Provably Good and Practically Efficient Algorithms for CMP Dummy Fill

Chunyang Feng[1], Hai Zhou[1,2], Changhao Yan[1], Jun Tao[1], Xuan Zeng[1*]
[1]State Key Lab of ASIC & System, Microelectronics Dept., Fudan University, China
[2]EECS, Northwestern University, U.S.A.

***Abstract*—To reduce chip-scale topography variation in Chemical Mechanical Polishing (CMP) process, dummy fill is widely used to improve the layout density uniformity. Previous researches formulated the dummy fill problem as a standard Linear Program (LP). However, solving the huge linear program formed by real-life designs is very expensive and has become the hurdle in deploying the technology. Even though there exist efficient heuristics, their performance cannot be guaranteed. In this paper, we develop a dummy fill algorithm that is both efficient and with provably good performance. It is based on a fully polynomial time approximation scheme by Fleischer [4] for covering LP problems. Furthermore, based on the approximation algorithm, we also propose a new greedy iterative algorithm to achieve high quality solutions more efficiently than previous Monte-Carlo based heuristic methods. Experimental results demonstrate the effectiveness and efficiency of our algorithms.**

Categories and Subject Descriptors:
J.6 [Computer-Aided Engineering]: Computer-Aided Design
General Terms: Design, Algorithms
Keywords: Design for Manufacturability, Dummy Fill Problem, Covering Linear Programming

I. Introduction

Chemical Mechanical Polishing (CMP) is widely used as the primary planarizing technique in the fabrication of integrated circuits. Despite being a predominant planarizing technique, CMP is known to suffer from undesired pattern dependent problems. Previous studies show that post-CMP topography is strongly dependent on the underlying feature density [11]. To achieve layout density uniformity, dummy fill is a highly recommended technique by foundries to increase the density of sparse regions.

In general, layout density control consists of two phases: density analysis and fill synthesis. Density analysis determines the area available for filling, while fill synthesis computes the amount of dummy fills for each density tile of the layout [3]. In this paper, we address the main problem of the dummy fill synthesis.

The existing work in the area of fill synthesis can be classified into two categories [10]: *linear-programming* (LP) based approaches and Monte-Carlo (or greedy) based heuristic approaches. Two objectives, the Min-Var objective and Min-Fill objective are proposed. The Min-Var objective seeks the most uniform density distribution possible, and the Min-Fill objective seeks to minimize the dummy feature insertion cost while satisfies a density uniformity constraint [8]. Kahng et al. [7] proposed the first LP formulation for the Min-Var objective. Tian et al. [13] gave the first LP formulation for the Min-Fill objective. Although LP solvers produce optimal solution for these formulations, the runtime is too expensive, in the order of n^3, where n is the number of variables in the LP. As stated in [8], in the fixed-dissection paradigm, if the window size $w = 200\mu m$, and each window is divided into $r = 4$ steps, then for the problem with a chip of $20mm$ on each side, there would be 160000 variables. This renders the LP-based method infeasible. This problem will get even worse with the density window size getting smaller. Recent experiments found that a window of about $50\mu m$ to $60\mu m$ is necessary to obtain the pattern density for copper CMP process [9]. Therefore, solving the linear program has become the computational bottleneck in the fill synthesis problem.

To alleviate this difficulty, several Monte-Carlo or greedy based heuristic approaches were proposed [1], [2], [14]. These methods select a tile according to certain criteria in each iteration, and fill it with a predetermined amount of dummy fills. These methods have been shown to take less runtime. However, due to their heuristic nature, their performance cannot be guaranteed or even bounded. Moreover, there is no clear guidance for the determination of the predefined filled amount during each iteration. If a large amount of fill is added into a tile during each iteration as in the greedy method [2], an excessive amount of total fill could be inserted in the fill. If only a single filling geometry is added per iteration as in the Monte Carlo approach [1], the runtime might become too long.

To address these issues in the existing approaches, the present work

- proposes, for the first time, a *covering linear program* (CLP) formulation for the Min-Fill objective of the dummy fill problem, and presents a Fully Polynomial Time Approximation Scheme (FPTAS) for the Min-Fill CLP problem based on the work of Fleischer [4]. Experimental results show that the new FPTAS algorithm is highly practical.
- develops a new greedy iterative method based on the concept of the new FPTAS algorithm for solving the Min-Fill problem, which improves the existing heuristic methods both in performance and runtime.

