RePlAce: Advancing Solution Quality and Routability Validation in Global Placement

Chung-Kuan Cheng, *Fellow, IEEE*, Andrew B. Kahng, *Fellow, IEEE*, Ilgweon Kang, *Member, IEEE*, and Lutong Wang, *Student Member, IEEE*

***Abstract*— The Nesterov's method approach to analytic placement [27] [28] [29] has recently demonstrated strong solution quality and scalability. We dissect the previous implementation strategy of [29] and show that solution quality can be significantly improved using two levers: *constraint-oriented local smoothing*, and *dynamic step size adaptation*. We propose a new density function that comprehends local overflow of area resources; this enables a constraint-oriented local smoothing at per-bin granularity. Our improved dynamic step size adaptation automatically determines step size and effectively allocates optimization effort to significantly improve solution quality without undue runtime impact. Our resulting global placement tool, RePlAce, achieves an average of 2.00% HPWL reduction over all best known ISPD-2005 and ISPD-2006 benchmark results, and an average of 2.73% over all best known MMS benchmark results, without any benchmark-specific code or tuning. We further extend our global placer to address routability, and achieve on average 8.50% to 9.59% scaled HPWL reduction over previous leading academic placers for the DAC-2012 and ICCAD-2012 benchmark suites. To our knowledge, RePlAce is the first work to achieve superior solution quality across all the ISPD-2005, ISPD-2006, MMS, DAC-2012 and ICCAD-2012 benchmark suites with a single global placement engine.**

I. Introduction

Placement is a fundamental, critical step in the physical design of integrated circuits (ICs) [19]. Placement solution quality directly impacts overall design quality of results (QoR) with respect to timing closure, die utilization, routability, and design turnaround time; these in turn affect the classic metrics of operating frequency, yield, power consumption and cost. Despite significant improvement in placement algorithms over the past decades [31], efficient and effective placement remains a challenging issue [2].

Among all academic placers, recent electrostatics-based placement (ePlace) implementations [27] [28] [29] achieve benchmark solutions that rank among the best known in terms of half-perimeter wirelength (HPWL). ePlace is a flat, nonlinear analytical global placement engine with electrostatics-based global-smooth density cost function and Nesterov's method nonlinear optimizer. The density cost function enables effective movement of standard cells and macros over fixed instances, blockages and large macros. The density cost is solved numerically by a fast Fourier transform (FFT) with high accuracy and $O(n \log n)$ complexity. Nesterov's method provides accelerated convergence, with steplength dynamically predicted via Lipschitz constant. A backtracking method effectively prevents steplength overestimation. ePlace is capable of standard-cell placement [28], mixed-size placement [29], and 3D-IC mixed-size placement [30]. Instance size differences between standard cells and macros are addressed by an approximated nonlinear preconditioner. Yet despite these and other previous efforts, our present work demonstrates the availability of significant further improvement over *best known* HPWL results for standard academic benchmarks. Furthermore, ePlace [27] [28] [29] cannot produce routable placements, e.g., for *SUPERBLUE12* [39] ePlace routing hotspots demand 211.73% of the routing supply (contrast this with RePlAce's RC value of 102.43% in Table VII). In RePlAce, we add optimization of routability in global routing to the Nesterov's approach, achieving substantial *scaled HPWL* improvements over previous leading academic placers for the DAC-2012 [39] and ICCAD-2012 [40] benchmark suites.

Density function and density penalty factor. Conventional global placement methodology seeks to minimize wirelength subject to density constraints which mitigate instance overlaps. The density constraints can be transformed to yield an unconstrained objective with a density penalty factor, as shown in Equation (1). Previous nonlinear placers [6] [20] [28] apply the density penalty factor globally across the entire placement region, with the penalty factor increased proportionally [20] [28], or at a constant rate [6], until the end of global placement. Such approaches suffer from the "global" nature of their iterations, which can overlook the fine-grain spatial and temporal behavior of the placement procedure. In other words, a globally-applied penalty factor can be insensitive to density variations across the placement region, and a fixed schedule for growth of the density penalty factor will not discern between early and late stages of global placement. This can lead to unnecessary suboptimality of solutions.

