An Analytical Placement Algorithm with Routing topology Optimization

Min Wei[1], Xingyu Tong[1], Zhijie Cai[1], Peng Zou[1], Zhifeng Lin[2] and Jianli Chen[1]
[1]State Key Lab of ASIC & System, Fudan University, Shanghai 200433, China
[2]Center for Discrete Mathematics and Theoretical Computer Science, Fuzhou University, Fuzhou 350108, China
mwei21@m.fudan.edu.cn; linzhifeng@fzu.edu.cn; chenjianli@fudan.edu.cn;

2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC) | 979-8-3503-9354-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/ASP-DAC58780.2024.10473827

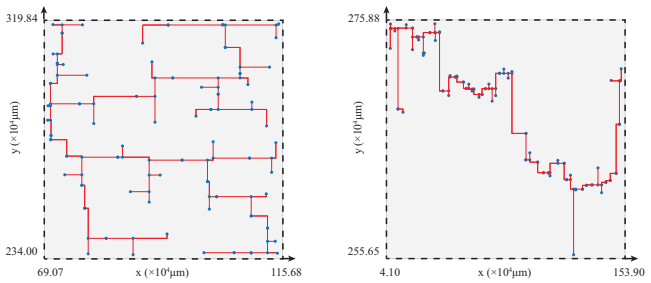
***Abstract*—Placement is a critical step in the modern VLSI design flow, as it dramatically determines the performance of circuit designs. Most placement algorithms estimate the design performance with a half-perimeter wirelength (HPWL) and target it as their optimization objective. The wirelength model used by these algorithms limits their ability to optimize the internal routing topology, which can lead to discrepancies between estimates and the actual routing wirelength. This paper proposes an analytical placement algorithm to optimize the internal routing topology. We first introduce a differential wirelength model in the global placement stage based on an ideal routing topology RSMT. Through screening and tracing various segments, this model can generate meaningful gradients for interior points during gradient computation. Then, after global placement, we propose a cell refinement algorithm and further optimize the routing wirelength with swift density control. Experiments on ICCAD2015 benchmarks show that our algorithm can achieve a 3% improvement in routing wirelength, 0.8% in HPWL, and 23.8% in TNS compared with the state-of-the-art analytical placer.**

I. Introduction

Throughout the physical implementation procedure of integrated circuits, placement sets the tone in essential metrics such as timing and routing congestion. As it is time-consuming to evoke accurate evaluation tools (Static timing analysis and early global routing) frequently in every placement action, generally, placers focus on optimizing total wirelength as a substitute. To further reduce computational complexity, most placers use a fast approximation—half-perimeter wirelength (HPWL) [1, 2, 3, 4, 5] or its variants, such as the Bound2Bound net model [6, 7] and the linearized quadratic wirelength [8].

However, for nets with pins more than 3, HPWL models distort the RSMT-based topology in routing [9]. Not only a shorter routing wirelength implies shorter RC segments in the RC tree, resulting in a possible delay decrease in wire connections; But also a routing topology with less crisscrossing alleviate the congestion pressure for the following routers. Consequently, such inconsistency between HPWL and routing topology leads to a mismatch in optimization intentions, whereas routing wirelength is more related to the ultimate optimization goals.

Some algorithms address high-fanout nets by assigning them a higher net weight [10]. Yet, the increased weight only poses attention to the bounding box, leaving the internal crisscrossing untouched. Figure 1 indicates a possible direction for routing topology optimization. As the evaluation metric, the rectilinear Steiner minimal tree (RSMT) is an optimal topology that can accurately approximate the routing wirelength. For a fair comparison, two distributions are scaled into the same size. Figure (1a) depicts a topology with loosely distributed interior points. In order to generate connections for all pins, its Steiner tree spreads to the whole bounding box, resulting in a much longer RSMT wirelength (StWL). At first glance, Figure (1b) has a longer HPWL. However, thanks to its densely packed topology, its StWL is 53.5% less than Figure (1a). In short, it is profitable to fuse the routing topology optimization into the objective function.



(a) Loosely distributed topology. StWL = 4.75e6μm, HPWL = 1.32e6μm, ratio = 3.59.

(b) Densely packed topology. StWL = 2.21e6μm, HPWL = 1.70e6μm, ratio = 1.30.

Fig. 1: Comparison of the routing topology.

As for the constraint, since analytical algorithms usually relax the constraint into the optimization function as a penalty term, it is challenging for them to give a solution completely without constraint violations. Hence, analytical approaches are typically designed for global placement (GP) to provide a rough position for each movable cell. The constraints are handled by the subsequent legalization (LG) stage, where the cells are moved to the legal sites without overlapping. The abrupt transition between the GP and LG stages may also deteriorate the original intention in routing topology optimization. Therefore, a buffer stage Post-GP would contribute to the smooth transition [4]. In addition, Post-GP can refine the GP solution for some implicit metrics such as routability [11]. Similarly, we can also keep and improve the routing wirelength with the help of Post-GP.