The rest of this paper is organized as follows. In Section II, we describe the Min-Fill LP formulation of the dummy fill problem. In Section III, we first give the formal formulation of the Min-Fill CLP problem, and then present the provably good and efficient approximation algorithm. In Section IV, we propose a new greedy iterative method based on the approximation algorithm. The experimental results are presented in Section V, and Section VI concludes the paper.

II. Problem Formulation

According to several widely accepted chip-scale CMP models [11], [6], the post-CMP topography is proportional to the feature density

*Corresponding author. E-mail: xzeng@fudan.edu.cn

DAC 2009, July 26 - 31, 2009, San Francisco, California, USA

within a given window. Thus, to improve the CMP quality, dummy fill is adopted to reduce the layout density variation. However, dummy fill also introduces undesirable side effects. Therefore, it is desirable to uniform the layout density and at the same time minimize the total insertion amount of dummy fill.

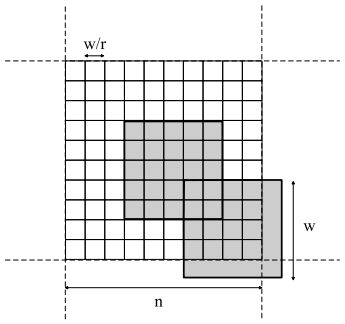


Fig. 1. The layout is discretized into $\frac{nr}{w} \times \frac{nr}{w}$ tiles ($r = 5$), and each $w \times w$ window contains r^2 tiles.

As in [3], [13], the Min-Fill objective of dummy fill problem can be described as: Given a design rule-correct layout in an $n \times n$ layout region, along with a window size $w < n$, and upper (U) and lower (L) bounds on the feature density in any window, add dummy fill features to create a filled layout such that the insertion cost is minimized while the density of any window remains in the give range (L, U).

Due to the uncountable number of windows in the layout, to make the filling problem tractable, the standard practice is to consider only a finite set of overlapping $w \times w$ windows of a fixed r-dissection [3], where r determines the window shift step w/r, as illustrated in Fig. 1. The $n \times n$ layout is partitioned into tiles $T_{ij}, i, j = 0, 1, \cdots, (nr/w) - 1$, with tile size w/r, such that each window W_{ij} centered at T_{ij}, $i, j = 0, 1, \cdots, (nr/w) - 1$, consists of r^2 tiles. Note that windows are "wrapped around" the layout, that is, a window that overlaps with the upper edge of the layout also contains tiles on the bottom of the layout. This is used to model the wafer with consecutive chips.

The density of the window W_{ij} in the "fixed-dissection" can be calculated using the following equation

$$\rho_w(i,j) = \sum_{k=i-r/2}^{i+r/2} \sum_{l=j-r/2}^{j+r/2} [d(k,l) \times f(k-i, l-j)], \qquad (1)$$

where $f()$ is the CMP filter function and $d(i,j)$ is the feature density in tile T_{ij}. The shape of the filter function depends on the used CMP models. In [7], the square function is used, which means the window density is simply equal to the average of tile densities in the window, while a low-pass filter function is incorporated in [13].

Note that Eq. (1) does not consider the multi-layer cumulative effect of topography thickness [13]. Even though only the single-layer dummy fill problem is discussed in this paper, our formulations and algorithms can be easily extended to multi-layer case.

By now, if we arrange both the tile density d and the window density ρ_w in vector forms, then the Min-Fill LP formulation proposed in [13] is given as follows:

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & l \le \rho_w \le u \\ & 0 \le x \le \textit{slack} \end{aligned} \qquad (2)$$

In the above formulation, x is a vector of size $(\frac{nr}{w})^2$ representing the dummy fill amounts in tiles. The term c is referred to as the insertion cost factor. If all the elements in c are equal, then the objective becomes to minimize the total amount of dummy fills. Terms l and u are also both vectors with size of $(\frac{nr}{w})^2$. All the elements of l equals to L, and all the elements of u equals to U, representing the window density constraints. The window density ρ_w can be expressed as a system of linear equations based on Eq. (1)

$$\rho_w = Ad = A(x + x^0), \qquad (3)$$

where x^0 is the original tile density before filling. A is the window matrix, giving the mapping from the tile density to the window density.

The term *slack* is the density upper bound of dummy fills for each tile, which is determined in the density analysis phase. For example, a coupling-constrained density analysis algorithm was proposed in [15] which identifies feasible locations and amounts of dummy fills such that the fill-induced coupling capacitance is bounded. Based on [15], $slack$ can be easily derived and the final dummy fill solution is guaranteed to satisfy both the density rules and coupling constraints.

