}200\}$ such that $1{,}100 + 1{,}200 = 2{,}300$. Note that if we have the assumption $s \leq c \cdot x_{max}$, where c is some constant, the problem can be solved in $O(n^c)$ time. However, in general, there is no such assumption. Note that without the bounded constraint $\forall i \in [n] 2 \cdot x_i > x_{\max}$, the BSS problem becomes the **subset sum** problem, which is in NP-complete [Arora and Barak 2009]. In the following, we will prove that with this additional constraint, the subset sum problem is still NP-complete. For the simplicity of later explanation, let S denote the set of n numbers. Note that, we can assume that all the numbers are integer numbers.

THEOREM 1. BSS *problem is NP-complete.*

The proof is in the Appendix.

In the following, we will show that even a simpler version of the OSP problem is NP-hard. In the simpler version, there is only one row in the stencil, and each character $c_i \in C$ is with the same length w. Besides, for each character, its left blank and right blank are symmetrical.

Definition 1 (*Minimum Packing*). Given a subset of characters $C' \in C$, its minimum packing is the packing with the minimum stencil length.

LEMMA 1. *Given a set of characters $C = \{c_1, c_2, \ldots, c_n\}$ placed on a single row stencil. If both left and right blanks are s_i for each character $c_i \in C$, then the minimum packing is with the following stencil length*

$$n \cdot w - \sum_{i\in[n]} s_i + \max_{i\in[n]}\{s_i\}. \tag{3}$$

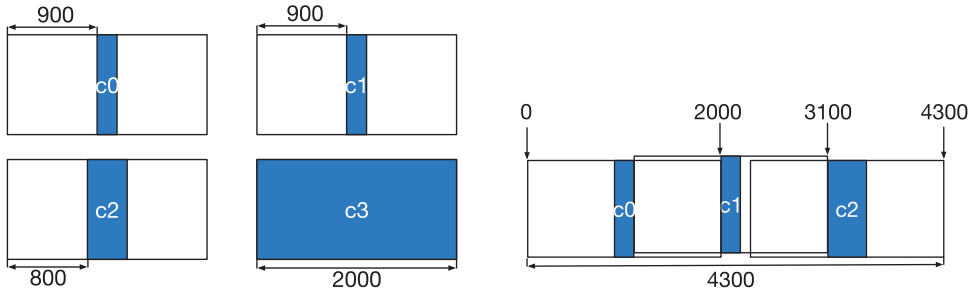

Fig. 3. (a) OSP instance for the BSS instance $S = \{1{,}100, 1{,}200, 2{,}000\}$ and $s = 2{,}300$. (b) The minimum packing is with stencil length $M + s = 2{,}000 + 2{,}300 = 4{,}300$.

PROOF. Without loss of generality, we assume that $s_1 \geq s_2 \geq \cdots \geq s_n$. We prove by recursion that in a minimum length packing, the overlapping blank is $f(n) = \sum_{i=2}^{n} s_i$. If there are only two characters, it is trivial that $f(2) = s_2$. We assume that when $p = n - 1$, the maximum overlapping blank $f(n-1) = \sum_{i=2}^{n-1} s_i$. For the last character c_n, the maximum sharing blank value is s_n. Since for any $i < n$, $s_i \geq s_n$, we can simply insert it at either the left end or the right end and find the incremental overlapping blank s_n. Thus, $f(n) = f(n-1) + s_n = \sum_{i=2}^{n} s_i$. Because the maximum overlapping blank for all characters is $\sum_{i=2}^{n} s_i$, we can see the minimum packing length is the same as in Equation (3). □

LEMMA 2. BSS $\leq_p$ OSP.

PROOF. Given an instance of BSS with s and $S = \{x_1, x_2, \ldots, x_n\}$, we construct an OSP instance as follows:

—The stencil length is set to $M + s$, where $M = \max_{i \in [n]}\{x_i\}$.
—For each $x_i \in S'$, in the OSP there is a character c_i whose width is M and both left and right blanks are $M - x_i$. As defined in Problem 2, $2 \cdot x_i > x_{max}$, which means $x_i > M/2$. Therefore, the sum of left blank and right blank is less than or equal to M.
—We introduce an additional character c_0, whose width size is M, and both left and right blanks are $M - \min_{i \in [n]}\{x_i\}$.
—The VSB writing time of character c_0 is set to $\sum_{i \in [n]} x_i$, while the VSB writing time for each character c_i is set to x_i. The CP writing times are set to 0 for all characters.
—There is only one region, and each character c_i repeats one time in the region.

For instance, given initial set $S = \{1{,}100, 1{,}200, 2{,}000\}$ and $s = 2{,}300$, the constructed OSP instance is shown in Figure 3.

We will show the BSS instance $S = \{x_1, x_2, \ldots, x_n\}$ has a subset that adds up to s if and only if the constructed OSP instance has minimum packing length $M + s$ and total writing time smaller than $\sum x_i$.

($\Rightarrow$ part) After solving the BSS problem, a set of items S' are selected that they add up to s. We denote "m" as the number of items in set S'. For each $x_i \in S'$, the corresponding character c_i is also selected into the stencil. Besides, since the system writing time for c_0 is $\sum x_i$, it is trivial to see that in the OSP instance the c_0 must be

selected. Due to Lemma 1, the minimum total packing length is

$$(m+1)\cdot M - \sum_{i\in S'}(M-x_i) = M + \sum_{i\in S'} x_i = M + s.$$

Meanwhile, the minimum total writing time in the OSP is $\sum_{i\in[n]} x_i - s$.

($\Leftarrow$ part) We start from an OSP instance with minimum packing length $M+s$ and total writing time smaller than $\sum x_i$, where a set of characters $C' \in C$ are selected. Since the total writing time must be smaller than $\sum x_i$, character $c_0 \in C'$. For all characters in set $c_i \in C'$ except c_0, we select x_i into the subset $S' \in S$, which adds up to s. □

THEOREM 2. OSP *is NP-hard.*

PROOF. Directly from Lemma 2 and Theorem 1. □

The OSP problem is a special case of the MOSP problem when region number K is set to 1. Therefore, from Theorem 2 we can see that the MOSP problem is NP-hard as well.

4. ALGORITHMS FOR MOSP

In the MOSP problem, each character implements one standard cell, and the enclosed circuit patterns of all the characters have the same height. This problem can be viewed as a combination of character selection and single row ordering problems. Different from a two-step heuristic proposed in Yuan et al. [2012], we show that these two problems can be solved simultaneously through a unified ILP formulation (4).

$$\min \quad T \tag{4}$$

$$\text{s.t} \quad T \geq \sum_{i\in[n]} t_{ik}\cdot a_i - \sum_{i\in[n]}\sum_{j\in[m]} b_{ij}\cdot t_{ik}(a_i-1) \qquad \forall k\in[K] \tag{4a}$$

$$0 \leq x_i \leq W - w \qquad \forall i\in[n] \tag{4b}$$

$$\sum_{j\in[m]} b_{ij} \leq 1 \qquad \forall i\in[n] \tag{4c}$$

$$x_i + w_{ii'} - x_{i'} \leq W(2 + p_{ii'} - b_{ij} - b_{i'j}) \qquad \forall i,i'\in[n], j\in[m] \tag{4d}$$

$$x_{i'} + w_{i'i} - x_i \leq W(3 - p_{ii'} - b_{ij} - b_{i'j}) \qquad \forall i,i'\in[n], j\in[m] \tag{4e}$$

$$b_{ij}, b_{i'j}, p_{ii'} \in \{0,1\} \qquad \forall i,i'\in[n], j\in[m] \tag{4f}$$

In formulation (4), W is the stencil width, and m is the number of rows. For each character c_i, w_i, and x_i are its width and its x-position, respectively. If and only if c_i is assigned to row j, $b_{ij}=1$. Otherwise, $b_{ij}=0$.

