ePlace: Electrostatics-Based Placement Using Fast Fourier Transform and Nesterov's Method

JINGWEI LU, University of California, San Diego
PENGWEN CHEN, National Chung Hsing University
CHIN-CHIH CHANG, LU SHA, DENNIS JEN-HSIN HUANG,
and CHIN-CHI TENG, Cadence Design Systems, Inc.
CHUNG-KUAN CHENG, University of California, San Diego

We develop a flat, analytic, and nonlinear placement algorithm, *ePlace*, which is more effective, generalized, simpler, and faster than previous works. Based on the analogy between placement instance and electrostatic system, we develop a novel placement density function *eDensity*, which models every object as positive charge and the density cost as the potential energy of the electrostatic system. The electric potential and field distribution are coupled with density using a well-defined Poisson's equation, which is numerically solved by spectral methods based on fast Fourier transform (FFT). Instead of using the conjugate gradient (CG) nonlinear solver in previous placers, we propose to use Nesterov's method which achieves faster convergence. The efficiency bottleneck on line search is resolved by predicting the steplength using a closed-form equation of Lipschitz constant. The placement performance is validated through experiments on the ISPD 2005 and ISPD 2006 benchmark suites, where ePlace outperforms all state-of-the-art placers (Capo10.5, FastPlace3.0, RQL, MAPLE, ComPLx, BonnPlace, POLAR, APlace3, NTUPlace3, mPL6) with much shorter wirelength and shorter or comparable runtime. On average, of all the ISPD 2005 benchmarks, ePlace outperforms the leading placer BonnPlace with 2.83% shorter wirelength and runs 3.05× faster; and on average, of all the ISPD 2006 benchmarks, ePlace outperforms the leading placer MAPLE with 4.59% shorter wirelength and runs 2.84× faster.

Categories and Subject Descriptors: B.7.2 [**Integrated Circuits**]: Design Aids; J.6 [**Computer-Aided Engineering**]: Computer-Aided Design (CAD)

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Electrostatics, density function, analytic placement, nonlinear optimization, Poisson's equation, spectral methods, fast Fourier transform, Nesterov's method, preconditioning

ACM Reference Format:
Jingwei Lu, Pengwen Chen, Chin-Chih Chang, Lu Sha, Dennis Jen-Hsin Huang, Chin-Chi Teng, and Chung-Kuan Cheng. 2015. ePlace: Electrostatics-based placement using fast fourier transform and nesterov's method. ACM Trans. Des. Autom. Electron. Syst. 20, 2, Article 17 (February 2015), 34 pages.
DOI: http://dx.doi.org/10.1145/2699873

This article is partially based on our prior works "FFTPL: An Analytic Placement Algorithm Using Fast Fourier Transform for Density Equalization" published in IEEE 10th International Conference on ASIC (ASICON2013) and "ePlace: Electrostatics-Based Placement Using Nesterov's Method" published in IEEE/ACM 51st Design Automation Conference (DAC2014).
This work was supported in part by NSF CCF-1017864.
Author's addresses: J. Lu (corresponding author), Department of Computer Science and Engineering, University of California, San Diego, 9500 Gilman Drive, La Jolla, CA 92093; email: jlu@cs.ucsd.edu; P. Chen, Department of Applied Mathematics, National Chung Hsing University, No. 250, Guoguang Rd, Nan District, Taichung City, Taiwan; C.-C. Chang, L. Sha, D. J.-H. Huang, and C.-C. Teng, Cadence Design Systems, 2655 Seely Avenue, San Jose, CA 95134; C.-K. Cheng, Department of Computer Science and Engineering, University of California, San Diego, 9500 Gilman Drive, La Jolla, CA 92093.

DOI: http://dx.doi.org/10.1145/2699873

1. INTRODUCTION

Placement plays an important role in the VLSI physical design automation [Kahng et al. 2010; Lu 2010] for both random logic [Lu et al. 2013] and datapath-intensive components [Zhuang et al. 2013]. Placement performance largely impacts the downstream stages of power grid design [Wang et al. 2013], clock tree synthesis [Lu et al. 2012a], power optimization [Lu et al. 2012b], global detail routing [Lu and Sham 2013], postlayout simulation [He et al. 2012], and design variability [Zheng et al. 2014]. As the technology node enters the deep nanometer scale [ITRS 2011] with billion-transistor integration, the performance of the placement engine becomes dominant on the overall quality of the design. Lots of research on placement has been proposed in recent years [Markov et al. 2012]. The quality of placement results is usually evaluated by the total half-perimeter wirelength (HPWL), that correlates with timing [Lu et al. 2010]. and routability [Sham et al. 2009; Han et al. 2011]. HPWL is widely used in modern research developments [Kim and Markov 2012; Kim et al. 2012, 2010; Viswanathan et al. 2007a, 2007b; Chen et al. 2008; Kahng and Wang 2006; Chan et al. 2006] and public placement contests [Nam et al. 2005; Nam 2006].

Traditional placement methods can be generally divided into four categories, namely: (1) stochastic simulation; (2) min-cut partition; (3) quadratic minimization; and (4) nonlinear optimization, respectively. *Stochastic* approaches are usually based on simulated annealing techniques, of which one representative work is Timberwolf [Sechen and Sangiovanni-Vincentelli 1986]. Uphill climbing is probabilistically accepted to rescue the placer from local optima. Despite high solution quality, stochastic placement has high complexity and low convergence rate that induce poor scalability to large circuits. *Min-cut* approaches recursively simplify the problem by partitioning the instance (netlist and placement region) into smaller subinstances. Local optimum algorithms [Caldwell et al. 2000] are usually employed when the problem instance becomes sufficiently small. State-of-the-art works include Capo [Roy et al. 2006], Dragon [Taghavi et al. 2005], and Fengshui [Agnihorti et al. 2005]. However, improper partitioning at early stages could induce unrecoverable quality loss to the final solution. *Quadratic* approaches approximate the net length using a quadratic function which can be linearized by various net models [Spindler et al. 2008]. The differentiability enables gradient-based minimization techniques [Press et al. 2007]. Density equalization is performed by adding pseudopins and nets to the physically overlapped cells with a linear term introduced to the cost function [Eisenmann and Johannes 1998]. By solving the linear system, cells are iteratively dragged away from overfilled regions. State-of-the-art quadratic placers include FastPlace3.0 [Viswanathan et al. 2007b], RQL [Viswanathan et al. 2007a], SimPL [Kim et al. 2010], MAPLE [Kim et al. 2012], ComPLx [Kim and Markov 2012], BonnPlace [Struzyna 2013], and POLAR [Lin et al. 2013]. Despite high placement efficiency, the solution quality and robustness usually lag behind nonlinear placers. *Nonlinear* approaches refer to those algorithms based on a framework of nonlinear optimization. Wirelength and density are modeled using smooth mathematical functions, thus gradients can be analytically calculated. Wirelength models mainly include the log-sum-exp model [Naylor et al. 2001] and the weighted average model [Hsu et al. 2011]. Density models mainly include the bell-shaped function [Naylor et al. 2001], Gaussian equation [Chen et al. 2008], and Helmholtz equation [Chan et al. 2005]. The partial differential equation (PDE) can be solved by Green's function [Cong et al. 2008] or finite difference method [Chan et al. 2005]. By Lagrange relaxation or penalty method, the grid density constraints are integrated into the objective function and solved by the nonlinear CG method. State-of-the-art nonlinear placers include APlace3 [Kahng and Wang 2006], NTUPlace3 [Chen et al. 2008], and mPL6 [Chan et al. 2006]. Due to the high complexity of modeling functions,

nonlinear approaches employ multilevel cell clustering to simplify the problem and accelerate the algorithm. However, the quality overhead is not negligible.

In this work, we develop a flat analytic algorithm *ePlace* [Lu et al. 2014] for nonlinear global placement. ePlace is more effective, generalized, simpler, and faster than previous approaches. In contrast to the multilevel framework in prior nonlinear placers, our algorithm conducts placement on the flat netlist. Moreover, we develop a novel density function *eDensity* [Lu et al. 2013] modeling the placement instance as an electrostatic system for density equalization. Unlike hierarchical density grid structures used in prior works, ePlace sticks to a flat density grid with constantly high resolution. Compared to previous nonlinear placers [Kahng and Wang 2006; Chan et al. 2006; Chen et al. 2008], ePlace avoids quality loss due to suboptimal cell clustering and low density resolution, especially at early placement iterations. The density function is formulated as the system potential energy, while the density gradient is defined as the electric repulsive force. A modified Poisson's equation is proposed to couple the charge density with electric potential and field distribution, and a Neumann boundary condition is enforced to maintain the legality of the global placement solution. Based on the previous definition, a fast numerical method is proposed to solve Poisson's equation using spectral methods [Skollermo 1975] and well satisfies the boundary condition and makes the local density gradient aware of global density information. The time complexity is only $O(m \log m)$, where m is the total number of movable elements. Besides, we propose to use Nesterov's method [Lu et al. 2014] for nonlinear placement optimization. The steplength is determined as the inverse of the Lipschitz constant, which is dynamically predicted without computation overhead. The placement efficiency is improved by more than $2\times$ compared to the CG method (with line search). We further enhance the performance of the nonlinear solver using a preconditioning technique to statically approximate the Hessian matrix of the objective function. All the preceding innovations are integrated into the flat nonlinear placement algorithm ePlace, which is validated through experiments on the ISPD 2005 [Nam et al. 2005] and ISPD 2006 [Nam 2006] benchmark suites with high placement quality and efficiency achieved.

The remainder of the article is organized as follows. In Section 2, we review the previous nonlinear placement works and discuss the existing problems. In Section 3, we discuss the analogy of placement instance to the electrostatic system as well as the formulation of the density penalty and gradient. We propose a well-defined Poisson's equation in Section 4 with a fast numerical solution based on spectral methods. In Section 5, we propose to use Nesterov's method for solving the nonlinear placement problem with dynamic prediction of the Lipschitz constant and discuss our preconditioning technique. In Section 6, we discuss our global placement algorithm ePlace which is empirically validated in Section 7. We conclude the work in Section 8 and discuss future research directions.

2. ESSENTIAL CONCEPTS OF PLACEMENT AND PRIOR NONLINEAR ALGORITHMS

In this section, we introduce the essential concepts and problem formulation of analytic global placement. We then discuss the basic methods and existing problems of prior nonlinear optimization algorithms.

2.1. Essential Concepts of Placement

A placement instance is formulated as a hypergraph $G = (V, E, R)$, where V denotes the set of vertices (cells), E the set of hyperedges (nets), and R the placement region, respectively. We use V_m and V_f to denote the movable cells and fixed macros in the node set V. Let $n = |V_m|$ denote the number of movable placement objects. A *legal solution* satisfies the following three requirements.

—Every cell is accommodated using enough free sites in the placement region.
—Every cell is horizontally aligned with the boundaries of one placement row.
—There is no overlap between cells or macros.

Based on the legality constraint, a placer targets minimizing the total HPWL of all the nets. Let $\mathbf{v} = (\mathbf{x}, \mathbf{y})$ denote a placement solution, where $\mathbf{x} = \{x_i | i \in V_m\}$ and $\mathbf{y} = \{y_i | i \in V_m\}$ are the horizontal and vertical coordinates of all the cells. The HPWL of each net e is denoted as $HPWL_e(\mathbf{v})$ and defined in Eq. (1).

$$HPWL_e(\mathbf{v}) = \max_{i,j \in e} |x_i - x_j| + \max_{i,j \in e} |y_i - y_j|. \tag{1}$$

The total HPWL is then computed as $HPWL(\mathbf{v}) = \sum_{e \in E} HPWL_e(\mathbf{v})$ and we have the placement problem defined in Eq. (2).

$$\min_{\mathbf{v}} HPWL(\mathbf{v}) \text{ such that } \mathbf{v} \text{ is a legal solution.} \tag{2}$$

2.2. Definition of Global Placement

Global placement is usually regarded as a problem of constrained optimization. The placement region is uniformly decomposed into a set of $m \times m$ rectangular grids (bins) denoted as B. Based on a placement solution $\mathbf{v}$, let $\rho_b(\mathbf{v})$ denote the density of each grid b as expressed in Eq. (3).

$$\rho_b(\mathbf{v}) = \sum_{i \in V} l_x(b,i) l_y(b,i). \tag{3}$$

Here $l_x(b,i)$ and $l_y(b,i)$ denote the horizontal and vertical overlaps between the grid b and the cell i. Both $l_x(b,i)$ and $l_y(b,i)$ exhibit a rectangular shape not differentiable at boundary points. As Eq. (4) shows, a global placement problem targets a solution $\mathbf{v}$ with minimum total HPWL, subject to the constraint that the density $\rho_b(\mathbf{v})$ of all the grids is equal to or below a predetermined target placement density ρ_t.

$$\min_{\mathbf{v}} HPWL(\mathbf{v}) \text{ s.t. } \rho_b(\mathbf{v}) \leq \rho_t, \forall b \in B. \tag{4}$$

2.3. Wirelength Smoothing

As Eq. (1) shows, the wirelength function $HPWL(\mathbf{v})$ is not differentiable and hard to minimize. As a result, various smoothing techniques have been developed to improve the differentiability, thus convergence rate. Here we only discuss the horizontal part of the wirelength smoothing function while the vertical part can be obtained in a similar way.

A Log-Sum-Exp (LSE) wirelength model is proposed in Naylor et al. [2001] and widely used in recent nonlinear placers [Chan et al. 2006; Chen et al. 2008; Kahng and Wang 2006]. For each net $e = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ with n pins, the LSE function approximates the horizontal span $HPWL_e$ as shown.

$$W_e(\mathbf{v}) = \gamma \left(\ln \sum_{i \in e} \exp\left(\frac{x_i}{\gamma}\right) + \ln \sum_{i \in e} \exp\left(\frac{-x_i}{\gamma}\right) \right). \tag{5}$$

Here γ is the smoothing parameter that can be used to control the modeling accuracy.[1] As discussed in Wang et al. [2009], the modeling error is upper bounded by $\varepsilon_{LSE}(e) \leq \gamma \ln n$.

[1]The HPWL smoothing parameter γ cannot be set arbitrarily small due to the computation precision constraint.

A Weighted Average (WA) wirelength model is proposed in Hsu et al. [2011]. Eq. (6) shows the horizontal function of net e

$$W_e(\mathbf{v}) = \left(\frac{\sum_{i \in e} x_i \exp(x_i/\gamma)}{\sum_{i \in e} \exp(x_i/\gamma)} - \frac{\sum_{i \in e} x_i \exp(-x_i/\gamma)}{\sum_{i \in e} \exp(-x_i/\gamma)} \right), \tag{6}$$

where similarly γ is used for accuracy control. Hsu et al. [2011] show that the modeling error is upper bounded by $\varepsilon_{WA}(e) \leq \frac{\gamma \Delta x}{1+\exp \Delta x/n}$, which is roughly half that of $\varepsilon_{LSE}(e)$. In this work, we use the WA wirelength model for our nonlinear placement optimization.