The Min-Fill LP in Eq. (2) imposes density constraints only on windows in the "fixed-dissection". Therefore, density constraints can still be violated by other windows within the layout. Fortunately, for the window density calculated using the square filter function, the discrepancy is exactly characterized in the following two theorems [8].

Theorem 1: Suppose all $w \times w$ windows in the "fixed-dissection" of the layout have area density at least L and at most U. Then any $w \times w$ window has density at least $L - \frac{1}{r} + \frac{1}{4r^2}$ and at most $U + \frac{1}{r} - \frac{1}{4r^2}$, and these bounds are tight.

Theorem 2: Suppose all $\frac{w}{r} \times \frac{w}{r}$ tiles in the "fixed-dissection" of the layout have area density at least L and at most U. Then the exact lower bound on the area density of any $w \times w$ window equals

$$\frac{(r-1)^2}{r^2} L + \frac{4(r-1)}{r^2} \max\{L - 0.5, 0\} + \frac{4}{r^2} \max\{L - 0.75, 0\}$$

and the exact upper bound equals

$$\frac{(r+1)^2}{r^2} U - \frac{4(r-1)}{r^2} \max\{U - 0.5, 0\} - \frac{4}{r^2} \max\{U - 0.25, 0\}$$

Although Eq. (2) can be solved exactly by standard LP solvers, these solvers are often too time-consuming for large problems. The number of variables and the number of constraints in Eq. (2) are both $O((\frac{nr}{w})^2)$. As discussed in previous section, the window size of $50\mu m - 100\mu m$ is often used to calculate the window density. If the shift step $r = 4$ or 5, even for a medium-sized chip with $n = 1mm$, the total number of variables will be $O(10^4)$, making the standard LP method too expensive to finish in reasonable time.

III. Fast Approximation Scheme

A. The Min-Fill CLP Formulation

If we change the Min-Fill LP in Eq. (2) slightly by moving the upper bound u on the window density to the tile density, we will get the following formulation:

$$\begin{aligned} \text{minimize} \quad & P(x) = c^T x \\ \text{subject to} \quad & Ax \ge b \\ & x \le s \\ & x \ge 0 \end{aligned} \qquad (4)$$

where $b = \max\{l - Ax_0, \mathbf{0}\}$. It is the density lower bound of dummy fills for each window. The dummy density upper bound for each tile is redefined as

$$s = \min\{u - x_0, \textit{slack}\}. \qquad (5)$$

The formulation in Eq. (4) is a standard *covering linear program* (CLP) due to the nonnegative nature of matrix A and vectors b, c and s (assuming the original layout density does not exceed U).

The reason that we can simply apply density upper bound on tiles instead of windows is that the Min-Fill CLP formulation can still guarantee similar density bounds on any $w \times w$ windows in the layout as the Min-Fill LP formulation. According to Theorem 1 and 2, the lower and upper density bounds of any $w \times w$ window in the layout of the Min-Fill LP formulation is $L - \frac{1}{r} + \frac{1}{4r^2}$ and $U + \frac{1}{r} - \frac{1}{4r^2}$, while the Min-Fill CLP formulation can ensure that the density of any $w \times w$ window in the layout can be bounded between $L - \frac{1}{r} + \frac{1}{4r^2}$ and $\frac{(r+1)^2}{r^2}U - \frac{4(r-1)}{r^2}\max\{U - 0.5, 0\} - \frac{4}{r^2}\max\{U - 0.25, 0\}$.

B. Provably Good Algorithm for Dummy Fill

The advantage of formulating the Min-Fill problem as a CLP problem is to leverage the efficient Fully Polynomial Time Approximation Scheme (FPTAS) for such a problem. Comparing with standard LP solvers, the scheme is much more efficient and produces better results if given longer running time. Our algorithm is mainly based on Fleischer [4]; the pseudo-code is given in Alg. 1.

Algorithm 1 FPTAS for Dummy Fill

Input: window matrix A, density bound b, cost factor c, tile bound s, precision ε
Output: filling density x

Initialize $x = \delta/C$
find window p with min relative density $\rho(p)/b(p)$
while $P(x) < \theta$ **do**
  $\alpha = (1+\varepsilon)\rho(p)/b(p)$
  while $\rho(p)/b(p) < \alpha$ and $P(x) < \theta$ **do**
    if tile $j \in$ window p and $x(j) < \alpha s(j)$ **then**
      put j in set $Q(p)$
    end if
    compute min weighted density price η among $Q(p)$
    for $j \in Q(p)$ **do**
      $x(j) = x(j)(1 + \varepsilon\eta/price(j,p))$
    end for
    find window p with min relative density $\rho(p)/b(p)$
  end while
  if $P(x/\alpha) < P(x^*/\alpha^*)$ **then**
    $x^* = x$, $\alpha^* = \alpha$
  end if
end while
return x^*/α^*

The general idea of the algorithm is very simple. It will iteratively find a window whose density is relatively low, and increase the window density by increasing the tile densities in the window. Increasing the density of a tile may also cause other window density to increase, giving an unexpected side-effect. A key feature of the approach is to compensate such a wrong decision not by removing the fill, but by scaling down its impact. In other words, the x value will keep increasing, but at the end of day, only x/α will be used as the solution, for a scaling factor of α.