In this work, we propose an analytical routing-topology-aware wirelength model to achieve a densely packed topology in the GP stage. Moreover, inspired by the multilevel placement [12] proposed by Chang *et al.*, we introduce a cell refinement algorithm in the Post-GP stage. The main contributions of this paper are summarized as follows:

- We propose a unique RSMT segmentation method to extract optimizable structures from the internal routing topology, which provides a theoretical basis for our wirelength model.
- We propose a novel wirelength model that can directly refine the topology inside a specific net and effectively optimize the routing wirelength during the GP stage.
- We propose an anchor-based cell refinement framework that effectively handles the density constraint. We also design a universal objective formulation for this framework, guaranteeing effectiveness and convergence.
- Through RSMT topology, we develop a heuristic anchor generation strategy for further optimizing the internal routing topology in the Post-GP stage.
- Compared with the state-of-the-art analytical placer, we can achieve a 3% improvement in routing wirelength, 0.8% in HPWL, and 23.8% in TNS.

Authorized licensed use limited to: National Yang Ming Chiao Tung University. Downloaded on May 21,2025 at 04:59:39 UTC from IEEE Xplore. Restrictions apply.

The remainder of this paper is organized as follows. Section II details our analysis of the HPWL model and its drawbacks as an objective function in the placement stage. Section III explains our routing-topology-aware wirelength model and anchor-based cell refinement framework. Section IV demonstrates our placement results and makes a comparison with state-of-the-art analytical placer. Finally, Section V concludes our algorithm.

II. Preliminaries

This section gives the basic concept of analytical placement, focusing mainly on the HPWL objective function and its analytical approximation. In addition, we shall dive into the net topology of placement-based HPWL and routing-based RSMT, preparing for our looking-ahead optimization algorithm of routing wirelength in the placement stage.

A. Wirelength-driven Analytical Placement

As mentioned in the introduction, analytical placement works by optimizing an objective function with variants of cell positions $(\mathbf{x}, \mathbf{y})$. In other words, the placers construct a minimization problem of the total wirelength under the cell density constraints. Many previous works relax the density constraints into a density penalty term to simplify the problem. The relaxed objective function is formulated as follows:

$$\min_{(\mathbf{x},\mathbf{y})} \sum_{e \in E} WL(e; \mathbf{x}, \mathbf{y}) + \lambda D(\mathbf{x}, \mathbf{y}), \tag{1}$$

where E is the complete set of nets of the design, $(\mathbf{x}, \mathbf{y})$ are cell locations, and $\lambda D(\mathbf{x}, \mathbf{y})$ is the density penalty term with the density penalty weight λ.

As the relaxed function is determined, it can be optimized with solvers such as conjugate gradient descent [5] and Nesterov accelerated gradient descent [3, 4, 13].

B. HPWL Wirelength Model and Analytical Approximation

HPWL approximates the net wirelength according to the minimum bounding box that encloses all cells concerned, the formulation of which is as follows:

$$HPWL(\mathbf{x}, \mathbf{y}) = \sum_{e \in E} (max_{i,j \in e} |x_i - x_j| + max_{i,j \in e} |y_i - y_j|), \tag{2}$$

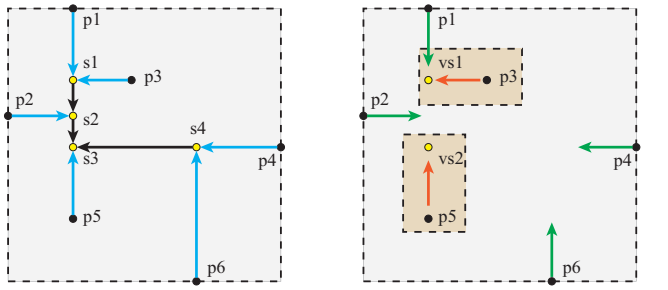
where i, j are different pins of the specific net e.

The calculation process of HPWL is so concise that it only needs to traverse the pin set once. However, the HPWL function in Equation (2) is neither smooth nor convex, which means it's hard to minimize it directly through gradient descent. A widely used approximation of HPWL is the weighted average (WA) function [2], which can be expressed as follows:

$$WA(\mathbf{x}, \mathbf{y})$$

$$= \sum_{e \in E} \left(\frac{\sum_{i \in e} x_i \exp(x_i/\gamma)}{\sum_{i \in e} \exp(x_i/\gamma)} - \frac{\sum_{i \in e} x_i \exp(-x_i/\gamma)}{\sum_{i \in e} \exp(-x_i/\gamma)} + \frac{\sum_{i \in e} y_i \exp(y_i/\gamma)}{\sum_{i \in e} \exp(y_i/\gamma)} - \frac{\sum_{i \in e} y_i \exp(-y_i/\gamma)}{\sum_{i \in e} \exp(-y_i/\gamma)} \right), \tag{3}$$

where γ is the parameter to control the approximation. The difference between exp weight values increases sharply as γ approaches 0. Therefore, these weights choose the minimum and maximum terms in the x and y directions, and the WA wirelength will converge to the HPWL value. Aside from good proximity, the continuity of the WA function can provide appropriate gradients for all cells during the gradient generation process.



(a) Segments division in a generated RSMT

(b) Routing-topology-aware gradients

Fig. 2: Segments division and gradient schematic.

C. Analysis of HPWL and RSMT Topology

The HPWL model ignores the distribution of interior points. Hence, interior pins lose track of the practical gradient guidance. As shown in Figure (2b), the green arrows represent the wirelength gradients in the HPWL model, where points p_5, p_3 will not get any gradient guidance.