2.4. Density Penalty

As Eq. (4) shows, a legal global placement solution requires all the $|B|$ grid density constraints to be satisfied simultaneously, where $|B|$ could be of million-scale or even larger in modern IC design. As a result, all the constraints are usually cast into a single penalty function $N(\mathbf{v})$ as shown in Eq. (7). By definition, all the $|B|$ density constraints will be satisfied if and only if we have $N(\mathbf{v}) = 0$.

$$\rho_b(\mathbf{v}) \leq \rho_t, \forall b \in B \;\Leftrightarrow\; N(\mathbf{v}) = 0 \tag{7}$$

Quadratic placement approaches usually model the density penalty as a linear or quadratic function that can be easily integrated into their objective function. The penalty in UPlace [Yao et al. 2005] is explicitly devised as a weighted sum of all the frequency components of the density function. Specifically, $N(\mathbf{v}) = \sum_{u,v} w_{u,v} a_{u,v}^2$, where u and v are the discrete frequency indexes, $w_{u,v}$ are the weight factors, and $a_{u,v}$ are the frequency coefficients. Notice that each frequency component is a differentiable wave function, of which the smooth curve can help direct gradient-based optimization in an effective way. The aforesaid penalty is fitted into a quadratic form and integrated into the objective function. Other quadratic placers [Eisenmann and Johannes 1998; Spindler et al. 2008; Viswanathan et al. 2007b; Kim et al. 2012; Kim and Markov 2012; Lin et al. 2013] modify the netlist by introducing anchor points, which implicitly produce the density penalty terms for the quadratic cost function.

Nonlinear placers have no constraints on the order of modeling functions, thus are able to design the penalty in more flexible ways. APlace3 [Kahng and Wang 2006] and NTUPlace3 [Chen et al. 2008] use a quadratic penalty function with respect to grid density, as Eq. (8) shows.

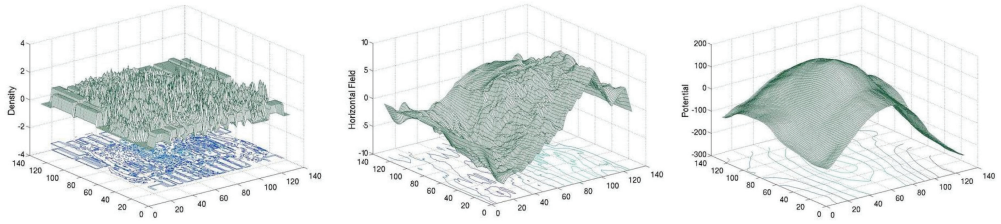
$$N(\mathbf{v}) = \sum_{b \in B} \left(\widetilde{\rho}_b(\mathbf{v}) - \rho_t \right)^2. \tag{8}$$

As the original density function $\rho_b(\mathbf{v})$ is not differentiable and hard to optimize, a smoothed density function $\widetilde{\rho}$ is used here by employing a "bell-shaped" local smoothing technique [Naylor et al. 2001]. In contrast to the penalty method discussed before, mPL6 [Chan et al. 2006] directly applies Lagrange multipliers to all the density constraints. The density function in Chan et al. [2006] is smoothed in a global scale by using the Helmholtz equation [Chan et al. 2005, Eq. (7)].

In this work, we model the placement instance as an electrostatic system and devise the density penalty $N(\mathbf{v})$ to be the system potential energy. In the remaining part of the article, we will use $N(\mathbf{v})$ to denote both density penalty and system energy. This modeling methodology is discussed in detail in Section 3 in terms of how the density penalty and gradient are defined. A fast numerical solution to the density and potential related Poisson's equation (Eq. (20)) is proposed in Section 4.

2.5. Nonlinear Optimization Formulation

Based on the smooth wirelength function $W(\mathbf{v})$ and density penalty function $N(\mathbf{v})$, nonlinear global placers [Chen et al. 2008; Kahng and Wang 2006] formulate the



(a) electric density **(b)** horizontal electric field **(c)** electric potential

Fig. 1. The snapshots of electric density, horizontal field, and potential distribution extracted at iteration 50. The placement is driven by only density force and conducted on the ISPD05 ADAPTEC1 benchmark.

objective function $f(\mathbf{v})$ using a penalty factor λ as follows:

$$\min_{\mathbf{v}} f(\mathbf{v}) = W(\mathbf{v}) + \lambda N(\mathbf{v}). \tag{9}$$

As both the wirelength function and the density penalty are smoothed and thus differentiable, gradient-based optimization methods [Shewchuk 1994] are used in prior nonlinear placers [Chen et al. 2008; Kahng and Wang 2006] to produce high-quality numerical solutions. Alternatively, Lagrange multipliers are also used [Chan et al. 2006] to formulate the objective function in a different form as

$$\min_{\mathbf{v}} f(\mathbf{v}) = W(\mathbf{v}) + \sum_{b \in B} \lambda_b |\widetilde{\rho}_b(\mathbf{v}) - \rho_t|. \tag{10}$$

Here λ_b denotes the multiplier on the density constraint of the bin b. This approach might consume longer runtime due to the computation demand on the multipliers. Multilevel cell clustering is employed in all the previous nonlinear placers [Chan et al. 2006; Chen et al. 2008; Kahng and Wang 2006] to accelerate the placement algorithm. Despite the efficiency improvement, the quality overhead due to suboptimal clustering is not negligible.

3. EDENSITY: A NOVEL DENSITY FUNCTION BY ELECTROSTATIC SYSTEM MODELING

We propose a novel formulation of the density penalty and gradient function, called *eDensity*, by modeling the entire placement instance as a two-dimension a independent electrostatic system. The distribution of electric potential and field is determined by all the elements in the system. Each node i (a cell or a macroblock) in the netlist is transformed into a positively charged particle (also denoted as i). The electric quantity q_i of the particle is set to be the node area A_i. The motion of a movable cell i is driven by the electric force $F_i = q_i \boldsymbol{\xi_i}$ formulated by the Lorentz force law, where $\boldsymbol{\xi_i}$ is the local electric field. Similarly, the cell potential energy N_i is calculated as $N_i = q_i \psi_i$, where ψ_i is the electric potential at cell i. The correlation between the original placement instance and the transformed electric system is illustrated in Figure 2. By Coulomb's law, the electric field and potential at cell i are the superposition of the contribution from all the remaining cells in the system. An example of charge density $\rho(x, y)$, horizontal electric field $\xi_x(x, y)$, and potential $\psi(x, y)$ distribution in the entire placement region R is shown in Figure 1.

3.1. System Modeling Using Electrostatic Equilibrium

Based on the system modeling, we correlate the global placement constraint of an even density distribution with the system state of the electrostatic equilibrium. The electric force helps direct the charge (cell) movement towards the equilibrium state. By Gauss

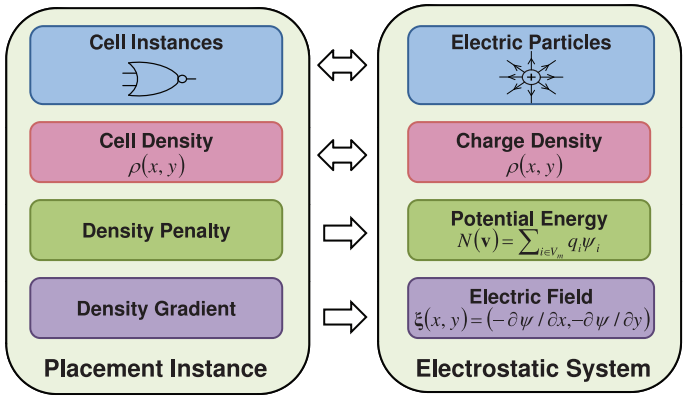


Fig. 2. The placement instance is modeled as an electrostatic system. Each movable cell or fixed macro is transformed into a positive charge with the electric quantity set to be the node area. The density force is set as the electric force that drives cells apart from each other. The target of density equalization is equivalent to the system state of electrostatic equilibrium.

law, the electric field equals the negative gradient of the potential as shown:

$$\boldsymbol{\xi}(x, y) = \left(\xi_x, \xi_y\right) = -\nabla\psi(x, y) = \left(-\frac{\partial\psi(x, y)}{\partial x}, -\frac{\partial\psi(x, y)}{\partial y}\right), \tag{11}$$

while the charge density equals the divergence of the electric field:

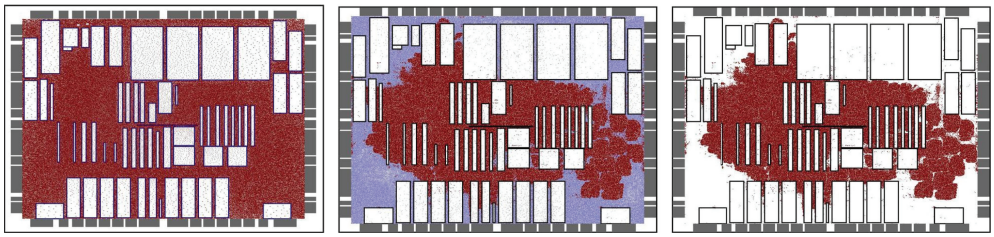
$$\rho(x, y) = \nabla \cdot \boldsymbol{\xi}(x, y) = -\nabla \cdot \nabla\psi(x, y) = -\left(\frac{\partial^2\psi(x, y)}{\partial x^2} + \frac{\partial^2\psi(x, y)}{\partial y^2}\right). \tag{12}$$

An electrostatic system with only positive charges will introduce only repulsion forces. The corresponding equilibrium state would have all the cells distributed along the chip boundaries where the global placement constraint is violated. As a result, we remove the direct-current (DC) component (i.e., the zero-frequency component) from the density distribution $\rho(x, y)$ to produce negative charges, while the integral of the density function over the placement region becomes zero. Specifically, since our density function transforms all the objects into positive charges, a positive-charge density distribution is thus produced. However, after removing the DC component from the spatial charge density distribution, underfilled placement regions with electric quantity below the original DC level become negatively charged. Meanwhile, the overfilled regions remain positively charged but with reduced electric quantity (DC is deducted from the original quantity). Cells at positively charged (i.e., highly overfilled) regions are attracted to the negatively charged regions, where the positive and negative charges neutralize each other. Meanwhile, cells at negatively charged regions will mostly keep still. In the end, the system reaches the electrostatic equilibrium state zero charge density over the entire placement region, while the total potential energy is reduced to zero. As a result, we model the placement density penalty and gradient using the system potential energy and electric field, respectively.

3.2. Density Penalty and Gradient Formulation

The total potential energy equals the sum of potential energy over all the charged elements of a new set V', which includes not only movable and fixed nodes from V, but also newly added fillers and dark nodes, as discussed next.

Filler insertion. Let A_m denote the total area of all the movable nodes, while A_{ws} denotes the total area of whitespace. The target of even density distribution will overly



(a) placement without filler insertion (b) placement with filler insertion (c) placement with filler insertion (removed after placement finishes)

Fig. 3. The distribution of standard cells and fillers at the end of global placement. Macros, standard cells, and fillers are shown by black rectangles, red dots, and blue dots, respectively. The total wirelength becomes shorter as fillers populate up whitespace, thus squeezing cells to be placed closer. The placement is conducted on the ISPD05 ADAPTEC1 benchmark using Nesterov's method.

spread the cells, thus increasing the wirelength, if we have the target density $\rho_t > \frac{A_m}{A_{ws}}$. Similar to Adya et al. [2003] and Chan et al. [2006] we add fillers into the system, all of equal size (rectangles), movable, and disconnected (with zero pins). Let V_{fc} denote the set of filler cells. The total area of filler cells is denoted as A_{fc} and defined as

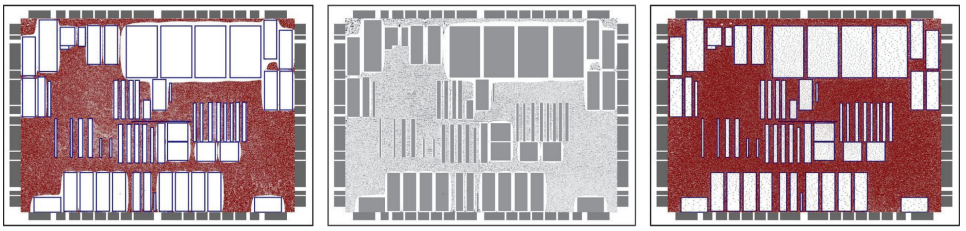
$$A_{fc} = \rho_t A_{ws} - A_m. \tag{13}$$

We illustrate the effect of filler insertion in Figure 3. The additional density force due to filler insertion will squeeze the cells to be placed closer to their connected neighbors with the density constraint still satisfied. The size of each filler i is denoted as A_i, which is determined based on the area distribution of the movable cells. Specifically, we set the filler size to be the average size of the mid-80% of movable cells. The remaining top and bottom 10% largest and smallest cells are considered as noise factors and filtered out. All the fillers are removed from the final solution of global placement.

Dark node insertion. As a generalized approach, our method could handle any irregularly shaped placement region without loss of quality or efficiency. Suppose the entire placement instance comprises a set of rectangular regions for cell placement. We impose a uniform grid R to cover all the placement regions. The total space within R but not belonging to any placement region will be decomposed into a set of rectangles, each modeled as a *dark node*, which is processed in the same way as that of a fixed object in the problem instance. Let V_d denote the set of all dark nodes and A_d the total area of all the dark nodes. Movable nodes will be stopped by the repelling force from the dark nodes when they are approaching the boundaries of any placement regions.

Density scaling. After the insertion of filler cells, we have the target density $\rho_t = \frac{A_m + A_{fc}}{A_{ws}}$. The area A_i of each fixed or dark node i must be scaled by the target density ρ_t, in order to maintain a globally equalized density distribution, otherwise, the density force becomes higher than that of cells and fillers and repels cells away, while the whitespace around the fixed nodes is emptied with wirelength overhead induced as Figure 4 shows. Notice that our density scaling method will not introduce any legalization issues. The electric quantity of each fixed or movable large macro is scaled down to the target placement density. Regions filled by small standard cells or covered by large macros will have the same charge density; there is no additional density force to drag cells away from macros. Without density scaling, it is impossible to achieve an even charge density distribution over the entire domain.