In more detail, the relative density of a window i is given by $\rho(i)/b(i)$, where $\rho = Ax$. The minimal relative density among all the windows will be iteratively increased. The increases are grouped into phases, each of which will ensure that the minimal relative density is at least α, and α will be increased by at least a factor of $1 + \varepsilon$ from phase to phase. This is given by the outer loop of Alg. 1. The inner loop will boost the minimal relative density by increasing x in the window p of the minimal relative density. Since each tile j has a density upper bound $s(j)$, we will only increase $x(j)$ satisfying $x(j)/\alpha < s(j)$. All such tiles are collected in Q. The price of increasing unit density of window p by tile j is given by $price(j,p) = c(j)/A(p,j)$. Obviously, we should increase more on tiles with lower prices. However, since we have upper bounds on tiles, the price should be weighted by $\frac{\alpha s(j) - x(j)}{\varepsilon x(j)}$. Denoting by η the minimal weighted price among all tile in $Q(p)$, each $x(j)$ in $Q(p)$ will be increased by a factor of $\varepsilon\eta/price(j,p)$. With such an update rule, tiles with lower prices will be rewarded with larger factors, benefiting $P(x)/\alpha$.

The iterations will continue till $P(x)/\alpha$ is within ε-optimal or, equivalently, $P(x) \geq \theta$ with a properly chosen θ. It should be noticed that during iterations x/α is always a feasible solution. Therefore, the best solution will be kept for the final answer.

C. Algorithm Analysis

We are going to show that the algorithm described in the previous subsection approximates the optimal solution within a constant factor. Due to space limitation, we will only be able to give a qualitative analysis of the algorithm.

The approximation algorithm simultaneously solves the Min-Fill CLP and its dual LP. The dual solution is used in proving the approximation guarantee of the algorithm. The dual of the Min-Fill CLP is:

$$\begin{aligned} \text{maximize} \quad & D(y,z) = b^T y - s^T z \\ \text{subject to} \quad & Ay - z \leq c \\ & y \geq 0 \\ & z \geq 0 \end{aligned} \tag{6}$$

where y, z are both variables of the dual problem.

According to the duality theory, the strong duality holds for linear programs. Our approximation algorithm obtains exactly feasible solutions x and y, z to both primal and dual problems with $P(x)/D(y,z) \leq 1 + \varepsilon$. This means that we will get both feasible solutions and the solution is ε-optimal from the exact solution.

The algorithm maintains dual variables (y, z) during α-phases. The increase of $P(x)$ each time by the increase of the primal variable x is balanced in $D(y,z)$ by the increase of the dual variable y. However, if the variable $x(j)$ has already reached its upper bound, the dual variable $z(j)$ will be increased by a proper amount, so that the increases of both dual variables $y(j)$ and $z(j)$ cancel each other and make $D(y,z)$ unchanged. This update procedure guarantees that the ratio of $P(x)$ and $D(y,z)$ is within a $1 + \varepsilon$ factor, so that the ε-optimality of the algorithm can be proved via LP duality theory. The whole update procedure of the dual variables is shown as follows

$$\begin{aligned} & y(p) = y(p) + \eta \\ & \textbf{for } j \notin Q(p) \textbf{ do} \\ & \quad z(j) = z(j) + \eta A(p,j) \\ & \textbf{end for} \end{aligned} \tag{7}$$

The above segment of instructions could be put between line (12) and line (13) of Alg. 1. Since they are not necessary in finding the optimal solution to the Min-Fill problem, they are omitted in the implementation.

If $Q(p) = \emptyset$ during the iteration, the algorithm will get stuck. However, it also means that the primal problem is infeasible. That is because if the problem is feasible, then $As \geq b$. Thus in each iteration the following inequality will hold

$$\begin{aligned} b(p)\alpha &> \textstyle\sum_{j=1}^{m} A(p,j)x(j) \geq \sum_{j \notin Q(p)} A(p,j)x(j) \\ &\geq \alpha \textstyle\sum_{j \notin Q(p)} s(j)A(p,j), \end{aligned} \tag{8}$$

where m is the number of tiles, which equals to $(\frac{nr}{w})^2$. Therefore, we have

$$b(p) > \sum_{j \notin Q(p)} A(p,j)s(j),$$

which implies that the problem is infeasible if $Q(p) = \emptyset$.