To extract optimizable structures, we first segment the topology into two parts: the *trunks* and the *branches*. Segments directly connected to the input and output pins are referred to as the *branches*, whereas the remaining part composed of Steiner points is called the *trunks*. Figure (2a) illustrates a 6-pin net with trunks marked in black and branches in blue. Specifically, the fundamental internal RSMT topology is derived by FLUTE [14] in this work. Other RSMT-generating algorithms shall also work.

Branches all have a physical pin connected in the RSMT structure; thus, they can be directly optimized by moving the connected cells. For the trunk part, however, pins at both ends are Steiner points with volatile positions in different topologies. In other words, the trunk wirelength can only be optimized implicitly. As the topology changes, the status of segments will constantly switch between trunks and branches. Hence, shrinking only the branch parts is feasible to achieve global optimization.

III. Our Proposed Algorithm

In this section, we propose our StWL optimization algorithm in stages of GP and post-GP, as summarized in Figure 3. Section III-A elaborates on the routing-topology-aware wirelength model and its gradient generation in the analytical GP. Section III-B gives our cell refinement framework in the Post-GP stage and introduces the heuristic anchor generation method used to optimize the StWL.

A. StWL Optimization in Global Placement

Our optimization objectives in the analytical placement framework are net topology and its corresponding wirelength, especially the internal routing topology. In this section, we have introduced a wirelength model which focuses the routing topology optimization and enables it to generate reasonable gradients.

1) Differentiable StWL Approximation: Similar to HPWL, StWL is neither smooth nor convex as well. Besides, it does not have a concise expression like Equation (2) for HPWL, and the wirelength calculation must depend on its corresponding topology. To give the formula to calculate the StWL, we adopt the segmentation in Section II-C and add all segments up:

$$StWL(e) = \sum_{i \in e} |x_i - x_{si}|, \tag{4}$$

Authorized licensed use limited to: National Yang Ming Chiao Tung University. Downloaded on May 21,2025 at 04:59:39 UTC from IEEE Xplore. Restrictions apply.

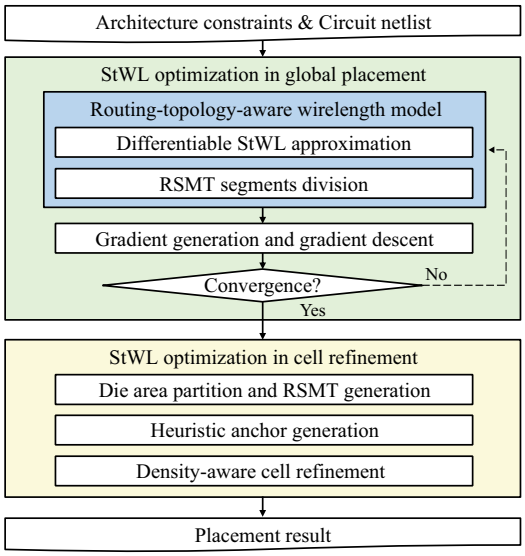


Fig. 3: Framework of our algorithm.

where i is the inner point of net e, including the steiner points, and si is the parent node of point i in Steiner tree. Each segment can be analyzed separately and seen as a 2-pin net. Therefore, all analytical approximations applicable to HPWL can also handle these independent segments. Here is the weighted average approximation of horizontal StWL for net e:

$$StWL(e) \approx \sum_{i \in e} \left(\frac{x_i \exp(x_i/\gamma) + x_{si} \exp(x_{si}/\gamma)}{\exp(x_i/\gamma) + \exp(x_{si}/\gamma)} - \frac{x_i \exp(-x_i/\gamma) + x_{si} \exp(-x_{si}/\gamma)}{\exp(-x_i/\gamma) + \exp(-x_{si}/\gamma)} \right), \tag{5}$$

where γ is a similar smoothing parameter used in $WA(\mathbf{x}, \mathbf{y})$. StWL in the y direction can be calculated similarly.

This approximation can give an accurate fitting result. Let $\varepsilon(i)$ be the estimation error for segment i in the x direction. Without loss of generality, we assume that $x_i > x_{si}$. Considering that exp wights are always positive. For segment i, we have:

$$\begin{aligned} \varepsilon(i) = & \left(x_i - \frac{x_i \exp(x_i/\gamma) + x_{si} \exp(x_{si}/\gamma)}{\exp(x_i/\gamma) + \exp(x_{si}/\gamma)} \right) \\ & + \left(x_{si} - \frac{x_{si} \exp(x_i/\gamma) + x_i \exp(x_{si}/\gamma)}{\exp(x_i/\gamma) + \exp(x_{si}/\gamma)} \right) \\ = & \frac{2\Delta x}{1 + \exp(\Delta x/\gamma)} \end{aligned} \tag{6}$$

As a result, we have the following property:

***Property** 1:* For each segement, its error is no more than $\frac{2\Delta x}{1+\exp(\Delta x/\gamma)}$.

2) RSMT segments division: All segments in the RSMT topology are included in Equation (5). However, considering the optimality and multiple solutions of the RSMT topology, it is time-consuming and unnecessary to optimize all parts by a single model. Therefore, we need to give a proper segment division to ensure the effectiveness and simplicity of the model. Before we can perform the division, it is necessary to establish the following assumptions: **The Steiner points in the topology can be considered fixed in each iteration.** Typically, this assumption is satisfied for most iterations in the GP stage.