Potential energy computation. Let $V' = V_m \cup V_f \cup V_{fc} \cup V_d$ denote the set of all elements in the system. For each node $i \in V'$, let ρ_i, $\boldsymbol{\xi}_i$, and ψ_i denote the electric density, field, and potential at the point where the node i locates. Given a placement solution $\mathbf{v}$ for both



(a) placement without macro density scaling **(b)** density distribution without macro density scaling **(c)** placement with macro density scaled by the target density ρ_t

Fig. 4. Without macro area scaling, the bin density at the macroblocks becomes higher than the target density ρ_t. As a result, the density force pushes the cells away from macros, inducing underfilled whitespace around macros and wirelength overhead.

movable cells V_m and filler cells V_{fc}, the total potential energy N is defined in Eq. (14).

$$N(\mathbf{v}) = \frac{1}{2}\sum_{i\in V'} N_i = \frac{1}{2}\sum_{i\in V'} q_i\psi_i. \tag{14}$$

As the system energy equals the sum of mutual energy of all the pairs of charges, we have a factor of $\frac{1}{2}$ for the energy of each single charge. We cast the numerous grid density constraints into a single energy constraint of zero system energy ($N(\mathbf{v}) = 0$). Our density penalty is different from those of all previous formulations [Chan et al. 2006; Chen et al. 2008; Kahng and Wang 2006] where it consists of a complete electrostatic system model with all the according physics laws strictly applied. By using the penalty factor λ, we could produce an unconstrained optimization problem as shown:

$$\min_{\mathbf{v}} f = W(\mathbf{v}) + \lambda N(\mathbf{v}), \tag{15}$$

where $W(\mathbf{v})$ is by Eq. (6) and $f(\mathbf{v})$ is the objective cost function to minimize. As both $W(\mathbf{v})$ and $N(\mathbf{v})$ are smooth, we can generate the gradient vector by differentiating Eq. (15) as follows:

$$\nabla f(\mathbf{v}) = \nabla W(\mathbf{v}) + \lambda\nabla N(\mathbf{v}) = \left(\frac{\partial W}{\partial x_1}, \frac{\partial W}{\partial y_1}\cdots\right)^T - \lambda\left(q_1\xi_{1_x}, q_1\xi_{1_y}, \ldots\right)^T. \tag{16}$$

Modeling of density force orientation and magnitude remains a long-term controversial topic [Markov et al. 2012] in the analytic placement domain. For quadratic placement, it remains unclear where to introduce the anchor point for each cell in order to produce a proper dragging force. An ad-hoc force scaling is proposed in Eisenmann and Johannes [1998], while in RQL [Viswanathan et al. 2007a] the top 10% highest density force is empirically cut off to improve the quality. SimPL [Kim et al. 2010], MAPLE [Kim et al. 2012], and ComPLx [Kim and Markov 2012] determine the anchor points by recursive netlist bipartitioning, while the density force relies on initial condition and cutline determination. Without restriction on the function order, the density force formulation in nonlinear placement is of higher freedom. However, the bell-shaped smoothing technique [Naylor et al. 2001] employed in Chen et al. [2008] and Kahng and Wang [2006] incorporates only local information into force modeling, thus it is difficult for the placers to identify a global path of cell movement. Parameter adjustment in the smoothing function could help include remote density information, but is highly case dependent and would consume more engineering effort and cause robustness issues. The algorithm in mPL6 [Chan et al. 2006] uses a more generalized approach with density force derived from potential differentiation. However, it lacks the electrostatics modeling methodology that helps cast all the density constraints

into one single energy function, as Eq. (14) shows. All of the existing problems indicate further improvement space for the density force formulation. Our analytic approach handles the problem by following the Lorentz force law, specifically:

—the density force orientation on each cell aligns with that of the steepest descent of the density penalty (system potential energy);
—the density force magnitude on each cell is determined by its contribution to the reduction of the density penalty; as Eq. (11) shows, and
—the system density force vector is well balanced with the wirelength force vector using a single penalty factor, as Eq. (15) shows.

As a result, our approach models the density force in a systematic way and is validated by the experimental results in Section 7, with shorter wirelength and high efficiency.

4. POISSON'S EQUATION AND NUMERICAL SOLUTION

Based on our eDensity formulation in Section 3, we propose Poisson's equation to couple the charge density with electric potential and field. The Neumann boundary condition is used to enforce the legality of the global placement solution. The Poisson's equation is numerically solved using spectral methods with high accuracy yet low complexity. Moreover, we propose a technique to locally smooth the density over discrete grids.

4.1. Well-Defined Poisson's Equation

By Gauss' law, the electric potential distribution $\psi(x, y)$ can be coupled with the density function $\rho(x, y)$ using Poisson's equation, as Eq. (17) shows.

$$\nabla \cdot \nabla \psi(x, y) = -\rho(x, y), (x, y) \in R, \tag{17}$$

Here the density function equals the negative of the divergence of the gradient vector of the potential function. Let $\hat{\mathbf{n}}$ denote the outer normal vector of the placement region R, and ∂R denote the boundary. When cells are moving towards the borderline of the placement region, the movement should be slowed down or stopped in order to prevent cells from moving outside. The electric (density) force thus diminishes towards zero while approaching the boundary of the density function domain. As a result, we use the Neumann boundary condition, which requires a zero boundary gradient as shown:

$$\hat{\mathbf{n}} \cdot \nabla \psi(x, y) = \mathbf{0}, (x, y) \in \partial R. \tag{18}$$

Besides, the integral of the density function $\rho(x, y)$ and the potential function $\psi(x, y)$ over the entire placement region R is set to be zero, as shown:

$$\iint_R \rho(x, y) = \iint_R \psi(x, y) = 0. \tag{19}$$

Therefore, all the constant factors introduced by the indefinite integration from density to field and potential become zero, moreover; Eq. (19) ensures the unique solution to the PDE in Eq. (17). The problem due to the ill-defined PDE in Eisenmann and Johannes [1998] is thus overcome. Based on all the preceding definitions, we have our well-defined Poisson's equation constructed as follows:

$$\begin{cases} \nabla \cdot \nabla \psi(x, y) = -\rho(x, y), \\ \hat{\mathbf{n}} \cdot \nabla \psi(x, y) = \mathbf{0}, (x, y) \in \partial R, \\ \iint_R \rho(x, y) = \iint_R \psi(x, y) = 0. \end{cases} \tag{20}$$

There are several quadratic placement works [Eisenmann and Johannes 1998; Spindler et al. 2008] in literature where the Poisson's equation is used. However,

the PDE solution is only used to determine the location of anchor points. Some nonlinear placers [Chan et al. 2006] use the Helmholtz equation to include two orders of derivatives to the smoothed density function. To guarantee a unique PDE solution, a linear term is added to the equation with a self-tuned multiplier. Unlike all the previous PDE-based placement approaches, our method is based on a complete system model. The density penalty is formally formulated as the system potential energy. The Poisson's equation is used to compute the electric field which, together with the electric quantity, determines the density gradient by strictly following the Lorentz force law. The uniqueness of our PDE solution is promised by enforcing a zero integral of the potential, which not only simplifies the integration but also avoids the introduction of extra noise due to the linear term in Chan et al. [2006].

4.2. Fast Numerical Solution Using Spectral Methods

We propose a numerical solution using spectral methods [Skollermo 1975] to effectively and efficiently solve the Poisson's equation in Eq. (20). Spectral methods express the solution to some PDE as the summation of basis functions (e.g., sinusoid and cosine waveforms) and choose the coefficients in the sum to satisfy the PDE and boundary conditions. A sinusoid function is an odd and periodic function. It diminishes to zero at the boundary of each period, which could naturally satisfy the Neumann condition as stated in Eq. (18). As a result, we use a sinusoid wave function as the basis function to express the electric field. As the density and potential functions are the derivative and integral of the field function, we use the cosine wave as the basis function to express them. Based on such a decomposition in the frequency domain, we use spectral methods to solve the Poisson's equation.

For expression using discrete cosine transformation (DCT), we modify the original density function $\rho(x, y)$ to an even and periodic form $\rho_{DCT}(x, y)$. Therefore, the new function can be decomposed into a group of cosine waveforms oscillating at different frequencies and constructed by DCT. Electric field and potential functions can be constructed by DCT and discrete sinusoidal transform (DST) in a similar way. The specific modification to the density function is as follows. Suppose the placement region R is uniformly decomposed into an $m \times m$ grid structure, thus the density function $\rho(x, y)$ is defined within the domain of $[0, m-1] \times [0, m-1]$. We mirror the density wave to the negative half-plane such that the function domain is extended to $[-m, m-1] \times [-m, m-1]$, while the density function becomes even. Then we periodically extend the domain of the density function to $[-\infty, +\infty] \times [-\infty, +\infty]$. Based on these two modifications, the new density function $\rho_{DCT}(x, y)$ can be expressed using DCT as follows.

Let u and v denote integer indexes ranging from 0 to $m-1$. The frequency components are defined as $w_u = 2\pi \frac{u}{m}$ and $w_v = 2\pi \frac{v}{m}$, respectively. We use $a_{u,v}$ to denote the coefficient of each basis wave function of DCT. By definition, all the $m \times m$ coefficients can be generated by the integral of the density function multiplied by the basis wave functions over the 2D grid. The solution to each coefficient is shown in Eq. (21).

$$a_{u,v} = \frac{1}{m} \sum_{x=0}^{m-1} \sum_{y=0}^{m-1} \rho(x, y) \cos(w_u x) \cos(w_v y). \tag{21}$$

All the prior coefficients can be rapidly computed by invoking the FFT library only once. Using these cosine coefficients, the new density function $\rho_{DCT}(x, y)$ can be expressed as a sum of cosine waves as shown:

$$\rho_{DCT}(x, y) = \sum_{u=0}^{m-1} \sum_{v=0}^{m-1} a_{u,v} \cos(w_u x) \cos(w_v y), \tag{22}$$

which can also be rapidly computed using one time of inverse FFT library invocation.

Based on Eqs. (17) and (19) and the cosine expression of the density function in Eq. (22), we have the solution to the potential function $\psi_{DCT}(x, y)$ as shown:

$$\psi_{DCT}(x, y) = \sum_{u=0}^{m-1} \sum_{v=0}^{m-1} \frac{a_{u,v}}{w_u^2 + w_v^2} \cos(w_u x) \cos(w_v y), \tag{23}$$

which well satisfies Eq. (17). By Gauss law, the electric field vector is the negative gradient of the potential function as in Eq. (11). Based on the solution to the potential function in Eq. (23), we can obtain the solution to the electric field $\boldsymbol{\xi}(x, y) = (\xi_{X_{DSCT}}, \xi_{Y_{DCST}})$ in the form of DCT and DST as in Eq. (24).

$$\begin{cases} \xi_{X_{DSCT}} = \sum_u \sum_v \frac{a_{u,v} w_u}{w_u^2 + w_v^2} \sin(w_u x) \cos(w_v y), \\ \xi_{Y_{DCST}} = \sum_u \sum_v \frac{a_{u,v} w_v}{w_u^2 + w_v^2} \cos(w_u x) \sin(w_v y). \end{cases} \tag{24}$$

Notice that the horizontal component $\xi_{X_{DSCT}}$ is constructed by sinusoid waves for the horizontal field, which diminishes to zero while reaching the end of a period, thus the horizontal boundary of the placement region. Similar construction is conducted on the vertical field $\xi_{Y_{DCST}}$. Library support to the preceding numerical solutions can be found in various FFT packages [Ooura 2001].

UPlace [Yao et al. 2005] also employs DCT to transform the density function into the frequency domain. The authors form the density penalty using a weighted sum of all the frequency components, where the biased weights between different frequencies would help improve the density equalization. In our approach, the DCT and DST are used in spectral methods to generate the solution to the partial differential equations, where the density penalty and gradient are modeled as system potential energy and electric force. As a result, our approach is different from UPlace in the formulation of both density penalty and gradient.

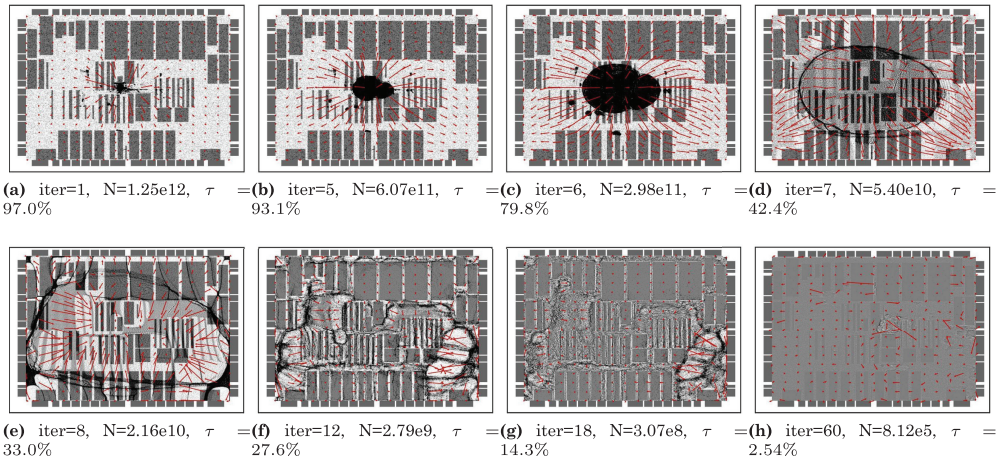
4.3. Convergence

Our density function formulation is based on the analogy between an electrostatic system and a placement instance. For the general case, the traditional bin packing problem has been proved NP-hard [Coffman et al. 1997], thus it is intractable to prove its convergence. However, for the homogeneous case, that is, where all the objects are of equal size, we can show the convergence through analogy of the charge distribution. Assume the final density distribution is not even, then from Eq. (21) we know that there must be some density frequency coefficients $a_{u,v} \neq 0$. As a result, we have the respective electric field coefficients $\frac{a_{u,v} w_u}{w_u^2 + w_v^2} \neq 0$ and $\frac{a_{u,v} w_v}{w_u^2 + w_v^2} \neq 0$, which means that ξ_x and ξ_y do not equal zero. The electric force will then keep pushing the system potential energy to drop by gradient descent, until finally a globally even density distribution is achieved.

4.4. Behavior and Complexity Analysis

An example of discrete density and field distributional in a two-dimensional a plane is shown in Figure 5. The distribution of the electric field changes across different iterations according to the variation of the density distribution, therefore the electric field dynamically directs the cells to underfilled regions. From the figure, we can also find that the electric field diminishes at the boundaries of the placement region. As also shown in Figure 1(b), such behavior satisfies the Neumann condition and the demand of global placement.

Suppose that we totally have n' cells ($n' = |V_m| + |V_{fc}|$) and an $m \times m$ grid imposed on the placement region. The total complexity of our numerical solution has two sources of contribution: (1) density computation and (2) potential and field computation.



(a) iter=1, N=1.25e12, τ = 97.0% **(b)** iter=5, N=6.07e11, τ = 93.1% **(c)** iter=6, N=2.98e11, τ = 79.8% **(d)** iter=7, N=5.40e10, τ = 42.4%

(e) iter=8, N=2.16e10, τ = 33.0% **(f)** iter=12, N=2.79e9, τ = 27.6% **(g)** iter=18, N=3.07e8, τ = 14.3% **(h)** iter=60, N=8.12e5, τ = 2.54%

Fig. 5. Snapshots of the density distribution $\rho(x, y)$ (grayscale) and the field distribution $\boldsymbol{\xi}(x, y)$ (red arrows) produced by *eDensity*. The placement is driven by only density force and conducted on the ISPD05 ADAPTEC1 benchmark, using Nesterov's method with preconditioning. Total potential energy and total density overflow are denoted by N and τ, respectively.

Density computation. At each iteration, the density function is generated by the following two steps:

—traversing all the elements in B to clear the cell density and cell area occupation to zero; and
—traversing all the cells in $V_m \cup V_{fc}$ to determine the area contribution of each cell to the according bins that overlap the cell.

The first step consumes $O(m^2)$ time, while the second step consumes $O(n')$ time. Totally it would consume $O(n' + m^2)$ time to generate the density distribution at each iteration.

Potential and field computation. At each iteration, we need to invoke the FFT library four times to solve Eqs. (21), (23), and (24), respectively. Each 2D FFT library call consumes $O(m^2 \log m^2) = O(2m^2 \log m) = O(m^2 \log m)$ time, thus the total complexity is $O(m^2 \log m)$.