Constraint (4a) is derived from Equations (1) and (2). Constraint (4b) is for the x position of each character. Constraint (4c) is to make sure one character can be inserted into at most one row. Constraints (4d) and (4e) are used to check position relationship between c_i and c_i'. Here $w_{ii'} = w_i - o_{ii'}$ and $w_{i'i} = w_{i'} - o^h_{i'i}$, where $o_{ii'}$ is overlapping when candidates c_i and c_i' are packed together. Only when $b_{ij} = b_{i'j} = 1$, that is, both characters c_i and $c_{i'}$ are assigned to row j, can one of the two constraints, (4d) and (4e), could be active. Besides, all the $p_{ii'}$ values are self-consistent. For example, for any three characters c_1, c_2, c_3 being assigned to row j, that is, $b_{1j} = b_{2j} = b_{3j} = 1$, if c_1 is on the left of c_2 ($p_{12}=0$) and c_2 is on the left of c_3 ($p_{23}=0$), then c_1 should be on the left of c_3 ($p_{13}=0$). Similarly, if c_1 is on the right of c_2 ($p_{12}=1$) and c_2 is on the right of c_3 ($p_{23}=1$), then c_1 should be on the right of c_3 ($p_{13}=1$) as well.

Since ILP is a well known NP-hard problem, directly solving it may suffer from long runtime penalty. One straightforward speedup method is to relax the ILP into

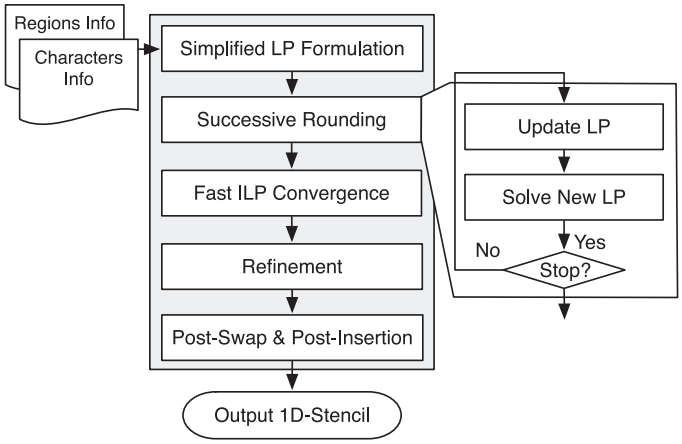

Fig. 4. E-BLOW overall flow for MOSP problem.

the corresponding linear programming (LP) through replacing constraints (4f) by the following:

$$0 \leq b_{ij}, b_{i'j}, p_{ii'} \leq 1.$$

It is obvious that LP relaxation provides a lower bound to the ILP solution. However, we observe that the solution of the relaxed LP could be like this: for each i, $\sum_{j\in[m]} b_{ij} = 1$ and all the $p_{ii'}$ are assigned 0.5. Although the objective function is minimized and all the constraints are satisfied, this LP relaxation provides no useful information to guide future rounding, that is, all the characters are selected ($b_{ij} = 1$) and no ordering relationship is determined ($p_{ii'} = 0.5$).

To overcome the limitation of the above rounding, we propose a novel successive rounding framework to search for a near-optimal solution in reasonable runtime. As shown in Figure 4, the overall flow includes several steps. In Section 4.1, the simplified formulation will be discussed, and its LP rounding lower bound will be proved. In Section 4.2, the details of successive rounding will be introduced. In Section 4.3, the Fast ILP convergence technique will be presented. In Section 4.4, the refinement process will be proposed. At last, to further improve the performance, in Section 4.5, the post-swap and post-insertion techniques will be discussed.

4.1. Simplified ILP Formulation

As discussed above, solving the ILP formulation (4) is very time consuming, and the related LP relaxation may be bad in performance. To overcome the limitations of formulation (4), we introduce a simplified ILP formulation, whose LP relaxation can provide good lower bound. The simplified formulation is based on a *symmetrical blank* (S-Blank) assumption: the blanks of each character are symmetric, that is, left blank equals to right blank. s_i is used to denote the blank of character c_i. Note that for different characters c_i and $c_{i'}$, their blanks s_i and $s_{i'}$ can be different.

At first glance the S-Blank assumption may lose optimality. However, under S-Blank assumption, the ILP formulation can be effectively simplified to provide a reasonable rounding bound, theoretically. Compared with the previous heuristic framework [Yuan et al. 2012], the proved rounding bound provides a better guideline for a global view search. In addition, to compensate the inaccuracy in the asymmetrical blank cases, E-BLOW provides a refinement (see Section 4.4).

The simplified ILP formulation is shown in Formula (5).

$$\max \sum_{i\in[n]} \sum_{j\in[m]} b_{ij} \cdot p_i \tag{5}$$

$$\text{s.t.} \sum_{i\in[n]} (w - s_i) \cdot b_{ij} \le W - B_j \qquad \forall j \in [m] \tag{5a}$$

$$B_j \ge s_i \cdot b_{ij} \qquad \forall i \in [n],\ j \in [m] \tag{5b}$$

$$\sum_{j\in[m]} b_{ij} \le 1 \qquad \forall i \in [n] \tag{5c}$$

$$b_{ij} = 0 \text{ or } 1 \qquad \forall i \in [n],\ j \in [m] \tag{5d}$$

In the objective function of Formula (5), each character c_i is associated with one **profit value** p_i. The p_i value is to evaluate the overall system writing time improvement if character c_i is selected. Through assigning each character c_i with one particular profit value, we can simplify the complex constraint (4*a*). More details regarding the profit value setting would be discussed in Section 4.2. Besides, due to Lemma 1, constraint (5*a*) and constraint (5*b*) are for row width calculation, where (5*b*) is to linearize *max* operation. Here B_j can be viewed as the maximum blank space of all the characters on row j. Constraint (5*c*) implies each character can be assigned into at most one row. It's easy to see that the number of variables is $O(nm)$, where n is the number of characters and m is the number of rows. Generally speaking, single character number n is much larger than row number m. Thus, compared with basic ILP formulation (4), the variable number in (5) can be reduced dramatically.

In our implementation, we set blank value s_i to $\lceil (sl_i + sr_i)/2 \rceil$, where sl_i and sr_i are c_i's left blank and right blank, respectively. Note that here the ceiling function is used to make sure that under the S-Blank assumption, each blank is still integral.[1] Although this setting may lose some optimality, E-BLOW provides post-stage to compensate the inaccuracy through incremental character insertion.

Now we will show that the LP relaxation of (5) has reasonable lower bound. To explain this, let us first look at a similar formulation (6) as follows:

$$\max \sum_{i\in[n]} \sum_{j\in[m]} b_{ij} \cdot p_i \tag{6}$$

$$\text{s.t.} \sum_{i\in[n]} (w - s_i) \cdot b_{ij} \le W - s_{max} \qquad \forall j \in [m] \tag{6a}$$

$$(5c) - (5d),$$

where s_{max} is the maximum horizontal blank length of all characters; that is,

$$s_{max} = \max_{i\in[n]} \{s_i\}.$$

Program (6) is a multiple knapsack problem [Martello and Toth 1990]. A multiple knapsack is similar to a knapsack problem, with the difference that there are multiple knapsacks. In formulation (6), each p_i can be rephrased as $(w_i - s_i) \times ratio_i$.

LEMMA 3. *If each $ratio_i$ is the same, the multiple knapsack problem (6) can find a* 0.5*−approximation algorithm using the LP rounding method.*

[1]We also investigated $s_i = \lfloor (sl_i + sr_i)/2 \rfloor$, and found that, usually, the current setting (ceiling function) can achieve better performance.