This assumption essentially splits the net into a fixed *trunk* part and a changeable *branch* part as analyzed in Section II-C. Equation (5) can also prove this feature: The *trunk* wirelength comes from the addition and subtraction between fixed coordinates, which is also a constant.

Under this assumption, we comprehensively analyze the gradient values calculated during the gradient generation of the smoothing StWL in Equation (5). It is worth noting that for pins that stay on the boundary, the gradient values and directions given by the WA model are pretty close to those provided by Equation (5). Figure (2b) gives an example. We use the green arrow to mark the gradient from the WA wirelength model. All these gradients are also parts of the gradient from the smoothing StWL model. Therefore, to take advantage of the computational efficiency of the WA model, we choose to use a hybrid wirelength model concerning RSMT topology in the GP phase:

$$WL(\mathbf{x}, \mathbf{y}) = WA(\mathbf{x}, \mathbf{y}) + \sum_{e \in E} StWL(e_{in}), \tag{7}$$

where e_{in} corresponds to the inner part of net e. It can be found that for 2-pin and 3-pin nets, Equation (7) degenerates to the WA model, which avoids a large number of routing topology analyses. Gradients from Equation (7) is shown in Figure (2b). The arrows between p_3 and vs_1, p_5 and vs_2 in the brown box are the inner gradient for this 6-pin net.

3) Gradient descent of our wirelength model: For pins insides, objective function Equation (7) can provide a gradient to minimize the length of their corresponding segments; for pins on the net boundary, their shrinking gradient is provided by the $WA(\mathbf{x}, \mathbf{y})$ term of Equation (7). Figure 4 shows the transformation of routing topology throughout gradient descent iterations. We applied the hybrid routing-topology-aware wirelength model from iteration 200 to make a comparison with the mere WA model. After 50 iterations, while the net remains littered with the WA model, our routing-topology-aware wirelength model has clustered cells around the trunk part and compacted the net internal structure.

B. StWL Optimization in Cell Refinement

Analytical placement is announced as "convergence" when cells are scattering with tolerable density violations. At this time, further iterations only perform trivial cell displacement. Aside from gradient descent, there is still room for RSMT refinement with our routing-topology-aware wirelength model. In the Post-GP stage, we propose a placement violation refinement algorithm with a heuristic RSMT optimization strategy.

1) Framework of our refinement algorithm: Moving cells to their wirelength-optimal region hastily would obviously deteriorate the low overlap distribution given by analytical global placement. Hence, we posed bin-level density control in cell movement to keep the incremental refinement swift and plausible.

A branch and bound method is employed for swift density check. Initially, the whole die is divided into four identical coarse bins, as shown in Figure 6. Then, we use a quadtree to store the available area of the four bins in each node. The child node regions will continue to be divided until the finest bin granularity (typically 5-10 times the row height) is reached, finishing the quadtree construction. Note that cells in the same bin are considered to share the same coordinates at each granularity. For a new cell, our algorithm traverses the quadtree to find potential regions with enough area. If the node bin does not have enough area, it will drop out of the set of candidate regions. At each level, the final optimal region that the traversing algorithm shall step into is determined by a pessimistic cost function based on RSMT topology, which is introduced in Section III-B2 After the cell is moved into

Authorized licensed use limited to: National Yang Ming Chiao Tung University. Downloaded on May 21,2025 at 04:59:39 UTC from IEEE Xplore. Restrictions apply.

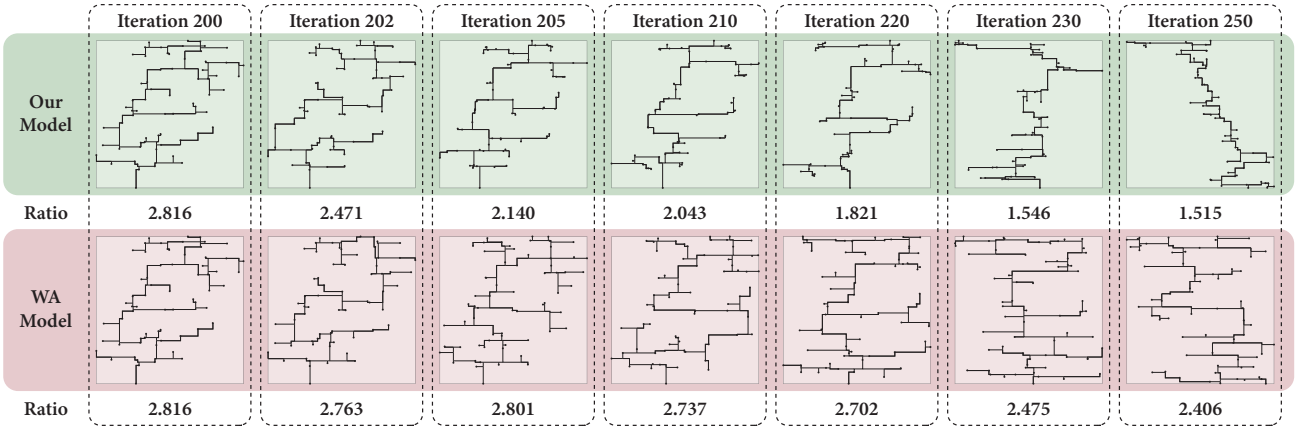


Fig. 4: Snapshots of the RSMT topology from the routing-topology-aware wirelength model (green shading) and WA model (pink shading). Two placements are all driven by the WA model in the first 200 iterations. The routing-topology-aware wirelength model is introduced in Iteration 201 and compared with the WA model on a 65-pin net from industrial benchmark design1. The ratio gives the concentration degree of interior points, which is calculated by StWL/HPWL.