In general, our numerical solution has a computation complexity of $O(n' + m^2 \log m)$ for each placement iteration. As the number of grids is usually at the same scale as the number of cells (to ensure accuracy after discretization), we have $O(n') = O(m^2)$ and the total complexity is essentially $O(m^2 \log m)$ or $O(n' \log n')$. Addition of fillers could slightly increase the computation time, but would not change the overall complexity. All the fillers are equally sized towards the average size of standard cells and will all be upsized to that of a single bin if utilization is small, thus the total number of fillers will not exceed $O(m^2)$. Moreover, as the number of fillers is at essentially the same order of that of movable placement objects, we have $n = O(n')$, thus the overall complexity is still $O(n \log n)$, where n is the number of movable placement objects.

There are many numerical solutions used in literature for the placement density function. Green's function is used in Eisenmann and Johannes [1998] to solve the PDE using 2D convolution. However, the computation complexity is high, with $O(n^2)$ total runtime consumed. Bell-shaped density smoothing is used in Chen et al. [2008] and Kahng and Wang [2006] where, by default, the density gradient is aware of only local information. Global density variation could be included in local gradient computation by parameter adjustment in the smoothing function. However, as the gradient computation on each cell would take $O(m^2) = O(n)$ time, the total time is

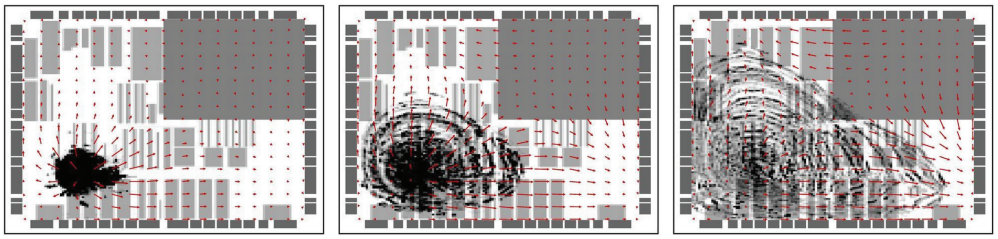


Fig. 6. Illustration of dynamically adjusted density force across different placement iterations. All the cells are initially squeezed into the lower-left subregion with an obstacle placed at the upper-right subregion. The local density gradient could immediately respond to the remote density variation and identify a global motion path for each overlapped cell to some remaining whitespace on the chip.

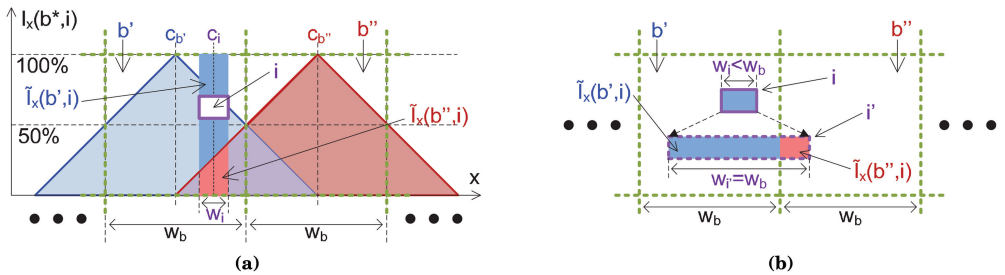


Fig. 7. A one-dimensional illustration of our local density smoothing technique. Here the cell width is smaller than the bin width ($w_i < w_b$). We enlarge the cell to the dimension of one bin. As a result, movement of i at any time will always change the overlaps between i, b', and b'', thus changing the density of b' and b'' simultaneously. There is no local smoothing applied when $w_i \geq w_b$.

still $O(n^2)$. Our PDE solution with spectral methods provides better performance than the aforesaid numerical solutions, as it is aware of global density information while only taking $O(n \log n)$ time for each iteration. The density variation could be instantly propagated to all the placement grids due to the frequency decomposition in Eq. (22). As shown in Figure 6, the local density gradient could be immediately adjusted based on the cell redistribution at remote areas.

4.5. Local Smoothness over Discrete Grids

Global smoothness by eDensity is achieved via Eq. (11) and Eq. (12). However, as the physical dimension of each density bin is usually larger than that of cells, local cell movement within a bin cannot be reflected in the density cost function, where smoothness is degraded. As a result, we propose a local smoothing technique such that the density function by Eq. (14) could well reflect any infinitely small movement of cells within each bin. A one-dimensional example is shown in Figure 7. Here w_i and w_b are the widths of cell i and bin b, and c_i and c_b are the coordinates of the centers of cell i and bin b, respectively. $l_x(i, b)$ and $\widetilde{l}_x(i, b)$ are the original and smoothed horizontal overlaps between the cell and the bin, so we have

$$\widetilde{l}_x(i,b) = \begin{cases} \left(1.0 - \frac{c_i - c_b}{w_b}\right) \times w_i & : c_i \in [c_b - w_b, c_b + w_b] \\ 0 & : c_i \in (-\infty, c_b - w_b) \cup (c_b + w_b, +\infty). \end{cases} \tag{25}$$

As the cell is being shifted rightwards, the contribution to the density of b' is linearly reduced, while the contribution to the density of b'' is linearly increased, respectively. The total contribution of i to the two neighboring bins (b' and b'') is constant and

equals w_i when the center of the cell c_i locates between the centers of the two bins $c_{b'}$ and $c_{b''}$. The smoothing effect is equivalent to the combination of cell dimension stretching and cell density lowering, which keeps the objective cost function analytic. Specifically, for each cell i, we conduct the local density smoothing as follows:

—if $w_i < w_b$, stretch the cell width from w_i to w_b and reduce the cell density from 1.0 to w_i/w_b;
—if $w_i \geq w_b$, keep the original cell width and density.

As a result, this smoothing technique is consistent over different granularities and cell dimensions. Notice that our local smoothing technique is being used at every iteration when updating the density map. It costs constant time for each object since only finite neighboring bins are affected by each object, thus the computation complexity is not changed.

5. NONLINEAR OPTIMIZATION

Global placement is proved an NP-complete problem [Garey et al. 1976]. Developments of prior heuristics are mostly directed by mathematical derivation for quality and efficiency. As Eq. (9) shows, the objective function consists of a convex wirelength function [Hsu et al. 2011] and usually a nonconvex density function [Naylor et al. 2001], where the property of nonconvexity challenges the performance of modern convex programming methods. In this section, we first briefly introduce the Conjugate Gradient (CG) method that is widely used in previous nonlinear placement works [Kahng and Wang 2006; Chen et al. 2008], and discuss the efficiency bottleneck on the line search. Then we propose Nesterov's method to solve the nonlinear problem and illustrate our technique of Lipschitz constant prediction that determines the steplength in constant time. To the best of our knowledge, our work is the first in literature to incorporate Nesterov's method and Lipschitz constant prediction into global placement optimization. A comparison of placement quality and efficiency by using these two optimization methods in ePlace is shown in Section 7, where Nesterov's method could outperform the CG method with 2.28% shorter wirelength and 2.21× speedup, on average, for all the ISPD05 benchmarks. In the end, we discuss our preconditioning technique.

5.1. Conjugate Gradient Method with Line Search

Details of the CG method in one iteration are illustrated in Algorithm 1. The Polak-Ribiere method is used to update β_k for correlation with previous search directions, as line 2 shows. β_k is reset to zero when the conjugacy is lost. The search direction is computed at line 3. We use line search to determine the steplength, and the best solution along the search path $\mathbf{d}_k$ and within the search interval α_k^{max} is obtained. In our approach, golden section search (GSS) is used to implement line search[2] as line 4 shows. The new solution for the current iteration is computed at line 5 and used as the initial solution for the next iteration, while CG would converge after a number of such iterations. CG targets optimization of locally quadratic functions. The closer f is to a quadratic form, the faster CG would converge. Otherwise, CG would easily lose the conjugacy with β reset to zero (line 2). As discussed in Shewchuk [1994], the local error rate of the CG method is bounded as $\|e_{(k)}\| \leq 2(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1})^k \|e_{(0)}\|$, where $\|e_{(k)}\|$ is the error at the kth iteration and κ is the condition number of the Hessian matrix of the objective function, respectively. On the other side, the global convergence rate by the CG method

[2]Within one iteration, the length of the search interval is recursively reduced by the golden ratio 0.618 in each step until the interval length is below α_k^{min}.

ALGORITHM 1: CG-Solver at kth iteration

Input: initial solution $\mathbf{v}_k$
objective function $f_k = f(\mathbf{v}_k)$ maximal and minimal search interval α_k^{max} and α_k^{min}
Output: local optimal solution $\mathbf{v}_{k+1}$
1: gradient vector $\nabla f_k = \nabla f(\mathbf{v}_k)$
2: Polak-Ribiere parameter $\beta_k = \max\left\{\frac{\nabla f_k^T(\nabla f_k - \nabla f_{k-1})}{\|\nabla f_{k-1}\|^2}, 0\right\}$
3: search direction $\mathbf{d}_k = -\nabla f_k + \beta_k \mathbf{d}_{k-1}$
4: steplength $\alpha_k = GSS\left(\mathbf{v}_k, f_k, \mathbf{d}_k, \alpha_k^{max}, \alpha_k^{min}\right)$
5: new solution $\mathbf{v}_{k+1} = \mathbf{v}_k + \alpha_k \mathbf{d}_k$
6: **return** $\mathbf{v}_{k+1}$

cannot exceed $O(1/k)$ [Nemirovskii and Yudin 1983]. Despite the wide usage of CG in previous nonlinear placers, there are still several existing problems.

—The major runtime bottleneck of nonlinear placement lies on the line search at line 4, where the cost function is repeatedly evaluated at different points along the search direction. Profile statistics in Section 7 show that, on the placement of ADAPTEC1, line search takes about 63% of the total runtime of global placement and about 50% of the total placement turnaround, respectively. As a result, line search becomes a roadblock to the pursuit of higher placement efficiency.
—At each iteration, the CG method requires the steplength to be at the zero gradient point along the search direction. However, GSS could only locate the local minimal point, while the actual zero gradient point may fall beyond the range of the search interval. As a result, such inaccurate steplength would prevent the CG method from matching its expected performance.
—The objective function of placement is highly nonlinear where the local cost behavior is usually far from a quadratic form. It becomes fairly easy to lose the conjugacy with respect to previous search directions, while the current search direction is repeatedly reset to that of the negative gradient ($\beta_k = 0$ at line 2), degrading the performance of the CG method to that of the gradient descent method.

As line search is time consuming and could dominate nonlinear placement efficiency [Kahng and Wang 2006], there are attempts in literature to use steplength prediction [Chen et al. 2008] instead. Specifically, as shown by Chen et al. [2008, Eq. (12)], steplength is modeled as $\alpha_k = \frac{sw_b}{\|d_k\|_2}$, where w_b is the bin dimension and $\|d_k\|_2$ the Euclidean norm of the search direction vector. s is a constant factor that is tuned between 0.2 and 0.3 to obtain a good trade-off between runtime and quality. In this work, we propose a novel and systematic approach to dynamically estimate the steplength, based on the local smoothness of the gradient function. Specifically, we use Nesterov's method as the nonlinear solver and Lipschitz constant prediction to determine the steplength. The optimizer could be beneficial in terms of both convergence rate and solution quality, simultaneously. The results in Section 7 show that our approach could outperform Kahng and Wang [2006] and Chen et al. [2008] by roughly 14% and 10% shorter wirelength and 10× and 1.5× speedup, on average, on all the ISPD05 and ISPD06 benchmarks.

5.2. Nesterov's Method with Lipschitz Constant Prediction

We propose to use Nesterov's method for nonlinear global placement optimization. Similar to the CG method, Nesterov's method requires only a first-order gradient and linear memory cost with respect to the problem size. Nesterov's method targets solving a convex programming problem in Hilbert space H. Unlike most convex programming

methods, Nesterov's method constructs a minimizing sequence of points $\{\mathbf{u}_k\}_0^\infty$ that is not relaxational. Algorithm 2 illustrates one iteration of the method on a typical problem $\min\{f(\mathbf{u})|\mathbf{u} \in H\}$ with a nonempty set U^* of minima. Here $\mathbf{u}$ is the solution

ALGORITHM 2: Nesterov-Solver at kth iteration

Input: major solution $\mathbf{u}_k$, reference solution $\mathbf{v}_k$, optimization parameter a_k and objective function $f_k = f(\mathbf{y}_k)$.
Output: new solutions $\mathbf{u}_{k+1}$ and $\mathbf{v}_{k+1}$
1: gradient vector $\nabla f_k = \nabla f(\mathbf{v}_k)$
2: steplength $\alpha_k = \arg\max_{\alpha}\{f_k - f(\mathbf{v}_k - \alpha\nabla f_k) \geq 0.5\alpha\|\nabla f_k\|^2\}$
3: new solution $\mathbf{u}_{k+1} = \mathbf{v}_k - \alpha_k\nabla f_k$
4: parameter update $a_{k+1} = \left(1 + \sqrt{4a_k^2 + 1}\right)/2$
5: new reference solution $\mathbf{v}_{k+1} = \mathbf{u}_{k+1} + (a_k - 1)(\mathbf{u}_{k+1} - \mathbf{u}_k)/a_{k+1}$
6: **return** $\mathbf{u}_{k+1}$

to the convex programming problem, $\mathbf{v}$ is a reference solution that determines the steplength, a is an optimization parameter, and α is the steplength, respectively. At the beginning ($k = 0$), the method starts from an initial solution $\mathbf{v}_0 \in H$ and sets $a_0 = 1$, $\mathbf{u}_0 = \mathbf{v}_0$ and $\alpha_0 = \frac{\|\nabla f(\mathbf{v}_0) - \nabla f(\mathbf{z})\|}{\|\mathbf{v}_0 - \mathbf{z}\|}$, respectively. $\mathbf{z}$ is an arbitrary point in H and $\mathbf{z} \neq \mathbf{v}_0$. All the preceding vectors and scalars will be iteratively updated. At line 2, the steplength α_k is maximized in order to accelerate the convergence. The new solution $\mathbf{u}_{k+1}$ is updated at line 3 based on the initial reference solution $\mathbf{v}_k$. The new optimization parameter a_{k+1} is updated at line 4, while the new reference solution $\mathbf{v}_{k+1}$ is updated at line 5 based on the solution $\mathbf{u}$ and parameter a.

The convergence rate of Nesterov's method in Algorithm 2 is proved $O(1/k^2)$ in Nesterov [1983], where k is the number of iterations. Notice that Nesterov's method [Nesterov 1983] is the firs in literature to achieve $O(1/k^2)$ convergence rate, which is proved the upper bound of convergence rate for the first-order optimization methods [Nemirovskii and Yudin 1983]. The expected convergence rate requires that the steplength α_k satisfies Eq. (26) at every single iteration.

$$f(\mathbf{v}_k) - f(\mathbf{v}_k - \alpha_k\nabla f(\mathbf{v}_k)) \geq 0.5\alpha_k\|\nabla f(\mathbf{v}_k)\|^2. \tag{26}$$

An upper-bounded error rate of Nesterov's method is shown in Eq. (27).

