ePlace: Electrostatics Based Placement Using Nesterov's Method

Jingwei Lu[1], Pengwen Chen[2], Chin-Chih Chang[3], Lu Sha[3], Dennis J-.H. Huang[3], Chin-Chi Teng[3], Chung-Kuan Cheng[1]
[1]Department of Computer Science and Engineering, University of California, San Diego
[2]Department of Applied Mathematics, National Chung Hsing University, [3]Cadence Design Systems, Inc.
jlu@cs.ucsd.edu, pengwen@nchu.edu.tw, {chinchih, lusha, dhuang, ccteng}@cadence.com, ckcheng@ucsd.edu

***Abstract*—ePlace is a generalized analytic algorithm to handle large-scale standard-cell and mixed-size placement. We use a novel density function based on electrostatics to remove overlap and Nesterov's method to minimize the nonlinear cost. Steplength is estimated as the inverse of Lipschitz constant, which is determined by our dynamic prediction and backtracking method. An approximated preconditioner is proposed to resolve the difference between large macros and standard cells, while an annealing engine is devised to handle macro legalization followed by placement of standard cells. The above innovations are integrated into our placement prototype ePlace, which outperforms the leading-edge placers on respective standard-cell and mixed-size benchmark suites. Specifically, ePlace produces 2.83%, 4.59% and 7.13% shorter wirelength while runs $3.05\times$, $2.84\times$ and $1.05\times$ faster than BonnPlace, MAPLE and NTUplace3-unified in average of ISPD 2005, ISPD 2006 and MMS circuits, respectively.**

I. Introduction

Placement remains crucial and challenging in VLSI physical design and could significantly impacts routing [11] and timing closure. Modern ASIC has thousands of large macros and millions of standard cells, of which the complexity challenges the capability of existing placers. Various categories of standard-cell placement algorithms have been proposed in literature. **Min-cut** approaches [16] simplify the problem by recursive partitioning, while quality loss due to suboptimal circuit and whitespace partition is hard to recover. **Quadratic** approaches [7]–[9], [18] approximate wirelength and density using quadratic functions, nevertheless, the low modeling order restricts the solution quality and robustness. Instead, **nonlinear** approaches [1], [4], [6] use high-order wirelength [5], [14] and density [4], [14] cost functions, where multi-level cell clustering is applied to reduce netlist complexity but introduce quality overhead.

Prior mixed-size placement can be divided into three categories. **Two-stage** methods [2], [3] conduct floorplanning followed by standard-cell placement, while the limited cell information usually induces suboptimal floorplan solution. **Constructive** (floorplan-guided) methods [16], [21] have standard cells grouped into soft blocks and optimized by floorplanner with incremental placer, where suboptimal clustering causes inevitable quality loss. **One-stage** methods remain popular among analytic placement algorithms [1], [4], [7], [20]. Macros and standard cells are placed simultaneously to avoid above limitations. However, unbalanced gradient due to problem complication usually causes placement hard to converge.

In this work, we develop ePlace, a generalized one-stage, flat, analytic nonlinear placement algorithm. Despite broad spectrum of topological and physical attributes, all the movable objects are equalized in the optimizer's perspective and handled in exactly the same way to ensure high and stable performance over different benchmarks. Our specific contributions are listed as follows.

DAC '14, June 01 - 05 2014, San Francisco, CA, USA
Copyright 2014 ACM 978-1-4503-2730-5/14/06 $15.00.
http://dx.doi.org/10.1145/2593069.2593133

- We provide detail analysis on the novel density function developed in our prior work [10].
- We use Nesterov's method as the nonlinear solver with steplength dynamically predicted via Lipschitz constant. A backtracking method is developed to improve the prediction accuracy.
- We develop an approximated preconditioner to resolve the gap between standard cells and macros.
- We devise an annealing-based macro legalizer to directly control macro shifting. A standard cell-only global placement follows to resolve the quality overhead induced during macro legalization.
- We integrate all the innovations into ePlace, a generalized placement framework, with promising experimental results obtained on ISPD 2005 [13], ISPD 2006 [12] and MMS [21] benchmarks.

The remainder is organized as follows. Section II introduces the background knowledge. Section III provides an overview of ePlace. Section IV analyzes the placement density function. Section V discusses the development of global placement. Section VI introduces our annealing-based macro legalizer and the standard-cell global placer. Experiments and results are shown in Section VII. We conclude in Section VIII.

II. Essential Concepts

Given a placement instance $G = (V, E, R)$ with n objects V (standard cells and macros), nets E and region R, placement is formulated as a constrained optimization. The **constraint** desires a solution $\mathbf{v} = \{x_1, \ldots, x_n, y_1, \ldots, y_n\}$ to accommodate objects with sufficient sites but zero overlap or density violation. Global placement uniformly decomposes the region into $n \times n$ rectangular grids (bins) denoted as B. For every grid b, its density $\rho_b(\mathbf{v})$ should not exceed the bound ρ_t (benchmark specific). The **objective** is usually set as the total half-perimeter wirelength (HPWL) of all the nets. Let $W_e(\mathbf{v})$ denote the HPWL of each net e, the total HPWL $W(\mathbf{v})$ is

$$W(\mathbf{v}) = \sum_{e \in E} W_e(\mathbf{v}) = \sum_{e \in E} \left(\max_{i,j \in e} |x_i - x_j| + |y_i - y_j| \right). \quad (1)$$

As a result, the cost function is formulated as

$$\min_{\mathbf{v}} W(\mathbf{v}) \text{ s.t. } \rho_b(\mathbf{v}) \leq \rho_t, \forall b \in B. \quad (2)$$

Analytic methods conduct placement using gradient-based optimization. As HPWL is not differentiable, we use the weighted-average [5] model for **wirelength smoothing** as below

$$\widetilde{W}_{e_x}(\mathbf{v}) = \frac{\sum_{i \in e} x_i \exp(x_i/\gamma)}{\sum_{i \in e} \exp(x_i/\gamma)} - \frac{\sum_{i \in e} x_i \exp(-x_i/\gamma)}{\sum_{i \in e} \exp(-x_i/\gamma)}, \quad (3)$$

here $\widetilde{W}_e(\mathbf{v}) = \widetilde{W}_{e_x} + \widetilde{W}_{e_y}$ and γ controls the modeling accuracy. A **density penalty function** helps incorporate the $|B|$ constraints in Eq. (2) to enhance analyticity. In literature, quadratic placers [7], [8], [19], [20] introduce anchor points to model the density gradient as a linear term, while nonlinear placers [1], [4], [6] smooth the density distribution by bell-shape functions [14]. Instead, we use the electrostatics based density function proposed in our prior work [10] to formulate the problem as

$$\min_{\mathbf{v}} f(\mathbf{v}) = \widetilde{W}(\mathbf{v}) + \lambda N(\mathbf{v}), \quad (4)$$

where $N(\mathbf{v})$ is the density function and λ is the penalty factor.