For brevity, we omit the proof, but detailed explanations can be found in Dawande et al. [2000]. When all $ratio_i$ are the same, formulation (6) can be approximated to a max-flow problem. In addition, if we denote α as $\min\{ratio_i\}/\max\{ratio_i\}$, we can achieve the following Lemma:

LEMMA 4. *The LP rounding solution of (6) can be a* $0.5\alpha-$ *approximation to optimal solution of (6).*

PROOF. First we introduce a modified formulation to program (6), where each p_i is set to $\min\{p_i\}$. In other words, in the modified formulation, each $ratio_i$ is the same. Let OPT and OPT' be the optimal value of (6) and the optimal value of the modified formulation, respectively. Let APR' be the corresponding LP rounding result in the modified formulation. According to Lemma 3, $APR' \geq 0.5 \cdot OPT'$. Since $\min\{p_i\} \geq p_i \cdot \alpha$, we can achieve the following relationships:

$$\begin{aligned} OPT' &= \sum_{i\in[n]} \sum_{j\in[m]} (b_{ij} \cdot \min\{p_i\}) \\ &\geq \sum_{i\in[n]} \sum_{j\in[m]} (b_{ij} \cdot p_i \cdot \alpha) \\ &= \alpha \cdot \sum_{i\in[n]} \sum_{j\in[m]} (b_{ij} \cdot p_i) \\ &= \alpha \cdot OPT \end{aligned}$$

That is, $OPT' \geq OPT$. In summary, $APR' \geq 0.5 \cdot OPT' \geq 0.5\alpha \cdot OPT$. □

The difference between (5) and (6) is the right side values at (5α) and (6α). Blank spacing is relatively small compared with the row length. That is, $W \gg s_{max}$ and $\forall j \in [m], W \gg B_j$, which means $W - s_{max} \approx W - B_j$. Thus, we can expect that program (5) has a similar rounding performance as program (6α). In practical test cases, the measured α values range from 0.1 to 0.2. Although the α value may not be theoretically promising, the effectiveness of the LP relaxation will be verified in the experimental results.

4.2. Successive Rounding

In this section, we propose a successive rounding algorithm to solve program (5), iteratively. Successive rounding uses a simple iterative scheme, in which fractional variables are rounded one after the other until an integral solution is found [Johnson et al. 2000]. The ILP formulation (5) becomes an LP if we relax the discrete constraint to a continuous constraint as: $0 \leq b_{ij} \leq 1$.

The details of successive rounding are shown in Algorithm 1. At first, we set all b_{ij} as *unsolved*, since none of them are assigned to rows (line 1). The LP is updated and solved iteratively (lines 3–4). For each new LP solution, we search for the maximal b_{ij} and store it in b_{pq} (line 6). Then, we find all b_{ij} that are closest to the maximum value b_{pq}, that is, $b_{ij} \geq b_{pq} \times \beta$. In our implementation, β is set to 0.9. For each of the selected variables b_{ij}, we try to pack c_i into row j, and set b_{ij} as *solved* (line 9). Note that when one character c_i is assigned to one row, all b_{ij} would be set as solved. Therefore, the variable number in updated LP formulation would continue to decrease. This procedure repeats until no appropriate b_{ij} can be found. One key step of Algorithm 1 is the p_i update (line 3). For each character c_i, we set its p_i as follows:

$$p_i = \sum_{k\in[K]} \frac{t_k}{t_{max}} \cdot (a_i - 1) \cdot t_{ik}, \tag{7}$$

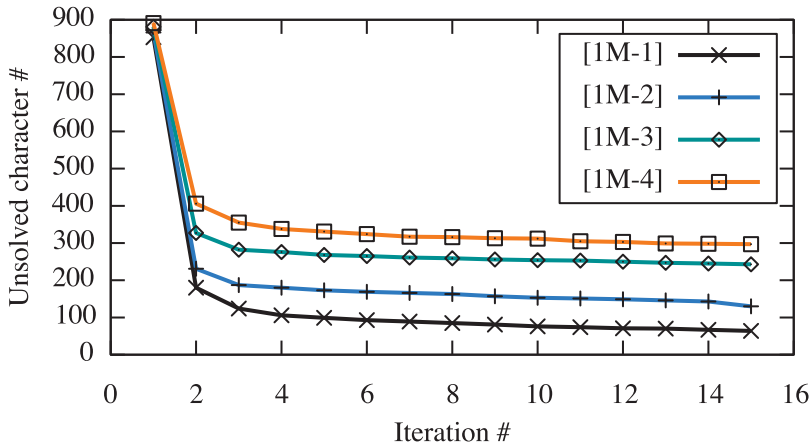

Fig. 5. Unsolved character numbers along the LP iterations for test cases 1M-1, 1M-2, 1M-3, and 1M-4.

ALGORITHM 1: SuccRounding ( )

Input: ILP Formulation (5)

Set all b_{ij} as unsolved;
repeat
  Update p_i for all unsolved b_{ij};
  Solve relaxed LP of (5);
  repeat
    $b_{pq} \leftarrow \max\{b_{ij}\}$;
    for all $b_{ij} \geq b_{pq} \times \beta$ **do**
      if c_i can be assigned to row r_j **then**
        $b_{ij} \leftarrow 1$ and set it as solved;
        Update capacity of row j;
      end if
    end for
  until cannot find b_{pq}
until no unsolved b_{ij}

where t_k is the current writing time of region k, and $t_{max} = \max_{k \in [K]}\{t_k\}$. Through applying the p_i, the region k with longer writing time would be considered more during the LP formulation. During successive rounding, if character c_i is not assigned to any row, p_i would continue to be updated so that the total writing time of the whole MCC system can be minimized.

4.3. Fast ILP Convergence

In each LP iteration of successive rounding, we select some characters into rows and set these characters as solved. In the next LP iteration, only unsolved characters are considered in formulation. Thus, the number of unsolved characters continues to decrease through the iterations. For four test cases (1M-1 to 1M-4), Figure 5 illustrates the number of unsolved characters in each iteration. We observe that in early iterations, more characters are assigned to rows in one iteration. However, when the stencil is almost full, fewer of b_{ij} can be close to 1. In other words, in late iterations only few characters are assigned into stencil; thus, the successive rounding requires more iterations.

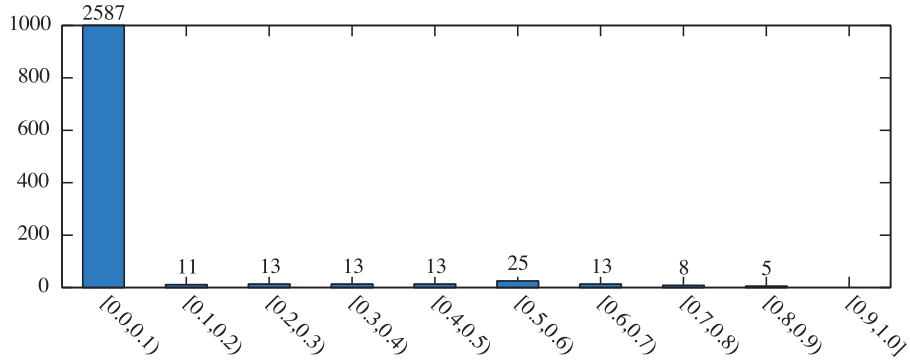

Fig. 6. For test case 1M-1, solution distribution in last LP, where most values are close to 0.

ALGORITHM 2: Fast ILP Convergence (δ^-, δ^+)

```
1:  for all b_ij in relaxed LP solutions do
2:      if b_ij < δ^- then
3:          Set b_ij as solved;
4:      end if
5:      if b_ij > δ^+ then
6:          Assign c_i to row j;
7:          Set b_ij as solved;
8:      end if
9:  end for
10: Solve ILP formulation (5) for all unsolved b_ij;
11: if b_ij = 1 then
12:     Assign c_i to row j;
13: end if
```

To overcome this limitation so that the successive rounding iteration number can be reduced, we present a convergence technique based on fast ILP formulation. The basic idea is that when we observe that only a few characters are assigned into rows in one LP iteration, we stop successive rounding in advance and call fast ILP convergence to assign all left characters. Note that in Kuang and Young [2014a], an ILP formulation with a similar idea was also applied. In our implementation, we trigger ILP convergence if in one iteration, less than 10% of variables are transferred from unsolved to solved. This condition is to detect when successive rounding becomes not that effective.

The details of the ILP convergence are shown in Algorithm 2, where δ^- and δ^+ are two user defined parameters. First, we check each b_{ij} (lines 1–9). If $b_{ij} < \delta^-$, then we assume character c_i would not be assigned to row j, and set b_{ij} as solved. Similarly, if $b_{ij} > \delta^+$, we assign c_i to row j and set b_{ij} as solved. For those unsolved b_{ij}, we build up ILP formulation (5) to assign final rows (lines 10–13).