THEOREM 5.1. *Suppose $f(\mathbf{u})$ is a convex function in $C^{1,1}(H)$ and $U^* \neq \emptyset$, where $C^{1,1}(H)$ means that the gradient function $\nabla f(\mathbf{u})$ is of Lipschitz continuity. We have $\mathbf{u}^* \in U^*$ and L is the Lipschitz constant of the gradient function $\nabla f(\mathbf{u})$. The following assertions are true based on the solution $\mathbf{u}_k$ output by Algorithm 2.*

$$f(\mathbf{u}_k) - f(\mathbf{u}^*) \leq \frac{4L\|\mathbf{v}_0 - \mathbf{u}^*\|^2}{(k+2)^2}. \tag{27}$$

Here we define the Lipschitz constant L of the gradient function ∇f as follows.

Definition 5.2. Given $f \in C^{1,1}(H)$, L is the Lipschitz constant of ∇f, if $\forall\mathbf{u}, \mathbf{v} \in H$ we have

$$\|\nabla f(\mathbf{u}) - \nabla f(\mathbf{v})\| \leq L\|\mathbf{u} - \mathbf{v}\|, \tag{28}$$

and $\nabla f(\mathbf{u})$ is thus of Lipschitz continuity. At each iteration, the inequality in Eq. (26) must be satisfied to achieve $O(1/k^2)$ convergence rate. Similar to line search in the CG method, Nesterov [1983] uses bisection search to determine the maximum steplength.

At each iteration, the objective function would be evaluated for $O(\log L)$ times, which increases the complexity to $O(n \log n \log L)$. Instead, we use steplength prediction to accelerate our placement algorithm. As discussed in Nesterov [1983], if the Lipschitz constant of the gradient function is known, we can set the steplength as the inverse of Lipschitz constant to satisfy Eq. (26) without convergence overhead. However, the estimating the exact Lipschitz constant for the objective function of global placement is difficult due to the following issues

—The objective function is nonconvex due to the energy (density) function, thus the requirement for Theorem 5.1 is not satisfied.
—The wirelength function is iteratively changed due to the dynamically adjusted smoothing coefficient (γ in Eq. (6)).
—The penalty factor (λ in Eq. (15)) on the energy (density) function is iteratively changed for runtime force balancing.

As a result, we propose a method to dynamically approximate the Lipschitz constant $\widetilde{L}_k$ iteratively. Based on Eq. (28), we select $\mathbf{x}$ to be the current reference solution ($\mathbf{y}_k$) and $\mathbf{y}$ to the reference solution at the last iteration ($\mathbf{y}_{k-1}$). The Lipschitz constant for $\nabla f(\mathbf{y}_k)$ is approximated as follows.

$$\widetilde{L}_k = \frac{\|\nabla f(\mathbf{y}_k) - \nabla f(\mathbf{y}_{k-1})\|}{\|\mathbf{y}_k - \mathbf{y}_{k-1}\|}. \tag{29}$$

Our approximation method is effective and efficient because:

—there is no additional computation cost introduced, as both $\nabla f(\mathbf{y}_k)$ and $\nabla f(\mathbf{y}_{k-1})$ are known; and
—the two solutions $\mathbf{y}_k$ and $\mathbf{y}_{k-1}$ are supposed to be close to each other; therefore $\|\mathbf{y}_k - \mathbf{y}_{k-1}\|$ is relatively small compared to $\|\mathbf{x} - \mathbf{y}\|$ by randomly selecting $\mathbf{x}$ and $\mathbf{y}$ (this prevents underestimation of $\widetilde{L}_k$, thus overestimation of the steplength α_k).

The results in Section 7 show that our placement algorithm using Nesterov's method with Lipschitz constant prediction could simultaneously improve the runtime and wirelength by 2.21× and 2.28%, compared to those by the CG method together with line search.

5.3. Preconditioning

Preconditioning reduces the condition number of a problem, which is transformed to be more suitable for a numerical solution. Traditional preconditioning techniques compute and inverse the Hessian matrix (H_f) of the objective function (f). Preconditioning has very wide applications in quadratic placers [Viswanathan et al 2007a, 2007b; Kim et al. 2010; Kim and Markov 2012; Lin et al. 2013] but zero attempts in nonlinear placers [Chan et al. 2006; Chen et al. 2008; Kahng and Wang 2006], because the density function is not convex. A preconditioned gradient vector $\nabla f_{pre} = H_f^{-1} \nabla f$ can smooth the numerical optimization to converge in fewer iterations. Nevertheless, the objective function of global placement is highly nonlinear and iteratively changed. Moreover, the problem instance is usually of millions of objects, where the complexity of $O(n^2)$ makes iterative computation of the Hessian matrix fairly expensive and indeed impractical. As a result, we select a Jacobi preconditioner with only diagonal

terms of the Hessian matrix being used, as Eq. (30) shows.

$$\mathbf{H}_{\mathbf{f}_{\mathbf{x},\mathbf{x}}} = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix} \approx \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & 0 & \cdots & 0 \\ 0 & \frac{\partial^2 f}{\partial x_2^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix} = \widetilde{\mathbf{H}}_{\mathbf{f}_{\mathbf{x},\mathbf{x}}}. \tag{30}$$

We have a similar definition on $\widetilde{\mathbf{H}}_{\mathbf{f}_{\mathbf{y},\mathbf{y}}}$ and can construct $\widetilde{\mathbf{H}}_{\mathbf{f}}$ based on them. By Eq. (15) we have $\frac{\partial^2 f(\mathbf{v})}{\partial x_i^2} = \frac{\partial^2 W(\mathbf{v})}{\partial x_i^2} + \lambda \frac{\partial^2 N(\mathbf{v})}{\partial x_i^2}$, and we concisely approximate $\frac{\partial^2 W(\mathbf{v})}{\partial x_i^2}$ and $\frac{\partial^2 N(\mathbf{v})}{\partial x_i^2}$ to ensure functionality of the preconditioner. Differentiating the wirelength function in Eq. (6) by two orders is computationally expensive and we use the vertex degree of object i instead:

$$\frac{\partial^2 W(\mathbf{v})}{\partial x_i^2} = \sum_{e \in E_i} \frac{\partial^2 W_e(\mathbf{v})}{\partial x_i^2} \Rightarrow |E_i|, \tag{31}$$

where E_i denotes the net subset incident to the object i. The nonconvexity of the density function in Eq. (14) disables the traditional preconditioner from achieving the expected performance. Eq. (32) shows its two-order differentiation

$$\frac{\partial^2 N(\mathbf{v})}{\partial x_i^2} = q_i \frac{\partial^2 \psi_i(\mathbf{v})}{\partial x_i^2} = q_i \frac{-\partial \xi_{i_x}(\mathbf{v})}{\partial x_i} \Rightarrow q_i. \tag{32}$$

Here we use the linear term q_i as the density preconditioner.

$$\widetilde{\mathbf{H}}_{\mathbf{f}_{\mathbf{x},\mathbf{x}}} = \begin{pmatrix} |E_1| + \lambda q_1 & 0 & \cdots & 0 \\ 0 & |E_2| + \lambda q_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & |E_n| + \lambda q_n \end{pmatrix}. \tag{33}$$

This we have the preconditioned gradient $\nabla f_{pre} = \widetilde{\mathbf{H}}_{\mathbf{f}}^{-1} \nabla f$. Section 7 shows that our preconditioner could improve the wirelength by 2.42% with essentially the same runtime on average of all the ISPD05 benchmarks.

6. GLOBAL PLACEMENT ALGORITHM

The entire flow of ePlace is shown in Figure 8, where our algorithm accounts for the middle stage of global placement. The global placement is based on the input solution $\mathbf{v}_{ip}$ from the initial placement stage, where the quadratic wirelength is minimized using a bound-2-bound (B2B) net model [Spindler et al. 2008]. A linear CG solver is used with Jacobi preconditioning for acceleration [Kim et al. 2010]. After global placement completes, all the fillers are removed from the solution $\mathbf{v}_{gp}$, which is then legalized and discretely optimized using FastDP [Pan et al. 2005]. As discussed in Section 5, both the CG method and Nesterov's method are used to solve the unconstrained optimization problem in Eq. (15). A self-adaptive parameter adjustment method (introduced in Section 6.1) is incorporated to improve the quality and convergence rate. Finally, we discuss the global placement algorithm in Section 6.2.

6.1. Self-Adaptive Parameter Adjustment

Grid dimension. Our approach uses fixed grid dimension throughout global placement. There is naturally a trade-off between granularity and efficiency. A coarser grid induces

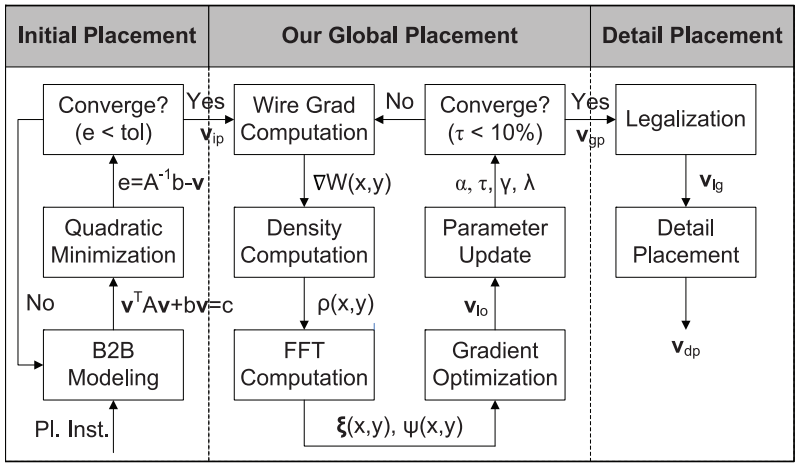


Fig. 8. The entire flow of ePlace, including initial quadratic wirelength minimization, our novel global placement algorithm, and detailed placement with legal solution generated.

higher efficiency but lower accuracy, and vice versa. From experiments we observe that a coarser grid causes additional problems, for instance, more cells undertake the same density force. These cells clot together and motion in the same trace, inducing density oscillation between adjacent regions and impeding cell spreading. In our approach, we determine the grid dimension based on the number of cells in the netlist and inserted fillers. As the FFT package from Ooura [2001] requires the grid dimension m to be a power of 2, we set $m = \lceil \log_2 \sqrt{n'} \rceil$ and upper bound m by 1024 due to efficiency concerns.

Steplength. As discussed in Section 5, in Nesterov's method the steplength is determined by the inverse of the approximated Lipschitz constant, as shown in Eq. (29). In the CG method, the steplength is determined by line search that locates the local minimal cost along the conjugate direction within a search interval. The length of the search interval α_k^{max} is dynamically adjusted as follows. The initial value is determined as linearly proportional to the bin dimension, specifically $\alpha_0^{max} = \kappa w_b$, where w_b is the grid width. In practice, we set $\kappa = 0.044$ to achieve best placement quality, and α_k^{max} is iteratively updated based on the optimal steplength α_{k-1}, as Eq. (34) shows.

$$\alpha_k^{max} = \max\left(\alpha_0^{max}, 2\alpha_{k-1}\right), \alpha_k^{min} = 0.01\alpha_k^{max}. \tag{34}$$

Here α_k is the steplength for the kth iteration generated by GSS, as line 4 of Algorithm 1 shows. Notice that, in practice, α_k may not be the exact local optimal, as line search will stop when the interval reduces to α_k^{min}. Moreover, if $f(x)$ is a multimodal function within the search interval, GSS may perform like a "random perturbation" and even increase the cost under pathological conditions. Despite its suboptimality, such occasional random perturbation will actually be useful. Solutions could escape from local optimum with uphill climbing actions due to GSS and, as a result, GSS remains an effective and efficient line search option.

Penalty factor. In our approach, we set the initial value of the penalty factor λ_0 by Eq. (35) in order to balance the forces of wirelength and density. This method is also used in Chen et al. [2008] and Kahng and Wang [2006]. Here $W_{x_i} = \frac{\partial W}{\partial x_i}$ and $W_{y_i} = \frac{\partial W}{\partial y_i}$, while ξ_{x_i} and ξ_{y_i} denote the horizontal and vertical electric field at node i, respectively.

$$\lambda_0 = \frac{\sum_{i \in V_m'} (|W_{x_i}| + |W_{y_i}|)}{\sum_{i \in V_m'} q_i (|\xi_{x_i}| + |\xi_{y_i}|)} \tag{35}$$

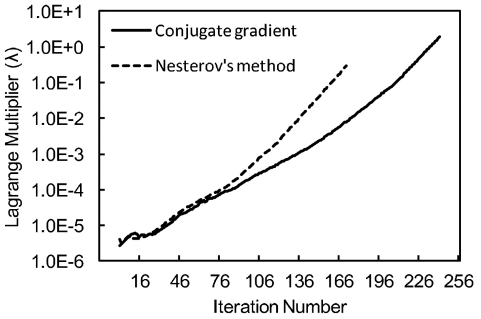


Fig. 9. Illustration of the iterative variation of penalty factor using the CG method with line search and Nesterov's method with Lipschitz constant approximation. The penalty factor increases almost monotonically under nonlinear optimization by both methods. The placement is conducted on the ISPD05 ADAPTEC1 benchmark, where Nesterov's method consumes fewer iterations than the CG method.

Traditional approaches usually multiply the penalty factor λ by a constant number (2.0 in Chen et al. [2008] and Kahng and Wang [2006]), when the optimization converges locally. However, as wirelength and density are changed at every iteration, the penalty factor should be updated immediately in order to adapt to the real-time variation. In our approach, we iteratively update the penalty factor by setting $\lambda_k = \mu_k \lambda_{k-1}$. The multiplier μ_k is based on the iterative HPWL variation $\Delta HPWL_k = HPWL(\mathbf{v}_k) - HPWL(\mathbf{v}_{k-1})$ as shown:

$$\mu_k = \mu_0^{-\frac{\Delta HPWL_k}{\Delta HPWL_{ref}}+1.0}, \tag{36}$$

where μ_0 is a predetermined fixed number and $\Delta HPWL_{ref}$ is the expected wirelength increase per iteration. In practice, we set $\mu_0 = 1.1$ and $\Delta HPWL_{ref} = 3.5 \times 10^5$ for best quality. The multiplier μ_k is upper and lower bounded by 1.1 and 0.75 in order to damp out the transient noise during the optimization flow. The experimental results show that the penalty factor iteratively increases under the nonlinear optimization of both the CG method and Nesterov's method, as illustrated in Figure 9.

Density overflow. Global placement usually terminates when the overlap is small enough. The remaining work is handled by the downstream legalizer and detail placer. Similar to NTUPlace3 [Chen et al. 2008] and mPL6 [Cong et al. 2008], we use the density overflow τ defined in Eq. (41) as the stopping criterion.

$$\tau = \frac{\sum_{b \in B} \max(\rho'_b - \rho_t, 0) A_b}{\sum_{i \in V_m} A_i} \tag{37}$$

Here A_b is the area of the grid b, while A_i is that of movable cell i. Also ρ'_b denotes the density of grid b due to only movable cells. The global placer terminates when the overflow τ is less than τ_{min}. The experimental results show that the total potential energy N is well correlated with the density overflow τ, as illustrated in Figure 10(a). The potential energy decreases exponentially while the density overflow decreases linearly.

