Abstract—We propose an electrostatics-based placement algorithm for large-scale mixed-size circuits (ePlace-MS). ePlace-MS is generalized, flat, analytic and nonlinear. The density modeling method eDensity is extended to handle the mixed-size placement. We conduct detailed analysis on the correctness of the gradient formulation and the numerical solution, as well as the rationale of dc removal and the advantages over prior density functions. Nesterov’s method is used as the nonlinear solver, which shows high yet stable performance over mixed-size circuits. The steep length is set as the inverse of Lipschitz constant of the gradient function, while we develop a backtracking method to prevent overestimation. An approximated nonlinear preconditioner is developed to minimize the topological and physical differences between large macros and standard cells. Besides, we devise a simulated annealer to legalize the layout of macros and use a second-phase global placement to reoptimize the standard cell layout. All the above innovations are integrated into our mixed-size placement prototype ePlace-MS, which outperforms all the related works in literature with better quality and efficiency. Compared to the leading-edge mixed-size placer NTUplace3, ePlace-MS produces up to 22.98% and on average 8.22% shorter wirelength over all the 16 modern mixed-size benchmark circuits with the same runtime.

Index Terms—Analytic placement, electrostatic analogy, fast Fourier transform (FFT), Lipschitz constant, Nesterov’s method, nonlinear optimization, Poisson’s equation, preconditioning, spectral methods.

I. INTRODUCTION

Placement remains crucial and challenging in VLSI physical design [22], [31]. Placement performance impacts downstream design phases of clock tree synthesis [27], dynamic power minimization [28], global and detailed routing [30], etc. The advent of billion-transistor integration [14] makes the placement performance dominant on the overall quality. Placement quality is usually evaluated by the total half-perimeter wirelength (HPWL), which correlates with timing [29], [48], routability [10], [41], and power [26], subject to the constraint of zero overlap among circuit components. Such problem formulation is broadly used among research developments [3], [5], [15], [17], [18], [45], [46] and well honored by public placement benchmarks [33], [34], [47]. Over thousands of predesigned IP blocks, macros and memory units are embedded to shorten the total design turnaround, where the topological and physical differences with standard cells are huge. The high design complexity and complication continuously challenge the capability of mixed-size placers.

Prior mixed-size placement algorithms form three categories. Two-stage methods use two separated phases of floorplanning and placement. Location and orientation of macros are determined and fixed at first, while placement follows to optimize only standard cells in the global scale. MP-tree [6] packs macros along the chip boundaries. A constraint-graph (CG) algorithm [4] uses mathematical programming to optimize placement, macro positions, and orientations. However, the limited information of standard cell distribution misguides the floorplanner at early stages, inducing suboptimal floorplan solution to degrade the overall quality.

Constructive (floorplan-guided) approaches combine the advantages of floorplan and placement. The floorplanner simultaneously optimizes both macros and soft blocks (clusters of standard cells). An incremental placement then spread standard cells in local scale. Capo [39] repeatedly invokes a fixed-outline floorplanner over the top-down placement framework, providing guidance to macro shifting. FLOP [47] groups cells into soft blocks to produce initial floorplan solution. Incremental global [46] and detailed placement further spread and legalize the standard cells within local scale. Nonetheless, the intrinsic limitation of partitioning and clustering usually induce suboptimal solutions in the placement perspective. Optimization space of standard cell placement could be substantially shrunk with quality loss hard to recover.

One-stage solution remains popular among most modern placement algorithms [13], [15], [17], [18], [21], [46]. Macros and standard cells are being placed simultaneously where the limitations discussed above can be well avoided. FastPlace3.0 [46] performs selective grid resizing to accommodate large macros with more whitespace. ComPLx [17] shreds macros into small objects with sizes similar to that of the standard cells. After placement finishes, each macro is reconstructed based on the gravity center of instances belonging to it. APlace3 [15] reshapes the smoothing curve of the density function to distinguish the smoothness of macro movement with that of standard cells. NTUplace3 [13]
incorporates rotational and flipping components into the gradient function, which enables simultaneous optimization on the location of all the movable objects as well as the orientation of macros. As mentioned in [47], macro and standard cell co-placement challenges the capability of modern analytic placement approaches. Despite largest search space, nevertheless, the substantial topological and physical differences between macros and standard cells might introduce gradient imbalance and cause the solution hard to converge.

In this paper, we develop a generalized one-stage, flat, and analytic nonlinear algorithm for mixed-size placement, based on the infrastructure of our early works Fast Fourier Transform based PLacement (FFTPL) [24] and ePlace [23] for standard-cell-based placement, as well as its extension to the mixed-size circuits [25]. As the major difficulty of mixed-size placement remains in the broad spectrum of topological and physical attributes among all the movable objects (i.e., standard cells and large macros), our innovation of nonlinear preconditioning well equalizes them in the solver’s perspective. As a generalized algorithm, ePlace-MS handles standard cells and macros in exactly the same way (see macro shifting when declustering [13], soft block formation by standard cells [46], [47], special macro density smoothing [15], [18], macro shredding [17], etc.) to ensure high and stable performance over various integrated circuits with potentially quite different structures of the design. Our contributions are listed as follows.

1. We extend our prior density function eDensity [24], [25] to model mixed-size integrated circuits in a generalized way. Besides, we provide detailed analysis and proof on eDensity with:
   a) rationale of its dc removal;
   b) correctness of the density gradient formulation;
   c) correctness of the numerical solution;
   d) advantages over literature density functions.

2. We extend Nesterov’s method as the nonlinear solver to handle mixed-size placement, with steplength dynamically predicted via Lipschitz constant. Moreover, we develop a backtracking method to effectively prevent steplength overestimation.

3. We develop an approximated nonlinear pre-conditioner to resolve the substantial topological and physical gap between standard cells and macros. The solution quality is significantly improved with negligible runtime overhead.

4. We devise an annealing-based macro legalizer providing direct control to the macro shifting. A second-phase standard cell-only global placement (cGP) is proposed to resolve the quality overhead induced during macro legalization.

5. We integrate all the innovations into ePlace-MS, an electrostatics-based prototype for mixed-size placement, with promising experimental results obtained on the modern mixed-size (MMS) [47] circuits. Specifically, ePlace-MS outperforms the leading placer NTUplace3 [13] with 8.22% shorter wirelength and the same runtime on average of all the 16 MMS benchmarks [47].

The remainder is organized as follows. Section II introduces the background knowledge. Section III provides an overview of ePlace-MS. Section IV analyzes our placement density function eDensity on the mixed-size circuits. Section V discusses the nonlinear placement optimization. Section VI proposes a nonlinear pre-conditioner to equalize macros with standard cells. Section VII introduces our annealing-based macro legalization. Section VIII discusses the second-phase cGP. Experiments and results are shown in Section IX. We conclude in Section X.