III. Placement Overview

Figure 1 shows the flowchart of ePlace. Given a placement instance, ePlace quadratically minimizes the total wirelength at the mixed-size initial placement (mIP) stage. The initial solution $\mathbf{v}_{mIP}$ is of low wirelength but high overlap. Based on target density ρ_t, our mixed-size global placer (mGP) populates extra whitespace with unconnected fillers, and co-optimizes all the objects (standard cells, macros and fillers) together. After mGP, we remove fillers and fix standard cells, then invoke the annealing engine mLG to legalize macros. In the second-phase global placement (cGP), we retrieve all the fillers and distribute them appropriately, then free standard cells and co-place them with fillers to further reduce the wirelength. Finally, in cDP we invoke the detail placer in [4] to legalize and discretely optimize the standard-cell layout.

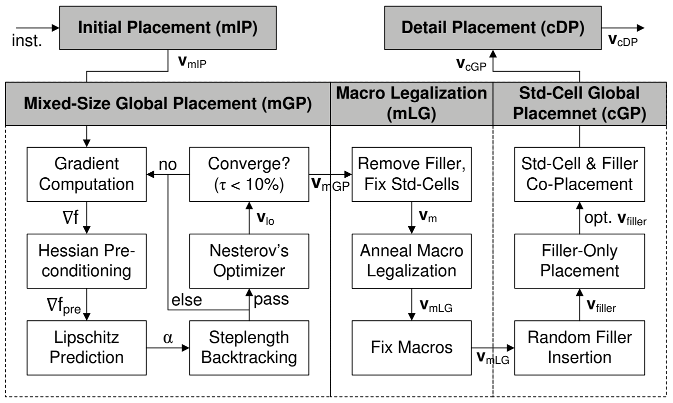


Fig. 1: The flowchart of ePlace.

ePlace disallows rotation or flipping of macros to follow contest protocols [12], [13] and lithography requirements. However, it has the flexibility to integrate the rotational and flipping gradients [4]. Deadspace allocation is not considered but can be realized by appropriate macro inflation.

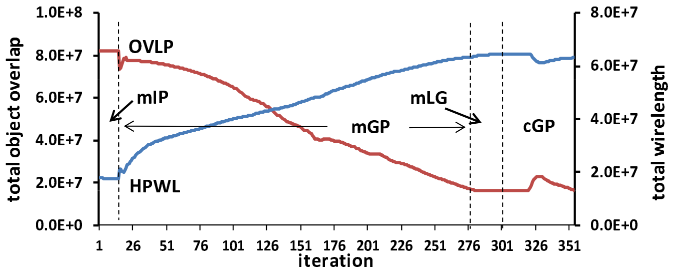


Fig. 2: Total HPWL and object overlap (OVLP) at different stages and iterations of placement on MMS ADAPTEC1.

ePlace maximally expands the design space for mGP while shrinks it for mLG and cGP. The major optimization effort is budgeted on mGP to produce optimal layout of global placement. In contrast, minor layout perturbation is expected in mLG and cGP. As Figure 2 shows, the constrained optimization focuses on the mGP stage and terminates when overlap is small enough (density overflow $\tau \leq 10\%$). ePlace is built upon our recent placement work FFTPL [10] with the same parameter setting[1], filler formation, etc..

IV. Density Function Analysis

Our prior work [10] develops a novel density function based on the electrostatic analogy. Modeling every object as a charge, the density function $N(\mathbf{v})$ in Eq. (5) is modeled as the total electric potential energy. The electric force spreads objects apart and reduces total energy to zero in the end, where the electrostatic equilibrium state (i.e., even density distribution) is reached. Compared to previous analytic approaches, the density function enables ePlace to achieve the minimum density overflow, as shown in Table I and Table II.

$$N(\mathbf{v}) = \sum_{i\in V} N_i(\mathbf{v}) = \sum_{i\in V} q_i\psi_i(\mathbf{v}), \tag{5}$$

here q_i is the electric quantity of object i (equal to its area), and ψ_i is the local potential. A well-defined Poisson's equation in Eq. (6) is proposed to correlate spatial density and potential distribution. Neumann boundary condition (i.e., zero gradient at the boundary) is enforced to prevent objects from moving outside the placement region R. We remove the zero-frequency component from the spatial density and potential distribution, in order to couple the equilibrium state with even charge distribution within the placement domain, rather than charge distribution only along the placement boundary. Also, the constant term during integral operation can be ignored.

$$\begin{cases} \nabla\cdot\nabla\psi(x,y) = -\rho(x,y), \\ \hat{\mathbf{n}}\cdot\nabla\psi(x,y) = \mathbf{0},\ (x,y)\in\partial R, \\ \iint_R \rho(x,y) = \iint_R \psi(x,y) = 0. \end{cases} \tag{6}$$

Here x and y are spatial coordinates, $\rho(x,y)$ and $\psi(x,y)$ denote spatial density and potential distribution, $\hat{\mathbf{n}}$ is the outer norm vector at the boundary ∂R. $\xi(x,y) = \nabla\psi(x,y)$ is the spatial field distribution. The electric force on each object i equals $q_i\xi_i(\mathbf{v})$, where ξ_i is the local field and can be decomposed into horizontal (ξ_{i_x}) and vertical (ξ_{i_y}) components. Our density function $N(\mathbf{v})$ is generalized without special handling of fixed blocks. The global smoothness of $N(\mathbf{v})$ (by Eq. (5) and (6)) indicates that movement of any object will change the global potential map thus energy of all the objects. The gradient of density function w.r.t. the horizontal movement of object i is

$$\frac{\partial N(\mathbf{v})}{\partial x_i} = \frac{\partial N_i}{\partial x_i} + \frac{\partial\left(\sum_{j\neq i} N_j\right)}{\partial x_i} = q_i\frac{\partial\psi_i}{\partial x_i} + \sum_{j\neq i} q_j\frac{\partial\psi_j}{\partial x_i}. \tag{7}$$