At first glance, the ILP formulation may be expensive to solve. However, we observe that in our convergence (Algorithm 2), typically the variable number is small. Figure 6 illustrates the solution distribution in the last LP formulation. We can see that most of the values are close to 0. In our implementation δ^- and δ^+ are set to 0.1 and 0.9, respectively. For this case, although the LP formulation contains more than 2,500 variables, our fast ILP formulation results in only 101 binary variables.

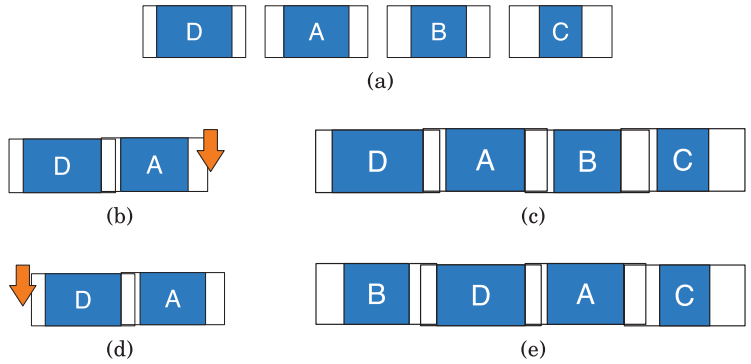

Fig. 7. Greedy-based single row ordering. (a) At first, all candidates are sorted by blank space. (c) One possible ordering solution where each candidate chooses the right end position. (e) Another possible ordering solution.

4.4. Refinement

Once some characters are assigned to the rows, the single row ordering problem [Yuan et al. 2012; Guo et al. 2015] adjusts the relative locations of input p characters to minimize the total width. Under the S-Blank assumption, because of Lemma 1, this problem can be optimally solved through the following two-step greedy approach.

(1) All characters are sorted decreasingly by blanks;
(2) All characters are inserted one by one. Each one can be inserted at either left end or right end.

One example of the greedy approach is illustrated in Figure 7, where four character candidates A, B, C, and D are to be ordered. In Figure 7(a), they are sorted decreasingly by blank space. Then, all the candidates are inserted one by one. From the second candidate, each insertion has two options: left side or right side of all inserted candidates. For example, if A is inserted at the right of D, B has two insertion options: one is at the right side of A (Figure 7(b)) and another is at the left side of A (Figure 7(d)). Given different choices of candidate B, Figures 7(c) and 7(e) give corresponding final solutions. Since, from the second candidate each one has two choices, by this greedy approach n candidates will generate 2^{n-1} possible solutions.

For the asymmetrical cases, the optimality does not hold anymore. To compensate the loss, E-BLOW consists of a refinement stage. For n characters $\{c_1, \ldots, c_n\}$, single row ordering can have $n!$ possible solutions. We avoid enumerating such huge solutions, and take advantage of the order in symmetrical blank assumption. That is, we pick up one best solution from the 2^{n-1} possible ones. Note that to search for optimal single row packing, a straightforward method is to enumerate all $n!$ options. But in E-BLOW, we only consider the 2^{n-1} options with the following two reasons. (1) The runtime complexity of $n!$ is unacceptable in practice. For example, if a single row contains 30 characters, then the enumeration will generate more than 2.6×10^{32} options. (2) Our preliminary results on small test cases show that the solution quality loss is negligible. In particular, in the experimental result we will show that E-BLOW can achieve optimal solutions for small test cases.

The refinement is based on dynamic programming and the details are shown in Algorithm 3. Refine(k) generates all possible order solutions for the first k characters $\{c_1, \ldots, c_k\}$. Each order solution is represented as a set (w, l, r, O), where w is the total length of the order, l is the left blank of the left character, r is the right blank of the right character, and O is the character order. At the beginning, an empty solution set

ALGORITHM 3: Refine(k)

Input: k characters $\{c_1, \ldots, c_k\}$;
1: **if** $k = 1$ **then**
2: Add $(w_1, sl_1, sr_1, \{c_1\})$ into S;
3: **else**
4: Refine($k - 1$);
5: **for** each partial solution (w, l, r, O) **do**
6: Remove (w, l, r, O) from S;
7: Add $(w + w_k - \min(sr_k, l), sl_k, r, \{c_k, O\})$ into S;
8: Add $(w + w_k - \min(sl_k, r), l, sr_k, \{O, c_k\})$ into S;
9: **end for**
10: **if** size of $S \geq \gamma$ **then**
11: Prune inferior solutions in S;
12: **end if**
13: **end if**

S is initialized (line 1). If $k = 1$, then an initial solution $(w_1, sl_1, sr_1, \{c_1\})$ would be generated (line 2). Here, w_1, sl_1, and sr_1 are the width of the first character c_1, the left blank of c_1, and the right blank of c_1. If $k > 1$, then *Refine*(k) will recursively call *Refine*(k-1) to generate all old, partial solutions. All these partial solutions will be updated by adding candidate c_k (lines 5–9). We propose pruning techniques to speed-up the dynamic programming process. Let us introduce the concept of inferior solutions. For any two solutions $S_A = (w_a, l_a, r_a, O_a)$ and $S_B = (w_b, l_b, r_b, O_b)$, we say S_B is **inferior** to S_A, if and only if $w_a \geq w_b$, $l_a \leq l_b$, and $r_a \leq r_b$. Those inferior solutions would be pruned during the pruning section (lines 10–12). In our implementation, γ is set to 20.

4.5. Post-Swap and Post-Insertion

A post-swap stage is applied to further improve the performance, where some unselected characters would be swapped with some characters on stencil if the swaps can improve the system writing time. The procedure of post-swap is detailed in Algorithm 4. The input consists of a set of characters on stencil IN $= \{i_1, \ldots, i_m\}$, and a set of characters not selected OUT $= \{o_1, \ldots, o_n\}$. The post-swap is implemented using a greedy flavor with two steps. In the first step, all the characters are sorted based on profit values (lines 1–2). The characters in IN are in ascending order, while the characters in OUT are in descending order. Please refer to Equation (7) about the profit value. The idea is that if an unselected character is with high profit value, assigning it into stencil may help the whole system writing time a lot. Similarly, if a character on stencil

ALGORITHM 4: Post-Swap

Input: m characters on stencil IN $= \{i_1, \ldots, i_m\}$;
Input: n characters not selected OUT $= \{o_1, \ldots, o_n\}$;
1: sort IN in ascending order based on profit value;
2: sort OUT in descending order based on profit value;
3: **for** each $i_p \in$ IN **do**
4: **for** each $o_q \in$ OUT **do**
5: **if** Swap (i_p, o_q) can improve performance **then**;
6: Label i_p as unselected, while o_q as selected on stencil;
7: Remove o_q from OUT;
8: Break;
9: **end if**
10: **end for**
11: **end for**

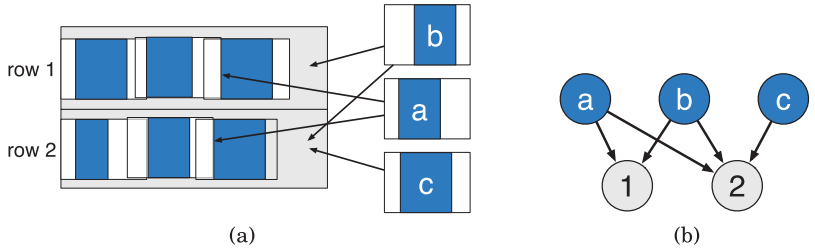

Fig. 8. Example of maximum weighted matching-based post character insertion. (a) Three additional characters a, b, c and two rows. (b) Corresponding bipartite graph to represent the relationships among characters and rows.

is with low profit value, we tend to remove it from stencil so that more high value characters can be pushed in. In the second step (lines 3–11), we traverse two sets of characters to check whether swapping characters $i_p \in$ IN and $o_q \in$ OUT can improve the performance. If so, then character i_p is unselected from stencil, while character o_q is assigned into stencil.