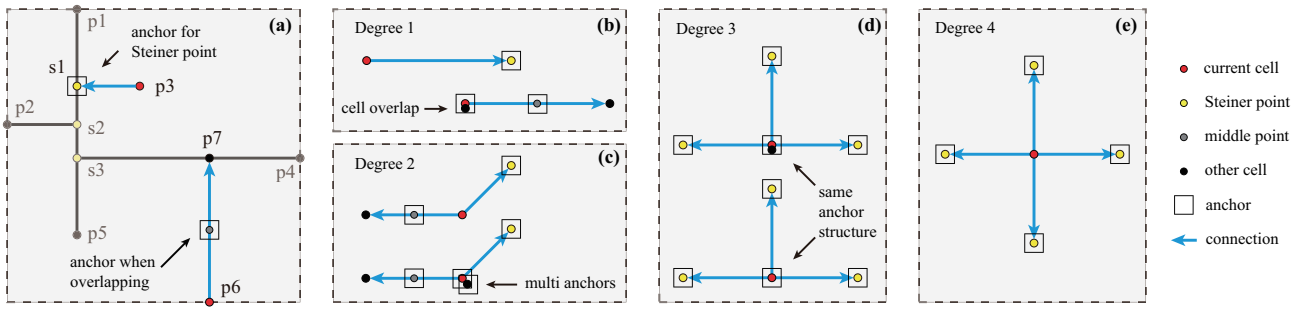


Fig. 5: Anchor generation based on RSMT topology. (a) gives an illustration of anchor generation inside a 7-pin net with point $\mathbf{p_7}$ coincides with one Steiner point. (b)-(e) give the anchor generation methods for conditions with 1-4 connections linked to the current cell.

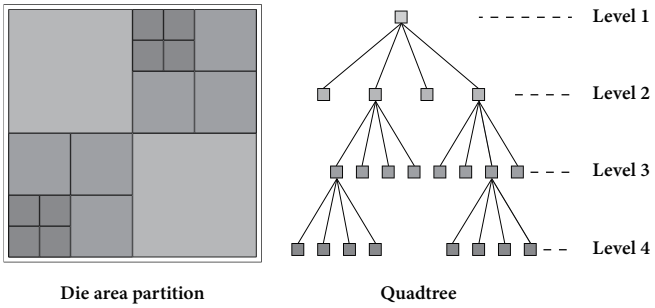


Fig. 6: Quadtree for swift density check.

its optimal region, the density of nodes affected throughout the quadtree will update accordingly. Steps involved in this optimization are described in Algorithm 1.

2) Heuristic anchor generation: The RSMT topology is constructed with FLUTE in advance at each granularity as it is hard to update incrementally on account of one cell movement. In addition, the Steiner points and cells may share the same coordinates at the bin-level granularity. We refer to the position of an instance as the bin-level coordinate. To optimize the RSMT wirelength in a non-deteriorating way, we would like to move the cell without altering the RSMT topology. Given a certain cell, we decompose its related segments in RSMT topology into rectilinear connections to RSMT-based **anchors**.

An accumulation of Manhattan wirelength between a cell and its anchors serves as the cost function in determining the optimal child node in the quadtree:

$$F(\mathbf{x},\mathbf{y}) = \sum_{v\in V}\sum_{g\in G_v} w_g(d_x(g)+d_y(g)), \tag{8}$$

where G_v is the complete set of anchors for cell v, w_g is the

Algorithm 1 Density-aware cell refinement.

Input: Anchors G_v for each cell v
Output: Cell position $(\mathbf{x},\mathbf{y})$ with a minimized wirelength

```
Store the remaining area to quadtree T
for each cell v in V do
    Set N_0 the root node of T
    for each level l in T do
        V_Nl ← nodes with enough area to accommodate cell v in 4 successors of N_{l-1}
        Find the optimal node N_l for cell v in V_Nl
    end for
    Set N_opt the leaf node with the minimum wirelength
    Calculate the wirelength gain ΔF(x*, y*) when moving cell v to bin N_opt
    if ΔF(x*, y*) < 0 then
        Move cell v to (x*, y*)
        Update remaining area on quadtree T
    end if
end for
```

Authorized licensed use limited to: National Yang Ming Chiao Tung University. Downloaded on May 21,2025 at 04:59:39 UTC from IEEE Xplore. Restrictions apply.

TABLE I: Benchmark statistics and experimental results on ICCAD2015 contest. **WL**: in $(\times 10^4)\mu$m. **RT**: in second. **WNS**: in ns. **TNS**: in $(\times 10^2)$ns.