II. Essential Concepts

Given a placement instance $G = (V, E, R)$ with a set $V$ of $n$ movable objects (standard cells and macros), nets $E$ and placeable region $R$, the placement is formulated as a constrained optimization problem. The constraint desires a solution $v = (x, y)^T = (x_1, \ldots, x_n, y_1, \ldots, y_n)^T$ to accommodate all the objects with sufficient sites but zero overlap or density violation. Global placement uniformly decomposes the entire region $R$ into $m \times m$ rectangular grids (bins) which are denoted as grid set $B$. For every grid $g \in B$, the density $\rho(g)$ should not exceed a predetermined upper-bound $\rho_t$, which is named target density and usually design specific. The objective is set as minimizing the total HPWL of all the nets. Let $\text{HPWL}_e(v)$ denote the HPWL of the net $e$, the total HPWL is expressed as

$$
\text{HPWL}(v) = \sum_{e \in E} \text{HPWL}_e(v)
$$

Here $i$ and $j$ are any two of all the objects connected by the net $e$. As a result, the nonlinear optimization of global placement is formulated as (2) and proved to be NP-complete [9]

$$
\min_v \text{HPWL}(v) \text{ s.t. } \rho_b(v) \leq \rho_t, \forall b \in B.
$$

Analytic methods conduct placement using gradient-based optimization. As the function $\text{HPWL}(v)$ in (1) is not differentiable, various quadratic [7], [20] and nonlinear [12], [35] wirelength smoothing techniques have been proposed in literature to ensure the differentiability thus analytic placement to work. In this paper, we use both the log-sum-exp (LSE) [35] model and the weighted-average (WA) [12] model (as two options for ePlace-MS) to approximate HPWL via a closed-form smooth function $W(e, v)$, as shown in

$$
W_{\epsilon_1}(v) = \left( \log \sum_{i \in e} \exp(x_i/\gamma) + \log \sum_{i \in e} \exp(-x_i/\gamma) \right)
$$

$$
W_{\epsilon_2}(v) = \frac{\sum_{i \in e} x_i \exp(x_i/\gamma)}{\sum_{i \in e} \exp(x_i/\gamma)} - \frac{\sum_{i \in e} x_i \exp(-x_i/\gamma)}{\sum_{i \in e} \exp(-x_i/\gamma)}.
$$

Here, $W_{\epsilon}(v) = W_{\epsilon_1}(v) + W_{\epsilon_2}(v)$ and $W(v) = \sum_{e} W_e(v)$. $\gamma$ is used to control the modeling accuracy, i.e., smaller $\gamma$ will improve the accuracy of approximation but decrease the function smoothness, vice versa. In this paper, we use ePlace-MS-WA and ePlace-MS-LSE to denote the usage of each wirelength model in our algorithm, respectively.

A density penalty function helps incorporate the $|B|$ constraints in (2) in order to achieve analyticit. Modern quadratic placers [17], [18], [21], [43], [46] are all based on the density force formulation proposed in [7], where anchor points are inserted to drag cells away from over-filled region, modeling the density force as a constant term in the gradient function. In contrast, nonlinear placers formulate the density gradient as a component independent to wirelength. APlace3 [15] and
NTUplace3 [13] ensure differentiability of the density distribution via bell-shaped curve [35] with local smoothness, while mPL6 [3] uses Helmholtz function targeting global density smoothness. In this paper, we use the electrostatics-based density modeling method eDensity function developed in our prior work FFTPL [24], which will be thoroughly analyzed in Section IV with proof of correctness. The numerous grid density constraints are all relaxed by using this single penalty function, while we have the unconstrained placement optimization problem defined as

\[
\min_v f(v) = W(v) + \lambda N(v)
\]

where \( N(v) \) [defined in (6)] is the density function and \( \lambda \) is the penalty factor for adjusting the ratio between wirelength and density.

III. PLACEMENT OVERVIEW

Fig. 1 shows the flowchart of ePlace-MS. Given a placement instance, ePlace-MS quadratically minimizes the total wirelength at the first stage of mixed-size initial placement (mIP). The initial solution \( v_{mIP} \) is of low wirelength but high overlap. Based on the target density \( \rho_t \), our mixed-size global placer (mGP) populates extra whitespace with unconnected fillers, then iteratively co-optimizes all the objects (standard cells, macros, and fillers) together. After mGP, we remove all the fillers, fix the standard-cell layout, then invoke the annealing engine macro legalization (mLG) to legalize the location of all the macros. In the second-phase global placement (cGP), we retrieve all the fillers and distribute them appropriately, then free standard cells and co-place them with fillers to further reduce the wirelength. Finally, in the standard-cell detailed placement (cDP), we invoke the detailed placer in [13] to legalize and discretely optimize the standard-cell layout.

ePlace-MS does not allow rotation or flipping of any objects due to the lithography issue. However, it has the flexibility to smoothly integrate the rotational and flipping gradients [13] to guide placement optimization iteratively. Deadspace allocation is also not considered in this paper, while it can be effectively realized in ePlace-MS via appropriate macro inflation.

ePlace-MS maximally expands the design space for mGP with the major optimization effort budgeted on mGP, since all the objects (standard cells, macros, and fillers) are allowed to move and can be optimized simultaneously. In contrast, the design spaces for mLG and cGP are relatively shrunk, as only macros or standard cells are allowed to move with other objects fixed thus acting as constraints, which actually constrains the search space of mixed-size placement solution.

Specifically, only minor layout perturbation is expected to perform changes within local scale. As Fig. 2 shows, the constrained optimization focuses on the mGP stage and terminates when overlap is small enough. The entire placement framework is built upon our recent work of FFTPL [24] and ePlace [23] with similar initialization and iterative adjustment of parameters. Grid dimension \( m \) is statically determined as \( m = \lceil \log_2 \sqrt{n} \rceil \) and upper-bounded by 1024, where \( n' = |V'| \) is the number of movable macros, standard cells and fillers [23], [24]. Penalty factor \( \lambda \) is initially set as (10) of [5]. We iteratively update \( \lambda_k = \mu_k \lambda_{k-1} \) in mGP to balance the wirelength and density forces, where \( \mu_k = 1.1 - \frac{(\Delta HPWL_{k}/\Delta HPWL_{REF})+1.0}{10} \) based on the HPWL variation \( \Delta HPWL_{k} = HPWL(\tilde{v}_k) - HPWL(\tilde{v}_{k-1}) \). In practice, we set \( \Delta HPWL_{REF} = 3.5 \times 10^3 \) and bound \( \mu_k \) by [0.75, 1.1].

Density overflow \( \tau \) is used as the stopping criterion. We terminate mGP when \( \tau \leq 10% \) and cGP when \( \tau \leq 7\% \), respectively. Wirelength coefficient \( \gamma \) is used to smooth the HPWL. We set the smoothing parameter as \( \gamma = 8.0 w_h \times 10^{30/9 \times (r-0.1) - 1.0} \) to encourage global movement at early iterations and convergence at later iterations. Fillers are used to balance the electrostatic dc component in the global scale. The total area of fillers equals the total whitespace multiplies target density then subtracted by the total area of all the movable objects. All the fillers are equally sized to be the average physical dimensions of all the standard cells. More details of parameter adjustment or filler formation can be found in [23] and [24].

IV. DENSITY FUNCTION ANALYSIS

In this section, we prove the correctness of the density gradient formulation and numerical solution, and analyze the rationale of dc removal and the advantages over previous placement density methods. Given its high performance on standard cell circuits, we extend the density function to handle the mixed-size placement in a generalized way. Fig. 3 shows the progression via a density-only mixed-size placement by ePlace-MS, where standard cells and macros are smoothly co-optimized toward even density distribution. Tables II and III show that our density function has the best performance with shortest wirelength and smallest density overflow versus all the mixed-size placers in [13], [17], [46], and [47].

A. Density Function

A placement density function is developed in [24] based on the electrostatic analogy, which is therefore

Here OVLP denotes physical overlap among all the objects. Computation costs \( O(n \log n) \) time via scanline and segment-tree data structure.
and

\( \rho \) Iter \( = \) density thus the best performance of our density function in Table III, indicating the fewest violations to the target density function achieves the minimum density overflow as shown.

