Cell Density-driven Detailed Placement with Displacement Constraint

Wing-Kai Chow, Jian Kuang, Xu He, Wenzan Cai, Evangeline F. Y. Young
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
{wkchow,jkuang,xhe,wzcai,fyyoung}@cse.cuhk.edu.hk

ABSTRACT

Modern placement process involves global placement, legalization, and detailed placement. Global placement produce a placement solution with minimized target objective, which is usually wire-length, routability, timing, etc. Legalization removes cell overlap and aligns the cells to the placement sites. Detailed placement further improves the solution by relocating cells. Since target objectives like wire-length and timing are optimized in global placement, legalization and detailed placement should not only minimize their own objectives but also preserve the global placement solution. In this paper, we propose a detailed placement algorithm for minimizing wire-length, while preserving the global placement solution by cell displacement constraint and target cell density objective. Our detailed placer involves two steps: *Global Move* that allocates each cell into a bin/region that minimizes wire-length, while not overflowing the target cell density. *Local Move* that finely adjust the cell locations in local regions to further minimize the wire-length objective. With large-scale benchmarks from ICCAD 2013 detailed placement contest [7], the results show that our detailed placer, *RippleDP*, can improve the global placement results by 13.38% – 16.41% on average under displacement constraint and target placement density objective.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids — Placement and routing

Keywords

Detailed placement, legalization, displacement constraint, cell density

1. INTRODUCTION

Modern placement process involves global placement, legalization and detailed placement. Global placement produces a placement solution with minimized target objective, which is usually wire-length, routability, timing, etc. Legalization removes cell overlap and aligns the cells to the placement sites. Detailed placement further improves the solution by relocating cells.

Traditionally, the major objective of placement is to minimize the total wire-length. However, as VLSI technology advances, wire-length is no longer the only objective. Modern placement is usually an optimization problem with multiple objectives, including wire-length, routability, timing, and power, etc.. One of the hard constraint of the placement problem is non-overlapping and placement site alignment. However, satisfying such constraint during the global placement stage is very complicated. Therefore, legalization is usually applied to the global placement solution to satisfy these constraints. Detailed placement is then applied as an independent step to further optimize the design objectives like wire-length.

In recent decade, many works have focused on the global placement problem. Different approaches have been proposed, including analytical placement [3][5], force-directed placement [13], partitioning-based approach [1] and simulated annealing-based approaches [14]. After global placement, we can assume that the global placer has already distributed the cells over the placement region well, such that the target objective is minimized. Therefore, the following legalization process should target at legalizing the solution with minimum perturbation, i.e., to minimize the cell displacement from the global placement solution.

In contrast to global placement, there are much less work in detailed placement. Although almost all placers have deploy detailed placement step, many of them share common detailed placement technique and even the same independent detailed placer. mPL6 [3], FengShui [1], NTUPlace [5] shares the same techniques of sliding window-based cell swapping; FastPlace [15], SimPL [8], MAPLE [9], Ripple [4] directly use FastPlace-DP [12] as the detailed placer.

Recently, Li et al. [10] proposed a mixed integer programming models to optimize detailed placement. Although it can achieve high quality, the run-time is too long and it is not flexible for other additional objectives. Liu et al.[11] proposed an routability-driven placement optimizer that can improve the placement solution in terms of wire-length and routability.

However, the previous works on detailed placement do not consider the impact of cell movement to the global placement solution. As most of the modern global placers target at multiple objectives, in detailed placement stage, the placer should either target at the same set of objectives or minimize

ISPD'14, March 30–April 2, 2014, Petaluma, CA, USA.

http://dx.doi.org/10.1145/2560519.2560523.

the perturbation to the solution. Since integrating different objectives into the current detailed placer is difficult, we believe that a detailed placer that have minimal impact to the global placement solution can be more flexible and thus more useful to modern placement problem.

In this paper, we proposed a novel way of detailed placement, that can minimize wire-length while maintaining a specified target placement density, under constraint of limited perturbation to global placement solution.

The main contributions of our work includes:

- A legality-maintaining framework that keeps the solution legal throughout the detailed placement.
- A displacement-constrained *Global Move* to allocate each cell into a bin such that the total density-awared scaled wire-length is minimized.
- A displacement-constrained *Local Move* to finely adjust the cell positions to further minimize scaled wire-length.
- An extension to the algorithm in [12] that optimally solves *fixed order single segment placement problem*, to also consider placement density and maximum displacement constraint.

The rest of the paper is organized as follows: Section 2 gives the formulation of the detailed placement problem as well as the objective. Section 3 provides a detailed description of our approach to solve the problem. Section 4 shows the experimental results and Section 5 concludes the paper.