In practice, if we find $Q(p) = \emptyset$ in the execution, we know that the density lower bound L cannot be achieved even when all the available area in window p is filled. We can thus fill all the tiles in window p, and reduce the problem size by resetting all the other bounds as $b(i) - \sum_{j:A(p,j)>0} A(i,j)s(j)$.

Theorem 3: The FPTAS for Dummy Fill in Alg. 1 finds a $(1+\omega)$-approximation solution for the Min-Fill CLP in $O(\varepsilon^{-2} m \log(mC))$ iterations by choosing

$$\begin{aligned} \delta &= ((C(1+\varepsilon))^{1-\varepsilon} m)^{-1/\varepsilon} \\ \varepsilon &< \min(0.15,\ \omega/4) \\ \theta &= 1 \end{aligned}$$

where

$$C = \frac{||c||_\infty}{\min_{j:c(j)>0} c(j)}$$

and $m = (\frac{nr}{w})^2$ is the number of tiles.

The detailed proof of the above theorem can be found in [5], [4].

IV. A New Greedy Iterative Approach

In this section, a new greedy iterative method will be developed based on the approximation algorithm in the previous section. The new heuristic method builds on the concept of window density increase of the approximation algorithm. The main difference of two algorithms is that the greedy method abandons the α-phases procedure, making it more efficient and simpler to implement. The greedy iterative method is described in Alg. 2.

Similar to the approximation algorithm, in each iteration, it picks the most underfilled window, and inserts dummy features into the tiles in that window. In more detail, the method maintains three sets T, W and $Q(p)$. A tile will be put in T if and only if either it belongs to a window which has already achieved the density upper bound U, or all its available area has been filled. In each iteration, the algorithm only increases the density of tiles in $Q(p)$, which collects all the tiles of window p that does not belong to set T. Set W is used to ensure the method contains no infinite loop. Initially, all windows are collected in W. If $Q(p) = \emptyset$ during the iteration, which means the density of the window p cannot be improved any more, then window p will be removed from set W.

If the user decides to apply a pre-defined dummy fill pattern to the layout, the final CLP solution may need to be rounded so that an integer number of dummy fills can be filled. The new greedy method can easily handle this rounding issue by applying randomized rounding [12] during each iteration, and its final solution is still close to the optimal. Note that in the implementation of the method, the rounding step should ensure that at least one single fill pattern be filled.

Compared with the previous Monte Carlo and Greedy methods [2], which either insert a single filling geometry or insert the maximal possible amount during each iteration, the new greedy method uses the weighed price concept of the approximation algorithm to decide the actual filled amount in each tile, making it closer to optimal, since the dynamic information of the layout is considered and used in each insertion, instead of a static and brute-force insertion.

Algorithm 2 The New Greedy Iterative Method for Dummy Fill

Input: window matrix A, density bounds L and U, cost factor c, tile bound $slack$, precision ε
Output: filling amount x

Initialize $x = 0$, $T = \emptyset$, put all windows in W
find window p with the minimum density in W
while $\rho_w(p) < L$ **do**
  if tile $j \in$ window p and tile $j \notin T$ **then**
    put j in set $Q(p)$
  end if
  if $Q(p) = \emptyset$ **then**
    remove p from W
  else
    compute min weighted density price η among $Q(p)$
    for $j \in Q(p)$ **do**
      $\gamma = \eta / price(j,p)$
      δ = rounding($\varepsilon \gamma x(j)$)
      $x(j) = x(j) + \delta$
      if $x(j) \geq slack(j)$ **then**
        put the tile j in set T
      end if
    end for
  end if
  for $i \in \{1, \ldots, m\}$ **do**
    if $\rho_w(i) \geq U$ **then**
      put all the tiles in window i in set T
    end if
  end for
  find the window p with the minimum density in W
end while
return x

V. Experimental Results

We have implemented both the fast approximation scheme and the greedy algorithm in C++. All simulations given in this section were performed on a Unix workstation with 3.0 GHz processor and 2 GB memory. For comparison, we use the GNU Linear Programming Kit as our standard LP solver.