**Charging the density function**

\( \) charge, the density function named eDensity. Modeling every object as a positive

\( \) overflow are denoted by density characterized by grayscale. Total potential energy and total density only density forces (denoted by red arrows) with the magnitude of the grid placement on the MMS ADAPTEC1 benchmark. The placement is driven by algorithms [3], [13], [15], [17], [21], [39], [46], [47], our den-

\( \) reached. Compared to all the previous mixed-size placement end. The electrostatic equilibrium state is coupled with

\( \) is the electric quantity of the charge \( i \), it equals the area of the respective object \( i \). \( \psi_i \) is the local potential. Also, as

\( \) system energy equals the sum of mutual potential energy between all the pairs of charges, we have a factor of 1/2 for the energy of each single charge. A well-defined Poisson’s equation in (7) correlates the density distribution \( \rho(x, y) \) with the potential distribution \( \psi(x, y) \), where \( x \) and \( y \) are spatial coordinates. We enforce Neumann boundary condition (i.e., zero gradient at the boundary of the density function or placement domain) to prevent objects from moving outside the placement region \( R \). Specifically, the horizontal density gradients along the two vertical boundaries are equivalent to zero, vice versa,

\[
N(v) = \frac{1}{2} \sum_{i \in V} N_i(v) = \frac{1}{2} \sum_{i \in V} q_i \psi_i(v). \quad (6)
\]

such that movement toward the placement boundaries will be gradually slowed down and finally stopped

\[
\begin{align*}
\vec{n} \cdot \nabla \psi(x, y) &= -\rho(x, y) \\
\int_R \rho(x, y) = \int_R \psi(x, y) &= 0.
\end{align*}
\quad (7)
\]

Here \( \vec{n} \) is the outer normal vector at the boundary \( \partial R \). We use \( \vec{\xi}(x, y) = \nabla \psi(x, y) \) to denote the electric field distribution. The electric force on each charge \( i \) equals \( q_i \vec{\xi}_i(v) \), where \( \vec{\xi}_i = (\xi_{ix}, \xi_{iy}) \) is the local field vector and can be decomposed into its horizontal (\( \xi_{ix} \)) and vertical (\( \xi_{iy} \)) components.
Our density function \( N(\mathbf{v}) \) is generalized. In contrast to prior nonlinear placers [13], [15], there is no special handling or smoothing applied to movable macros or fixed blocks. Please refer to Section IV-E for a more detailed advantage analysis. The global smoothness of \( N(\mathbf{v}) \) [by (6) and (7)] indicates that the local movement of any object will change the potential map in the global scale. The potential energy of all the objects will thus be changed by the movement of any single object \( i \).

B. DC Removal

eDensity correlates even distribution of placement objects with the electrostatic equilibrium state. However, since all the objects are mapped to positively charged particles (with electric quantity set as the object area) as Fig. 4(a) shows, the electric forces applied on all the charges are purely repulsive. Such repulsive force will keep pushing all the charges toward infinity. On the other hand, if moving outside the placement domain \( \Delta x \) of the object \( i \), the local movement of any object will change the potential energy of all the objects. Let \( \rho_i \) denote the dc of the global density distribution, i.e., the constant terms in the gradient of \( N(\mathbf{v}) \). Since \( \rho_i \) equals the average grid density of \( \sum_b \rho_b = 0 \) after removing dc (\( \rho_{\text{avg}} \)) from \( \rho_i \), we have \( \sum_b \rho_b = 0 \) if \( \forall b \in B \). As a result, we have \( \sum_b \rho_b = 0 \) after removing dc (\( \rho_{\text{avg}} \)) from \( \rho_i \), \( \forall b \in B \). Similarly, the density gradient of \( N(\mathbf{v}) \) with respect to the vertical movement of \( i \) is \( q_i \xi_i(\mathbf{v}) \). As a result, \( q_i \xi_i(\mathbf{v}) \) is consistent with the gradient descent of the density cost (potential energy) function.

C. Correctness of Gradient Formulation

As discussed in Section IV-A, we use \( q_i \xi_i \) as the gradient of the density function \( N(\mathbf{v}) \) with respect to the horizontal movement of the charge \( i \). However, by directly differentiating \( N(\mathbf{v}) \) with respect to \( x_i \), we obtain the following formula:

\[
\frac{\partial N(\mathbf{v})}{\partial x_i} = \frac{1}{2} \left( \frac{\partial N_i(\mathbf{v})}{\partial x_i} + \frac{\partial \left( \sum_{j \neq i} N_j(\mathbf{v}) \right)}{\partial x_i} \right)
\]

(8)

which is different from \( q_i \xi_i \), with one extra term. By the nature of electrostatics, the potential at each charge \( i \) is the superposition of the potential contributed by all the remaining charges in the system. Let \( N_i \) denote the potential energy of charge \( i \) contributed by \( j \), vice versa. For an electrostatic system defined on the 2-D plane, we have

\[
N_i(\mathbf{v}) = - \frac{q_i q_j}{2\pi \epsilon_0} \ln \left( \frac{r_{ij}(\mathbf{v})}{r_{\text{ref}}} \right) = N_j(\mathbf{v})
\]

(9)

where \( r_{ij} \) is the physical distance between the two charges \( i \) and \( j \) based on the placement solution \( \mathbf{v} \). \( r_{\text{ref}} \) is the reference distance where the potential by charge \( i \) (j) diminishes to zero, in this paper, we see it as the dimension of placement domain \( R \). As a result, we have \( N_i(\mathbf{v}) = N_j(\mathbf{v}) \). Thus the mutual potential energy of each pair of charges \( i \) and \( j \) are equivalent. By the principle of potential superposition, we have

\[
N_i(\mathbf{v}) = \sum_{j \neq i} N_j(\mathbf{v}) = \sum_{j \neq i} N_j(\mathbf{v}).
\]

(10)

Therefore

\[
\frac{\partial N(\mathbf{v})}{\partial x_i} = \frac{1}{2} \left( \frac{\partial N_i(\mathbf{v})}{\partial x_i} + \frac{\partial \left( \sum_{j \neq i} N_j(\mathbf{v}) \right)}{\partial x_i} \right)
\]

(11)

so, \( q_i \xi_i(\mathbf{v}) \) is the actual gradient of \( N(\mathbf{v}) \) with respect to the horizontal movement \( \Delta x_i \) of the object \( i \). Similarly, the density gradient of \( N(\mathbf{v}) \) with respect to the vertical movement of \( i \) is \( q_i \xi_i(\mathbf{v}) \). As a result, \( q_i \xi_i(\mathbf{v}) \) is consistent with the gradient descent of the density cost (potential energy) function.

D. Correctness of Numerical Solution

Poison’s equation in (7) is solved via spectral methods [43] using a 2-D fast Fourier transform (FFT) applied to the spatial domain. Sinusoidal waveform approaches zero at the end of each function period, such behavior well matches the Neumann condition \( \mathbf{n} \cdot \nabla \psi(x, y) = 0 \), \( \forall(x, y) \in \partial R \) in (7), which requires zero gradient along the boundaries. As a result, we apply discrete sinusoidal transformation (DST) to the spatial field distribution \( \xi(x, y) \). As the electric potential and density distribution are the integral and derivative of the field, i.e., \( \nabla \psi(x, y) = -\xi(x, y) \) and \( \rho(x, y) = \nabla \cdot \xi(x, y) \), we reconstruct them via discrete cosine transformation (DCT). Based on an even mirroring and periodic extension, we have the DCT coefficients \( a_{j,k} \) of the spatial density distribution \( \rho(x, y) \) as

\[
a_{j,k} = \frac{1}{m^2} \sum_{x=0}^{m-1} \sum_{y=0}^{m-1} \rho(x, y) \cos(w_j x) \cos(w_k y)
\]

(12)

where \( w_j \) and \( w_k \) are frequency components. The density \( \rho(x, y) \) can then be spatially expressed as

\[
\rho(x, y) = \sum_{j=0}^{m-1} \sum_{k=0}^{m-1} a_{j,k} \cos(w_j x) \cos(w_k y).
\]

(13)

As \( \nabla \cdot \psi(x, y) = -\rho(x, y) \), we have the spatial potential distribution expressed as

\[
\psi(x, y) = \sum_{j=0}^{m-1} \sum_{k=0}^{m-1} a_{j,k} \cos(w_j x) \cos(w_k y).
\]

(14)