Wirelength coefficient. In our approach, we use the WA model [Hsu et al. 2011] in Eq. (6) to smooth the wirelength function. WA outperforms the traditional LSE model by roughly 2× in accuracy. The experiments show that quality and convergence are sensitive to the smoothing parameter γ. Our approach relaxes the smoothing parameter at early iterations, such that more cells are encouraged to be globally moved out of the high-density regions. At later stages when local movement dominates, the parameter is reduced to make the smoothed wirelength W approach $HPWL$. Meanwhile, the density

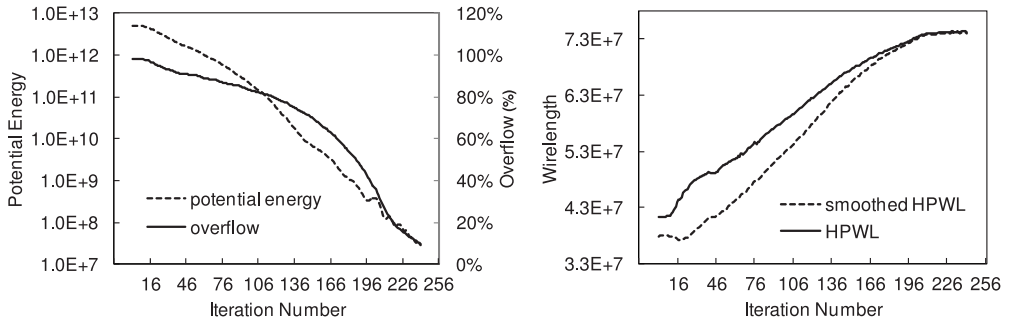


Fig. 10. Illustration of: (a) Total overflow ratio τ and potential energy N; (b) *HPWL* and smoothed wirelength W. The overflow decreases with linear rate while the energy decreases with exponential rate. The smoothed wirelength approximates HPWL better when the density overflow approaches the lower limit (τ_{min}). The global placement uses the CG method and is conducted on the ISPD05 ADAPTEC1 benchmark.

of a smaller grid is more sensitive towards cell movement, and vice versa. Therefore, we set the smoothing parameter γ to be the function of both the density overflow τ and the grid size w_b. By reducing the smoothing parameter, we only enable the motion of HPWL-insensitive cells that are locally shifted to resolve the remaining overlap. Here, by "HPWL-insensitive cells" we are referring to those cells whose movement will not change the HPWL of their incident nets, that is, cells locate relatively far away from the boundaries of the net bounding box. At later iterations, we only expect minor perturbation to the placement layout such that the solution will converge smoothly. As a result, since we target to enhance the accuracy of wirelength modeling, the respective wirelength force becomes stronger to allow only small-scale cell movement, thus minor layout perturbation. The iterative correlation of the smoothed wirelength to the HPWL is shown in Figure 10(b), where the smoothed wirelength converges to HPWL in the end. Our empirical studies show that modeling γ as a linear function of bin dimension w_b yet an exponential function of density overflow τ achieves the best quality. As the density overflow usually starts from around 100% and ends with 10% (our stopping criterion), we set $\gamma(\tau = 1.0) = 80w_b$ and $\gamma(\tau = 0.1) = 0.8w_b$ by empirical tuning. The function of the smoothing parameter γ in terms of density overflow τ is then modeled as

$$\gamma(\tau) = 8.0w_b \times 10^{k\tau+b}. \tag{38}$$

Based on the value of $\gamma(1.0)$ and $\gamma(0.1)$ as mentioned earlier, it is easy to derive that $k = \frac{20}{9}$ and $b = -\frac{11}{9}$, respectively.

6.2. Global Placement

The detail flow of our global placement method ePlace is shown in Algorithm 3. The objective function f_k is formulated at line 5, while wirelength gradient ∇W_k and density distribution $\rho_k(x, y)$ are computed at line 6. The FFT library [Ooura 2001] is invoked at line 7 to generate the distribution of field $\boldsymbol{\xi}_k(x, y)$ and potential $\psi_k(x, y)$. The density (energy) gradient ∇N_k is computed at line 8, while the total gradient ∇f_k is computed at line 9. The nonlinear solver (NL-Solver) is invoked at line 10 with the current solution $\mathbf{v}_k$, and the solution $\mathbf{v}_{k+1}$ for the next iteration is output by the nonlinear solver and used to update the parameters at line 11. The stopping criterion is evaluated at line 12 to determine whether the solution converges. Finally, the global placement solution $\mathbf{v}_{gp}$ is output to the legalizer and detail placer at line 17.

We illustrate the process of global placement in Figure 11 using snapshots of cell and filler distribution at eight intermediate iterations. Nesterov's method is used for the

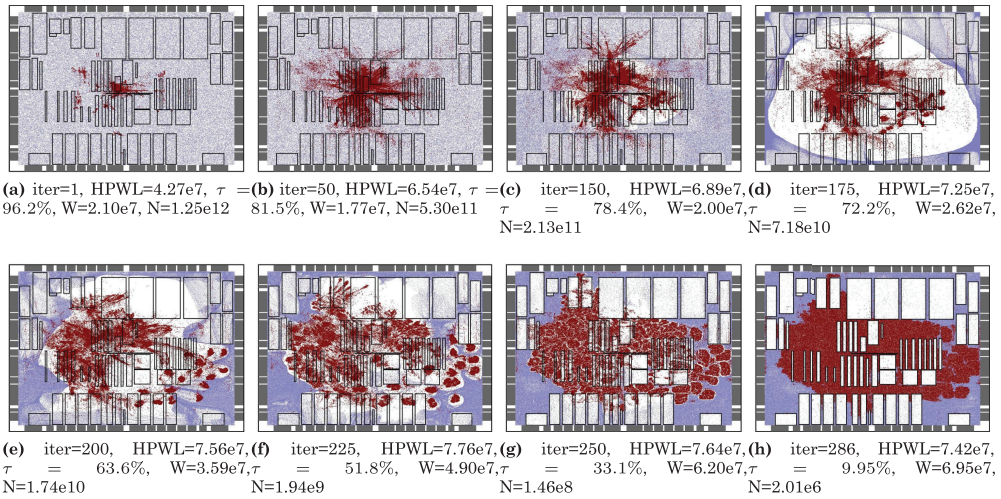


Fig. 11. Snapshots of cell and filler distribution during global placement progression. Standard cells, macros, and fillers are shown by red points, black rectangles, and blue points, respectively. The placement is conducted on the ISPD05 ADAPTEC1 benchmark by ePlace using Nesterov's method with preconditioning.

ALGORITHM 3: ePlace

Input: initial placement solution $\mathbf{v}_0 = \mathbf{v}_{ip}$
uniform chip decomposition into $m \times m$ grid
minimum overflow τ_{min}
maximum iterations $k_{max} = 3000$
Output: global placement solution $\mathbf{v}_{gp}$

1: $m \times m$ decomposition over R
2: initialize λ_0 by Eq. (35)
3: initialize $\alpha_0^{max} = 0.044 w_b$
4: **for** $k = 1 \rightarrow k_{max}$ **do**
5: $f_k = f(\mathbf{v}_k) = W(\mathbf{v}_k) + \lambda_k N(\mathbf{v}_k)$
6: compute wirelength gradient ∇W_k and density ρ_k
7: $(\psi_k, \boldsymbol{\xi}_k) = \text{FFTsolver}(\rho_k)$
8: compute energy (density) gradient $\nabla N_k = \mathbf{q}\boldsymbol{\xi}_k$
9: $\nabla f_k = \nabla W_k + \lambda_k \nabla N_k$
10: $\mathbf{v}_{k+1} = \text{NL-Solver}(\mathbf{v}_k, f_k, \nabla f_k, \alpha_k^{max}, 0.01\alpha_k^{max})$
11: update α_{k+1}^{max}, λ_{k+1}, τ_{k+1}, γ_{k+1} by Eq. (34), (36), (41), (38)
12: **if** $\tau_{k+1} \leq \tau_{min}$ **then**
13: $\mathbf{v}_{gp} = \mathbf{v}_{k+1}$
14: break
15: **end if**
16: **end for**
17: **return** $\mathbf{v}_{gp}$

nonlinear optimization with dynamic prediction of the Lipschitz constant to determine the steplength. The initial placement solution $\mathbf{v}_{ip}$ is shown in Figure 11(a), where standard cells are placed at the central region and filler cells are randomly distributed over the entire placement region R. At later iterations, standard cells spread away from overfilled regions, and the density force pushes the disconnected fillers towards the boundary of the placement region. In the end, all the standard cells converge to a stable location with acceptable system energy (density penalty) and wirelength overhead.

7. EXPERIMENTS AND RESULTS

We implement our algorithm using the C programming language and execute the program in single-thread mode on a Linux machine with an Intel i7 920 2.67 GHz CPU and 12GB memory. In our experiments, we use the benchmark suites from Nam et al. [2005] and Nam [2006] as published in the ISPD 2005 and ISPD 2006 placement contests, respectively. As denoted in Nam et al. [2005] and Nam [2006], the benchmark circuits preserve the physical structure of real ASIC designs. We also use the evaluation policies and scripts in Nam et al. [2005] and Nam [2006] as they have become common criteria and are widely admitted in modern placement works, to rank the performance of different placers in our experiments. Moreover, we set the minimum density overflow $\tau_{min} = 10\%$ as the stopping criterion of global placement for all the benchmarks. We apply the same setting of parameters to all the testcases. In other words, there is no parameter tuning towards specific benchmarks.

Global placement plays the dominant role in the overall placement solution quality. However, the global placement result is illegal and it is relatively hard to tell how much wirelength penalty will be introduced when legalizing it. There are placers in literature, such as SimPL [Kim et al. 2010] and ComPLx [Kim and Markov 2012], that consist of only global placement algorithm development and invoke a detail placement engine from other works to legalize and discretely optimize their solutions. As a result, we follow the custom in literature to conduct a performance comparison between legalized solutions. Specifically, we use the detail placer in Chen et al. [2008] to perform legalization and detail placement in our global placement solution.

We compare the performance of our work with ten cutting-edge placers of different categories: Capo10.5 [Roy et al. 2006] (min-cut), FastPlace3.0 [Viswanathan et al. 2007b], RQL [Viswanathan et al. 2007a], MAPLE [Kim et al. 2012], ComPLx (v13.07.30) [Kim and Markov 2012], BonnPlace [Struzyna 2013], POLAR [Lin et al. 2013] (quadratic) APlace3 [Kahng and Wang 2006], NTUPlace3 [Chen et al. 2008] and mPL6 [Chan et al. 2006] (nonlinear). We have applied and obtained the sourcecode or binary from seven of the aforesaid ten placers, each compiled and executed in our local machine. The executables of RQL, MAPLE, and BonnPlace are not available due to their industrial use and other issues and, as a result, their solution quality and runtime results are cited from the according publications [Viswanathan et al. 2007a; Kim et al. 2012; Struzyna 2013]. The performance of Capo10.5 and APlace3 on the ISPD 2006 benchmark suite is obtained from the respective contest result [Nam 2006].

7.1. Results on ISPD 2005 Benchmark Suite

The circuit statistics of the ISPD 2005 benchmark suite are shown in Table I. Notice that the design scale is up to two million cells, which well represents modern IC design complexity. As there is no specific density constraint, the density upper bound in Table I is set as 100% for every benchmark. Notice that one out of the total of eight circuits (BIGBLUE3) has movable macros, where the physical dimension and logic effort differ quite a lot from those of standard cells. Such objects further challenge the existing placers to provide stable performance under different circuit characteristics.

All the experimental results are shown in Table II and Table III with HPWL in $\times 10^6$ and CPU in minutes. The experiments are executed in single-thread mode (except for POLAR, which consumes up to four CPUs simultaneously) with the solution quality evaluated using the official scripts from Nam et al. [2005]. For our placement framework ePlace, we include three different configurations to study its performance in detail:

—*CG*: ePlace using consists of a conjugate gradient method for nonlinear optimization and line search for steplength determination;

Table I. Circuit Statistics of the ISPD 2005 Placement Benchmark Suite [Nam et al. 2005]

Circuits	# Objects	# Standard Cells	# Movable Macros	# Fixed Macros	# Nets	Density (%)	Util. (%)	Density Bound (%)
ADAPTEC1	211447	210904	0	543	221142	75.71	57.34	100
ADAPTEC2	255023	254457	0	566	266009	78.59	44.32	100
ADAPTEC3	451650	450927	0	723	466758	74.58	33.68	100
ADAPTEC4	496045	494716	0	1329	515951	62.71	27.18	100
BIGBLUE1	278164	277604	0	560	284479	54.19	44.67	100
BIGBLUE2	557866	534782	0	23084	577235	61.88	37.90	100
BIGBLUE3	1096812	1093034	2485	1293	1123170	85.52	56.23	100
BIGBLUE4	2177353	2169183	0	8170	2229886	65.14	44.06	100

—*Nes*: ePlace using is Nesterov's method for nonlinear optimization and Lipschitz constant prediction for steplength determination; and
—*Nes-Pre* is composed of ePlace using Nesterov's method for nonlinear optimization, Lipschitz constant prediction for steplength determination, and preconditioning for search space reshaping.

It is relatively difficult for ePlace-CG and ePlace-Nes to handle large macros (e.g., BIGBLUE3) as the density force is linearly proportional to the object area by Eq. (16), while movable macros significantly differ from standard cells with much higher magnitude of gradient. As a result, an unpreconditioned gradient makes macros with large area and high incidence degree bounce between opposite placement boundaries, causing the solution to oscillate and become hard to converge within a limited number of iterations[3]. To prevent divergence of nonlinear placement optimization, in ePlace-CG and ePlace-Nes, we disable the movement of objects with area larger than 500× the average object area. By preconditioning, we relieve the imbalance between object gradients and make the search space more spherical, thus all the objects are allowed to move in ePlace-Nes-Pre. Among the preceding three options, ePlace-CG has the worst solution quality and placement efficiency, where ePlace-Nes could outperform it by roughly 2.28% shorter wirelength and 2.21× speedup on average. Using preconditioning could further reduce the wirelength by 2.42%, while the runtime is not increased. By default, we use Nesterov's method together with gradient preconditioning in ePlace.

Compared to the performance of all ten placers from our local experiments or according publications as shown in Table II, ePlace-Nes-Pre generates the best placement solutions with the shortest total wirelength in all eight benchmarks. On average, ePlace-Nes-Pre improves the total wirelength by 21.14%, 10.00%, 5.40%, 3.21%, 4.50%, 2.83%, 3.08%, 14.33%, 12.05%, and 8.33% over Capo10.5, FastPlace3.0, RQL, MAPLE, ComPLx, BonnPlace, POLAR, APlace3, NTUPlace3, and mPL6, respectively.

ePlace is faster than all the previous nonlinear placers. Specifically, ePlace-Nes-Pre outperforms APlace3, NTUPlace3, and mPL6 with 9.13×, 1.40×, and 3.78× speedup, even if using multilevel clustering for problem simplification while we are conducting placement on the original flat netlist. At the coarsest level, a hierarchical placer will usually place about only 1000 clusters, which is 0.1% of that of the original netlist. However, despite zero netlist coarsening, our placer runs faster than the previous multilevel works, hence validates the efficiency of our placement algorithm. ePlace-Nes-Pre is slower than some of the previous quadratic placement approaches because all the computation-intensive steps in nonlinear optimization are not included in quadratic placers. For instance, in nonlinear placement, since the objective cost and gradient function are both of very high order, they consume the greatest portion of runtime

[3]In ePlace, we set 3000 as the upper limit of iterations.