By the nature of electrostatics, the potential at each charge i is the superposition of potential contributed by the remaining charges in the system. As a result, the mutual potential energy of each pair of charges i and j are equivalent. Let N_{i_j} denote the potential energy of charge i contributed by j, where $N_i = \sum_{j\neq i} N_{i_j}$. For the electrostatic system on the two-dimensional plane, we have $N_{i_j} = N_{j_i} = \frac{q_i q_j}{2\pi\epsilon_0}\ln(r_{i,j})$, where $r_{i,j}$ is the physical distance between the two charges i and j. As a result, we have

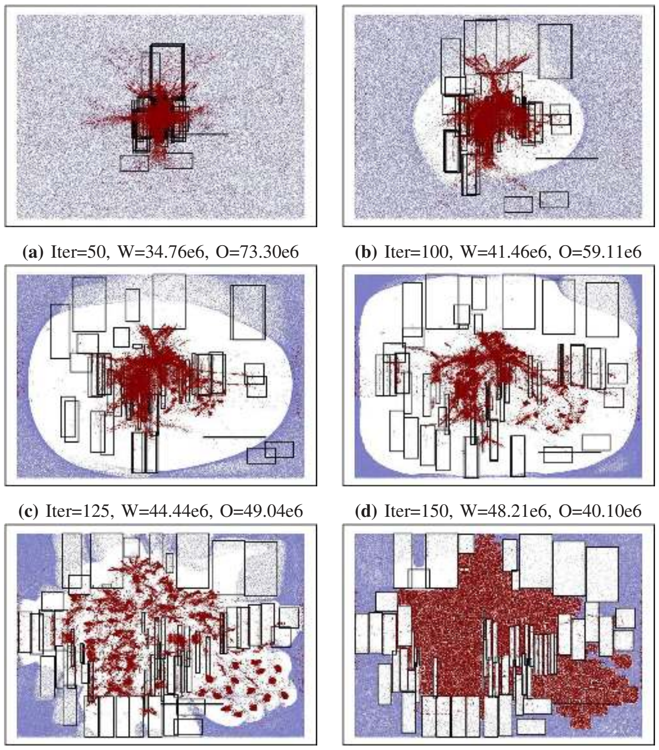
$$\frac{\partial\left(\sum_{j\neq i} N_j\right)}{\partial x_i} = \frac{\partial\left(\sum_{j\neq i} N_{j_i}\right)}{\partial x_i} = \frac{\partial N_i}{\partial x_i} \Rightarrow \frac{\partial N}{\partial x_i} = 2\frac{\partial N_i}{\partial x_i}, \tag{8}$$

and $2\frac{\partial N_i}{\partial x_i} = 2q_i\frac{\partial\psi_i}{\partial x_i} = 2q_i\xi_{i_x}$ is the density gradient. Similarly, the density gradient of $N(\mathbf{v})$ w.r.t. vertical movement of i is $2q_i\xi_{i_y}$. We solve Eq. (6) by spectral methods to obtain $\psi(x,y)$ thus $\xi_x(x,y)$ and $\xi_y(x,y)$. The time complexity is only $O(n\log n)$ by fast Fourier transform (FFT) [10]. The well-formulated density gradient, global density smoothness and low computational complexity enables ePlace to conduct placement on the flat netlist and the flat density grid of high resolution. Compared to prior nonlinear placers [1], [4], [6] with multi-level netlist clustering and grid coarsening, ePlace avoids quality loss due to suboptimal clustering and low density resolution.

V. Mixed-Size Global Placement (mGP)

mGP conducts simultaneous optimization on both macros and standard cells in a smooth way as Figure 3 shows. As a generalized approach, mGP handles macros and standard cells in exactly the same way (c.f. macro shifting when declustering [4], soft block formation

[1]E.g., grid decomposition, initialization and iterative adjustment of wirelength coefficient γ, density overflow τ and penalty factor λ, etc..



(a) Iter=50, W=34.76e6, O=73.30e6 **(b)** Iter=100, W=41.46e6, O=59.11e6

(c) Iter=125, W=44.44e6, O=49.04e6 **(d)** Iter=150, W=48.21e6, O=40.10e6

(e) Iter=200, W=56.99e6, O=31.56e6 **(f)** Iter=265, W=63.37e6, O=16.48e6

Fig. 3: Snapshots of mGP progression on MMS ADAPTEC1 with standard cells, macros and fillers shown by red points, black rectangles and blue points. Total wirelength and object overlap are denoted by W and O, respectively.

by standard cells [20], [21], particular macro density smoothing [6], [8], macro shredding and pseudo-net insertion [7], etc.). In each iteration, we compute the gradient and preconditioner, predict the Lipschitz constant, and adjust steplength via backtracking. Nesterov's method solves the nonlinear problem iteratively till convergence.

A. Existing Problems

Line search remains the major runtime bottleneck in Conjugate Gradient method[2], which is used in prior nonlinear placers [6]. Moreover, in practice the steplength determined by line search usually fails to satisfy the optimizer's conjugacy requirement [17], i.e., the following search direction may not be orthogonal (w.r.t. the Hessian matrix) to all previous ones, while the theoretical convergence rate can not be expected. Instead of line search, [4] statically determines the steplength via upper-bounded moving distance per iteration, where solution quality is promised by sacrificing convergence rate. As a result, a systematic solution with dynamic steplength adjustment and theoretical support becomes desirable.

B. Nesterov's Method with Lipschitz Constant Prediction

In this work, we use Nesterov's method [15] for nonlinear optimization, as Algorithm 1 shows. There are two placement solutions $\mathbf{u}_k$ and $\mathbf{v}_k$ concurrently updated at each iteration k, only $\mathbf{u}$ is output as final solution in the end. α_k is the steplength while a_k is an optimization parameter. Initially, we set $a_0 = 1$ and have both $\mathbf{u}_0$ and $\mathbf{v}_0$ set as $\mathbf{v}_{mIP}$. $BkTrk$ denotes steplength backtracking (Section V-C), ∇f_{pre} denotes the preconditioned gradient (Section V-D). Instead of line search, we compute the steplength through a closed-form formula of the Lipschitz constant of the gradient defined as below.