Notice that for every pair of horizontal and vertical frequency components \( \cos(w_j x) \) and \( \cos(w_k y) \) from density \( \rho(x, y) \) and potential \( \psi(x, y) \), we have the Poisson’s equation
in (7) well satisfied in the numerical perspective as shown below
\[
\nabla \cdot \nabla \psi(x, y) = \frac{\partial^2 \psi(x, y)}{\partial x^2} + \frac{\partial^2 \psi(x, y)}{\partial y^2} = \\
\left( - \sum_j \sum_k \frac{a_{jk} w_j^2}{w_j^2 + w_k^2} \cos(w_j x) \cos(w_k y) \right) \\
+ \left( - \sum_j \sum_k \frac{a_{jk} w_k^2}{w_j^2 + w_k^2} \cos(w_j x) \cos(w_k y) \right) \\
= - \sum_j \sum_k a_{jk} \cos(w_j x) \cos(w_k y) = -\rho(x, y).
\]

We remove the dc component \(\rho_{\text{avg}}\) from \(\rho(x, y)\) by setting \(a_{0,0} = 0\). The spatial field distribution is expressed as
\[
\xi(x, y) = \sum_j \sum_k \frac{a_{jk} w_j}{w_j^2 + w_k^2} \sin(w_j x) \cos(w_k y),
\]
\[
\tilde{\xi}(x, y) = \sum_j \sum_k \frac{a_{jk} w_k}{w_j^2 + w_k^2} \cos(w_j x) \sin(w_k y),
\]
which also satisfies (7) in the numerical perspective since
\[
\nabla \psi(x, y) = \left( \frac{\partial \psi(x, y)}{\partial x}, \frac{\partial \psi(x, y)}{\partial y} \right) = \\
\left( - \sum_j \sum_k \frac{a_{jk} w_j}{w_j^2 + w_k^2} \sin(w_j x) \cos(w_k y) \right) \\
- \sum_j \sum_k \frac{a_{jk} w_k}{w_j^2 + w_k^2} \cos(w_j x) \sin(w_k y) \right) = \\
( -\xi_1(x, y), -\xi_2(x, y) ) = -\xi(x, y).
\]

Given \(|V| = n^2\) movable objects in the netlist, ePlace-MS decomposes the placement region \(R\) into \(m \times m\) grids, where \(m = \sqrt{n^2}\), to have one object per grid on average. The above 2-D-FFT computation thus costs exactly \(O(n^2 \log n)\) runtime per iteration. As the number of fillers is at essentially the same order of the number of all the standard cells and movable macros, the complexity is essentially \(O(n \log n)\). The well-formed density gradient, global density smoothness and low computational complexity enables ePlace-MS to conduct placement on the flat netlist and the flat density grid with constantly high resolution. Compared to all the prior mixed-size nonlinear placers [3], [13], [15] with multilevel netlist clustering and grid coarsening, ePlace-MS avoids quality loss due to the suboptimal clustering and low density resolution especially at early iterations.

### E. Advantage Analysis

Density force modeling remains quite a controversial problem [31] in quadratic placement, where the best location of anchor point for each object is usually unclear. Relaxed quadratic spreading and linearization [45] nullifies the top 10% density force vectors to suppress over-spreading of standard cells, while the empirical tuning lacks theoretical support and may not guarantee convergence. Reference [7] uses Green’s function to determine appropriate positions of cell anchors. The 2-D convolution makes the complexity to be \(O(n^2)\) thus is computationally expensive. Kraftwerk2 [44] determines the anchor position via solution to the Poisson’s equation. Due to the function order restriction in the quadratic placement infrastructure, the density cost is degraded from exponential to linear, which helps achieve convexity and efficiency but loses quality. SimPL [16] and ComPLx [17] determine the anchor position via recursive bi-partitioning, while convergence is theoretically promised via the primal-dual framework. Nevertheless, the solution quality is sensitive toward the initial solution. Moreover, it is hard to tell how much the optimum solution would follow the initial layout with minimum wirelength yet high overlap.

Nonlinear placement has no restriction on function orders thus ensures more flexibility in density modeling. However, the nonconvexity of the density function remains a headache to the nonlinear solvers. Bell-shape method [35] covers only adjacent grids in the local scale. Iterative grid uncoarsening is usually conducted in prior nonlinear placers [13], [15] to keep consistent with the scale of clustered netlist. However, the quality degradation due to low density resolution is not negligible. Besides, such local density smoothness would force objects to detour around obstacles thus inevitably lower the convergence rate. Notice that bell-shape method could realize fully global density smoothness by parameter adjustment, nevertheless, the regarding complexity scales up to \(O(n^2)\), which is numerically expensive. Helmholtz equation in [3] smooths the density in global scale with only \(O(n \log n)\) runtime complexity. However, sub-optimality in the choice of the linear factor \(\epsilon\) in the Helmholtz equation [2, eq. (7)] introduces noises. Moreover, there is no formulation of density gradient functions in [3], where up to millions of constraints are simultaneously applied to all the grid density, which complicates the problem, degrades the placement quality and efficiency.

eDensity concisely formulates the placement density problem using the closed-form equation in (6). By differentiating it we derive the gradient vector to direct density cost reduction, where by (5) only one penalty factor is needed for force balancing with wirelength. eDensity numerically solves the partial differential equation via the spectral methods [43] in (14) and (15). Based on the nice properties of FFT, it consumes exactly only \(O(n \log n)\) runtime per iteration. At each grid, the local electric potential and field are impacted by the global density distribution, while objects driven by density forces are able to freely move over blockages or macros, as Fig. 5 shows. Moreover, the global smoothness enables all the movable objects in over-filled regions to detect whitespace at remote area, as illustrated in Fig. 3, which helps quickly converge to the objective of even density. To this end, unlike all the prior methodologies in literature, eDensity approaches density equalization via directly simulating the behavior of a real electrostatic system, which in reality will always transfer toward the states of lower potential energy (until the energy decreases to zero), therefore theoretically guarantees the global convergence of eDensity. The nature of simulation enables us to use constantly high density resolution throughout the whole global placement, without any potential misguidance to the nonlinear solver.

### V. Nonlinear Optimization for MGP

Our prior work shows high performance of Nesterov’s method on placing standard cell-based circuits. In this section, we extend it to handle mixed-size placement, where we observe consistently good performance as shown by the experimental results in Section IX. In the framework of ePlace-MS (Fig. 1), MGP uses Nesterov’s method to smoothly conduct simultaneous optimization on both macros and standard cells, as Fig. 5 shows. As a generalized approach, MGP
Algorithm 1 Nesterov's Method in ePlace-MS

Input: $a_k$, $u_k$, $v_k$, $v_{k-1}$, $\nabla f_{pre}(v_k)$, $\nabla f_{pre}(v_{k-1})$
Output: $u_{k+1}$, $v_{k+1}$, $a_{k+1}$
1: $a_k = BkTrk (v_k, v_{k-1}, \nabla f_{pre}(v_k), \nabla f_{pre}(v_{k-1}))$
2: $u_{k+1} = v_k - a_k \nabla f_{pre}(v_k)$
3: $a_{k+1} = \frac{1 + \sqrt{4a_k^2 + 1}}{2}$
4: $v_{k+1} = u_{k+1} + (a_k - 1) (u_{k+1} - u_k) / a_{k+1}$
5: return

B. Nesterov's Method

The flow of Nesterov's method used in ePlace-MS is illustrated in Algorithm 1. We use Lipschitz constant prediction together with steplength backtracking to control the speed of optimization. $a_k$ is an optimization parameter which is iteratively updated. There are two concurrently updated solutions, $u_k$ and $v_k$, where only $u$ is output as the final solution (at the end of mGP and eGP), while $v$ is used for steplength prediction. $\nabla f_{pre}$ denotes the preconditioned gradient vector, which will be discussed in Section VI. Initially, we set $a_0 = 1$ and have both $u_0$ and $v_0$ set as $mIP$. $BkTrk$ denotes steplength backtracking as shown in Section V-D. The convergence rate of Nesterov's method is proven to be $O(1/k^2)$ in [37], which achieves the upper-bound of global convergence rate of first-order optimization methods [36], on condition that the steplength $a_k$ satisfies (16) at every single iteration $k$

$$f(v_k) - f(v_k - a_k \nabla f(v_k)) \geq 0.5 a_k \| \nabla f(v_k) \|^2.$$  \hspace{1cm} (16)

Bisection search is suggested by [37] to generate the maximal $a_k$ without violating the inequality in (16). Similar to line search, it usually introduces significant runtime overhead. As [37] claims, the function $f(v_k - a_k \nabla f(v_k))$ would be evaluated by $O(\log L)$ times along the search direction for a single iteration, increasing the complexity to $O(n \log n \log L)$. Here $L$ is the Lipschitz constant as defined in Definition 1. As a result, step length prediction becomes necessary to accelerate the optimization process.