In Algorithm 4, the character sorting (lines 1–2) needs $O(n\,logn + m\,logm)$, where n and m are the character numbers in IN and OUT, respectively. The for loops (lines 3–11) need $O(nm)$. Thus, the post-swap procedure can be finished in $O(nm)$ time.

After post-swap, a post-insertion stage is applied to further insert more characters into stencil. Different from the greedy insertion approach in Yuan et al. [2012] that new characters can be only inserted into one row's right end. We consider to insert characters into the middle part of rows. Generally speaking, the character with higher profit value (7) would have a higher priority to be inserted into rows. We propose a character insertion algorithm to insert some additional characters into the rows. The insertion is formulated as a maximum weighted matching problem [Galil 1986], under the constraint that for each row there is at most one character that can be inserted. Although this assumption may lose some optimality, in practice it works quite well, as the remaining space for a row is usually very limited.

Figure 8 illustrates one example of the character insertion. As shown in Figure 8(a), there are two rows (row 1, row 2) and three additional characters (a, b, c). Characters a and b can be inserted into either row 1 or row 2, but character c can only be inserted into row 2. It shall be noted that the insertion position is labeled by arrows. For example, two arrows from character a mean that a can be inserted into the middle of each row. We build up a bipartite graph to represent the relationships among characters and rows (see Figure 8(b)). Each edge is associated with a cost as the character's profit. By utilizing the bipartite graph, the best character insertion can be solved by finding a maximum weighted matching.

We build up a graph to transfer the post-insertion problem into a conventional maximum weighted matching problem. The time complexity of the graph construction is $O(mnC)$, where m is the total row number, n is the additional character number, and C is the maximum character number on each row. We propose two heuristics to speed up the graph construction process. First, to reduce n, we only consider those additional characters with high profits. Second, to reduce m, we skip those rows with very little empty space.

Based on the constructed graph, we carry out the maximum weighted matching to assign additional characters into rows. Solving the matching needs $O(|V| \cdot \log|V| \cdot |E|)$ [Galil 1986], where $|V|$ and $|E|$ are the vertex number and the edge number in the constructed graph, respectively.

5. EXPERIMENTAL RESULTS

E-BLOW is implemented in C++ programming language and executed on a Linux machine with two 3.0GHz CPU and 32GB Memory. GUROBI [Gurobi Optimization Inc. 2014] is used to solve ILP/LP. The benchmark suite (1D-1, ..., 1D-4) from Yuan et al. [2012] is tested for an OSP problem. To evaluate the algorithms for the MCC system, eight benchmarks (1M-x) are generated for the MOSP problem. In these new benchmarks, character projection (CP) number are all set to 10. For each small case (1M-1, ..., 1M-4) the character candidate number is 1,000, and the stencil size is set to $1{,}000\mu m \times 1{,}000\mu m$. For each larger case (1M-5, ..., 1M-8), the character candidate number is 4,000, and the stencil size is set to $2{,}000\mu m \times 2{,}000\mu m$. The size and the blank width of each character are similar to those in Yuan et al. [2012].

5.1. E-BLOW vs. Previous Work

In the first experiment, we compare E-BLOW with some previous works: the greedy method in Yuan et al. [2012], the heuristic framework in Yuan et al. [2012], and the algorithms in Kuang and Young [2014a]. Table II lists the comparison results. We have obtained the programs of Yuan et al. [2012] and executed them in our machine. It should be noted that Yuan et al. [2012] is targeting at a single CP system, and the MCC system is modified to optimize the total writing time of all the regions. The results of Kuang and Young [2014a] are directly from their paper. Column "**cand #**" is the number of character candidates, while column "**CP#**" is the number of character projections. For each algorithm, we report "**T**," "**char#**," and "**CPU(s)**," where "T" is the writing time of the EBL system, "char#" is the character number on final stencil, and "CPU(s)" reports the computational runtime. From Table II we can see E-BLOW achieves better performance than both the greedy method and the heuristic method in Yuan et al. [2012]. Compared with E-BLOW, the greedy method has 32% more system writing time, while Yuan et al. [2012] introduces 19% more system writing time. One possible reason is that, different from the greedy/heuristic methods, E-BLOW proposes mathematical formulations to search for global solution space. Additionally, due to the successive rounding scheme, E-BLOW is around 15× faster than the work in Yuan et al. [2012].

In Table II, E-BLOW is also compared with one recent OSP work [Kuang and Young 2014a]. We can see that E-BLOW is able to find stencil placements with better EBL system writing time for 10 out of 12 test cases. In addition, for all the MCC system test cases (1M-1, ..., 1M-8), E-BLOW outperforms [Kuang and Young 2014a]. One possible reason is that to optimize the overall throughput of the MCC system, a global view is necessary to balance the throughputs among different regions. E-BLOW utilizes the mathematical formulations to provide such global optimization. Although our runtimes are longer than those in Kuang and Young [2014b], they are still acceptable that on average, the runtime is only 8.6 seconds. Even for the largest test case, E-BLOW can be finished in 22 seconds. In addition, our performance is better than Kuang and Young [2014b], that it can reduce 2% of the EBL system writing time. For a specific design, the EBL system writing time could be several hours or several days, thus the 2% throughput improvement could be attractive compared to a 22 second computation time penalty. After all, E-BLOW and Kuang and Young [2014b] are complementary, that both of them provide good runtime-performance tradeoff with different foci.

5.2. Effectiveness of the Proposed Techniques

In the second experiment, we demonstrate the effectiveness of two proposed techniques, that is, the fast ILP convergence (Section 4.3) and the post-optimization

Table II. Compare E-BLOW with Previous Works

	cand	CP	Greedy in Yuan et al. [2012]			[Yuan et al. 2012]			[Kuang and Young 2014a]			E-BLOW		
	#	#	T	char#	CPU(s)	T	char#	CPU(s)	T	char#	CPU(s)	T	char#	CPU(s)
1D-1	1,000	1	64,891	912	0.03	42,524	943	11.59	**19,095**	940	0.01	19,602	939	1.64
1D-2	1,000	1	99,381	844	0.03	82,033	874	10.32	35,295	864	0.01	**34,974**	866	1.37
1D-3	1,000	1	165,480	748	0.03	147,353	774	7.35	69,301	757	0.01	**68,553**	763	2.01
1D-4	1,000	1	193,881	691	0.04	179,547	714	9.74	**92,523**	703	0.01	92,805	705	1.54
1M-1	1,000	10	63,811	912	0.03	45,521	943	12.25	39,026	938	0.01	**37,057**	945	3.03
1M-2	1,000	10	104,877	844	0.03	87,100	874	10.46	77,997	864	0.01	**74,770**	875	2.31
1M-3	1,000	10	172,834	748	0.03	153,932	774	6.24	138,256	758	0.56	**132,879**	774	2.64
1M-4	1,000	10	200,498	691	0.03	187,358	714	9.36	176,228	698	0.36	**169,670**	716	9.36
1M-5	4,000	10	274,992	3,604	0.43	225,124	3,683	530.17	204,114	3,660	0.03	**202,397**	3,680	20.32
1M-6	4,000	10	437,088	3,341	0.43	390,079	3,418	377.30	357,829	3,382	0.03	**348,855**	3,419	22.08
1M-7	4,000	10	650,419	3,000	0.42	604,168	3,071	305.37	568,339	3,016	0.59	**555,731**	3,071	17.64
1M-8	4,000	10	820,013	2,756	0.53	773,387	2,818	248.50	731,483	2,760	0.42	**716,440**	2,820	20.07
Avg.	-	-	270,680.4	1,590.9	0.17	243,177.2	1,633.3	128.22	209,123.8	1,611.7	0.17	204,477.8	1,631.1	8.67
Ratio	-	-	**1.32**	0.98	0.02	**1.19**	1.00	14.79	**1.02**	0.99	0.02	**1.0**	1.0	1.0