2. PROBLEM FORMULATION

Beginning with a legalized placement with all macros fixed, our detailed placement algorithm move the standard cell legally to minimize the total wire-length in terms of half perimeter wire-length (HPWL) defined as in (1), with consideration of user-specific placement density and maximum displacement constraint.

$$\text{HPWL} = \sum_{n \in N} (\max_{i \in n} x_i - \min_{i \in n} x_i + \max_{i \in n} y_i - \min_{i \in n} y_i) \quad (1)$$

2.1 Maximum Displacement

In most detailed placement algorithm, cell displacement is not constrained. However, unless the target objective in detailed placement is the same as that in global placement, detailed placement could ruin the optimized objective of global placement, such as timing and global routability. Therefore, detailed placement should perform local optimization instead of re-placing the cells globally. In our proposed algorithm, we consider displacement as a constraint, so that the resulting solution has similar cell distribution as in global placement, while the detailed placement objective is also optimized. Our placer formulate maximum displacement constraint D_{disp} as:

$$D_{disp} \geq \max_{c \in C}(|x_c - x_{c0}| + |y_c - y_{c0}|) \quad (2)$$

where C is the set of all movable cell, (x_c, y_c) is the current position of cell c , (x_{c0}, y_{c0}) is the original position of cell c in the given global placement solution.

2.2 Average Bin Utilization (ABU)

In [9], the author proposed the concept of *Average Bin Utilization (ABU)* as a metric to evaluate placement density. ABU_γ is defined as the average area utilization ratio of the top $\gamma\%$ bins of highest area utilization ratio. Following the metric of the ICCAD 2013 placement contest, overflow is defined as (3). Our placer formulates the overall placement density overflow as a weighted sum of $overflow_2$, $overflow_5$, $overflow_{10}$, $overflow_{20}$, and it is considered as a scale factor in evaluation of the wire-length. The objective of our placer is to minimize the scaled half perimeter wire-length (sHPWL) as defined in (5).

$$overflow_\gamma = \max(0, \frac{ABU_\gamma}{density_{target}} - 1) \quad (3)$$

$$scale = \frac{\sum_{\gamma \in \Gamma} w_\gamma overflow_\gamma}{\sum_{\gamma \in \Gamma} w_\gamma}, \Gamma = \{2, 5, 10, 20\} \quad (4)$$

$$\text{sHPWL} = (1 + scale) \cdot \text{HPWL} \quad (5)$$

In our implementation, we use $w_2 = 10, w_5 = 4, w_{10} = 2, w_{20} = 1$. Unlike other placers that formulate cell density as sum of area utilization overflow, we give higher weight to the bins with higher area utilization ratio. Therefore, for two solutions with the same value of total utilization overflow, the one with higher peak value results in higher scaling factor in our metric.

In summary, our placer formulate the detailed placement problem as follow:

$$\begin{array}{ll} min & \text{sHPWL} \\ s.t. & D_{disp} \geq \max_{c \in C}(|x_c - x_{c0}| + |y_c - y_{c0}|) \end{array} \quad (6)$$

3. OUR PROPOSED ALGORITHM

3.1 Overview

Algorithm 1 shows the flow of our placer. We first divide the whole placement region into regular rectangular bins. The resulting bins are the basic unit in evaluating placement density. In the first stage, *Global Move*, we try to move each cell to another bin to reduce the scaled half perimeter wire-length (sHPWL), while the bin should be within the range of maximum displacement constraint. We don't care the precise position in a bin at this stage and will randomly place the cells in the selected bins. The scaled wire-length will monotonically reduce throughout the process. We iteratively repeat Global Move until the improvement of an iteration is below a threshold.

In the second stage, *LocalMove*, we locally move the cells to the nearby placement sites. This stage composes of three sub-steps: *Vertical Move*, *Reordering*, and *Compaction*. Vertical Move targets at reducing the sHPWL, and it moves cells vertically within several rows above or below. Reordering also targets at reducing the sHPWL, and it changes the cell order within a few neighbouring cells horizontally. Compaction targets at reducing the HPWL without increasing the placement density overflow, and it adjusts cell positions horizontally without changing the cell orders. We iteratively repeat the three steps until the improvement of an iteration is below a certain threshold.

Before we go into details of our algorithm, we need to introduce a key operation in our algorithm called *Legalized Cell Move*.

Algorithm 1 Detailed Placement Algorithm

Detailed Placement

Input: Legal global placement solution
Output: Legal detailed placement solution
Let C be the set of all movable cells
Partition placement region into rectangular bins
repeat
  GlobalMove(C)
until $improvement < threshold_g$
repeat
  VerticalMove(C)
  Reorder(C)
  Compact(C)
until $improvement < threshold_l$

3.1.1 Legalized Cell Move

In many detailed placers, cell move is simply moving a cell to a position aligned to placement site, but the site may already be occupied by another cells and cell move can cause overlapping. They usually deploy an explicit legalization step to the resulting solution to remove overlaps by spreading the cells out. However, we found that such approach is harmful because of the following reasons: 1) Legalization usually increases wire-length; 2) Spreading can harm placement density; 3) It can bring a cell further away from its original location in the global placement solution, and violate the maximum displacement constraint. To resolve these issues, we proposed a novel technique of Legalized Cell Move. The idea is to keep our intermediate solution legal after every cell movement. To facilitate such requirement, we have an efficient "legalizability" check to ensure that a specific cell move can be leagalized with an affordable impact to the result, and also gives a corresponding legalization to commit the cell move. Therefore, cell move in our algorithm may involve movement of multiple cells instead of moving just one single cell.