C. Lipschitz Constant Prediction

Instead of line search, we compute the steplength through a closed-form formula of the Lipschitz constant of the gradient.

handles macros and standard cells in exactly the same way (see macro shifting at each netlist declustering layer [13], formation of soft blocks by standard cells [46], [47], special density smoothing of macros [15], [18], macro shredding [17], etc.).

In each iteration, we compute the gradient and preconditioner, predict the Lipschitz constant, and adjust steplength via backtracking. Nesterov’s method solves the nonlinear problem iteratively till convergence is reached.

A. Existing Problems

Line search remains the major runtime bottleneck in conjugate gradient method, which is widely used in prior nonlinear placers [15]. In practice, it is not guaranteed that the steplength output by line search could satisfy the conjugacy requirement [42]. Specifically, the vector of current search direction may not be orthogonal (with respect to the Hessian matrix of the cost function) to all the previous vectors. Therefore, local convergence rate of conjugate gradient method [11], $2(\sqrt{\kappa} - 1/\sqrt{\kappa} + 1)^k$, cannot be guaranteed. Instead of line search, Chen et al. [5] determined the steplength via upperbounding the Euclidean distance of objects movement per iteration by a constant number. Such prediction assumes underestimation of steplength, which in general slows down the placement convergence rate. Moreover, steplength overestimation could still occur at some special area of the search space where the gradient changes sharply, therefore degrades the solution quality. As a result, a systematic solution with dynamic steplength adjustment and theoretical support becomes quite necessary.

Fig. 5. Snapshots of mGP progression in ePlace-MS-WA on the MMS ADAPTEC1 benchmark with standard cells, macros, and fillers shown by red points, black rectangles, and blue points. Total wirelength, total potential energy, total energy overflow, total OVLP and total mOVLP are denoted by $W$, $N$, $\tau$, $O$, and $Om$, respectively. (a) Iter = 50, $W = 34.76e6$, $N = 26.55e11$, $\tau = 84.31\%$, $O = 73.30e6$, and $Om = 37.85e6$. (b) Iter = 100, $W = 41.46e6$, $N = 91.4e10$, $\tau = 67.59\%$, $O = 59.11e6$, and $Om = 24.40e6$. (c) Iter = 125, $W = 44.4e6$, $N = 35.2e10$, $\tau = 55.31\%$, $O = 49.04e6$, and $Om = 16.68e6$. (d) Iter = 150, $W = 48.21e6$, $N = 98.78e9$, $\tau = 44.60\%$, $O = 40.10e6$, and $Om = 98.40e5$. (e) Iter = 175, $W = 52.74e6$, $N = 22.57e9$, $\tau = 39.72\%$, $O = 36.8e6$, and $Om = 80.98e5$. (f) Iter = 200, $W = 56.99e6$, $N = 46.31e8$, $\tau = 33.15\%$, $O = 31.56e6$, and $Om = 43.09e5$. (g) Iter = 235, $W = 61.05e6$, $N = 34.78e7$, $\tau = 20.34\%$, $O = 22.76e6$, and $Om = 11.05e5$. (h) Iter = 265, $W = 63.37e6$, $N = 25.51e6$, $\tau = 9.67\%$, $O = 16.48e6$, and $Om = 58.11e4$.
Definition 1: Given a multivariate convex function $f(v) \in C^{1,1}(H)$, $\exists L > 0$ s.t. $\forall u, v \in H$

$$\|\nabla f(u) - \nabla f(v)\| \leq L\|u - v\|. \quad (17)$$

$H$ as Hilbert space is a generalized notion of Euclidean space, $C^{1,1}(H)$ requires $f(v)$ with Lipschitz continuous gradient. As our objective is nonconvex, we leverage Nesterov’s method in an approximate way. Reference [37] states that $\alpha_k = L^{-1}$ satisfies the steplength requirement specified in (16) but lacks a formal proof. The rationale behind is that smaller Lipschitz constant indicates higher smoothness of the gradient thus faster convergence can be achieved via larger steplength, vice versa. Here we provide a proof to the statement that $\alpha_k = L^{-1}$ always satisfies (16) as Theorem 1.

Theorem 1: Given convex $f \in C^{1,1}(H)$ and $L$ defined in Definition 1, $\alpha \leq L^{-1}$ satisfies (16).

Proof: $\forall u, v \in H$, we have

$$f(v) - f(u) - \langle \nabla f(u), v - u \rangle$$

$$= \int \frac{\nabla f(v')}{u} dv' - \langle \nabla f(u), v - u \rangle$$

$$= \int \nabla f(u + \tau(v - u))d(\tau(v - u)) - \langle \nabla f(u), v - u \rangle$$

$$\leq \int \|\nabla f(u + \tau(v - u)) - \nabla f(u)\|\cdot\|v - u\|d\tau$$

$$\leq \int L\cdot\|\tau(v - u)\|\cdot\|v - u\|d\tau$$

$$= 0.5L\|v - u\|^2$$

where the first and second inequalities hold based on the Cauchy–Schwartz inequality [1] and the definition of Lipschitz constant in (17), respectively. Equation (18) indicates that

$$f(v) \leq f(u) + \langle \nabla f(u), v - u \rangle + 0.5L\|v - u\|^2. \quad (19)$$

Let $u = v_k$ and $v = v_k - \alpha_k\nabla f(v_k)$, based on (19)

$$f(u) - f(v) = f(v_k) - f(v_k - \alpha_k\nabla f(v_k))$$

$$\geq \langle \nabla f(v_k), v - v - \alpha_k\nabla f(v_k) \rangle - 0.5L\|v - v\|^2$$

$$= \|\nabla f(v_k)\| - 0.5\alpha_k^2L\|\nabla f(v_k)\|^2$$

$$\geq \alpha_k\|\nabla f(v_k)\|^2 - 0.5\alpha_k^2L\|\nabla f(v_k)\|^2$$

$$\geq (\alpha_k - 0.5\alpha_k^2L)\|\nabla f(v_k)\|^2$$

$$= 0.5\alpha_k\|\nabla f(v_k)\|^2$$

where the second inequality holds if we have $L \leq \alpha_k^{-1}$. ■

As a result, $L^{-1}$ can be used as the steplength to accelerate the algorithm without convergence penalty. Exact Lipschitz constant is very expensive to compute (even more time consuming than line search). Moreover, static estimation will be invalidated through iterative change of the cost function, as both the wirelength coefficient $\gamma$ in (3) and penalty factor $\lambda$ in (5) are being iteratively adjusted in ePlace-MS (more details can be found in [24]). As a result, we approximate the Lipschitz constant and steplength as follows:

$$\tilde{L}_k = \frac{\|\nabla f(v_k) - \nabla f(v_{k-1})\|}{\|v_k - v_{k-1}\|}, \alpha_k = \frac{1}{\tilde{L}_k} \quad (21)$$

where only $v$ is used for Lipschitz constant prediction. The computation overhead is negligible since both $\nabla f(v_k)$ and $\nabla f(v_{k-1})$ are known thus there is no extra computation.