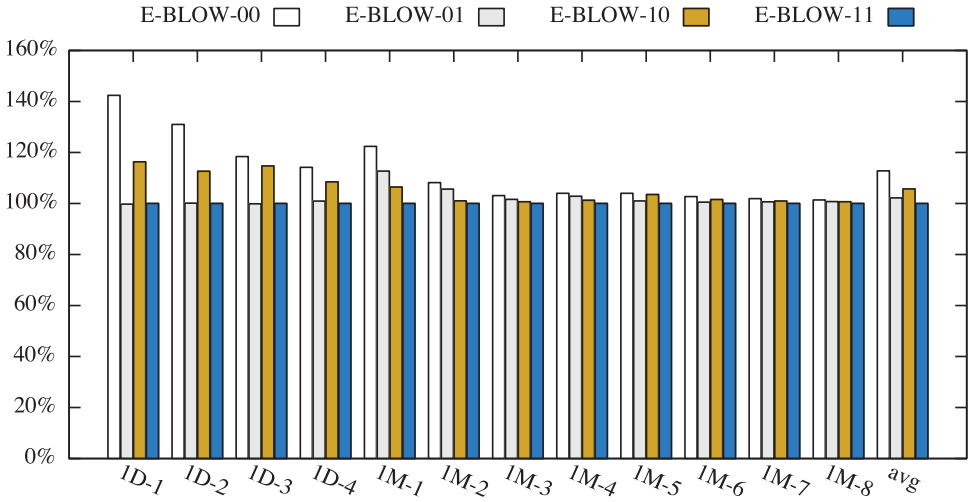

Fig. 9. Comparison of different E-BLOW versions on EBL system writing time.

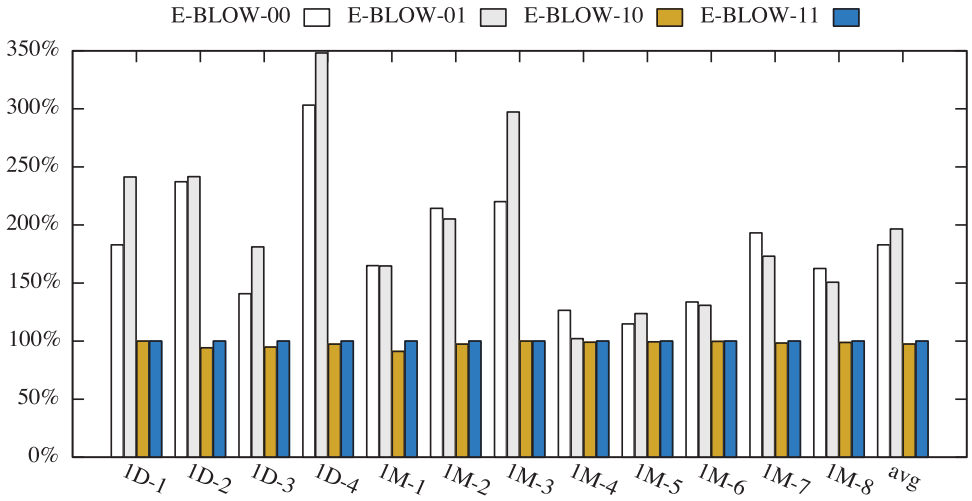

Fig. 10. Comparison of different E-BLOW versions on computational runtime.

(Section 4.5). To facilitate the comparison, we implemented four different versions of E-BLOW framework: **E-BLOW-00** is the framework where no fast ILP convergence or post-optimization is utilized; **E-BLOW-01** is with post-optimization but no fast convergence; **E-BLOW-10** is with fast convergence but no post-optimization; **E-BLOW-11** is integrated with both fast ILP convergence and post-optimization.

Figure 9 compares these four E-BLOW versions in terms of the EBL system writing time. As shown in Figure 9, through post-optimization, E-BLOW-01 outperforms E-BLOW-00 with around 10% system writing time improvement. Similarly, E-BLOW-11 outperforms E-BLOW-10 with around 6% system writing time improvement. We can see that post-optimization is effective in improving the E-BLOW performance.

Figure 10 compares these four E-BLOW versions in terms of computational runtime. Through fast ILP convergence, E-BLOW-10 outperforms E-BLOW-00 with nearly

Table III. ILP vs. E-BLOW

	cand	ILP				E-BLOW		
	#	binary#	T	char#	CPU(s)	T	char#	CPU(s)
1T-1	8	64	434	6	0.5	434	6	0.1
1T-2	10	100	1,034	6	26.1	1,034	6	0.2
1T-3	11	121	1,222	6	58.3	1,222	6	0.2
1T-4	12	144	1,862	6	1,510.4	1,862	6	0.2
1T-5	14	196	NA	NA	>3,600	2,758	6	0.1

two times speed-up. E-BLOW-11 can also achieve around two times speed-up against E-BLOW-01. From Figure 10, we can see that fast ILP convergence can effectively reduce E-BLOW framework CPU time.

5.3. E-BLOW vs. ILP

At last, we compare E-BLOW with the ILP formulations (4). Although the ILP formulations can find optimal solutions theoretically, they may suffer from runtime overhead and cannot get legal solutions for practical benchmarks. Therefore, we randomly generate five small benchmarks ("1T-x"). The sizes of all the character candidates are set to $40\mu m \times 40\mu m$. The row number is set to 1 and the row length is set to 200. The comparisons are listed in Table III, where column "cand #" is the number of character candidates. "**ILP**" and "**E-BLOW**" represent the ILP formulation and our E-BLOW framework, respectively. In ILP formulation, column "binary#" gives the binary variable number. For each mode, we report "T," "char#," and "CPU(s)," where "T" is the EBL system writing time, "char#" is the character number on the final stencil, and "CPU(s)" is the runtime. Note that in Table III, the ILP solutions are optimal.

Let us compare E-BLOW with the ILP formulation for test cases 1T-1, ... , 1T-5. E-BLOW can achieve the same results with ILP formulations; meanwhile, it is so fast that all cases can be finished in 0.2 seconds. Although ILP formulation can achieve optimal results, it is so slow that a case with 14 character candidates (1T-5) can not be finished in one hour.

Although the integral variable number for each case is not huge, we find that in the ILP formulations, the solutions of corresponding LP relations are vague. Therefore, expensive search methods may cause unacceptable runtimes. From these cases, ILP formulations are impossible to be directly applied in an OSP problem, as in the MCC system character number may be as large as 4,000.

6. CONCLUSION

In this article, we have proposed E-BLOW, a tool to effectively solve the OSP problem and the MOSP problem. A successive relaxation algorithm, a dynamic programming-based refinement, and a set of post-optimization techniques are proposed. Experimental results show that, compared with previous works, E-BLOW can achieve better performance in terms of both shot number and runtime. Note that in an MCC system, different regions tend to have specific stencils due to the consideration of system throughput improvement. However, if a shared stencil is so well-designed and optimized that such sharing can achieve very comparable throughput, we can even reduce the stencil design cost. In that situation, sharing stencil design could be attractive, especially for the companies that have a limited design budget. As EBL, including the MCC system, is widely used for mask making and also gaining momentum for direct wafer writing, we believe a lot more research can be done for not only stencil planning, but also EBL-aware design.

APPENDIX

PROOF OF THEOREM 1

LEMMA 5. BSS *problem is in NP.*

PROOF. It is easy to see that BSS problem is in NP. Given a subset of integer numbers $S' \in S$, we can add them up and verify that their sum is s in polynomial time. □

LEMMA 6. 3SAT $\leq_p$ BSS.

PROOF. In 3SAT problem, we are given m clauses $\{C_1, C_2, \ldots, C_m\}$ over n variables $\{y_1, y_2, \ldots, y_n\}$. Besides, there are three literals in each clause, which is the OR of some number of literals. Equation (8) gives one example of 3SAT, where $n = 4$ and $m = 2$.

$$(y_1 \vee \bar{y}_3 \vee \bar{y}_4) \wedge (\bar{y}_1 \vee y_2 \vee \bar{y}_4) \tag{8}$$

Without loss of generality, we can have the following assumptions:

(1) No clause contains both variable y_i and $\bar{y}_i$. Otherwise, any such clause is always true and we can just eliminate them from the formula.
(2) Each variable y_i appears in at least one clause. Otherwise, we can just assign any arbitrary value to the variable y_i.