Benchmark	Benchmark statistics				DREAMPlace [3]					Ours				
	Nets	Cells	Pins	Macro	RWL	HPWL	WNS	TNS	RT	RWL	HPWL	WNS	TNS	RT
superblue1	1216K	1210K	3767K	3787	8579.1	7559.0	-18.5	-257.0	281.8	8154.8	7682.0	-21.4	-221.0	485.5
superblue3	1225K	1213K	3905K	2074	9493.3	8797.0	-32.1	-83.3	356.8	9323.7	8779.0	-34.3	-71.7	563.7
superblue4	803K	796K	2498K	3471	6358.9	5892.0	-21.4	-185.7	206.7	6120.0	5797.0	-26.5	-134.0	270.3
superblue5	1101K	1087K	3243K	1872	9672.0	9083.0	-48.8	-192.9	422.0	9387.5	8934.0	-54.0	-163.1	439.6
superblue7	1934K	1932K	6372K	4910	12314.2	11180.0	-20.4	-155.1	532.3	11863.8	11000.0	-19.0	-145.2	838.4
superblue10	1898K	1876K	5560K	1696	18448.5	17440.0	-33.4	-822.7	554.3	18028.5	17190.0	-26.3	-692.0	570.1
superblue16	1000K	982K	3013K	101	8587.6	8115.0	-16.9	-416.8	192.9	8354.2	8002.0	-17.0	-252.8	281.4
superblue18	772K	768K	2559K	653	5370.4	4381.0	-20.3	-88.0	212.1	5330.9	4397.0	-15.6	-80.0	369.6
N.Average					1.030	1.008	1.019	1.238	0.726	1.000	1.000	1.000	1.000	1.000

Algorithm 2 Anchor generation for a certain net

Input: RSMT topology T of net e
Output: Multiset of anchors G_v for each cell v

```
Initialize map M: bin b ↦ RSMT-connected bins set M(b)
for each segment s = (b_from, b_to) in topology T do
    if ∃v' ∈ e, b_v' == b_to then
        Add mid(b_from, b_to) to M(b_from)
        Add mid(b_from, b_to) to M(b_to)
    else
        Add b_to to M(b_from)
        Add b_from to M(b_to)
    end if
end for
for each cell v in net e do
    G_v ← M(b_v)
    switch (size of M(b_v))
    case 1:
        if ∃v' ∈ e, b_v == b_v' then
            G_v ← G_v ∪ [b_v]
        end if
    case 2:
        if ∃v' ∈ e, b_v == b_v' then
            G_v ← G_v ∪ [b_v, b_v]
        end if
    case 3:
        G_v ← G_v ∪ [b_v]
    end switch
end for
```

weight of anchor connection g, $d_x(g)$, $d_y(g)$ are the spanning lengths of connection g in x and y directions. To normalize the cost function, we set the weight $w_g = 1/N_{anchor}$, where N_{anchor} is the number of anchors connected to this cell within this net.

For anchor generation, a straightforward strategy is to use the adjacent Steiner point as in Section III-A2. An example is p_3 in Figure (5a). Yet, this method is only applicable to some situations. If two cells connect directly in an RSMT, there will be no Steiner points for anchor generation. Empirically, anchors are generated at the midpoint of the segment, as shown in cellp_6 in Figure (5a), intending to pull the two cells closer without topology change. Besides, cells in the net concerned may be in the same bin in coarse granularity. At this point, the rectilinear wirelength is 0 between the two points. Hence, we need extra anchors to keep track of the potential wirelength increment of this segment.

Figure (5b-e) enumerates our anchor generation scheme for all situations. Algorithm 2 gives a detailed description of the anchor generation process. Map M in Algorithm 2 stores the bin-to-bin segments from the RSMT topology T;

TABLE II: Industrial benchmark statistics.

Benchmark	Benchmark Statistics			
	Nets	Cells	Pins	Macros
design1	21K	20K	73K	8
design2	90K	90K	338K	32
design3	90K	90K	380K	-
design4	118K	110K	564K	1024
design5	153K	149K	628K	45
design6	164K	163K	642K	-
design7	187K	187K	770K	-
design8	760K	760K	3151K	43
design9	1510K	1510K	4885K	7
design10	1511K	1511K	4886K	-
design11	1292K	1228K	5231K	70
design12	1292K	1228K	5231K	70
design13	1296K	1232K	5238K	70
design14	1313K	1249K	5278K	39

3) Convergence and complexity analysis: This procedure repeats until the reduction of $StWL(\mathbf{x}, \mathbf{y})$ in a single iteration falls below a threshold ϵ. This parameter stops the ping-pong effect caused by the mismatch from the bin-level coordinate. In practice, this value is usually set to 10 or more.

$$StWL(\mathbf{x_k}, \mathbf{y_k}) \leq StWL(\mathbf{x_{k-1}}, \mathbf{y_{k-1}}) - \epsilon. \tag{9}$$

With the help of hyper-parameter ϵ, the sequence of StWL that Algorithm 1 gives is a strictly decreasing sequence. Since StWL is a bounded function, this optimization process should always converge to a local minimum $(\mathbf{x}^*, \mathbf{y}^*)$.