D. Steplength Backtracking

We develop a backtracking method to enhance the prediction accuracy via preventing potential steplength overestimation by (21), which would unexpectedly misguide the nonlinear solver. Being used to generate the placement solutions for the current iteration $k$. Instead, our backtracking method predicts $\alpha_k$ using $v_k$ and $v_{k+1}$. At line 1 of Algorithm 2, we set the steplength computed by (21) as a temporary variable $\tilde{\alpha}_k$. The respective temporary solution $\tilde{v}_{k+1}$ (line 3) is used to produce a reference steplength. If it is exceeded by $\tilde{\alpha}_k$ (line 4), we update $\tilde{\alpha}_k$ and $\tilde{v}_{k+1}$ at lines 5 and 7 and do the backtracking circularly until the inequality at line 4 is satisfied. $v_k$ and $v_{k-1}$ are the placement solutions for the current iteration $k$ and the past iteration $k - 1$. $u_k$ is the other solution (at iteration $k$) simultaneously updated with $v_k$, as shown in Algorithm 1. $\epsilon = 0.95$ is the scaling factor to encourage earlier return of function BkTrk thus prevent over-backtracking, which could consume too much runtime with limited accuracy improvement. The runtime overhead is zero if the first check at line 4 is passed, since the newly computed gradient $\nabla f(v_{k+1})$ can be reused at the following iteration. Experiments show that the average number of backtracks per iteration over all the 16 MMS benchmarks [47] is only 1.037, indicating less than 4% runtime overhead on mGP. Disabling backtracking causes ePlace-MSWA (using the weighted-average wirelength model) to fail on MMS BIGBLUE4 and increase wirelength by 43.12% on average of the remaining 15 MMS benchmarks.

VI. PRECONDITIONING

This section introduces our development of the nonlinear preconformer, which is used by Nesterov’s method in Algorithm 1 and steplength backtracking in Algorithm 2. Preconditioning reduces the condition number of a problem, which is transformed to be more suitable for numerical solution. Traditional preconditioning techniques compute the
inverse of the Hessian matrix \( \mathbf{H}_f \) of the objective function \( f \). Preconditioning has broad application in quadratic placers \([17],[21],[45],[46]\) but none attempts in nonlinear placers \([3],[13],[15]\), mainly due to the nonconvexity of the density function. In this paper, we approximate the original Hessian \( \mathbf{H}_f \) with a positive definite diagonal matrix \( \mathbf{H}_f \) as the preconditioner. We multiply it to the gradient vector and use \( \mathbf{V}_{\text{pre}} = \mathbf{H}_f^{-1} \mathbf{V}_f \) to direct the nonlinear placement optimization. A preconditioned gradient vector \( \mathbf{V}_{\text{pre}} \) is used to stretch the function space to be more spherical in order to smooth and accelerate the numerical optimization. However, as the objective function of global placement is of large scale (usually millions of objects to place) and highly nonlinear, to compute the Hessian matrix becomes very expensive and indeed computationally impractical. As a result, we choose the Jacobi preconditioner using only the diagonal terms of the Hessian matrix \( \mathbf{H}_f \)

\[
\mathbf{H}_{fxx} \approx \mathbf{\tilde{H}}_{fxx} = \begin{pmatrix}
\frac{\partial^2 f}{\partial x_1^2} & 0 & \cdots & 0 \\
0 & \frac{\partial^2 f}{\partial x_2^2} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \frac{\partial^2 f}{\partial x_N^2}
\end{pmatrix}.
\]  

(22)

By (5), we have

\[
\mathbf{\tilde{H}}_f = \begin{pmatrix}
\mathbf{\tilde{H}}_{fxx} \\
0 \\
\mathbf{\tilde{H}}_{fyy}
\end{pmatrix} = \mathbf{\tilde{H}}_W + \lambda \mathbf{\tilde{H}}_N.
\]  

(23)

As \( (\partial^2 f(x_i)/\partial x_i^2) = (\partial^2 W(x_i)/\partial x_i^2) + \lambda (\partial^2 N(x_i)/\partial x_i^2) \), we need to separately compute or estimate \( (\partial^2 W(x_i)/\partial x_i^2) \) and \( (\partial^2 N(x_i)/\partial x_i^2) \) at every iteration.

A. Wirelength

Based on the LSE wirelength modeling equation shown in (3), we differentiate it to derive the gradient function of the wirelength of net \( e \) with respect to \( x_i \) as shown below:

\[
\frac{\partial W_e^{\text{LSE}}(x)}{\partial x_i} = \frac{\gamma}{\sum_{j \in E_e} \exp \left( \frac{x_j}{\gamma} \right)} \times \frac{\partial}{\partial x_j} \left( \sum_{j \in E_e} \exp \left( \frac{x_j}{\gamma} \right) \right)
\]

\[
+ \frac{\gamma}{\sum_{j \in E_e} \exp \left( -\frac{x_j}{\gamma} \right)} \times \frac{\partial}{\partial x_i} \left( \sum_{j \in E_e} \exp \left( -\frac{x_j}{\gamma} \right) \right)
\]

(24)

Via further differentiating (24) with respect to \( x_i \), we are able to derive the second-order gradient of the LSE function as below

\[
\frac{\partial^2 W_e^{\text{LSE}}(x)}{\partial x_i^2} = \frac{\exp \left( \frac{x_i}{\gamma} \right) \left( \sum_{j \in E_e} \exp \left( \frac{x_j}{\gamma} \right) \right)}{\gamma \left( \sum_{j \in E_e} \exp \left( \frac{x_j}{\gamma} \right) \right)^2}
\]

\[
+ \frac{\exp \left( -\frac{x_i}{\gamma} \right) \left( \sum_{j \in E_e} \exp \left( -\frac{x_j}{\gamma} \right) \right)}{\gamma \left( \sum_{j \in E_e} \exp \left( -\frac{x_j}{\gamma} \right) \right)^2}
\]

(25)

Similarly, we can derive the gradient function of the WA wirelength model by differentiating (4). However, the second-order differentiation of (4) is complicated, moreover, quite computationally expensive. As a result, we use the vertex degree of object \( i \) instead

\[
\frac{\partial^2 W_e^{\text{WA}}(x)}{\partial x_i^2} = -\sum_{e \in E_i} \frac{\partial^2 W_e(x)}{\partial x_i^2} = \sum_{e \in E_i} \frac{\partial^2 W_e(x)}{\partial x_i^2}.
\]

(26)

Since \( W(v) \) in both LSE and WA are strongly convex \([12],[35]\) and globally differentiable, the Hessian matrices are also positive definite with strictly positive eigenvalues. As a result, we can use the closed-form formula \( (\partial^2 W/\partial x_i^2) \) in (27) as the nonlinear wirelength preconditioner.

B. Density

By differentiating the density gradient function in (11), we could obtain the second-order derivative as below

\[
\frac{\partial^2 N(x)}{\partial x_i^2} = -q_i \frac{\partial}{\partial x_i} \sum_{\rho_i \neq \rho_i} = -q_i \rho_i.
\]

(28)


Fig. 6: Performance comparison of the three candidate density preconditioners versus a density-only placement on the MMS ADAPTEC1 benchmark.
distribution. The performance comparison of the three density preconditioners [no preconditioner, (29) and (28)] is shown in Fig. 6. Compared to the other two options, our proposed preconditioner using charge quantity $q_i$ (object area) achieves the highest effectiveness and efficiency in the convergence of the density cost minimization.