To test the performance of the proposed methods, we use both randomly generated layout cases and a real design case. For the randomly generated layout, similar to the procedure described in [14], we assume the maximal signal metal density is 50% in each tile, and use a random number uniformly distributed between 0 and 0.5 to represent the original tile density before filling. Further we assume that the dummy fills are square blocks separated by required spacing, and the maximal dummy fill density is also assumed to be 50%. So the density slack in each tile can be calculated as

$$\textit{slack(i)} = 0.5 \times \max(0, 1 - x_0(i) - \textit{slack\_reduction}). \quad (9)$$

As discussed before, when calculating the slack areas, the foundry design rules or other considerations such as the coupling capacitance constrains [15] have to be satisfied. This makes the allowable areas smaller than the empty areas. We model this by using a parameter *slack_reduction* that reduces the allowable area density. In the following experiments, *slack_reduction* is a random number between 0 and 0.2.

Throughout the experiments, for minimizing the total amount of dummy fill, we set each element of the cost factor c to 1. We also adopt the filter function used in [13] to calculate the density of each window:

$$f(x,y) = c_0 exp(c_1(x^2 + y^2)^{c_2}), \quad (10)$$

where $c_0 = 0.1$, $c_1 = -0.1$ and $c_2 = 1$.

A. Approximation Precision

In order to test the approximation precision of the approximation algorithm with respect to the parameter ε, we compare the solution of our algorithm with the solution of the LP method. In the test case, we assume the layout size $n = 800\mu m$, the window size $w = 100\mu m$, the step number $r = 5$, and the lower and upper density bounds are 50% and 80%, respectively. We solve the fractional CLP problem to get a fair comparison with the optimal LP solution. Here, two approximation ratios are defined for comparison. The first approximation ratio is used to compare the total inserted amount of dummy fill of these two methods, and it is defined as:

$$\textit{fill\_ratio} = \textit{abs}(\textit{CLP\_fills} - \textit{LP\_fills})/\textit{LP\_fills}, \qquad (11)$$

where *abs()* is the absolute value function, and *CLP_fills* and *LP_fills* represent the obtained total inserted amount of the CLP and LP approaches, respectively. The second one is used to compare the window density distribution, and is defined as:

$$\textit{std\_ratio} = \textit{abs}(\textit{CLP\_std} - \textit{LP\_std})/\textit{noFill\_std}, \qquad (12)$$

where *CLP_std*, *LP_std* and *noFill_std* represent the standard deviations of the window density distributions of the CLP solution, the LP solution, and the original layout, respectively.

Fig. 2 shows these approximation ratios with respect to the parameter ε for the test case. It is clear that the accuracy of the

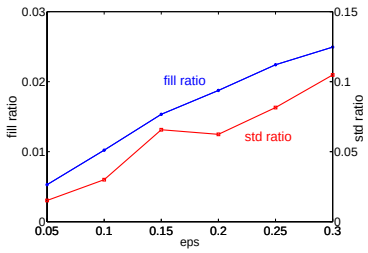


Fig. 2. The approximation ratios for different values of ε.

approximation algorithm is very high. For the total inserted amount, when $\varepsilon = 0.3$, the fill ratio is only 2.5%. As for the window density distribution, it is important to note that smaller ε leads to smaller window density variation, and the solution of our method is very close to the optimal LP solution when ε is small.

Fig. 3 illustrates the ratio of the primal solution $P(x)$ and the dual solution $D(y, z)$ of our approximation algorithm for the test case. The approximation bound $1 + 4\varepsilon$ is also drawn for comparison. It is important to note that the ratio of the primal and dual solutions is theoretically guaranteed to be no greater than $1 + 4\varepsilon$. In practice, it is much smaller as is shown in Fig. 3. This clearly demonstrates the effectiveness of our approximation algorithm.

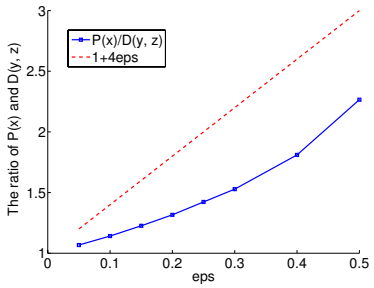


Fig. 3. The ratio of the primal and dual solutions.

B. Scalability

The fast speed is the significant advantage of our approximation method over the standard LP method. In the test cases, we keep w and r fixed as $100\mu m$ and 5, while the layout size n changes from $600\mu m$ to $1.6mm$. The CPU time comparison of these two methods is shown in Fig. 4. The horizontal coordinate is the number of unknowns, which is equal to $(\frac{nr}{w})^2$. The values of ε used in this experiment are 0.1, 0.2 and 0.3. The scalability of our approximation algorithm is clearly demonstrated in Fig. 4.