The concept of "legalizability" check is usually implicitly considered in other placers by calculating the whitespace remaining in the placement row and the amount of local whitespace near the target location. In contrast, our algorithm will capture the legalization impact more accurately and yet efficiently. The legalization impact is evaluated in our algorithm as the sum of cell displacement caused. When the sum of cell displacement caused by a cell move exceeds the limit, we consider the cell move is not "legalizable", and the cell move is discarded. Legalized Cell Move is used in Global Move and the Vertical Move step in Local Move, while the other steps do not need legalization as their results are always legal. The algorithm of Legalized Cell Move is shown in algorithm 2. To check if a cell move is legalizable only (without really moving the cells), we have another function named *IsLegalizable*, which is similar to *LegalizedCellMove* but without any actual cell move.

Algorithm 2 Legalized Cell Move

LegalizedMove

Input: Cell c_i, location (x, y) and width w_i of c_i
Output: True if the legalization is successful, False otherwise
Find the first cell c_l at the left of (x, y)
Find the first cell c_r at the right of (x, y)
Let c_l^L and c_l^R be the left and right boundary of cell c_l
Let c_r^L and c_r^R be the left and right boundary of cell c_r
Move cell c_i to (x, y)
$dist \leftarrow 0$
$L \leftarrow x - w_i/2$
while $c_l \neq \varnothing$ and $c_l^R > L$ **do**
  Move c_l leftward with distance $c_l^R - L$
  $dist \leftarrow dist + c_l^R - L$
  $L \leftarrow L - w_l$
  $c_l \leftarrow$the first cell at the left of c_l
end while
$R \leftarrow x + w_i/2$
while $c_r \neq \varnothing$ and $c_r^L < R$ **do**
  Move c_r rightward with distance $R - c_r^L$
  $dist \leftarrow dist + R - c_r^L$
  $R \leftarrow R + w_r$
  $c_r \leftarrow$the first cell at the right of c_r
end while
if $dist >$impact limit OR any cell violate the maximum displacement constraint OR any cell is at illegal position **then**
  Restore all cell positions
  return False
end if
return True

3.2 Global Move

Before the Global Move step, the whole placement region is partitioned into regular rectangular bins. In Global Move, we aim at moving each cell into the best bin that can minimize the sHPWL under the constraint of maximum cell displacement.

We can estimate the resulting scaled wire-length for each cell movement by assuming that all other cells are fixed. The objective at this stage is to place every cell into the bin with the lowest cost. Instead of testing every bin to make the decision of movement, we select a set of candidate bins. The bin selection is based on two regions: "optimal region" and "max-displacement region". *Optimal region* is proposed by [12]. *Optimal region* of a cell c is defined as the region bounded by the median of the x- and y-coordinates of c's associated net's bounding boxes. *max-displacement region* is defined as the movable range of each cell under the maximum displacement constraint. The bins overlapped with both regions are selected as candidate bins. It is possible that the two regions do not overlap, especially when the maximum displacement constraint is tight. In such a case, we will consider the eight neighbouring bins of the bin containing the cell as candidate bins.

In this stage, the precise cell location is not important. For each candidate bins selected, we pick n random placement sites in the bin that can accommodate the cell without violating the displacement constraint ($n = 4$ in our implementation). We calculate the cost when the cell is moved into the bin, assuming that all other cells are fixed. The cost s_c^b of moving cell c into a candidate bin b is calculated with the following formula:

$$s_c^b = \text{sHPWL}_c^b + w \cdot \text{density}_b \tag{7}$$

where sHPWL_c^b is the resulting scaled wire-length when cell c is moved to the random location in bin b. w is the weight

of bin density, density_b is the area utility of bin b after the cell movement.

The cost is a weighted sum of two parts. The first part is the sHPWL which reflect the resulting solution quality. The second part is the bin-density. The weight w in the second part is very small, such that the first term has much higher priority. However, the effect of the second term is still significant. It is because when there are several candidate bins that can produce similar sHPWL, the second term helps to evenly distribute the cell among the bins. It reduces the number of bins with critical cell density, which can cause overflow easily with a slight increase in bin utilization.

For each iteration, the above procedure is applied to each movable cell and it is repeated until the improvement in scaled wire-length is less than a threshold. In our implementation, the threshold is 0.2%. The overview of an iteration of Global Move is shown in algorithm 3.