C. Summary

As a result, we use $(\partial^2 f(v)/\partial x^2) \approx (\partial^2 W(v)/\partial x^2) + \lambda q_i$ to approximate the $i$th diagonal term of the placement preconditioner $H_{xx}$ with respect to horizontal charge movement, while the preconditioner for the vertical charge movement can be derived in a similar way. Disabling the preconditioner causes ePlace-MS to fail on nine out of the totally 16 MMS benchmarks, since macros significantly differ from standard cells with much higher magnitude of gradients. As a result, unpreconditioned gradient makes macros with large area and high incidence degree to bounce between opposite placement boundaries, causing the solution to oscillate and hard to converge within limited number of iterations. On average of the remaining seven MMS benchmarks, the wirelength is increased by 24.63%, indicating high effectiveness of our preconditioner.

VII. MLG

Based on the mGP solution $v_{mGP}$, MLG legalizes the macro layout via a simulated annealing (SA) [19] based approach, as Fig. 8 shows. Unlike traditional SA-based floorplanners and macro placers [4], [6], [32], [46] which perturb floorplan expression then physically realize it, MLG uses SA to directly control macro motion.

1) We expect a high-quality solution from mGP. Only local macro shifts are expected in MLG, the shrunk design space can be well explored by SA.

2) Our SA engine is more efficient with only minor position change to each single macro.

3) After each random perturbation of floorplan expression, the respective floorplan realization may cause significant layout change, which is time consuming and could induce unexpected quality degradation.

Similar to Timberwolf [40], however, MLG legalizes macros rather than detailedly place cells. As Fig. 7 shows, MLG can be decomposed into two levels. At each iteration of the outer loop (mLG iteration), we update the cost $f_{mLG}(v)$ by

$$f_{mLG}(v) = \text{HPWL}(v) + \mu_D D(v) + \mu_O O_m(v)$$

(30)

where HPWL($v$), $D(v)$, and $O_m(v)$ denote the total wirelength, total standard-cell area covered by macros and total mOVLP. MLG is set as constrained optimization.

---

3We set 3000 as the upper limit of iterations in ePlace-MS.

4Authorized licensed user limited to: Univ of Calif San Diego. Downloaded on October 23, 2023 at 22:32:53 UTC from IEEE Xplore. Restrictions apply.
Iterduct experiments on the MMS benchmarks [47], as shown in Table I. MMS benchmarks inherit the same netlists and den-
wirelength, potential energy, density overflow, and OVLP are denoted by cGP by ePlace-MS-W A on the MMS ADAPTEC1 benchmark. The total

filler redistribution (c) standard cell and filler co-optimization (d) after

LU

we set

λ
cGP to recover the aggressiveness of mGP. As cGP section

920 2.67 GHz CPU and 12 GB memory. The FFT package

and execute the program in a Linux machine with Intel i7

mGP iteration

λ

mGP iterations divided by 10.

m

in order to maintain the uniqueness of the analytic solution.

also fixed IO blocks inserted within the placement domain

benchmarks but have all the macros freed to place. There are

all the MMS benchmarks, the wirelength will be increased by

to compensate extra density violation by fillers. On average of

results to relocate fillers to their best sites. The resulting solution

standard cells fixed, a filler-only placement runs for 20 itera-

filler layout and may induce huge macro-filler overlap. As a

layout and converges faster than mGP. mLG is unaware of

Fig.9 shows, cGP causes small changes to the standard-cell

fixed, cGP uses the same nonlinear algorithm of mGP. As

second-phase global placement of standard cells. With macros

crashes on every MMS circuit as reported in [47] thus is not

machine. FLOP is not available due to IP and other issues, thus

obtained the binaries of four placers and executed them on our

categories of algorithms (as discussed in Section I) are

indicates further improvement space thus potentially better

unified-NR (with macro rotation and flipping disabled) fails

ect and buffers, therefore facilitate the following design

Following the contest policy in ISPD 2006 [33], there is a

benchmark-specific density upper-bound ρl for eight out of

the totally 16 circuits. This target density ρl helps produce

whitespace among circuit objects to accommodate intercon-

and buffers, therefore facilitate the following design stages of routing, timing correction, etc. By the benchmark protocol [33], exceeding ρl will penalize the wirelength by sHPWL = HPWL × (1 + 0.01 × τavg), where τavg denotes the

totally 16 circuits. This target density ρl helps produce

white space among circuit objects to accommodate intercon-

and buffers, therefore facilitate the following design stages of routing, timing correction, etc. By the benchmark protocol [33], exceeding ρl will penalize the wirelength by sHPWL = HPWL × (1 + 0.01 × τavg), where τavg denotes the scaled density overflow per bin and scaled HPWL (sHPWL) is the scaled wirelength. More detailed circuit statistics of MMS benchmarks can be found in [47]. After cGP is completed, ePlace-MS invokes the detailed placer in [13] for the legalization and cDP. There is no benchmark specific parameter tuning in this paper, and we use the official scripts from [47] to evaluate the performance of all the placers in our experiments.

Seven state-of-the-art mixed-size placers covering two
categories of algorithms (as discussed in Section I) are

included for performance comparison, namely, Capo10.5 [39],

FastPlace3.0 [46], ComPlx (v13.07.30) [17], POLAR [21],
mPL6 [3], FLOP [47], and NTUplace3-unified [13]. We have

obtained the binaries of four placers and executed them on our

FLOP is not available due to IP and other issues, thus we
cite their performance from [47]. Capo10.5 and mPL6 fail
to work with MMS benchmarks in our machine, so we cite

the respective results also from [47] instead. Also, APlace3 [15]
crashes on every MMS circuit as reported in [47] thus is not

inclusion in the results. MP-tree [6] and CG [4] are not avail-
able due to the industrial copyrights, while their results on

MMS benchmarks are also not available. However, as both of them have been outperformed by NTUplace3-unified [13] with on average 21% and 9% shorter wirelength (reported in [13, Table V]), we do not include them in our experiments.

The experimental results of HPWL and sHPWL on the

MMS circuits are shown in Table II. As shown in Table I,

there are no target density constraints for the first eight cir-
cuits (i.e., 100%) thus no density penalty on the wirelength.

In other words, HPWL equals sHPWL for the first eight

MMS testcases in Table II as marked with †. NTUplace3-

unified-NR (with macro rotation and flipping disabled) fails

on two MMS benchmarks (NEWBLUE3 and NEWBLUE7)

with the average wirelength, density overflow, and runtime

computed based on the other fourteen benchmarks. Compared
to all the placers in the experiments, ePlace-MS produces the

best solutions with the shortest wirelength for fourteen out of

the totally 16 testcases. Besides, it outperforms the leading-

edge mixed-size placer NTUplace3 [13] by up to 22.98% shorter

wirelength¹ and on average 8.22% shorter wirelength over all the MMS circuits. Notice that unlike NTUplace3, ePlace-MS does not allow macro rotation or flipping, which indicates further improvement space thus potentially better solution quality. The statistics of density overflow (i.e., the amount of violations to the testcase dependent target density ρl as specified in Table I) is shown in Table III. The respective results of Capo10.5, FLOP-NR, and mPL6 are not available from respective publications [47]. ePlace-MS obtains consistently the lowest density overflow at all the eight testcases (with predefined target density), showing the best performance of our density modeling method eDensity. The runtime statistics is shown in Table IV. On average of

cGP mitigates the quality overhead due to mLG via a

second-phase global placement of standard cells. With macros

fixed, cGP uses the same nonlinear algorithm of mGP. As

Fig. 9 shows, cGP causes small changes to the standard-cell

layout and converges faster than mGP. mGP is unaware of

filler layout and may induce huge macro-filler overlap. As a

result, fillers are retrieved and randomly distributed first. With

standard cells fixed, a filler-only placement runs for 20 itera-
tions to relocate fillers to their best sites. The resulting solution

with low density cost ensures placement of standard cells not
to compensate extra density violation by fillers. On average of

all the MMS benchmarks, the wirelength will be increased by

6.53% if we skip filler-only placement.

cGP then co-optimizes standard cells with fillers. The initial

penalty factor λcGP is determined based on that by the last

mGP iteration λmGP. As λ will be multiplied by up to 1.1,

we set λcGP = λmGP × 1.1m, using m buffering iterations for

cGP to recover the aggressiveness of mGP. As cGP section

of Fig. 2 shows, the wirelength (overlap) reduces (increases)

sharply for an optimal initial solution (similar to mIP). By

increasing λcGP iteratively, cGP reduces the overlap with mild

wirelength overhead. In practice, we set m as the number of

mGP iterations divided by 10.