Definition 1. $\forall$ *convex* $f(\mathbf{v}) \in C^{1,1}(H)$, $\exists L > 0$ *s.t.* $\forall \mathbf{u}, \mathbf{v} \in H$,

$$\|\nabla f(\mathbf{u}) - \nabla f(\mathbf{v})\| \leq L\|\mathbf{u} - \mathbf{v}\|. \tag{9}$$

H as Hilbert space is a generalized notion of Euclidean space, $C^{1,1}(H)$ requires $f(\mathbf{v})$ with Lipschitz continuous gradient. As our objective is non-convex, we leverage Nesterov's method in an approximate way. The convergence rate of Nesterov's method is $O(1/k^2)$ satisfying its steplength requirement (Eq. (4) of [15]). As [15] shows, $\alpha_k = L^{-1}$ satisfies the steplength requirement. The rationale behind is that smaller Lipschitz constant indicates higher smoothness of the gradient thus faster convergence can be achieved via larger steplength, vice versa. However, exact Lipschitz constant is expensive to compute, moreover, static estimation will be invalidated through iterative change of the cost function[3]. As a result, we approximate the Lipschitz constant and steplength as follows

$$\widetilde{L}_k = \frac{\|\nabla f(\mathbf{v}_k) - \nabla f(\mathbf{v}_{k-1})\|}{\|\mathbf{v}_k - \mathbf{v}_{k-1}\|},\ \alpha_k = \widetilde{L}_k^{-1}, \tag{10}$$

only $\mathbf{v}$ is used for Lipschitz constant prediction. The computation overhead is negligible since $\nabla f(\mathbf{v}_{k-1})$ and $\nabla f(\mathbf{v}_k)$ are known.

Algorithm 1 Nesterov's method in ePlace

```
Input: $a_k$, $\mathbf{u}_k$, $\mathbf{v}_k$, $\mathbf{v}_{k-1}$, $\nabla f_{pre}(\mathbf{v}_k)$, $\nabla f_{pre}(\mathbf{v}_{k-1})$
Output: $\mathbf{u}_{k+1}$, $\mathbf{v}_{k+1}$, $a_{k+1}$
1: $\alpha_k = BkTrk\left(\mathbf{v}_k, \mathbf{v}_{k-1}, \nabla f_{pre}(\mathbf{v}_k), \nabla f_{pre}(\mathbf{v}_{k-1})\right)$
2: $\mathbf{u}_{k+1} = \mathbf{v}_k - \alpha_k \nabla f_{pre}(\mathbf{v}_k)$
3: $a_{k+1} = \left(1 + \sqrt{4a_k^2 + 1}\right)/2$
4: $\mathbf{v}_{k+1} = \mathbf{u}_{k+1} + (a_k - 1)\left(\mathbf{u}_{k+1} - \mathbf{u}_k\right)/a_{k+1}$
5: return
```

C. Steplength Backtracking

We develop a backtracking method to enhance the prediction accuracy by preventing steplength overestimation, which misguides optimization. Being used to generate $\mathbf{v}_{k+1}$, however, α_k by Eq. (10) is predicted using $\mathbf{v}_k$ and $\mathbf{v}_{k-1}$. The iterative parameter adjustment in the cost function may deteriorate the prediction accuracy. As a result, our backtracking method predicts α_k using $\mathbf{v}_k$ and $\mathbf{v}_{k+1}$ instead. At line 1 of Algorithm 2, we set the steplength computed by Eq. (10) as a temporary variable $\hat{\alpha}_k$. The respective temporary solution $\hat{\mathbf{v}}_{k+1}$ (line 1) is used to produce a *reference steplength*. If it is exceeded by $\hat{\alpha}_k$ (line 2), we update $\hat{\alpha}_k$ and $\hat{\mathbf{v}}_{k+1}$ at line 3 and do the backtracking circularly until the inequality at line 2 is satisfied. Here $\epsilon = 0.95$ is

Algorithm 2 $BkTrk$

```
Input: solutions $\mathbf{v}_k$ and $\mathbf{v}_{k-1}$, gradients $\nabla f(\mathbf{v}_k)$ and $\nabla f(\mathbf{v}_{k-1})$
Output: steplength $\alpha_k$ and new solution $\mathbf{v}_{k+1}$
1: $\hat{\alpha}_k = \frac{\|\mathbf{v}_k - \mathbf{v}_{k-1}\|}{\|\nabla f(\mathbf{v}_k) - \nabla f(\mathbf{v}_{k-1})\|}$; $\hat{\mathbf{v}}_{k+1} = \mathbf{v}_k - \hat{\alpha}_k \nabla f(\mathbf{v}_k)$
2: while $\hat{\alpha}_k > \epsilon\left(\frac{\|\hat{\mathbf{v}}_{k+1} - \mathbf{v}_k\|}{\|\nabla f(\hat{\mathbf{v}}_{k+1}) - \nabla f(\mathbf{v}_k)\|}\right)$ do
3:     $\hat{\alpha}_k = \frac{\|\hat{\mathbf{v}}_{k+1} - \mathbf{v}_k\|}{\|\nabla f(\hat{\mathbf{v}}_{k+1}) - \nabla f(\mathbf{v}_k)\|}$; $\hat{\mathbf{v}}_{k+1} = \mathbf{v}_k - \hat{\alpha}_k \nabla f(\mathbf{v}_k)$
4: end while
5: $\mathbf{v}_{k+1} = \hat{\mathbf{v}}_{k+1}$; $\alpha_k = \hat{\alpha}_k$
6: return $\alpha_k$
```

the scaling factor to encourage earlier return of function $BkTrk$ thus prevent over-backtracking, which could consume too much runtime with limited accuracy improvement. The runtime overhead is zero if the first check at line 2 is passed, since the newly computed gradient $\nabla f(\hat{\mathbf{v}}_{k+1})$ can be reused at the following iteration. Experiments show that the average number of backtracks per iteration over all MMS benchmarks [21] is 1.037, indicating less than 4% runtime overhead on mGP. Disabling backtracking causes ePlace to fail on MMS BIGBLUE4 and increase wirelength by 43.12% in average of

[2]Our empirical studies on FFTPL [10] show that line search takes more than 60% of the total runtime on placing ADAPTEC1 of ISPD 2005.

[3]Wirelength coefficient γ in Eq. (3) and penalty factor λ in Eq. (4) are both iteratively adjusted.

the remaining 15 MMS benchmarks, showing the importance of our steplength backtracking method.