Table II. HPWL ($\times 10^6$) on the ISPD 2005 Benchmark Suite [Nam et al. 2005]

Categories	Min-Cut	Quadratic						Nonlinear			ePlace (nonlinear)		
Benchmarks	CP10.5	FP3	RQL*	MPE*	CPx	BPL*	POLAR	AP3	NP3	mPL6	CG	Nes	Nes-Pre
ADAPTEC1	87.80	78.34	77.82	76.36	77.73	76.87	77.21	78.35	80.29	77.93	76.46	74.66	**74.63**
ADAPTEC2	102.66	93.47	88.51	86.95	88.84	86.36	86.16	95.70	90.18	92.04	85.57	89.00	**84.84**
ADAPTEC3	234.27	213.48	210.96	209.78	203.45	202.00	201.30	218.52	233.77	214.16	202.16	201.76	**194.57**
ADAPTEC4	204.33	196.88	188.86	179.91	183.16	181.53	182.37	209.28	215.02	193.89	185.83	183.43	**179.02**
BIGBLUE1	106.58	96.23	94.98	93.74	94.41	94.85	94.67	100.02	98.65	96.80	91.64	91.05	**90.99**
BIGBLUE2	161.68	154.89	150.03	144.55	145.33	144.21	143.85	153.75	158.27	152.34	145.54	143.31	**141.83**
BIGBLUE3	403.36	369.19	323.09	323.05	337.66	317.17	324.53	411.59	346.33	344.25	359.00	326.77	**308.77**
BIGBLUE4	945.77	834.04	797.66	775.71	788.33	781.79	781.06	871.29	829.09	829.44	805.90	763.14	**753.20**
Average	21.14%	10.00%	5.40%	3.21%	4.50%	2.83%	3.08%	14.33%	12.05%	8.33%	4.70%	2.42%	0.00%

CP = Capo, FP = FastPlace, MPE = MAPLE, CPx = ComPLx, BPL = BonnPlace, AP = APlace, NP = NTUPlace. Cited results are marked with *. The HPWL and legality of all the solutions are evaluated by the official scripts [Nam et al. 2005]. Average results are normalized to ePlace-Nes-Pre.

Table III. Runtime (minutes) on the ISPD 2005 Benchmark Suite [Nam et al. 2005]

Categories	Min-Cut	Quadratic						Nonlinear			ePlace (nonlinear)		
Benchmarks	CP10.5	FP3	RQL*	MPE*	CPx	BPL*	POLAR	AP3	NP3	mPL6	CG	Nes	Nes-Pre
ADAPTEC1	48.33	2.92	5.19	18.46	2.97	18.28	2.78	48.88	7.17	23.27	9.17	4.65	4.28
ADAPTEC2	61.63	4.13	7.71	25.26	3.90	26.40	4.67	68.07	8.22	24.75	12.67	7.48	4.90
ADAPTEC3	133.43	9.53	16.80	61.25	8.62	48.50	7.72	186.67	18.53	73.97	45.40	20.77	13.73
ADAPTEC4	141.85	8.75	14.57	58.68	7.38	41.12	7.83	209.60	23.53	71.03	34.33	14.66	15.00
BIGBLUE1	77.90	4.57	7.27	27.47	4.93	26.22	3.72	64.05	14.30	30.05	23.63	5.38	6.25
BIGBLUE2	150.15	8.00	12.82	54.05	7.58	43.13	7.48	136.43	35.10	79.00	30.83	7.31	10.50
BIGBLUE3	373.87	21.05	31.13	117.00	20.75	114.68	20.95	289.78	38.77	104.63	107.75	32.69	28.68
BIGBLUE4	779.22	40.13	78.54	303.93	40.18	258.68	50.87	730.42	106.08	238.82	165.00	43.08	63.70
Average	8.94×	0.53×	0.91×	2.84×	0.52×	3.05×	0.52×	9.13×	1.40×	3.78×	2.21×	0.99×	1.00×

CP = Capo, FP = FastPlace, MPE = MAPLE, CPx = ComPLx, AP = APlace, NP = NTUPlace. Cited results are marked with *. Average results are normalized to ePlace-Nes-Pre.

Table IV. Circuit Statistics of the ISPD 2006 Placement Benchmark Suite [Nam 2006]

Circuits	# Objects	# Standard Cells	# Movable Macros	# Fixed Macros	# Nets	Density (%)	Util. (%)	Density Bound (%)
ADAPTEC5	843128	842482	0	646	867798	78.62	49.85	50
NEWBLUE1	330474	330037	64	337	338901	70.69	70.69	80
NEWBLUE2	441516	436516	3723	1277	465219	86.15	61.66	90
NEWBLUE3	494011	482833	0	11178	552199	84.71	26.33	80
NEWBLUE4	646139	642717	0	3422	637051	65.82	46.47	50
NEWBLUE5	1233058	1228177	0	4881	1284251	74.43	49.26	50
NEWBLUE6	1255039	1248150	0	6889	1288443	59.26	38.70	80
NEWBLUE7	2507954	2481372	0	26582	2636820	76.36	49.06	80

at each iteration. However, in quadratic placement, these two functions are of only second and first orders, where a numerical solution can be computed much faster. Specifically, ePlace-Nes-Pre runs 0.53×, 0.91×, and 0.52× slower than FastPlace3.0, RQL, and ComPLx, while the respective wirelength improvement is 10.00%, 5.40%, and 4.50%. MAPLE, BonnPlace, and POLAR have the best published results on the ISPD 2005 benchmark suite in literature, however, Table III shows, the average runtime of MAPLE, BonnPlace, and POLAR is 2.84×, 3.05×, and 0.52× that of our placer ePlace-Nes-Pre, respectively, while our wirelength improvement over these three placers is 3.21%, 2.83%, and 3.08%, respectively.

As Tables II and III show, on average, of all the ISPD 2005 benchmarks, preconditioning produces 2.42% shorter wirelength and consumes essentially the same runtime compared to the original placement. There are some special testcases, such as BIGBLUE3 of ISPD05, our placer fail to converge without preconditioning, that is, the runtime would approach infinity. As discussed before, we disable the movement of objects with size above certain threshold to force the convergence.

7.2. Results on ISPD 2006 Benchmark Suite

The circuit statistics of the ISPD 2006 benchmark suite [Nam 2006] are shown in Table IV. Notice that BonnPlace [Struzyna 2013] is specifically designed for the ISPD 2005 benchmark suite while its binary or results for the ISPD 2006 benchmarks are not available. As a result, we do not include it in the experiments. Similar to ISPD 2005, the design scale is up to 2.5 million objects thus represents the complexity of modern ASIC design. In contrast to ISPD 2005, there is a benchmark-specific density constraint. Violations of such constraint in placement solutions (i.e., exceeding the density upper bound) would induce a penalty on the total wirelength. Here, two out of the total of eight circuits (NEWBLUE1 and NEWBLUE2) have movable macros that challenge the placement performance stability across different object dimensions.

All the experimental results are shown in Table V and Table VI with scaled HPWL (sHPWL) in $\times 10^6$ and CPU in minutes. Here we just include ePlace-Nes-Pre (denoted as ePlace). Following the contest protocol, we define the scaled wirelength as

$$sHPWL = HPWL \times (1 + 0.01 \times \tau_s). \tag{39}$$

Here τ_s is the density penalty on the wirelength, and denotes the scaled density overflow per bin as defined next.

$$\tau_s = \left(\frac{\tau_{tot} A_{b'} \rho_t}{400 \sum_{i \in V_m} A_i} \right)^2 \tag{40}$$

Here $A_{b'}$ is the area of each uniform bin b', as defined by the contest organizer [Nam 2006] with both width and height equal to 10x the placement row height of each benchmark. A_i is the area of each movable object i, and τ_{tot} is the total density overflow

Table V. Scaled HPWL (sHPWL) ($\times 10^6$) and Original HPWL (in parentheses) on the ISPD 2006 Benchmark Suite [Nam 2006]

Categories	Min-Cut	Quadratic					Nonlinear			
Benchmarks	CP10.5*	FP3	RQL*	MPE*	CPx	POLAR	AP3*	NP3	mPL6	ePlace
ADAPTEC5	494.64 (491.60)	472.72 (437.01)	443.28 (405.73)	407.33 (N/A)	415.77 (407.26)	438.47 (389.82)	520.97 (449.61)	444.41 (345.82)	428.31 (423.96)	**397.53** (394.71)
NEWBLUE1	98.48 (98.35)	74.11 (73.34)	64.43 (64.21)	69.25 (N/A)	64.75 (64.10)	67.52 (66.11)	73.31 (73.26)	**61.01** (60.58)	72.62 (66.61)	62.31 (62.13)
NEWBLUE2	309.53 (308.64)	206.04 (204.00)	199.60 (196.74)	191.66 (N/A)	193.06 (191.07)	191.25 (187.81)	198.24 (197.42)	194.24 (191.31)	201.91 (199.05)	**182.69** (181.38)
NEWBLUE3	361.25 (361.21)	297.45 (295.83)	269.33 (269.13)	268.07 (N/A)	273.42 (270.91)	271.28 (267.64)	273.64 (273.63)	275.08 (274.94)	285.26 (283.40)	**266.80** (266.61)
NEWBLUE4	362.40 (358.28)	308.33 (295.86)	308.75 (268.07)	282.49 (N/A)	292.82 (288.65)	305.14 (273.96)	384.12 (377.55)	296.62 (260.98)	298.20 (293.22)	**276.13** (272.86)
NEWBLUE5	659.57 (657.40)	621.47 (579.67)	537.49 (473.14)	515.04 (N/A)	507.74 (498.94)	521.85 (462.18)	613.86 (545.90)	537.92 (446.89)	535.80 (528.02)	**492.62** (489.55)
NEWBLUE6	668.66 (668.33)	549.89 (544.31)	515.69 (494.30)	494.82 (N/A)	501.05 (495.39)	512.06 (483.99)	522.73 (522.58)	534.96 (533.45)	523.47 (516.21)	**464.44** (462.60)
NEWBLUE7	1518.75 (1518.49)	1105.58 (1091.20)	1057.80 (1031.33)	1032.60 (N/A)	1041.21 (1026.80)	1045.20 (998.95)	1098.90 (1098.26)	1096.16 (1074.54)	1085.68 (1072.86)	**989.96** (987.45)
Average sHPWL (Average HPWL)	43.73% (44.04%)	16.25% (13.40%)	7.99% (2.72%)	4.59% (N/A)	4.86% (4.01%)	7.16% (1.21%)	18.38% (14.67%)	7.74% (0.60%)	10.11% (8.28%)	0.00% (0.00%)

CP = Capo, FP = FastPlace, MPE = MAPLE, CPx = ComPLx, AP = APlace, NP = NTUPlace. Cited results are marked with *. The scaled HPWL, legality, and density penalty of the solutions are all evaluated by the official scripts [Nam 2006]. Average results are normalized to ePlace.

Table VI. Runtime (minutes) on the ISPD 2006 Benchmark Suite [Nam 2006]

Categories	Min-Cut	Quadratic			Nonlinear			
Benchmarks	CP10.5*	FP3.0	CPx	POLAR	AP3*	NP3	mPL6	ePlace
ADAPTEC5	161.97	21.00	16.70	14.48	337.78	64.53	97.30	34.18
NEWBLUE1	42.70	5.18	4.15	5.95	71.72	12.57	24.48	9.62
NEWBLUE2	94.03	8.80	9.70	8.53	92.22	22.80	61.28	10.10
NEWBLUE3	101.27	10.10	8.58	9.02	208.38	21.00	102.23	14.38
NEWBLUE4	115.43	13.22	11.05	10.17	249.70	38.92	67.75	22.92
NEWBLUE5	347.57	28.70	25.85	23.78	546.65	76.82	127.38	54.83
NEWBLUE6	308.08	20.85	20.52	22.27	485.40	67.60	120.83	52.33
NEWBLUE7	916.03	40.97	50.65	48.23	914.20	149.30	307.03	86.27
Average	6.68×	0.59×	0.55×	0.69×	10.21×	1.63×	3.71×	1.00×

CP = Capo, FP = FastPlace, CPx = ComPLx, AP = APlace, NP = NTUPlace. Cited results are marked with *. Average results are normalized to ePlace.

Table VII. Scaled Density Overflow on the ISPD 2006 Benchmark Suite [Nam 2006]

Categories	Min-Cut	Quadratic					Nonlinear			
Benchmarks	CP10.5*	FP3	RQL*	MPE*	CPx	POLAR	AP3*	NP3	mPL6	ePlace
ADAPTEC5	0.62	8.17	9.25	4.76	1.93	12.48	15.87	28.51	1.03	0.71
NEWBLUE1	0.13	1.04	0.34	1.05	1.02	2.13	0.06	0.70	9.02	0.28
NEWBLUE2	0.29	1.00	1.45	1.01	1.05	1.83	0.42	1.82	1.44	0.68
NEWBLUE3	0.01	0.55	0.07	0.77	0.93	1.36	0.00	0.05	0.66	0.07
NEWBLUE4	1.15	4.22	15.2	5.86	1.45	11.38	1.74	13.66	1.70	1.20
NEWBLUE5	0.33	7.21	13.6	4.05	1.76	12.91	12.45	20.37	1.47	0.63
NEWBLUE6	0.05	1.02	4.33	1.08	1.14	5.80	0.03	0.28	1.41	0.40
NEWBLUE7	0.02	1.30	2.57	1.70	1.40	4.63	0.06	2.01	1.19	0.25
Average	0.45×	5.90×	9.08×	5.46×	4.19×	13.77×	5.58×	12.29×	7.14×	1.00×

CP = Capo, FP = FastPlace, MPE = MAPLE, CPx = ComPLx, AP = APlace, NP = NTUPlace. Cited results are marked with *. All results are evaluated by the official scripts [Nam 2006]. Average results are normalized to ePlace.

amount, defined as

$$\tau_{tot} = \sum_{b' \in B'} \max(\rho'_{b'} - \rho_t, 0) A_{b'}, \tag{41}$$

using the contest-specified bin structure B' as mentioned before.

Compared to the quality of all the ten placers as shown in Table V, ePlace generates the best placement solution (with the shortest scaled wirelength) in seven out of the eight total benchmarks. On average, our placer improves total wirelength by 43.73%, 16.25%, 7.99%, 4.59%, 4.86%, 7.16%, 18.38%, 7.74%, and 10.11% over Capo, FastPlace3.0, RQL, MAPLE, ComPLx, POLAR, APlace3, NTUPlace3, and mPL6, respectively.

The runtimes of all the placers on the ISPD 2006 benchmark suite are shown in Table VI. Notice that the runtimes of RQL and MAPLE are not available, as the authors did not release them in the respective publications [Viswanathan et al. 2007a; Kim et al. 2012]. Compared to all three prior nonlinear placers, ePlace improves the efficiency by up to 10.21×. As discussed before, nonlinear placers lag behind quadratic ones in efficiency due to the computation of high-order gradient functions. However, such a gap is largely reduced by ePlace, that is, on average, of all eight ISPD 2006 circuits, state-of-the-art quadratic placers consume roughly 60% more runtime compared to ePlace.