Suppose we have n_v cells, n_e nets, and n_p pins in the design. And to control the density, we divide the die area into $m \times m$ uniform bins. Algorithm 1 needs to traverse the quadtree for each cell, the complexity of which is $O(n_v log(m))$. Algorithm 2 has a complexity of $O(n_p)$ in traversing all pins to generate the anchor set G_v. In total, our algorithm consumes $O(n_v log(m) + n_p)$ runtime to refine the placement in one iteration, which is relatively efficient.

IV. Experimental Results

Our routing-topology-aware wirelength model and cell refinement technique are implemented in C++. We analyze the performance of our method on both the ICCAD2015 incremental timing-driven placement contest benchmarks and industrial benchmarks. Table I summarizes the statistics of ICCAD2015 benchmarks, and comparison with the open-source placer DREAMPlace [3]. The test environment is a 64-bit Linux machine with 24-core Intel Xeon processors running at 2.4GHz; for a fair comparison, both placers run on a CPU with 8 threads. Since the legalization and detailed placement modules of DREAMPlace cannot work correctly on ICCAD2015 benchmarks [15], we utilize Innovus refinePlace for layout legalization and fine-tuning. We compared the routed wirelength by Innovus earlyGlobalRoute and the timing metrics by UI-timer2.0[16].

Authorized licensed use limited to: National Yang Ming Chiao Tung University. Downloaded on May 21,2025 at 04:59:39 UTC from IEEE Xplore. Restrictions apply.

TABLE III: Experimental Results on industrial benchmarks. **WL**: in $(\times 10^4)\mu$m. **RT**: in second. **WNS**: in ns. **TNS**: in $(\times 10^2)$ns.

Benchmark	WA Model							Ours						
	RWL	HPWL	TNS	WNS	H_OF	V_OF	RT	RWL	HPWL	TNS	WNS	H_OF	V_OF	RT
design1	54.0	40.3	-75.5	-9.1	0.00	0.00	8.7	45.7	39.3	-43.3	-5.8	0.00	0.00	17.0
design2	299.4	250.3	-6450.0	-64.6	0.00	0.04	21.8	286.4	251.9	-7120.0	-42.2	0.00	0.03	35.1
design3	129.6	70.0	-726.1	-9.5	0.00	0.00	28.0	113.9	71.0	-604.7	-6.7	0.00	0.00	132.5
design4	2049.8	1761.4	-36200.0	-31.2	1.76	0.12	65.3	2003.8	1737.5	-30900.0	-26.5	1.57	0.11	245.1
design5	1810.3	1484.5	-537.3	-3.1	0.41	0.12	40.0	1697.1	1432.2	-426.1	-2.1	0.39	0.07	91.6
design6	617.7	326.4	-24400.0	-41.4	0.00	0.00	29.8	577.8	348.4	-18300.0	-27.6	0.00	0.00	94.0
design7	397.0	141.8	-27400.0	-45.3	0.00	0.00	40.0	342.6	163.6	-18900.0	-24.9	0.00	0.00	184.6
design8	1298.8	598.6	-1190.0	-4.6	0.00	0.08	208.8	1171.6	655.1	-1180.0	-3.6	0.00	0.06	1056.4
design9	2094.0	1106.0	-2170.0	-105.4	0.00	0.00	426.5	1942.8	1165.7	-1930.0	-98.7	0.00	0.00	1477.3
design10	1577.9	802.4	-48900.0	-37.0	0.00	0.00	508.0	1434.6	876.4	-40200.0	-33.1	0.00	0.00	1621.0
design11	3219.8	1934.4	-37800.0	-19.5	8.57	23.27	498.6	2851.2	2032.7	-27500.0	-14.3	6.06	15.29	1245.1
design12	3193.2	1934.3	-28900.0	-40.2	5.58	8.94	486.4	2829.5	2032.6	-19800.0	-32.2	4.20	5.64	1310.7
design13	3245.0	1945.7	-53.8	-5.5	7.12	22.68	447.9	2863.9	2046.1	-48.1	-5.7	5.46	16.32	1319.9
design14	3258.1	1961.8	-28600.0	-40.3	2.64	2.84	440.3	2882.9	2058.0	-19700.0	-32.3	1.93	2.18	1233.1
N.Average	1.106	0.960	1.273	1.344	1.264	1.409	0.346	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Compared with DREAMPlace, our algorithm can achieve a 3.0% reduction in routing wirelength on average with a 0.8% HPWL reduction. Internal routing topology optimization can also bring great improvement in timing metrics. Compared with DREAMPlace, our algorithm achieves a 23.8% reduction in TNS. It needs to be noted that our algorithm does not perform any exceptional repairs for the most critical paths. Consequently, the average reduction in WNS is 1.9%, which is relatively smaller than the TNS reduction.

We further analyze the performance of our algorithm on industrial benchmarks with real routing layers and constraints. Table II summarizes the benchmark statistics. Unfortunately, DREAMPlace cannot accommodate the complex LEF structure of industrial benchmarks. Therefore, we validate our algorithm by turning off the StWL optimization in GP and the cell refinement algorithm in Post-GP. This control group is called the "WA Model", named after the wirelength model that takes effect in the GP stage. Experimental results are shown in Table III. On average, our algorithm can bring a 10.6% reduction in routing wirelength. In contrast to the great optimization of routing wirelength, our algorithm's HPWL is 4.0% larger than the WA Model, consistent with the HPWL characteristics of our aforementioned analysis. The routing congestion is also effectively alleviated due to the effective optimization of our algorithm in the routing wirelength. To give an average optimized ratio, we omitted benchmarks without overflow. Compared with WA Model, as shown in Table III, H_OV has been reduced by 26.4%, while V_OV has been reduced by 40.9%. For severely congested benchmarks, our algorithm can greatly alleviate their routing congestion.