D. Nonlinear Placement Preconditioning

Preconditioning has broad application in quadratic placers [7], [20] but zero attempts in nonlinear placers [1], [4], [6], mainly due to the non-convexity of the density function. In this work, we approximate the original Hessian H_f with a positive definite diagonal matrix $\widetilde{H}_f$ as the preconditioner. We apply it to the gradient vector and use $\nabla f_{pre} = \widetilde{H}_f^{-1}\nabla f$ to direct optimization. The horizontal part is

$$H_{f_x} \approx \widetilde{H}_{f_x} = \begin{pmatrix} \frac{\partial^2 f(\mathbf{v})}{\partial x_1^2} & 0 & \cdots & 0 \\ 0 & \frac{\partial^2 f(\mathbf{v})}{\partial x_2^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial^2 f(\mathbf{v})}{\partial x_n^2} \end{pmatrix} \tag{11}$$

By Eq. (4) 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. (3) by two orders is computationally expensive and we use the vertex degree of object i instead,

$$\frac{\partial^2 W(\mathbf{v})}{\partial x_i^2} = \sum_{e\in E_i}\frac{\partial^2 W_e(\mathbf{v})}{\partial x_i^2} \Rightarrow |E_i|, \tag{12}$$

where E_i denotes the net subset incident to the object i. The non-convexity of the density function in Eq. (5) disables the traditional preconditioner to achieve expected performance. Eq. (13) shows its two-order differentiation

$$\frac{\partial^2 N(\mathbf{v})}{\partial x_i^2} = q_i\frac{\partial^2 \psi_i(\mathbf{v})}{\partial x_i^2} = q_i\frac{-\partial \xi_{i_x}(\mathbf{v})}{\partial x_i} \Rightarrow q_i. \tag{13}$$

Here we use the linear term q_i as the density preconditioner. The total preconditioner is concisely formulated as $\widetilde{H}_f = |E_i| + \lambda q_i$, thus we have the preconditioned gradient $\nabla f_{pre} = (|E_i| + \lambda q_i)^{-1}\nabla f$. Disabling preconditioner causes ePlace to fail on nine MMS benchmarks, since macros have much larger area thus magnitude of gradient than standard cells. As a result, unpreconditioned gradient makes macros bounce between opposite placement boundaries, causing the solution to oscillate thus fail to converge within limited number of iterations (3000 in ePlace). In average of the remaining seven MMS benchmarks, the wirelength is increased by 24.63%, indicating the effectiveness of our preconditioner.

VI. Macro Legalization (mLG) & Standard-Cell Global Placement (cGP)

Based on the mGP solution $\mathbf{v}_{mGP}$, mLG legalizes and fixes the macro layout, while cGP mitigates the quality overhead due to mLG.

A. Macro Legalization (mLG)

Unlike traditional simulated annealing (SA) based floorplanners and macro placers [2], [3], [20] which perturb floorplan expression then physically realize it, mLG uses SA to directly control macro motion. We expect a high-quality solution from mGP. Only local macro shifts are needed in mLG, while the shrunk design space can be well explored by SA. As Figure 4 shows, mLG can be decomposed into two levels. At each iteration of the outer loop (mLG iteration) we update the cost function $f_{mLG}(\mathbf{v})$, as Eq. (14) shows.

$$f_{mLG}(\mathbf{v}) = W(\mathbf{v}) + \mu_D D(\mathbf{v}) + \mu_O O_m(\mathbf{v}), \tag{14}$$

where $W(\mathbf{v})$, $D(\mathbf{v})$ and $O_m(\mathbf{v})$ denote the total wirelength, total standard-cell area covered by macros and total macro overlap, respectively. We set mLG as a constrained optimization.

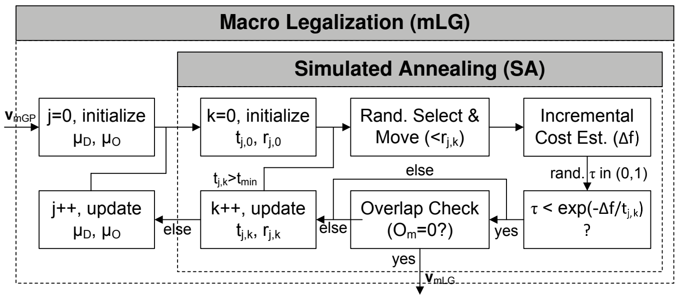
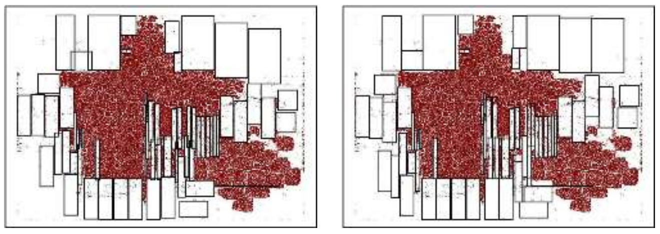


Fig. 4: Our two-level annealing-based macro legalizer.



(a) j=0, W=63.37e6, D=12.1e5, Om=6.1e5 **(b)** j=2, W=64.36e6, D=14.7e5, Om=0

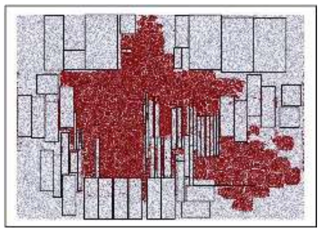
Fig. 5: Distribution of macros (a) before and (b) after mLG on MMS ADAPTEC1 with fixed standard-cell layout.

- **Objective** is to minimize $W(\mathbf{v}) + \mu_D D(\mathbf{v})$. Since penalty on $D(\mathbf{v})$ will be transformed to wirelength during cGP and cDP, we treat them equally in mLG thus statically set $\mu_D = \frac{W(\mathbf{v})}{D(\mathbf{v})}$.
- **Constraint** is zero macro overlap ($O_m(\mathbf{v}) = 0$). We set μ_O as the penalty factor and multiply it by κ at each mLG iteration to make the legalizer more aggressive on macro overlap reduction.