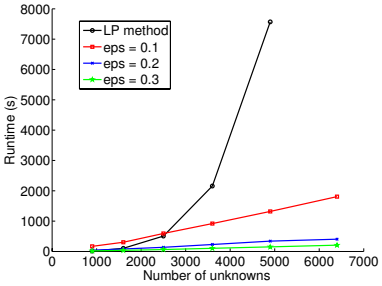


Fig. 4. The runtime comparison of our approximation method with LP method.

C. Rounding Issue

In this section, we use pre-defined squared dummy patterns to test the performance of our methods. The size of the dummy blocks used in this experiment is $0.5\mu m \times 0.5\mu m$.

The first experiment in this section is intended to make a comparison between the new FPTAS algorithm and the new greedy algorithm. We keep w and r fixed as $100\mu m$ and 5, while the layout size n changes from $200\mu m$ to $1000\mu m$. Speedup of the greedy method is computed by comparing to the runtime of the FPTAS algorithm, which is illustrated in Fig. 5. The values of ε used in this experiment are 0.1, 0.25 and 0.5. It can be seen that the greedy algorithm is more efficient than the approximation algorithm in practice, especially with smaller ε. Furthermore, there are only slight solution degradations in both total inserted amount and window density distribution for the greedy method compared to the FPTAS algorithm. The approximation ratios (*fill_ratio* and *std_ratio*) are no more than 5% for the test cases.

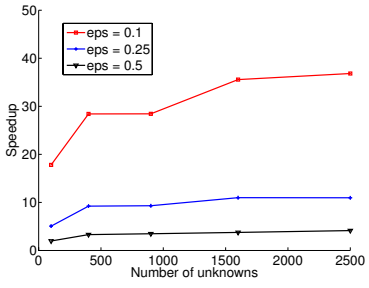


Fig. 5. The speedup of the greedy algorithm compared to the approximation algorithm.

In the second experiment, a comparison of our greedy algorithm with our implementation of the Monte-Carlo method [1] is conducted and reported in Table I. "MaxDens" is the maximum density of all the windows. "#Dummy" shows the number of inserted dummy fills. In the Monte-Carlo method, the priority of a tile p is chosen to be proportional to $U - \textit{MaxDens(p)}$, where *MaxDens(p)* is the maximum density over windows containing the tile p. In each iteration of the

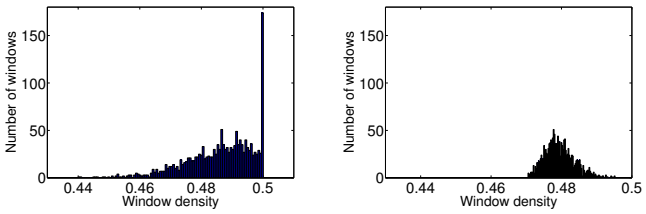
TABLE I
COMPARISON OF OUR GREEDY METHOD WITH MONTE-CARLO APPROACH

	Monte-Carlo				Greedy			
Test case (n/w/r)	MaxDens	MinDens	#Dummy	Time(s)	MaxDens	MinDens	#Dummy	Time(s)
800/200/4	0.5000	0.4434	951186	44.01	0.4946	0.4720	950365	0.71
1000/100/5	0.5000	0.4437	969389	154.82	0.4954	0.4719	947426	15.32
800/100/5	0.5000	0.4388	594805	79.37	0.4946	0.4701	574469	9.45
600/100/5	0.5000	0.4536	341082	39.77	0.4992	0.4773	337867	5.23
400/50/5	0.5000	0.4399	149358	20.10	0.4952	0.4705	145084	6.38
Average	0.5000	0.4439	601164	67.62	0.4958	0.4724	591040	7.42

Monte-Carlo method, a single filling geometry is inserted into a tile. The approximation parameter ε used in the greedy method is 0.1.

It should be noted that there are certain differences between the objectives of our methods and the Monte-Carlo method [1]. In the Monte-Carlo method, the window density upper bound is the input of the algorithm, and it tries to maximize the window density lower bound. Our methods, however, with the given density lower bounds, try to minimize the total number of dummy fills. In order to make a fair comparison, we use the medium value of the window density solution of the Monte-Carlo method as the density lower bound input to our greedy method.

We can observe that our greedy method performs better than the Monte-Carlo method under all known metrics. Fig. 6 shows the window density solutions of our greedy method and the Monte-Carlo method for one of the test cases. Our method achieved roughly 50% reduction on the density variation.



(a) The solution of the Monte-Carlo method. (b) The solution of the greedy method.