Algorithm 3 Global Move

```
Global Move
Let C be the set of all movable cells
Let B be the set of all bins
for all c ∈ C do
   Find optimal region R_o of cell c
   Find the max-displacement region R_d of cell c
   B' ← ∅
   for all b ∈ B do
      if b overlap R_o AND b overlap R_d then
         Add b into B'
      end if
   end for
   if |B'| < threshold then
      Add neighbouring bins into B'
   end if
   s_best ←cost when cell c is at original position
   for all b ∈ B' do
      for j ← 1 to 4 do
         Randomly pick a placement site (x_j, y_j) in b_i
         Calculate cost s of moving cell c to (x_j, y_j)
         if s < s_best AND IsLegalizable(c, (x_j, y_j)) then
            s_best ← s
            (x_best, y_best) ← (x_j, y_j)
         end if
      end for
   end for
   if s_best <original cost then
      LegalizedMove(s_best, (x_best, y_best)
   end if
end for
```

3.3 Local Move

After Global Move, all movable cells are allocated to appropriate bins that can minimize total scaled wire-length. However, since the position of cells are randomly chosen, there still have room to finely adjust the position in order to further minimize the wire-length. Our Local Move stage consists of three sub-steps: Vertical Move, Local Reordering and Compaction.

3.3.1 Vertical Move

The objective of Vertical Move is to reduce the sHPWL. For each cell, we try to evaluate the resulting sHPWL when the cell is vertically shifted and aligned to the several nearby rows. Then we choose to move the cell into the row with the maximum improvement in sHPWL, while the maximum displacement constraint is satisfied. The cells are also moved with Legalized Cell Move proposed in section 3.1.1.

In contrast to a similar operation *Vertical Swap* in [12], we fix the x-coordinate of the cell being moved, instead of searching for different positions in another row. This ensure that the improvement in sHPWL is produced by the decision of allocating the cell in that row, instead of by cell shifting in horizontally. Figure 1 illustrates such a a situation. In figure 1(a), shifting the cell in the x-direction produces an incorrect information that row $i-1$ is closer to the optimal location. In figure 1(b), fixing the x-coordinate when evaluating the direction of vertical movement can correctly identify the better row. Also, if we try a few more rows, we will not be blocked by wide cells or congested rows.

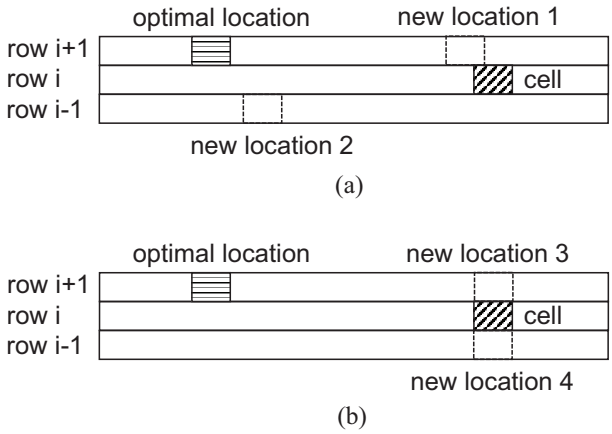

Figure 1: Example of incorrect row selection due to cell shifting.

Since Vertical Move only move vertically to the nearby rows, the sHPWL improvement is usually quite small. Therefore, when we apply Legalized Cell Move in this step, the impact of legalization may even be larger than the improvement. We could reduce the cell displacement limit, i.e., the *impact limit* in algorithm 2, to reduce this effect. To handle this issue robustly, we dynamically set the total cell displacement limit in Legalized Cell Move for Vertical Move, such that larger improvement in sHPWL of a cell move can afford larger legalization impact, i.e., impact limit. In our implementation, the legalization impact limit is set to be $1.2 \cdot \Delta sHPWL$ where $\Delta sHPWL$ is the improvement in scaled wire-length.

3.3.2 Local Reordering

After Vertical Move, the cells are located in a better row than before. The next two sub-steps horizontally shift the cells to further minimize wire-length. Many detailed placement algorithms use an approach of examining all permutation in a window to minimize wire-length. The idea of local reordering proposed in [12] is similar but it is restricted to all permutation of ordering three consecutive cells within the same placement segment. We use the similar approach: For every set of three consecutive cells, we try all six possi-

ble permutations of the cell order to find out the best order with minimum sHPWL.

3.3.3 Compaction

In Compaction, we try to move cell to optimal position while maintaining the cell order in a row. The problem is defined in [12] as *fixed order single segment placement problem* and it is optimally solved. However, as we have to consider placement density and the maximum displacement constraint for each cell, the original algorithm does not apply. In this paper, we extend the algorithm to consider maximum displacement and bin density.