At each iteration of the inner loop shown in Figure 4 (SA iteration), the annealer randomly picks a macro and randomly determine its motion vector within the search range. The cost difference Δf is then incrementally evaluated and we generate a random number $\tau \in (0, 1)$ to determine whether the new layout will be accepted by $\tau < \exp\left(-\frac{\Delta f}{t_{j,k}}\right)$. Here j and k denote the mLG and SA iteration indexes. The **temperature** $t_{j,k}$ at each iteration (j, k) is determined based on the maximum cost increase $\Delta f_{max}(j, k)$ that will be accepted by more than 50% probability, thus we set $t_{j,k} = \frac{\Delta f_{max}(j,k)}{\ln 2}$. We set $\Delta f_{max}(j, 0)$ ($\Delta f_{max}(j, k_{max})$) as $0.03 \times \kappa^j$ ($0.0001 \times \kappa^j$), denoting that cost increase by less than 3% (0.01%) at the first (last) SA iteration will be accepted by more than half chance. These parameters appear small but fit well into our framework, since only minor layout change is expected. Meanwhile, they are scaled up per mLG iteration to adapt to enhancement of penalty factor μ_O. We initialize $\Delta f_{max}(j, k)$ by $\Delta f_{max}(j, 0)$ and linearly decreased towards $\Delta f_{max}(j, k_{max})$. The **radius** $r_{j,k}$ of macro motion range is dependent on both the penalty factor and the amount of macros. Given m macros to legalize, we set $r_{j,0} = \frac{R_x}{\sqrt{m}} \times 0.05 \times \kappa^j$, which means the entire placement region R can be decomposed into m subregions, every macro can be moved within 5% of its assigned region at each time. Similar to the temperature, the radius is scaled by κ at each mLG iteration. In practice, we set $\kappa = 1.5$ to achieve good tradeoff between quality and efficiency.

B. Standard-Cell Global Placement (cGP)

Despite fixed macros, cGP uses the same algorithm as that of mGP. In contrast, cGP introduces only small changes to the standard-cell layout and converges much faster than mGP. mLG is unaware of existing fillers in $\mathbf{v}_{mGP}$ and may introduce substantial macro-to-filler overlap. As a result, we retrieve all the fillers and conduct a filler-only placement for 20 iterations to relocate them appropriately. The result is of minimal density cost such that subsequent placement of standard


(a) Iter=0, W=64.36e6, O=16.13e6

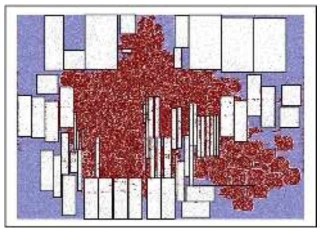

(b) Iter=51, W=63.04e6, O=16.29e6

Fig. 6: Distribution of standard cells and fillers (a) before and (b) after cGP on MMS ADAPTEC1 with fixed macro layout.

cells will not sacrifice wirelength for density. Experiments show that in average of all MMS benchmarks, the wirelength will be increased by 6.53% if we disable the filler-only placement. All the standard cells and fillers are then co-optimized by cGP. The initial penalty factor λ_{cGP}^{init} is determined based on the penalty factor λ_{mGP}^{last} at the last mGP iteration. As λ will be multiplied by up to 1.1 for maximal aggressiveness enhancement, we set $\lambda_{cGP}^{init} = \lambda_{mGP}^{last} \times 1.1^{-m}$ denoting that m buffering iterations are budgeted for cGP to recover the aggressiveness of mGP. This is shown in the cGP section of Figure 2, where the wirelength (overlap) reduces (increases) sharply to approach a low-wirelength initial solution for cGP (similar to what mIP does). By increasing λ_{cGP} iteratively, cGP reduces the existing overlap with small wirelength overhead. In practice, we set m as the number of mGP iterations divided by ten.

VII. Experiments and Results

We implement ePlace using C programming language and execute the program in a Linux machine with Intel i7 920 2.67GHz CPU and 12GB memory. To validate the performance of ePlace as a generalized algorithm, we conduct experiments on **ISPD 2005** [13] and **ISPD 2006** [12] (standard cell-based) benchmarks (with mLG and cGP disabled in ePlace), as well as experiments on modern mixed-size (**MMS**) benchmarks [21]. Notice that we use the original benchmarks in all the experiments (i.e., without modification to the circuits). There is a benchmark-specific density upper-bound ρ_t in ISPD 2006 to incorporate routability concern. By the contest protocol, exceeding ρ_t will induce wirelength penalty as $sHPWL = HPWL \times (1 + 0.01 \times \tau_{avg})$ where τ_{avg} denotes the scaled density overflow per bin. **MMS** benchmarks inherit the same netlists and density constraints from ISPD 2005 and ISPD 2006 benchmarks but have macros freed and fixed IO blocks inserted. Detailed MMS circuit statistics can be found in [21]. ePlace invokes the detail placer in [4] for legalization and detail placement of standard cells (cDP). There is no benchmark specific parameter tuning in our work, and we use the official scripts to evaluate the placement performance.

Twelve state-of-the-art standard-cell and mixed-size placers are included for performance comparison, namely, Capo10.5 [16], FastPlace3.0 [20], RQL [19], MAPLE [8], ComPLx (v13.07.30) [7], BonnPlace [18], POLAR [9], APlace3 [6], mPL6 [1], NTUplace3-unified [4], FLOP [21], FFTPL [10]. We have obtained the binaries of eight placers and executed them in our machine. RQL, MAPLE, BonnPlace and FLOP are not available due to IP and other issues, thus we cite their performance from respective publications. Capo10.5 and mPL6 fail to work with MMS benchmarks in our machine, so we cite the respective results from [21]. Also, APlace3 crashes on every MMS circuit as reported in [21] thus is not included in the respective experiments. MP-tree [3] and CG [2] are not available and have been outperformed by NTUplace3-unified [4] with 21% and 9% shorter wirelength, thus we do not include them in our experiments.

The results on ISPD 2005, ISPD 2006 and MMS circuits are shown in Table I, II and III, respectively. On ISPD 2005 circuits, ePlace produces the best solutions for all the eight cases and outperforms the leading placer BonnPlace by in average 2.83% shorter wirelength and 3.05× faster runtime. On ISPD 2006 circuits, ePlace produces the best solutions for seven of the eight cases and outperforms the leading placer MAPLE by in average 4.59% shorter wirelength and 2.84× faster runtime. On the MMS circuits, ePlace produces the best solutions for eleven of the sixteen cases. Despite none macro rotation or flipping, ePlace still outperforms the leading mixed-size placer NTUplace3-unified by in average 7.13% shorter wirelength with essentially the same runtime. ePlace also achieves the smallest density overflow among all the placers except for Capo10.5 on ISPD 2006 circuits, which lags behind ePlace by 43.73% wirelength. Figure 7 shows the **runtime breakdown** of ePlace on MMS benchmarks. mGP is the most effective placement stage (as Figure 2 shows) and it consumes the longest runtime. A further runtime breakdown of mGP shows that computation of density and wirelength gradients and other operations (Lipschitz constant prediction, parameter update, etc.) consume 57%, 29% and 14% runtime of mGP, respectively.