In addition, we also include the results of the scaled density overflow and original wirelength for comparison between all the placers, as shown in Table VII and Table V (in parentheses), respectively. Our placer could outperform eight out of the total of nine

placers with smaller scaled density overflow. For Capo10.5 with better density overflow, ePlace produces 44.04% and shorter original wirelength, where their respective scaled wirelength still lag behind ours by 43.73%. In terms of original HPWL, our placer outperforms eight out the nine placers in comparison. POLAR and NTUplace3 lag behind ePlace with 0.60% and 1.21% shorter original HPWL, however, their scaled density overflow is 13.77× and 12.29× greater than that of ePlace. As a result, the scaled wirelength of ePlace is 7.16% and 7.74% shorter than that of POLAR and NTUplace3, respectively.

7.3. Placement Runtime Breakdown

We use the timing profile of our placement algorithm ePlace on ADAPTEC1 to analyze the runtime bottleneck and improvement. The placement region is uniformly decomposed into a 512×512 grid. Using the CG method with line search (ePlace-CG), we find that 5.62% of the total runtime is consumed by initial placement (quadratic wirelength minimization), 14.68% by legalization and detail placement, while the remaining 79.70% is due to our global placement. A breakdown of the global placement execution shows that the runtime bottlenecks lie in the computation of wirelength gradient (6.89%), density gradient (20.88%), and function evaluation in line search (63.22%), while the remaining operations take 9.01% runtime. To improve the efficiency, step prediction can be used to replace the line search, and we use Nesterov's method to solve the runtime bottleneck. The steplength is predicted based on our method of dynamic Lipschitz constant approximation, where the runtime overhead is negligible. After replacing the CG method with Nesterov's method, the total runtime of ePlace-Nes is improved by 2.21×. Specifically, the runtime consumed by global placement is reduced to 54.72%, while the initial placement and detail placement consume 11.44% and 33.84%, respectively. The remaining bottlenecks mainly lie in the computation of wirelength gradient (19.72%) and density gradient (59.6%), while other miscellaneous operations cost a total of 20.68% time. To further accelerate the placement engine, we can extend the gradient computation to a parallel platform. The symmetric structure of the FFT algorithm for density gradient computation as well as the wirelength gradient computation [Cong and Zou 2009] would well fit the architecture of graphics processing unit (GPUs) [Moreland and Angel 2003] and distributed systems.

8. CONCLUSION

In this article, we propose a flat nonlinear global placement algorithm *ePlace*. Based on the development of a novel placement density formulation *eDensity*, the placement instance is converted into an electrostatic system to model density cost as the system potential energy. The electric potential and field distribution are correlated with the spatial density distribution via a well-defined Poisson's equation, and we use spectral methods based on fast Fourier transform to produce a fast and accurate numerical solution. We propose to use Nesterov's method as the nonlinear placement solver, which outperforms the CG solver with better quality and efficiency. A novel heuristic is developed to dynamically approximate the Lipschitz constant for steplength prediction. Our nonlinear preconditioning technique further enhances the solution quality with negligible runtime overhead. Experimental results on the ISPD 2005 and ISPD 2006 benchmarks validate the high performance of ePlace. More details on the ePlace framework and solutions can be found at the ePlace homepage [2014].

ePlace is a generalized and effective nonlinear placement algorithm. It resolves the traditional bottlenecks in nonlinear placement (low efficiency due to line search, suboptimality of netlist clustering, quality degradation via the coarse density grid at early stages, etc.), and shows that nonlinear placement has the capability to outperform cutting-edge quadratic placement algorithms [Lin et al. 2013; Kim and Markov 2012;

Kim et al. 2012; Struzyna 2013; Viswanathan et al. 2007a] with better solution quality and comparable or even shorter runtime. Compared to the state-of-the-art research innovations in placement literature such as Eisenmann and Johannes [1998], Naylor et al. [2001], etc., ePlace takes a new outlook on this traditional problem. Specifically, we study and leverage the analogy between placement and electrostatics while eDensity is actually conducting a simulation on the behavior of the equivalent electrostatic system. As a result, we could have global smoothness, fast convergence, and high quality, all achieved in a promising way.

In the future, we will explore opportunities in the parallel computing platform while gradient computation can be well accelerated via distributed system [He et al. 2014] or graphics processing units [Moreland and Angel 2003]. Furthermore, based on the current ePlace prototype of analytic nonlinear placement, we will enhance its capability to handle mixed-size large-scale circuits and extend the framework towards other design objectives, such as routability, timing, etc.

ACKNOWLEDGMENTS

The authors would like to thank: (1) Dr. N. Viswanathan and Professor C. Chu for the binary of FastPlace3.0 and FastPlace-DP; (2) Dr. M.-C. Kim and Professor I. L. Markov for the binary of SimPL and ComPLx; (3) T. Lin and Professor C. Chu for the binary of POLAR; (4) J. W.-T. Chan and Professor A. B. Kahng for the binary of APlace3; (5) Dr. M.-K. Hsu and Professor Y.-W. Chang for the binary of NTUplace3; (6) Professor J. Cong for the binary of mPL6; and (7) the support of NSF CCF-1017864.

REFERENCES

S. N. Adya, I. L. Markov, and P. G. Villarrubia. 2003. On whitespace and stability in mixed-size placement. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD'03)*. 311–318.

A. R. Agnihorti, S. Ono, C. Li, M. C. Yildiz, A. Khathate, C.-K. Koh, and P. H. Madden. 2005. Mixed block placement via fractional cut recursive bisection. *IEEE Trans. Comput.-Aided Des. Integr. Circ. Syst.* 24, 5, 748–761.

A. E. Caldwell, A. B. Kahng, and I. L. Markov. 2000. Optimal partitioners and end-case placers for standard-cell layout. *IEEE Trans. Comput.-Aided Des. Integr. Circ. Syst.* 19, 11, 1304–1313.

T. Chan, J. Cong, and K. Sze. 2005. Multilevel generalized force-directed method for circuit placement. In *Proceedings of the ACM International Symposium on Physical Design (ISPD'05)*. 185–192.

T. F. Chan, J. Cong, J. R. Shinnerl, K. Sze, and M. Xie. 2006. mPL6: Enhanced multilevel mixed-size placement. In *Proceedings of the ACM International Symposium on Physical Design (ISPD'06)*. 212–214.

T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang. 2008. NTUPlace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraint. *IEEE Trans. Comput.-Aided Des. Integr. Circ. Syst.* 27, 7, 1228–1240.

E. G. Coffman, M. R. Garey, and D. S. Johnson 1997. *Approximation Algorithms for Bin Packing: A Survey*. PWS Publishing, Boston, MA.

J. Cong, G. Luo, and E. Radke. 2008. Highly efficient gradient computation for density-constrained analytical placement. *IEEE Trans. Comput.-Aided Des. Integr. Circ. Syst.* 27, 12, 2133–2144.

J. Cong and Y. Zou. 2009. Parallel multi-level analytical global placement on graphics processing unit. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD'09)*. 681–688.

H. Eisenmann and F. M. Johannes. 1998. Generic global placement and floorplanning. In *Proceedings of the Annual Design Automation Conference (DAC'98)*. 269–274.

Eplace Homepage. 2014. http://vlsi-cuda.ucsd.edu/~ljw/ePlace/index.

M. R. Garey, D. S. Johnson, and L. Stockmeyer. 1976. Some simplified np-complete graph problems. *Theor. Comput. Sci.* 1, 3, 237–267.

S. K. Han, K. Jeong, A. B. Kahng, and J. Lu. 2011. Stability and scalability in global routing. In *Proceedings of the System Level Interconnect Prediction Workshop (SLIP'11)*. 1–6.

Q. He, W. Au, A. Korobkov, and S. Venkateswaran. 2014. Parallel power grid analysis using distributed direct linear solver. In *Proceedings of the IEEE International Symposium on Electromagnetic Compatibility (EMC'14)*.

Q. He, D. Chen, and D. Jiao. 2012. From layout directly to simulation: A first-principle guided circuit simulator of linear complexity and its efficient parallelization. *IEEE Trans. Component Packag. Manufact. Technol.* 2, 4, 687–699.

M.-K. Hsu, Y.-W. Chang, and V. Balabanov. 2011. TSV-aware analytical placement for 3d ic designs. In *Proceedings of the Annual Design Automation Conference (DAC'11)*. 664–669.

Itrs. 2011. http://www.itrs.net/Links/2011ITRS/Home2011.htm.

A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu. 2010. *VLSI Physical Design: From Graph Partitioning to Timing Closure*. Springer.

A. B. Kahng and Q. Wang. 2006. A faster implementation of aplace. In *Proceedings of the ACM International Symposium on Physical Design (ISPD'06)*. 218–220.

M.-C. Kim, D.-J. Lee, and I. L. Markov. 2010. SimPL: An effective placement algorithm. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD'10)*. 649–656.

M.-C. Kim and I. Markov. 2012. ComPLx: A competitive primal-dual lagrange optimization for global placement. In *Proceedings of the Annual Design Automation Conference (DAC'12)*. 747–752.

M.-C. Kim, N. Viswanathan, C. J. Alpert, I. L. Markov, and S. Ramji. 2012. MAPLE: Multilevel adaptive placement for mixed-size designs. In *Proceedings of the ACM International Symposium on Physical Design (ISPD'12)*. 193–200.

T. Lin, C. Chu, J. R. Shinnerl, I. Bustany, and I. Nedelchev. 2013. POLAR: Placement based on novel rough legalization and refinement. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD'13)*. 357–362.

J. Lu. 2010. Fundamental research on electronic design automation in vlsi design - Routability. Masters thesis, The Hong Kong Polytechnic University. http://hdl.handle.net/10397/4114.

J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-H. Huang, C.-C. Teng, and C.-K. Cheng. 2013. FFTPL: An analytic placement algorithm using fast fourier transform for density equalization. In *Proceedings of the IEEE International Conference on ASIC (ASICON'13)*. 1–4.

J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-H. Huang, C.-C. Teng, and C.-K. Cheng. 2014. ePlace: Electrostatics based placement using nesterov's method. In *Proceedings of the Annual Design Automation Conference (DAC'14)*.

J. Lu, W.-K. Chow, and C.-W. Sham. 2012a. A new clock network synthesizer for modern vlsi designs. *Integr. VLSI J.* 45, 2, 121–131.

J. Lu, W.-K. Chow, and C.-W. Sham. 2012b. Fast power- and slew-aware gated clock tree synthesis. *IEEE Trans. VLSI Syst.* 20, 11, 2094–2103.

J. Lu, W.-K. Chow, C.-W. Sham, and E. F.-Y. Young. 2010. A dual-mst approach for clock network synthesis. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASPDAC'10)*. 467–473.

J. Lu and C.-W. Sham. 2013. LMgr: A low-memory global router with dynamic topology update and bending-aware optimum path search. In *Proceedings of the International Symposium on Quality Electronic Design (ISQED'13)*. 231–238.

I. L. Markov, J. Hu, and M.-C. Kim. 2012. Progress and challenges in vlsi placement research. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD'12)*. 275–282.

K. Moreland and E. Angel. 2003. The fft on a gpu. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware (HWWS'03)*. 112–119.

G.-J. Nam. 2006. ISPD 2006 placement contest: Benchmark suite and results. In *Proceedings of the International Symposium on Physical Design (ISPD'06)*. 167.

G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter, and M. Yildiz. 2005. The ispd2005 placement contest and benchmark suite. In *Proceedings of the International Symposium on Physical Design (ISPD'05)*. 216–220.

W. C. Naylor, R. Donelly, and L. Sha. 2001. Non-linear optimization system and method for wire length and delay optimization for an automatic electric circuit placer. US Patent 6301693.

A. S. Nemirovskii and D. B. Yudin. 1983. *Problem Complexity and Method Efficiency in Optimization*. John Wiley and Sons.

Y. E. Nesterov. 1983. A method of solving a convex programming problem with convergence rate.

T. Ooura. 2001. General purpose fft package. http://www.kurims.kyoto-u.ac.jp/~ooura/fft.html.

M. Pan, N. Viswanathan, and C. Chu. 2005. An efficient and effective detailed placement algorithm. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'05)*. 48–55.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 2007. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press.

J. A. Roy, S. N. Adya, D. A. Papa, and I. L. Markov. 2006. Min-cut floor placement. *IEEE Trans. Comput.-Aided Des. Integr. Circ. Syst.* 25, 7, 1313–1326.

C. Sechen and A. Sangiovanni-Vincentelli. 1986. TimberWolf3.2: A new standard cell placement and global routing package. In *Proceedings of the Annual Design Automation Conference (DAC'86)*. 432–439.

C.-W. Sham, E. F.-Y. Young, and J. Lu. 2009. Congestion prediction in early stages of physical design. *ACM Trans. Des. Autom. Electron. Syst.* 14, 1, 12:1–18.

J. Shewchuk. 1994. An introduction to the conjugate gradient method without the agonizing pain. Tech. rep. CMU-CS-TR-94-125, Carnegie Mellon University.

G. Skollermo. 1975. A fourier method for the numerical solution of poisson's equation. *Math. Comput.* 29, 131, 697–711.

P. Spindler, U. Schlichtmann, and F. M. Johannes. 2008. Kraftwerk2 - A fast force-directed quadratic placement approach using an accurate net model. *IEEE Trans. Comput.-Aided Des. Integr. Circ. Syst.* 27, 8, 1398–1411.

M. Struzyna. 2013. Sub-quadratic objectives in quadratic placement. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE'13)*. 1867–1872.

T. Taghavi, X. Yang, and B.-K. Choi. 2005. Dragon2005: Large-scale mixed-size placement tool. In *Proceedings of the International Symposium on Physical Design (ISPD'05)*. 245–247.

N. Viswanathan, G.-J. Nam, C. J. Alpert, P. Villarrubia, H. Ren, and C. Chu. 2007a. RQL: Global placement via relaxed quadratic spreading and linearization. In *Proceedings of the Annual Design Automation Conference (DAC'07)*. 453–458.

N. Viswanathan, M. Pan, and C. Chu. 2007b. FastPlace3.0: A fast multilevel quadratic placement algorithm with placement congestion control. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASPDAC'07)*. 135–140.

L.-T. Wang, Y.-W. Chang, and K.-T. Cheng. 2009. *Electronic Design Automation: Synthesis, Verification and Test*. Morgan Kaufmann, San Fransisco.

X. Wang, W. Yueh, D. B. Roy, S. Narasimhan, Y. Zheng, S. Mukhopadhyay, D. Mukhopadhyay, and S. Bhunia. 2013. Role of power grid in side channel attack and power-grid-aware secure design. In *Proceedings of the Annual Design Automation Conference (DAC'13)*. 1–9.

B. Yao, H. Chen, C.-K. Cheng, N.-C. Chou, L.-T. Liu, and P. Suaris. 2005. Unified quadratic programming approach for mixed mode placement. In *Proceedings of the International Symposium on Physical Design (ISPD'05)*. 193–199.

Y. Zheng, A. Basak, and S. Bhunia. 2014. CACI: Dynamic current analysis towards robust recycled chip identification. In *Proceedings of the Annual Design Automation Conference (DAC'14)*. 1–6.

H. Zhuang, J. Lu, K. Samadi, Y. Du, and C.-K. Cheng. 2013. Performance-driven placement for design of rotation and right arithmetic shifters in monolithic 3d ics. In *Proceedings of the International Conference on Communications, Circuits and Systems (ICCCAS'13)*. 509–513.

Received November 2013; revised September 2014; accepted September 2014