The algorithm proposed in [12], [2] and [6] base on the fact that the HPWL wire-length function of the x-coordinate of a cell is a convex function. However, the sHWPL function of the x-coordinate of a cell is not convex due to the formulation of the overflow factor. Therefore, the objective in Compaction is different from that of the previous sub-steps. It now minimizes wire-length in HPWL instead of sHPWL, while the bin density overflow is maintained independently.

For each cell c in a segment s, we find two sets of values: *optimal region range* and *displacement range*. *Optimal region range* is found with the method proposed in [12], which represents the bounds on the x-coordinate that will result in the minimum wire-length. *Displacement range* is the bounds on the x-coordinate of a cell that is within the maximum displacement constraint within the segment. For all bins overlapping with the segment, we identify the set of critical bins that will have cell density overflow when the whole target segment is fully occupied by cells. The bins with such property are referred as *critical bins*, which means that moving cells into these bins may worsen the placement density. Therefore, moving cells into these bins should be restricted. Although this will constrain the wire-length improvement, it is necessary to guarantee that the primary objective of sHPWL does not get worsen.

Since we should not move cells into the critical bins, the actual movable range of a cell is not simply the displacement range. The next step is to find out the *actual movement range* of a cell c. It is found by removing the parts of the displacement range that overlap with the critical bins. However, moving a cell within a fragmented movable range is running time expensive, we will only keep the part that overlaps with the cell's x-coordinate, while other disjointed parts will be discarded. With the fact that the density overflow is usually very low in the Local Move stage, and we have tried to minimize the number of critical bins in the GlobalMove stage by equation 7, the quality loss due to this is not large. The procedure is visualized in figure 2. In the figure, the cell can move within the x-coordinate range of $range_{disp}$ without violating the maximum displacement constraint. bin_2 and bin_4 are identified as critical bins and the two bins span in range of $range_{cbin2}$ and $range_{cbin4}$. The subtraction of these from the displacement range results in two discontinuous parts (red and blue) and only the part overlapping with the cell's x-coordinate remain (red). This results in the actual movement range of the cell shown in the figure as $range_{actual}$.

The pseudo-code of the Compaction step is shown in algorithm 4. In Compaction, we shift every cell horizontally to its optimal location, i.e., a position in its optimal region range within the actual movement range. Placing a cell at any location within optimal region range produces the same

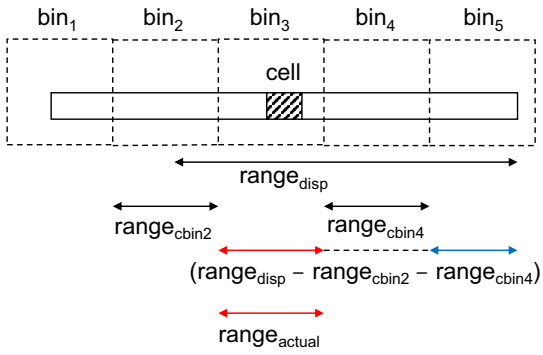

Figure 2: Example of actual movement range computation

optimal wire-length. In our implementation, we try to place the cell at the center of the optimal region. When the optimal region range and actual movement range do not overlap, we will shift the cell to the left or right end of its actual movement range that is closer to its optimal region. The resulting cell positions may overlap. If overlapping occurs, we will merge the overlapping cells to form a new supercell by abutting the cells in the original order. Then we will update the optimal region range of this super cell by considering all the pins on the cells within it. The actual movement range of this supercell is also updated based on the actual movement range of each individual cell inside. The above process is repeated until no new supercell or no cell overlap occurs.

When the segment has no critical overlapping bin, we can achieve the optimal solution with the given order of cells and the given maximum displacement constraint. The proof is shown as follow:

Theorem *Given a fixed order of cells, and given the maximum displacement constraint, the Compaction algorithm gives the optimal solution of minimum HPWL.*

Proof. Our algorithm is similar to the approach in [6] and [12]. The only difference is the existence of movement range coming from the maximum displacement constraint that limits the movement of a cell. Consider the actual movement range R_x of a cell (or cluster formed by merging 2 or more cells) x. If R_x has overlap with the optimal region range Q_x of x, we can simply put x within the overlapping region $R_x \cap Q_x$ and the HPWL w.r.t. cell x is optimal. Otherwise, since the wire-length function of x w.r.t. its x-coordinates (assumming center) is a convex function and we should put x within its movement range R_x closest to its optimal position Q_x to minimize HPWL. In this way, the HPWL w.r.t. cell x is also optimal. The correctness and optimality of the overall flow follow from [6] and [12]. □

4. EXPERIMENTAL RESULTS

To validate our proposed algorithm, the algorithm is implemented in C++. The experiments were performed on a 64-bit Linux Server with Intel Xeon 3.4GHz CPU and 32GB memory, using the detailed placement benchmarks provided by ICCAD13 detailed placement contest [7]. Each benchmark includes a placement problem, a legalized global place-