V. Conclusions

In this paper, we have studied the HPWL model and its analytical approximation in depth and analyzed the problem posed by this model. We have proposed a differential routing-topology-aware wirelength model and an efficient cell refinement strategy for optimizing routing wirelength in the Post-GP stage to solve these problems. With optimization in two stages, our algorithm outperforms state-of-the-art placer in routing wirelength and timing metrics. We have also shown the great potential of looking-ahead routing topology optimization for solving the congestion problem. Our strategy can be regarded as an effective upgrade of the existing placement framework. Our routing-topology-aware wirelength model is an improved objective function, and the refinement framework is a supplementary optimization after the GP stage. Therefore, our algorithm has extremely high practicality. Our future work lies in the heuristic optimization of the trunk part of the routing topology and the enhancement of this structure to optimize more metrics, such as power metrics and design rule violations in detailed routing.

References

[1] Lixin Liu, Bangqi Fu, Martin D. F. Wong, and Evangeline F. Y. Young. Xplace: an extremely fast and extensible global placement framework. In *Proceedings of the 59th Design Automation Conference*, 2022.

[2] Meng-Kai Hsu, Yao-Wen Chang, and Valeriy Balabanov. Tsv-aware analytical placement for 3d IC designs. In *Proceedings of the 48th Design Automation Conference*, 2011.

[3] Yibo Lin, Zixuan Jiang, Jiaqi Gu, Wuxi Li, Shounak Dhar, Haoxing Ren, Brucek Khailany, and David Z. Pan. Dreamplace: Deep learning toolkit-enabled GPU acceleration for modern VLSI placement. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 40(4):748–761, 2021.

[4] Chung-Kuan Cheng, Andrew B. Kahng, Ilgweon Kang, and Lutong Wang. RePlAce: Advancing solution quality and routability validation in global placement. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 38(9):1717–1730, 2019.

[5] Tung-Chieh Chen, Zhe-Wei Jiang, Tien-Chang Hsu, Hsin-Chen Chen, and Yao-Wen Chang. Ntuplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 27(7):1228–1240, 2008.

[6] Myung-Chul Kim, Dongjin Lee, and Igor L. Markov. Simpl: An effective placement algorithm. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 31(1):50–60, 2012.

[7] Peter Spindler, Ulf Schlichtmann, and Frank M. Johannes. Kraftwerk2 - A fast force-directed quadratic placement approach using an accurate net model. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 27(8):1398–1411, 2008.

[8] Georg Sigl, Konrad Doll, and Frank M. Johannes. Analytical placement: A linear or a quadratic objective function? In *Proceedings of the 28th Design Automation Conference*, 1991.

[9] Jarrod A. Roy and Igor L. Markov. Seeing the forest and the trees: Steiner wirelength optimization in placement. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 26(4):632–644, 2007.

[10] Andrew E. Caldwell, Andrew B. Kahng, Stefanus Mantik, Igor L. Markov, and Alexander Zelikovsky. On wirelength estimations for row-based placement. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 18(9):1265–1278, 1999.

[11] Jai-Ming Lin, Liang-Chi Zane, Min-Chia Tsai, Yung-Chen Chen, Che-Li Lin, and Chen-Fa Tsai. PPOM: an effective post-global placement optimization methodology for better wirelength and routability. *IEEE Trans. Very Large Scale Integr. Syst.*, 30(11):1783–1793, 2022.

[12] Chin-Chih Chang, Jason Cong, David Zhigang Pan, and Xin Yuan. Multilevel global placement with congestion control. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 22(4):395–409, 2003.

[13] Jingwei Lu, Pengwen Chen, Chin-Chih Chang, Lu Sha, Dennis Jen-Hsin Huang, Chin-Chi Teng, and Chung-Kuan Cheng. eplace: Electrostatics-based placement using fast fourier transform and nesterov's method. *ACM Trans. Design Autom. Electr. Syst.*, 20(2):17:1–17:34, 2015.

[14] Chris C. N. Chu and Yiu-Chung Wong. FLUTE: fast lookup table based rectilinear steiner minimal tree algorithm for VLSI design. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 27(1):70–83, 2008.

[15] Myung-Chul Kim, Jin Hu, Jiajia Li, and Natarajan Viswanathan. ICCAD-2015 CAD contest in incremental timing-driven placement and benchmark suite. In Diana Marculescu and Frank Liu, editors, *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 921–926, 2015.

[16] Tsung-Wei Huang, Guannan Guo, Chun-Xun Lin, and Martin D. F. Wong. Opentimer v2: A new parallel incremental timing analysis engine. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 40(4):776–789, 2021.

Authorized licensed use limited to: National Yang Ming Chiao Tung University. Downloaded on May 21,2025 at 04:59:39 UTC from IEEE Xplore. Restrictions apply.