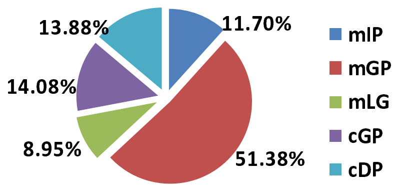


Fig. 7: The runtime breakdown of ePlace in average of MMS benchmarks.

VIII. Conclusion

ePlace is a generalized and effective placement algorithm to handle standard-cell and mixed-size circuits of large scale. Using the novel density function based on electrostatics, macros and standard cells are equalized by preconditioning and smoothly co-optimized by Nesterov's method with steplength determined by Lipschitz continuity. ePlace resolves the traditional bottlenecks in nonlinear placement (low efficiency due to line search, suboptimality of netlist clustering, etc.) and shows that nonlinear placement has the capability to outperform cutting-edge quadratic placement [8], [9], [18] with better quality and comparable or even shorter runtime. Our future work targets acceleration via parallel computation and extension towards other design objectives like timing, routability, and etc..

IX. Acknowledgement

The authors acknowledge the support of NSF CCF-1017864.

References

[1] T. F. Chan, J. Cong, J. R. Shinnerl, K. Sze, and M. Xie. mPL6: Enhanced Multilevel Mixed-Size Placement. In *ISPD*, pages 212–214, 2006.
[2] H.-C. Chen, Y.-L. Chunag, Y.-W. Chang, and Y.-C. Chang. Constraint Graph-Based Macro Placement for Modern Mixed-Size Circuit Designs. In *ICCAD*, pages 218–223, 2008.
[3] T.-C. Chen, P.-H. Yuh, Y.-W. Chang, F.-J. Huang, and D. Liu. MP-Trees: A Packing-Based Macro Placement Algorithm for Modern Mixed-Size Designs. *IEEE TCAD*, 27(9):1621–1634, 2008.
[4] M.-K. Hsu and Y.-W. Chang. Unified Analytical Global Placement for Large-Scale Mixed-Size Circuit Designs. *IEEE TCAD*, 31(9):1366–1378, 2012.
[5] M.-K. Hsu, Y.-W. Chang, and V. Balabanov. TSV-Aware Analytical Placement for 3D IC Designs. In *DAC*, pages 664–669, 2011.
[6] A. B. Kahng and Q. Wang. A Faster Implementation of APlace. In *ISPD*, pages 218–220, 2006.
[7] M.-C. Kim and I. Markov. ComPLx: A Competitive Primal-dual Lagrange Optimization for Global Placement. In *DAC*, pages 747–752, 2012.
[8] M.-C. Kim, N. Viswanathan, C. J. Alpert, I. L. Markov, and S. Ramji. MAPLE: Multilevel Adaptive Placement for Mixed-Size Designs. In *ISPD*, pages 193–200, 2012.
[9] T. Lin, C. Chu, J. R. Shinnerl, I. Bustany, and I. Nedelchev. POLAR: Placement based on Novel Rough Legalization and Refinement. In *ICCAD*, pages 357–362, 2013.

TABLE I: HPWL ($\times 10^6$) on the ISPD 2005 benchmark suite [13]. CP=Capo, FP=FastPlace, MPE=MAPLE, CPx=ComPLx, BPL=BonnPlace, AP=APlace, NP3U=NTUplace3-unified. Cited results are marked with *. All the results are evaluated by the official scripts [13].

Categories		Min-Cut	Quadratic						Nonlinear				
Benchmarks	# Cells	CP10.5	FP3.0	RQL*	MPE*	CPx	BPL*	POLAR	AP3	NP3U	mPL6	FFTPL	ePlace
ADAPTEC1	211K	88.70	78.34	77.82	76.36	77.73	76.87	77.21	78.35	80.29	77.93	76.46	**74.63**
ADAPTEC2	255K	103.50	93.47	88.51	86.95	88.84	86.36	86.16	95.68	90.18	92.04	85.57	**84.84**
ADAPTEC3	452K	235.78	213.48	210.96	209.78	203.45	202.00	201.30	218.52	233.77	214.16	202.16	**194.57**
ADAPTEC4	496K	205.97	196.88	188.86	179.91	183.16	181.53	182.37	209.28	215.02	193.89	185.83	**179.02**
BIGBLUE1	278K	107.58	96.23	94.98	93.74	94.41	94.85	94.67	100.02	98.65	96.80	91.64	**90.99**
BIGBLUE2	558K	163.75	154.89	150.03	144.55	145.37	144.21	143.85	153.75	158.27	152.34	145.54	**141.83**
BIGBLUE3	1097K	407.28	369.19	323.09	323.05	337.72	317.71	324.53	411.59	346.33	344.25	359.00	**308.77**
BIGBLUE4	2177K	952.20	834.04	797.66	775.71	788.30	781.79	781.06	871.29	829.09	829.44	805.90	**753.20**
Average HPWL		21.14%	10.0%	5.40%	3.21%	4.50%	2.83%	3.08%	14.33%	12.05%	8.33%	4.70%	0.00%
Average Runtime		8.94×	0.53×	0.91×	2.84×	0.52×	3.05×	0.52×	9.13×	1.40×	3.78×	2.21×	1.00×

TABLE II: Scaled HPWL ($\times 10^6$) on the ISPD 2006 benchmark suite [12]. CP=Capo, FP=FastPlace, MPE=MAPLE, CPx=ComPLx, AP=APlace, NP3U=NTUplace3-unified. Cited results are marked with *. All the results are evaluated by the official scripts [12].