Algorithm 4 Compaction

```
Compaction
for all s ∈ S do
  Let C be the set of all cells within a segment s
  Let Cluster be the set of all cluster
  Cluster ← C
  repeat
    Find the set of critical bin range B that overlap with segment s
    for all c ∈ Cluster do
      Find optimal region range for Cluster c
      Find displacement range for Cluster c
      Find actual movable range for Cluster c by subtracting each critical bin range from the displacement range.
    end for
    for all c ∈ Cluster do
      Shift each cluster c to a position within its actual movable range closest to its optimal region.
    end for
    for i ← 1 to |Cluster| − 1 do
      if c_i overlap with c_{i+1} then
        Merge c_{i+1} into c_i by abutting the two clusters.
        Remove c_{i+1} from Cluster
      end if
    end for
  until all cluster and cells are not overlapping
end for
```

ment solution as input, a target placement density and two maximum displacement constraints. One of the displacement constraints is tight, ranging from 150 – 220, while the other one is more relaxed, ranging from 15,000 – 25,000. Table 1 shows the information of all benchmarks being used in the experiments. In the table, *#cell* and *#net* denote the total number of movable cells and nets in the circuits; *Target Density* denotes the target density; *Overflow* denotes the weighted sum of overflow in terms of ABU; *HPWL* and *sHPWL* denote the wire-length and scaled wire-length of the given global placement solution; D_1 and D_2 are the two maximum displacement constraints specified by the contest.

4.1 On Placement Density

Table 2 and 3 show the detailed placement results under two different maximum displacement constraints. In the table, D_{max} and D_{avg} denote the maximum and the average displacement of the solutions. By comparing the overflow values in Table 2, 3 with the overflow values of given input global placement solution from Table 1, we can see that our detailed placer can effectively reduce the placement density overflow to nearly zero, while HPWL is also significantly reduced by 6.27% on average even under the tight displacement constraint, and 9.49% under the relaxed displacement constraint. In terms of sHPWL, the reduction is 13.38% under the tight constraint and 16.41% under the relaxed constraint.

4.2 On Maximum Displacement Constraint

We obtained the source code of the detailed placer FastPlace-DP from the author of [12]. FastPlace-DP does not consider maximum displacement and placement density. Therefore, we do the minimal adjustment to FastPlace-DP to make it consider displacement constraint in *Global Swap*, *Vertical Swap*, and *Local Reordering*. Cell moves in these steps are permitted only when the cell movement can satisfy the displacement constraint or it can reduce the violation to the displacement constraint [1]. However, it is difficult to integrate such constraint in FastPlace-DP's *Clustering* and *Legalization* steps. To consider displacement in these two steps indirectly, we partition long placement segments into shorter ones with length at most double of the maximum displacement constraint. In this way, the maximum possible movement distance within a segment in these two steps will be reduced. However, it will still violate the maximum displacement constraint in their solutions.

We perform two experiments to compare RippleDP with FastPlace-DP. The first experiment is done without considering the placement density and displacement constraint, i.e., the traditional detailed placement formulation. In this test, FastPlace-DP works without any modification, while our placer works with setting of target density to 1.0 and maximum displacement of infinity. The result in table 4 shows that our detailed placer performs slightly better on average than FastPlace-DP with reasonable run-time.

The second experiment is done considering the displacement constraint. We found that no matter whether the maximum displacement is tight or relaxed, the modified FastPlace-DP still will violate the constraint for all benchmarks. In the experiment, the maximum displacement constraint is set as twice the tight maximum displacement constraint, so as to have less violations in the results of FastPlace-DP. Placement density is still not considered in this experiment. The experimental result is shown in Table 5. D_{cons} denotes the maximum displacement constraint for each benchmark. The modified version of FastPlace-DP can now consider maximum displacement constraint to a certain extent, although it still violates the constraint. FastPlace-DP can achieve 7.45% of improvement on average. In our detailed placer, larger improvement of 8.32% in wire-length is observed, while the cell displacement constraint is strictly satisfied.

5. CONCLUSIONS

In this paper, we proposed a detailed placer targeting at optimizing the multi-objective modern placement problem. Unlike previous detailed placers, we consider placement density and also minimizing the perturbation to the global placement solution by integrating the maximum displacement constraint. Our two-step approach detailed placer composes of two stages: Global Move moves cells to bins to minimize the scaled half perimeter wire-length under the maximum displacement constraint. The Local Move stage locally adjusts the cell positions to further optimize the scaled wire-length. Vertical Move vertically move cells to different rows which produces lower sHPWL. Local Reordering locally find the best cell order to minimize the sHPWL. Compaction shifts cells to achieve the best cell locations under the maximum displacement constraint while maintaining placement density. Our experiments on large-scale benchmarks show that our proposed algorithm can effec-

[1]Violation of the displacement constraint is still possible since the movement restriction step we added is hard to be implemented in some steps of FastPlace-DP

Table 1: Details of ICCAD 2013 detailed placement contest benchmarks.