To convert a 3SAT instance to a BSS instance, we create two integer numbers in set S for each variable y_i, and three integer numbers in S for each clause C_j. All the numbers in set S and s are in base 10. Besides, $10^{n+2m} < y_i < 2 \cdot 10^{n+2m}$, so that the bounded constraints are satisfied. All the details regarding S and s are defined as follows.

—In the set S, all integer numbers are with $n + 2m + 1$ digits, and the first digit is always 1.
—In the set S, we construct two integer numbers t_i and f_i for the variable y_i. For both of the values, the n digits after the first '1' serve to indicate the corresponding variable in S. That is, the i^{th} digit in these n digits is set to 1 and all others are 0. For the next m digits, the j^{th} digit is set to 1 if the clause C_j contains the respective literal. The last m digits are always 0.
—In the set S, we also construct three integer numbers c_{j1}, c_{j2}, and c_{j3} for each clause C_j. In c_{jk}, where $k = \{1, 2, 3\}$, the first n digits after the first '1' are 0, and in the next m digits, all are 0 except the j^{th} index setting to k. The last m digits are all 0, except the j^{th} index setting to 1.
—$T = (n + m) \cdot 10^{n+2m} + s_0$, where s_0 is an integer number with $n + 2m$ digits. The first n digits of s_0 are 1; in the next, m digits all are 4; and in the last, m digits all are 1.

Based on the above rules, given the 3SAT instance in Equation (8), the constructed set S and target s are shown in Figure 11. Note that the highest digit achievable is 9, meaning that no digit will carry over and interfere with other digits.

CLAIM 1. *The* 3SAT *instance has a satisfying truth assignment iff the constructed* BSS *instance has a subset that adds up to s.*

Proof of ⇒ part of Claim: If the 3SAT instance has a satisfying assignment, we can pick a subset containing all t_i, for which y_i is set to true, and f_i, for which y_i is set to false. We should then be able to achieve s by picking the necessary c_{jk} to get 4's in the s. Due to the last m '1' in s, for each $j \in [m]$, only one would be selected from $\{c_{j1}, c_{j2}, c_{j3}\}$. Besides, we can see totally $n + m$ numbers would be selected from S.

Proof of ⇐ part of Claim: If there is a subset $S' \in S$ that adds up to s, we will show that it corresponds to a satisfying assignment in the 3SAT instance. S' must include exactly one of t_i and f_i; otherwise, the ith digit value of s_0 cannot be satisfied.

			y_1	y_2	y_3	y_4	C_1	C_2	A_1	A_2
t_1	=	1	1	0	0	0	1	0	0	0
f_1	=	1	1	0	0	0	0	1	0	0
t_2	=	1	0	1	0	0	0	1	0	0
f_2	=	1	0	1	0	0	0	0	0	0
t_3	=	1	0	0	1	0	0	0	0	0
f_3	=	1	0	0	1	0	1	0	0	0
t_4	=	1	0	0	0	1	0	0	0	0
f_4	=	1	0	0	0	1	1	1	0	0
c_{11}	=	1	0	0	0	0	1	0	1	0
c_{12}	=	1	0	0	0	0	2	0	1	0
c_{13}	=	1	0	0	0	0	3	0	1	0
c_{21}	=	1	0	0	0	0	0	1	0	1
c_{22}	=	1	0	0	0	0	0	2	0	1
c_{23}	=	1	0	0	0	0	0	3	0	1
s	=	6	1	1	1	1	4	4	1	1
s_0	=		1	1	1	1	4	4	1	1

Fig. 11. The constructed BSS instance for the given 3SAT instance in Equation (8).

If $t_i \in S'$, in the 3SAT we set y_i to true; otherwise, we set it to false. Similarly, S' must include exactly one of c_{j1}, c_{j2}, and c_{j3}; otherwise the last m digits of s cannot be satisfied. Therefore, all clauses in the 3SAT are satisfied and 3SAT has a satisfying assignment. □

For instance, given a satisfying assignment of Equation (8): $\langle y_1 = 0, y_2 = 1, y_3 = 0, y_4 = 0\rangle$, the corresponding subset S' is $\{f_1 = 110000100, t_2 = 101000100, f_3 = 100101000, f_4 = 100011100, c_{12} = 100002010, c_{21} = 100000101\}$. We set $s = (m + n) \cdot 10^{n+2m} + s_0$, where $s_0 = 11114411$, and then $s = 611114411$. We can see that $f_1 + t_2 + f_3 + f_4 + s_{12} + s_{21} = s$.

ACKNOWLEDGMENT

The authors would like to thank Zhao Song at University of Texas for helpful comments.

REFERENCES

Yukiyasu Arisawa, Hajime Aoyama, Taiga Uno, and Toshihiko Tanaka. 2010. EUV flare correction for the half-pitch 22nm node. In *Proceedings of SPIE*, Vol. 7636.

Sanjeev Arora and Boaz Barak. 2009. *Computational Complexity: A Modern Approach*. Cambridge University Press.

Sergey Babin, Andrew B. Kahng, Ion I. Mandoiu, and Swamy Muddu. 2003. Resist Heating Dependence on Subfield Scheduling in 50kV Electron Beam Maskmaking. In *Proceedings of SPIE*, Vol. 5130.

Tuck Boon Chan, Puneet Gupta, Kwangsoo Han, Abde Ali Kagalwalla, Andrew B. Kahng, and Emile Sahouria. 2014. Benchmarking of Mask Fracturing Heuristics. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 246–253.

Li-Wen Chang, Xinyu Bao, Bencher Chris, and H.-S. Philip Wong. 2010. Experimental demonstration of aperiodic patterns of directed self-assembly by block copolymer lithography for random logic circuit layout. In *IEEE International Electron Devices Meeting (IEDM)*. 33.2.1–33.2.4.

Chris Chu and Wai-Kei Mak. 2014. Flexible packed stencil design with multiple shaping apertures for e-beam lithography. In *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*. 137–142.

M. Dawande, J. Kalagnanam, P. Keskinocak, F. S. Salman, and R. Ravi. 2000. Approximation Algorithms for the Multiple Knapsack Problem with Assignment Restrictions. *Journal of Combinatorial Optimization* 4 (2000), 171–186. Issue 2.

Yixiao Ding, Chris Chu, and Wai-Kei Mak. 2014. Throughput Optimization for SADP and E-beam based Manufacturing of 1D Layout. In *ACM/IEEE Design Automation Conference (DAC)*. 51:1–51:6.

Peng Du, Wenbo Zhao, Shih-Hung Weng, Chung-Kuan Cheng, and Ronald Graham. 2012b. Character Design and Stamp Algorithms for Character Projection Electron-Beam Lithography. In *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*. 725–730.

Yuelin Du, Hongbo Zhang, Martin D. F. Wong, and Kai-Yuan Chao. 2012a. Hybrid lithography optimization with e-beam and immersion processes for 16nm 1D gridded design. In *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*. 707–712.

S.-Y. Fang, W.-Y. Chen, and Y.-W. Chang. 2013a. Graph-based subfield scheduling for electron-beam photomask fabrication. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* 32, 2 (2013), 189–201.

Shao-Yun Fang, Iou-Jen Liu, and Yao-Wen Chang. 2013b. Stitch-aware routing for multiple e-beam lithography. In *ACM/IEEE Design Automation Conference (DAC)*. 25:1–25:6.

Aki Fujimura. 2010. Design for E-Beam: Design insights for direct-write maskless lithography. In *Proceedings of SPIE*, Vol. 7823.

Takeshi Fujino, Yoshihiko Kajiya, and Masaya Yoshikawa. 2005. Character-build standard-cell layout technique for high-throughput character-projection EB lithography. In *Proceedings of SPIE*, Vol. 5853.

Zvi Galil. 1986. Efficient algorithms for finding maximum matching in graphs. *Comput. Surveys* 18, 1 (Mar. 1986), 23–38.