IX. EXPERIMENTS AND RESULTS

We implement ePlace-MS using C programming language

and execute the program in a Linux machine with Intel i7

920 2.67 GHz CPU and 12 GB memory. The FFT package

from [38] is used in ePlace-MS to perform DCT and DST

operations. To validate the performance of ePlace-MS, we

conduct experiments on the MMS benchmarks [47], as shown

in Table I. MMS benchmarks inherit the same netlists and
density constraints ρl from ISPD 2005 [34] and ISPD 2006 [33]
benchmarks but have all the macros freed to place. There are
also fixed IO blocks inserted within the placement domain

in order to maintain the uniqueness of the analytic solution.

Authorized licensed use limited to: Univ of Calif San Diego. Downloaded on October 23,2023 at 22:32:53 UTC from IEEE Xplore. Restrictions apply.
all the 16 MMS benchmarks, ePlace-MS runs faster than Capo10.5, FLOP, ComPLx, and mPL6, and shows essentially the same efficiency with NTUplace3. Despite longer runtime than FastPlace3.0 and POLAR, ePlace-MS produces on average 19.47% and 32.03% shorter wirelength. In general, ePlace-MS outperforms all the mixed-size placement algorithms in literature and achieves good results on both LSE and WA wirelength models, showing that our density function and nonlinear optimization algorithm have high and stable performance, which are not dependent on specific wirelength models.

Fig. 10 shows the CPU breakdown of ePlace-MS-WA on average of all the MMS benchmarks. mGP is the most effective placement stage (as Fig. 2 shows) and consumes the longest runtime. A further breakdown of mGP by Fig. 11 illustrates that computation of density and wirelength gradients and other operations (Lipschitz constant prediction, parameter update, etc.) consume 57%, 29%, and 14% runtime of mGP.

Authorized licensed use limited to: Univ of Calif San Diego. Downloaded on October 23,2023 at 22:32:53 UTC from IEEE Xplore. Restrictions apply.
TABLE IV

| Runtime (Minutes) on the MMS Benchmark Suite [47]. MAC = Macros, CP = CAPO, FP = FastPlace, CPX = COMPLX, NP3U = NTUPLACE3-UNIFIED, NR = No Rotation or Flipping of Macros. Cited Results Are Marked With * |
|-----------------|-----------------|----------------|-----------------|-----------------|----------------|
| Categories       | Constructive    | One-Stage      | ePlace-MS       |
|                  | CPU | PLOP | MACROS | FP3.0 | CPx | POLAR | mPLx | NP3U | NR | NP3U | LSE | WA |
| ADAPtec1         | 92.78 | 12.03 | 13.73 | 3.32 | 5.43 | 4.85 | 40.07 | 5.63 | 6.30 | 5.25 | 5.47 |
| ADAPtec2         | 122.88 | 20.32 | 19.87 | 3.28 | 21.55 | 9.73 | 47.83 | 8.45 | 6.30 | 7.58 | 7.43 |
| ADAPtec3         | 282.86 | 32.38 | 35.27 | 6.73 | 14.07 | 10.72 | 99.72 | 15.10 | 10.37 | 26.22 | 27.3 |
| ADAPtec4         | 291.73 | 39.97 | 40.45 | 6.80 | 16.87 | 12.52 | 99.92 | 9.40 | 8.92 | 56.60 | 29.35 |
| BIGBLUE1         | 140.97 | 28.93 | 33.45 | 5.05 | 4.25 | 4.58 | 47.75 | 12.07 | 10.50 | 7.85 | 7.82 |
| BIGBLUE2         | 294.12 | 39.55 | 51.83 | 5.32 | 49.80 | 12.28 | 203.37 | 17.08 | 16.78 | 13.97 | 13.70 |
| BIGBLUE3         | 9165.35 | 117.28 | 990.62 | 20.27 | 176.36 | 46.07 | 159.13 | 47.30 | 58.13 | 82.20 | 72.98 |
| BIGBLUE4         | 1825.79 | 230.27 | 327.85 | 36.73 | 126.02 | 56.63 | 397.80 | 115.95 | 92.17 | 141.37 | 204.15 |
| ADAPtec5         | 399.28 | 84.13 | 54.88 | 15.22 | 13.15 | 21.37 | 377.27 | 53.78 | 46.07 | 50.27 | 48.35 |
| NEWBLUE1         | 52.22 | 22.80 | 16.87 | 5.03 | 4.58 | 6.20 | 52.85 | 11.85 | 10.75 | 11.70 | 10.87 |
| NEWBLUE2         | 135.93 | 44.10 | 40.23 | 6.92 | 31.02 | 21.93 | 106.73 | 13.35 | 15.00 | 51.12 | 62.40 |
| NEWBLUE3         | 1222.32 | 38.93 | 45.95 | 12.08 | 36.30 | 39.27 | 293.72 | N/A | 58.06 | 30.97 | 17.33 |
| NEWBLUE4         | 109.82 | 42.92 | 40.92 | 9.05 | 10.57 | 13.88 | 162.20 | 28.05 | 32.07 | 28.27 | 29.73 |
| NEWBLUE5         | 275.80 | 146.68 | 152.72 | 21.60 | 35.78 | 30.62 | 413.43 | 82.03 | 77.50 | 55.47 | 63.40 |
| NEWBLUE6         | 301.27 | 157.50 | 159.38 | 18.37 | 21.25 | 28.02 | 218.53 | 62.97 | 65.73 | 112.62 | 69.65 |
| NEWBLUE7         | 723.10 | 312.75 | 418.40 | 50.53 | 73.75 | 66.77 | 528.00 | N/A | 116.03 | 392.02 | 191.47 |

X. CONCLUSION

ePlace-MS is a generalized and effective placement algorithm to handle mixed-size circuits of very large scale. Using the density function eDensity based on electrostatics analogy, macros and standard cells are equalized by preconditioning and smoothly co-optimized by Nesterov's method. Steplength is determined via Lipschitz continuity together with a backtracking strategy to prevent overestimation. Unlike all the approaches in literature, ePlace-MS treats standard cells and macros in exactly the same way. The experimental results on MMS benchmarks validate its high and stable performance. The analytic nature of ePlace-MS ensures smooth integration of other design objectives (timing, routability, thermal, etc.). More details of ePlace-MS can be found at its homepage [8].

ACKNOWLEDGMENT

The authors would like to thank Dr. J. Z. Yan and Prof. C. Chu for providing the MMS placement benchmark suite, Dr. M.-K. Hsu and Prof. Y.-W. Chang for providing the binary of NTUplace3-unified, Dr. N. Viswanathan and Prof. C. Chu for providing the binary of FastPlace3.0 and FastPlace-DP, Dr. M.-C. Kim and Prof. I. L. Markov for providing the binary of SimPL and ComPLx, T. Lin and Prof. C. Chu for providing the binary of POLAR, and Y. Wang for the assistance in placement visualization.

REFERENCES


[31] Dr. Cheng was the recipient of the Best Paper Award from the IEEE International Symposium on Quality Electronic Design in 2013, and the Best Paper Award nominations from ACM/IEEE Design Automation Conference in 2014 and ACM/IEEE Asia and South Pacific Design Automation Conference in 2010. He has been a Technical Reviewer for the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION, and the IEEETF Journal since 2011.

Authors' photographs and biographies not available at the time of publication.