Benchmarks	#Cell	#Net	Design Utility	Target Density	Overflow (before)	HPWL (before)	sHPWL (before)	D_1	D_2
superblue1	765102	822744	0.347	0.51	9.92	339099616	372736183	150	20000
superblue5	677416	786999	0.372	0.50	16.38	395464608	460241360	160	20000
superblue7	1271887	1340418	0.578	0.74	7.23	478403584	513005419	200	20000
superblue10	1045874	1158784	0.313	0.53	6.17	610708992	648403031	220	25000
superblue11	859771	935731	0.403	0.64	8.40	394250528	427363458	220	25000
superblue12	1278084	1293436	0.435	0.63	5.14	340540256	358032046	180	18000
superblue16	680450	697458	0.458	0.55	6.63	305835392	326101664	150	15000
superblue19	506097	511685	0.491	0.68	7.11	167200288	179086438	150	15000

Table 2: Detailed placement results with tight maximum displacement constraint. D_{max} and D_{avg} denote the maximum and average cell displacement from the global placement solution.

	D_{max}	D_{avg}	Overflow (after)	Δ Overflow	HPWL (after)	Δ HPWL	sHPWL (after)	Δ sHPWL	Runtime (sec)
superblue1	150	39	0.02	-99.84%	321339488	-5.24%	321388935	-13.78%	107.71
superblue5	160	47	0.66	-96.00%	377410080	-4.57%	379885066	-17.46%	125.26
superblue7	200	49	0.00	-100.00%	443692544	-7.26%	443692544	-13.51%	283.36
superblue10	220	50	0.00	-99.94%	579638464	-5.09%	579660007	-10.60%	176.43
superblue11	220	49	0.03	-99.61%	371626432	-5.74%	371746638	-13.01%	169.19
superblue12	180	46	0.00	-99.98%	306118368	-10.11%	306121044	-14.50%	324.18
superblue16	150	44	0.00	-100.00%	286653216	-6.27%	286653216	-12.10%	82.07
superblue19	150	37	0.03	-99.63%	157406768	-5.86%	157447729	-12.08%	118.99
Average				**-99.38%**		**-6.27%**		**-13.38%**	

tively optimize placement solution under different maximum displacement constraints and target densities.

6. REFERENCES

[1] A. Agnihotri, S. Ono, C. Li, M. Yildiz, A. Khatkhate, C.-K. Koh, and P. Madden. Mixed block placement via fractional cut recursive bisection. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(5):748–761, 2005.

[2] U. Brenner and J. Vygen. Faster optimal single-row placement with fixed ordering. In *Proceedings of the conference on Design, automation and test in Europe*, DATE '00, pages 117–121, New York, NY, USA, 2000. ACM.

[3] J. Cong and M. Xie. A robust mixed-size legalization and detailed placement algorithm. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(8):1349–1362, 2008.

[4] X. He, T. Huang, W.-K. Chow, J. Kuang, K.-C. Lam, W. Cai, and E. Young. Ripple 2.0: High quality routability-driven placement via global router integration. In *Design Automation Conference (DAC), 2013 50th ACM / EDAC / IEEE*, pages 1–6, 2013.

[5] M.-K. Hsu, Y.-F. Chen, C.-C. Huang, T.-C. Chen, and Y.-W. Chang. Routability-driven placement for hierarchical mixed-size circuit designs. In *Design Automation Conference (DAC), 2013 50th ACM / EDAC / IEEE*, pages 1–6, 2013.

[6] A. Kahng, P. Tucker, and A. Zelikovsky. Optimization of linear placements for wirelength minimization with free sites. In *Design Automation Conference, 1999. Proceedings of the ASP-DAC '99. Asia and South Pacific*, pages 241–244 vol.1, 1999.

[7] M.-C. Kim. IEEE CEDA / taiwan MOE, ICCAD 2013 contest. Retrieved October 10, 2013 from `http://cad_contest.cs.nctu.edu.tw/CAD-contest-at-ICCAD2013/problem_b`, 2013.

[8] M.-C. Kim, D. Lee, and I. L. Markov. Simpl: An effective placement algorithm. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 31(1):50–60, 2012.

[9] M.-C. Kim, N. Viswanathan, C. J. Alpert, I. L. Markov, and S. Ramji. Maple: multilevel adaptive placement for mixed-size designs. In *Proceedings of the 2012 ACM international symposium on International Symposium on Physical Design*, ISPD '12, pages 193–200, New York, NY, USA, 2012. ACM.

[10] S. Li and C.-K. Koh. Mixed integer programming models for detailed placement. In *Proceedings of the 2012 ACM international symposium on International Symposium on Physical Design*, ISPD '12, pages 87–94, 2012.