Categories			Min-Cut	Quadratic					Nonlinear			
Benchmarks	# Cells	ρ_t	CP10.5*	FP3.0	RQL*	MPE*	CPx	POLAR	AP3*	NP3U	mPL6	ePlace
ADAPTEC5	843K	0.5	494.64	472.72	443.28	407.33	415.77	438.47	520.97	444.41	428.31	**397.53**
NEWBLUE1	330K	0.8	98.48	74.11	64.43	69.25	64.75	67.52	73.31	**61.01**	72.62	62.31
NEWBLUE2	442K	0.9	309.53	206.04	199.60	191.66	193.06	191.25	198.24	194.80	201.91	**182.69**
NEWBLUE3	494K	0.8	361.25	297.46	269.33	268.07	273.42	271.28	273.64	275.08	285.26	**266.80**
NEWBLUE4	646K	0.5	362.40	308.35	308.75	282.49	292.82	305.14	384.12	296.62	298.20	**276.13**
NEWBLUE5	1233K	0.5	659.57	621.47	537.49	515.04	507.74	521.85	613.86	537.92	535.80	**492.62**
NEWBLUE6	1255K	0.8	668.66	549.87	515.69	494.82	501.05	512.06	522.73	534.96	523.47	**464.44**
NEWBLUE7	2508K	0.8	1518.75	1105.43	1057.80	1032.60	1041.21	1045.20	1098.90	1096.16	1085.68	**989.96**
Average Scaled HPWL			43.73%	16.25%	7.99%	4.59%	4.86%	7.16%	18.38%	7.74%	10.11%	0.00%
Average Runtime			6.68×	0.59×	N/A	N/A	0.55×	0.69×	10.21×	1.63×	3.71×	1.00×
Average Density Overflow			0.45×	5.90×	9.08×	5.46×	4.19×	13.77×	5.58×	12.29×	7.14×	1.00×

TABLE III: HPWL and scaled HPWL ($\times 10^6$) on the MMS benchmark suite [21]. Mac=Macros, CP=Capo, FP=FastPlace, CPx=ComPLx, NP3U=NTUplace3-unified, NR="no rotation or flipping of macros". Cited results are marked with *. All the results are evaluated by the official scripts [21].

Categories				Constructive		One-Stage						
Benchmarks	# Cells	# Mac	ρ_t	CP10.5*	FLOP*	FP3.0	CPx	POLAR	mPL6*	NP3U-NR	NP3U	ePlace
ADAPTEC1	211K	63	1.0	84.77	76.83	82.39	79.05	92.17	77.84	75.92	75.55	**67.15**
ADAPTEC2	255K	127	1.0	92.61	84.14	88.53	99.11	149.43	88.40	84.89	78.50	**77.37**
ADAPTEC3	452K	58	1.0	202.37	175.99	187.98	175.78	197.48	180.64	170.88	169.74	**164.50**
ADAPTEC4	496K	69	1.0	202.38	161.68	187.50	156.75	175.19	162.02	167.13	166.68	**148.39**
BIGBLUE1	278K	32	1.0	112.58	94.92	104.91	96.18	99.12	99.36	96.42	96.57	**86.82**
BIGBLUE2	558K	959	1.0	149.54	153.02	145.89	147.19	157.72	144.37	148.12	147.17	**130.18**
BIGBLUE3	1097K	2549	1.0	583.37	346.24	400.40	344.63	420.28	319.63	324.39	338.47	**302.29**
BIGBLUE4	2177K	199	1.0	915.37	777.84	775.43	772.53	814.07	804.00	797.17	799.66	**657.92**
ADAPTEC5	843K	76	0.5	565.88	357.83	338.77	338.67	380.45	376.30	295.24	**294.24**	315.76
NEWBLUE1	330K	64	0.8	110.54	67.97	73.91	65.26	70.68	66.93	**61.13**	61.25	62.56
NEWBLUE2	442K	3748	0.9	303.25	187.40	197.15	187.87	197.65	179.18	164.27	**163.76**	166.59
NEWBLUE3	494K	51	0.8	1282.19	345.99	325.72	**269.47**	601.17	415.86	N/A	280.92	304.24
NEWBLUE4	646K	81	0.5	300.69	256.54	270.70	256.97	277.60	277.69	231.59	**229.36**	229.95
NEWBLUE5	1233K	91	0.5	570.32	510.83	500.09	453.05	450.69	515.49	414.81	420.46	**393.21**
NEWBLUE6	1255K	74	0.8	609.16	493.64	512.19	452.83	475.78	482.44	471.51	474.86	**410.04**
NEWBLUE7	2508K	161	0.8	1481.45	1078.18	1016.10	1010.00	1107.59	1038.66	N/A	1100.84	**897.81**
Average (Scaled) HPWL				64.42%	14.31%	18.22%	10.90%	30.69%	16.13%	7.40%	7.13%	0.00%
Average Runtime				14.69×	2.14×	0.37×	1.15×	0.71×	6.34×	0.81×	1.05×	1.00×
Average Density Overflow				N/A	N/A	2.08×	1.69×	9.11×	N/A	5.37×	5.26×	1.00×

[10] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-H. Huang, C.-C. Teng, and C.-K. Cheng. FFTPL: An Analytic Placement Algorithm Using Fast Fourier Transform for Density Equalization. In *ASICON*, 2013.

[11] J. Lu and C.-W. Sham. LMgr: A Low-Memory Global Router with Dynamic Topology Update and Bending-Aware Optimum Path Search. In *ISQED*, pages 231–238, 2013.

[12] G.-J. Nam. ISPD 2006 Placement Contest: Benchmark Suite and Results. In *ISPD*, pages 167–167, 2006.

[13] G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter, and M. Yildiz. The ISPD2005 Placement Contest and Benchmark Suite. In *ISPD*, pages 216–220, 2005.

[14] W. C. Naylor, R. Donelly, and L. Sha. Non-Linear Optimization System and Method for Wire Length and Delay Optimization for an Automatic Electric Circuit Placer. In *US Patent 6301693*, 2001.

[15] Y. E. Nesterov. A Method of Solving A Convex Programming Problem with Convergence Rate $O(1/k^2)$. *Soviet Math*, 27(2):372–376, 1983.

[16] J. A. Roy, S. N. Adya, D. A. Papa, and I. L. Markov. Min-Cut Floorplacement. *IEEE TCAD*, 25(7):1313–1326, 2006.

[17] J. Shewchuk. An Introduction to the Conjugate Gradient Method without the Agonizing Pain. In *CMU-CS-TR-94-125,*, 1994.

[18] M. Struzyna. Sub-Quadratic Objectives in Quadratic Placement. In *DATE*, pages 1867–1872, 2013.

[19] N. Viswanathan, G.-J. Nam, C. J. Alpert, P. Villarrubia, H. Ren, and C. Chu. RQL: Global Placement via Relaxed Quadratic Spreading and Linearization. In *DAC*, pages 453–458, 2007.

[20] N. Viswanathan, M. Pan, and C. Chu. FastPlace3.0: A Fast Multilevel Quadratic Placement Algorithm with Placement Congestion Control. In *ASPDAC*, pages 135–140, 2007.

[21] J. Z. Yan, N. Viswanathan, and C. Chu. Handling Complexities in Modern Large-Scale Mixed-Size Placement. In *DAC*, 2009.