Fig. 6. Comparison of the window density distributions.

D. Test on Real Layout Design

We also perform experiments on a real design case. The layout size is $854\mu m \times 243\mu m$, and it has 202238 rectangles. We duplicate the layout 3 times, and assume the window size is $100\mu m$ and step size $r = 5$. So there are totally 1849 variables. It takes the LP solver 375 seconds to get the optimal solution, while our approximation algorithm with $\varepsilon = 0.25$ only takes 3 seconds. The standard deviation of the window density distribution of the original layout is 0.0567. After filling by our approximation algorithm, it is reduced to 0.0065.

VI. CONCLUSIONS

In this paper, we present the first covering linear program formulation for the dummy fill problem with Min-Fill objective, and propose a probably good and efficient algorithm based on the recent fast approximation scheme [4]. The efficiency and the scalability of the algorithm are clearly demonstrated in the experimental results. Moreover, based on the approximation algorithm, a new greedy iterative method is proposed. Simulation results show that our greedy method outperforms the Monte-Carlo method [1] both in performance and runtime. We are currently developing multicore parallel dummy fill algorithms based on the approximation scheme.

ACKNOWLEDGMENTS

This research is supported partially by NSFC research project 60676018 and 60806013, China National Basic Research Program under the grant 2005CB321701, China National Major Science and Technology special project 2008ZX01035-001-06 during the 11th five-year plan period, the doctoral program foundation of Ministry of Education of China under 200802460068, the International Science and Technology Cooperation program foundation of Shanghai under 08510700100, the program for Outstanding Academic Leader of Shanghai, and NSF under CCF-0238484 and CCF-0811270.

REFERENCES

[1] Y. Chen, A. B. Kahng, G. Robins, and A. Zelikovsky. Monte-Carlo algorithms for layout density control. In *Proceedings of ASP-DAC*, pages 523–528, 2000.

[2] Y. Chen, A. B. Kahng, G. Robins, and A. Zelikovsky. Practical iterated fill synthesis for CMP uniformity. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 671–674, 2000.

[3] Y. Chen, A. B. Kahng, G. Robins, and A. Zelikovsky. Area fill synthesis for uniform layout density. *IEEE Trans. on CAD*, 21(10):1132–1147, 2002.

[4] L. Fleischer. A fast approximation scheme for fractional covering problems with variable upper bounds. In *Proceedings of ACM-SIAM symposium on Discrete algorithms*, pages 1001–1010, 2004.

[5] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractinal packing problems. In *Proceedings of IEEE Symposium on Foundations of Computer Science*, pages 300–309, 1998.

[6] T. E. Gbondo-Tugbawa. *Chip-Scale Modeling of Pattern Dependencies in Copper Chemical Mechanical Polishing Processes*. PhD thesis, Massachusetts Institute of Technology, 2002.

[7] A. Kahng, G. Robins, A. Singh, and A. Zelikovsky. Filling algorithms and analyses for layout density control. *IEEE Trans. on CAD*, 18(4):445–462, 1999.

[8] A. B. Kahng and K. Samadi. CMP fill synthesis: A survey of recent studies. *IEEE Trans. on CAD*, 27(1):3–19, 2008.

[9] S. Lakshminarayanan, P. J. Wright, and J. Pallinti. Electrical characterization of the copper CMP process and derivation of metal layout rules. *IEEE Trans. on Semiconductor Manufacturing*, 16(4):668–676, 2003.

[10] M. Mukherjee and K. Chakraborty. A randomized greedy method for rectangular-pattern fill problems. *IEEE Trans. on CAD*, 27(8):1376–1384, 2008.

[11] D. O. Ouma. *Modeling of Chemical Mechanical Polishing for Dielectric Planarization*. PhD thesis, Massachusetts Institute of Technology, 1998.

[12] P. Raghavan and C. D. Thompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1978.

[13] R. Tian, D. F. Wong, and R. Boone. Model-based dummy feature placement for oxide chemical mechanical polishing manufacturability. In *Proceedings of ACM/IEEE Design Automation Conference*, pages 667–670, 2000.

[14] X. Wang, C. C. Chiang, J. Kawa, and Q. Su. A min-variance iterative method for fast smart dummy feature density assignment in chemical-mechanical polishing. In *Proceedings of ISQED*, pages 258–263, 2005.

[15] H. Xiang, L. Deng, R. Puri, K.-Y. Chao, and M. D. F. Wong. Fast dummy-fill density analysis with coupling constraints. *IEEE Trans. on CAD*, 27(4):633–642, 2008.