[11] W.-H. Liu, C.-K. Koh, and Y.-L. Li. Optimization of placement solutions for routability. In *Proceedings of the 50th Annual Design Automation Conference*, DAC '13, pages 153:1–153:9, 2013.

[12] M. Pan, N. Viswanathan, and C. Chu. An efficient and effective detailed placement algorithm. In *Computer-Aided Design, 2005. ICCAD-2005. IEEE/ACM International Conference on*, pages 48–55, 2005.

[13] P. Spindler, U. Schlichtmann, and F. Johannes. Kraftwerk2 – a fast force-directed quadratic placement approach using an accurate net model. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(8):1398–1411, 2008.

Table 3: Detailed placement results with relaxed maximum displacement constraint.

	D_{max}	D_{avg}	Overflow (after)	Δ Overflow	HPWL (after)	Δ HPWL	sHPWL (after)	Δ sHPWL	Runtime (sec)
superblue1	19991	107	0.01	-99.94%	300900000	-11.27%	300918158	-19.27%	204.96
superblue5	19607	128	0.13	-99.24%	364778272	-7.76%	365234290	-20.64%	200.9
superblue7	19999	101	0.00	-100.00%	428872320	-10.35%	428872320	-16.40%	422.3
superblue10	25000	104	0.00	-100.00%	566647104	-7.21%	566648189	-12.61%	261.6
superblue11	24997	105	0.02	-99.79%	361081056	-8.41%	361145182	-15.49%	233.2
superblue12	18000	75	0.00	-100.00%	296077440	-13.06%	296077440	-17.30%	452.85
superblue16	15000	102	0.00	-100.00%	278006080	-9.10%	278006080	-14.75%	147.26
superblue19	15000	87	0.03	-99.52%	152526080	-8.78%	152578076	-14.80%	150.76
Average				**-99.81%**		**-9.49%**		**-16.41%**	

Table 4: Comparison with the original FastPlace-DP without any constraint.

	FastPlace-DP					**Ours**				
	D_{max}	D_{avg}	HPWL	Δ HPWL	Runtime (sec)	D_{max}	D_{avg}	HPWL	Δ HPWL	Runtime (sec)
superblue1	11019	164	300702240	-11.32%	189.24	33563	115	297145152	-12.37%	219.14
superblue5	16889	176	358064096	-9.46%	147.45	28640	140	355578496	-10.09%	175.58
superblue7	25327	169	421648896	-11.86%	334.72	31594	113	422021664	-11.79%	438.21
superblue10	19573	161	564782784	-7.52%	214.72	37981	115	560863744	-8.16%	247.94
superblue11	40207	151	358554720	-9.05%	188.88	41590	112	354556832	-10.07%	200.96
superblue12	24361	211	286595904	-15.84%	407.25	25009	100	288083072	-15.40%	721.18
superblue16	25511	138	274610464	-10.21%	108.29	24744	112	273619520	-10.53%	165.41
superblue19	22631	129	150141792	-10.20%	86.14	24000	100	148739280	-11.04%	189.42
Average				**-10.68%**					**-11.18%**	

Table 5: Comparison with the modified FastPlace-DP with displacement constraint. D_{cons} denotes the maximum displacement constraint specified in the experiments.

		FastPlace-DP					**Ours**				
	D_{cons}	D_{max}	D_{avg}	HPWL	Δ HPWL	Runtime (sec)	D_{max}	D_{avg}	HPWL	Δ HPWL	Runtime (sec)
superblue1	300	1199	72	318601280	-6.04%	140.90	300	55	314771776	-7.17%	136.43
superblue5	320	902	84	372861888	-5.72%	121.51	320	69	367689920	-7.02%	124.95
superblue7	400	1165	83	436650720	-8.73%	249.43	400	68	434180640	-9.24%	354.51
superblue10	440	1417	89	576288000	-5.64%	185.75	440	69	571392576	-6.44%	203.06
superblue11	440	1072	85	369094208	-6.38%	163.61	440	66	363791616	-7.73%	174.24
superblue12	360	1335	86	298825088	-12.25%	298.56	360	66	297952704	-12.51%	485.73
superblue16	300	934	77	282755712	-7.55%	156.05	300	62	280808032	-8.18%	116.44
superblue19	300	629	67	154998144	-7.30%	84.59	300	54	153431216	-8.24%	147.48
Average					**-7.45%**					**-8.32%**	

[14] T. Taghavi, X. Yang, and B.-K. choi. Dragon2005: large-scale mixed-size placement tool. In *Proceedings of the 2005 international symposium on Physical design*, ISPD '05, pages 245–247, New York, NY, USA, 2005. ACM.

[15] N. Viswanathan, M. Pan, and C. Chu. Fastplace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control. In *Design Automation Conference, 2007. ASP-DAC '07. Asia and South Pacific*, pages 135–140, 2007.