Jhih-Rong Gao, Bei Yu, and David Z. Pan. 2014. Self-aligned double patterning layout decomposition with complementary e-beam lithography. In *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*. 143–148.

Daifeng Guo, Yuelin Du, and Martin D. F. Wong. 2015. Polynomial time optimal algorithm for stencil row planning in E-Beam lithography. In *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*. 658–664.

Gurobi Optimization Inc. 2014. Gurobi Optimizer Reference Manual. http://www.gurobi.com. (2014).

Rimon Ikeno, Takashi Maruyama, Tetsuya Iizuka, Satoshi Komatsu, Makoto Ikeda, and Kunihiro Asada. 2013a. High-throughput electron beam direct writing of VIA layers by character projection using character sets based on one-dimensional VIA arrays with area-efficient stencil design. In *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*. 255–260.

Rimon Ikeno, Takashi Maruyama, Satoshi Komatsu, Tetsuya Iizuka, Makoto Ikeda, and Kunihiro Asada. 2013b. A structured routing architecture and its design methodology suitable for high-throughput electron beam direct writing with character projection. In *ACM International Symposium on Physical Design (ISPD)*. 69–76.

Ellis L. Johnson, George L. Nemhauser, and Martin W. P. Savelsbergh. 2000. Progress in linear programming-based algorithms for integer programming: An exposition. *INFORMS Journal on Computing* 12, 1 (2000), 2–23.

Andrew B. Kahng, Chul-Hong Park, Xu Xu, and Hailong Yao. 2008. Layout decomposition for double patterning lithography. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 465–472.

Andrew B. Kahng, Xu Xu, and Alex Zelikovsky. 2006. Fast Yield-Driven Fracture for Variable Shaped-Beam Mask Writing. In *Proceedings of SPIE*, Vol. 6283.

Jian Kuang and Evangeline F. Y. Young. 2014a. A Highly-Efficient Row-Structure Stencil Planning Approach for E-Beam Lithography with Overlapped Characters. In *ACM International Symposium on Physical Design (ISPD)*. 109–116.

Jian Kuang and Evangeline F. Y. Young. 2014b. Overlapping-aware Throughput-driven Stencil Planning for E-Beam Lithography. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 254–261.

Yibo Lin, Bei Yu, Yi Zou, Zhuo Li, Charles J. Alpert, and David Z. Pan. 2016. Stitch Aware Detailed Placement for Multiple E-Beam Lithography. In *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*. 186–191.

Xu Ma, Shangliang Jiang, and Avideh Zakhor. 2011. A Cost-Driven Fracture Heuristics to Minimize Sliver Length. In *Proceedings of SPIE*, Vol. 7973.

Wai-Kei Mak and Chris Chu. 2014. E-Beam Lithography Character and Stencil Co-Optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* 33, 5 (May 2014), 741–751.

S. Manakli, H. Komami, M. Takizawa, T. Mitsuhashi, and L. Pain. 2009. Cell projection use in mask-less lithography for 45nm & 32nm logic nodes. In *Proceedings of SPIE*, Vol. 7271.

Silvano Martello and Paolo Toth. 1990. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Inc.

Takashi Maruyama, Yasuhide Machida, Shinji Sugatani, Hiroshi Takita, Hiromi Hoshino, Toshio Hino, Masaru Ito, Akio Yamada, Tetsuya Iizuka, Satoshi Komatsue, Makoto Ikeda, and Kunihiro Asada. 2012. CP element based design for 14nm node EBDW high volume manufacturing. In *Proceedings of SPIE*, Vol. 8323.

T. Maruyama, M. Takakuwa, Y. Kojima, Y. Takahashi, K. Yamada, J. Kon, M. Miyajima, A. Shimizu, Y. Machida, H. Hoshino, H. Takita, S. Sugatani, and H. Tsuchikawa. 2008. EBDW technology for EB shuttle at 65nm node and beyond. In *Proceedings of SPIE*, Vol. 6921.

Hai Pham Dinh Minh, Tetsuya Iizuka, Makoto Ikeda, and Kunihiro Asada. 2006. Shot minimization for throughput improvement of character projection electron beam direct writing. In *Proceedings of SPIE*, Vol. 6921.

David Z. Pan, Bei Yu, and J.-R. Gao. 2013. Design for Manufacturing With Emerging Nanolithography. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* 32, 10 (2013), 1453–1472.

Hans C. Pfeiffer. 2009. New prospects for electron beams as tools for semiconductor lithography. In *Proceedings of SPIE*, Vol. 7378.

Masahiro Shoji, Tadao Inoue, and Masaki Yamabe. 2010. Extraction and utilization of the repeating patterns for CP writing in mask making. In *Proceedings of SPIE*, Vol. 7748.

Makoto Sugihara, Taiga Takata, Kenta Nakamura, Ryoichi Inanami, Hiroaki Hayashi, Katsumi Kishimoto, Tetsuya Hasebe, Yukihiro Kawano, Yusuke Matsunaga, Kazuaki Murakami, and Katsuya Okumura. 2006a. Cell Library Development Methodology for Throughput Enhancement of Character Projection Equipment. *IEICE Transactions on Electronics* E89-C (2006), 377–383.

Makoto Sugihara, Taiga Takata, Kenta Nakamura, Ryoichi Inanami, Hiroaki Hayashi, Katsumi Kishimoto, Tetsuya Hasebe, Yukihiro Kawano, Yusuke Matsunaga, Kazuaki Murakami, and Katsuya Okumurae. 2007. Technology mapping technique for throughput enhancement of character projection equipment. In *Proceedings of SPIE*, Vol. 6151.

M. Sugihara, T. Takata, K. Nakamura, Rx. Inanami, R. Inanami, H. Hayashi, K. Kishimoto, T. Hasebe, Y. Kawano, Y. Matsunaga, K. Murakami, and K. Okumura. 2006b. A character size optimization technique for throughput enhancement of character projection lithography. In *IEEE International Symposium on Circuits and Systems (ISCAS)*. 2561–2564.

Yunfeng Yang, Wai-Shing Luk, Hai Zhou, Changhao Yan, Xuan Zeng, and Dian Zhou. 2015. Layout decomposition co-optimization for hybrid e-beam and multiple patterning lithography. In *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*. 652–657.

Hiroshi Yasuda, Takeshi Haraguchi, and Akio Yamada. 2004. A proposal for an MCC (Multi-column cell with lotus root lens) system to be used as a mask-making e-beam tool. In *Proceedings of SPIE*, Vol. 5567.

Bei Yu, Jhih-Rong Gao, and David Z. Pan. 2013a. L-Shape Based Layout Fracturing for E-Beam Lithography. In *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*. 249–254.

Bei Yu and David Z. Pan. 2014. Layout Decomposition for Quadruple Patterning Lithography and Beyond. In *ACM/IEEE Design Automation Conference (DAC)*. 53:1–53:6.

Bei Yu, Kun Yuan, Jhih-Rong Gao, and David Z. Pan. 2013b. E-BLOW: E-Beam Lithography Overlapping aware Stencil Planning for MCC System. In *ACM/IEEE Design Automation Conference (DAC)*. 70:1–70:7.

Bei Yu, Kun Yuan, Boyang Zhang, Duo Ding, and David Z. Pan. 2011. Layout Decomposition for Triple Patterning Lithography. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–8.

Kun Yuan, Bei Yu, and David Z. Pan. 2012. E-Beam Lithography Stencil Planning and Optimization With Overlapped Characters. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* 31, 2 (Feb. 2012), 167–179.

Hongbo Zhang, Yuelin Du, M. D. Wong, and Rasit Topaloglu. 2011. Self-Aligned Double Patterning Decomposition for Overlay Minimization and Hot Spot Detection. In *ACM/IEEE Design Automation Conference (DAC)*. 71–76.

Hongbo Zhang, Yuelin Du, Martin D. F. Wong, Yunfei Deng, and Pawitter Mangat. 2012. Layout small-angle rotation and shift for EUV defect mitigation. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 43–49.

Received July 2015; revised December 2015; accepted January 2016