Routability-driven placement. Routability is a fundamental requirement of real-world global placement [1] [39] [40], as the placement process must provide a routable placement solution to the router. It is well-understood that the standard minimum total HPWL placement objective at some point becomes detrimental to routability. Previous works achieve improved routability via (i) congestion estimation, and (ii) congestion mitigation. Several works [4] [41] [44] are based on placement properties (e.g., Rent's parameter, pin density, net

Manuscript received August 24, 2017.

C. K. Cheng and I. Kang are with the Department of Computer Science and Engineering, University of California at San Diego, La Jolla, CA 92093, USA (E-mail: ckcheng@ucsd.edu; igkang@ucsd.edu).

A. B. Kahng is with the Departments of Computer Science and Engineering, and Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093, USA (E-mail: abk@ucsd.edu).

L. Wang is with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093, USA (E-mail: luw002@ucsd.edu).

overlapping, etc.), without considering actual routing. Probabilistic estimations assume a uniform wire density model [37], or pattern routing considering wire bends and vias [42]. More recent and effective constructive estimations used in recent routability-driven placers [10] [21] [23] are based on global routers [16] [25] [35]. To mitigate congestion, the works of [13] [17] [37] formulate a routability-driven objective function with multiple Lagrangian multipliers. The works of [9] [10] [21] [23] implement cell inflation with local refinement, or a rough legalizer during global placement, to spread overlapped instances. The DAC-2012 [39] and ICCAD-2012 [40] routability-driven global placement benchmark suites are the most recent academic evaluation frameworks that address the routability issue at the global routing stage, and are widely used to validate the performance of academic placers. Among all published results for these two benchmark suites, [7] [10] [13] [23] show leading-edge solution qualities in terms of scaled HPWL, considering routing congestion (RC) as a penalty factor to HPWL as defined in [39] [40]. A separate body of work (e.g., [8] [14]) addresses the routing-driven ISPD-2014 [45] and ISPD-2015 [3] benchmarks. However, quality of results heavily depends on detailed placers that are sensitive to details of technology and library cells, which is beyond scope for a global placement framework such as ours.

In our present work, we achieve global placement solution quality well beyond published (best known) results for the ISPD-2005, ISPD-2006 and MMS benchmark suites via a new constraint-oriented local-density function (RePlAce-ld), and an improved dynamic step size adaptation (RePlAce-ds, RePlAce-ldds). With extensions to support routability as assessed in global routing (RePlAce-r), we achieve superior solution quality on the DAC-2012 and ICCAD-2012 benchmark suites. Our contributions are summarized as follows.

- We propose a new constraint-oriented local-density function for mixed-size placement that incorporates (i) a constraint-oriented local-density penalty factor for each bin (i.e., local Lagrangian multiplier for each bin), and (ii) a constraint-oriented local-density cost coefficient for each instance. Combining the previous global density function [29] with a new local density function that comprehends local *density overflow* per bin[1], we obtain a global placement with *constraint-oriented local smoothing* that achieves improved solution quality.
- We propose a methodology for density-penalty adaptation via an improved *dynamic step size adaptation* that automatically adjusts the density penalty factor based on the *HPWL curve* (i.e., trajectory of HPWL cost versus iteration count) observed in a *trial placement procedure*.[2] Our improved dynamic step size adaptation applies more fine-grained control at transition points on the HPWL curve. Compared to a constant small step size, we obtain better solution quality while saving runtime.
- We propose a layer-aware cell inflation technique, considering per-layer pin blockages, and integrate the official global router *NCTU-GR* [48] of the DAC-2012 and ICCAD-2012 benchmark suites for congestion estimation. We develop a simple but effective superlinear cell inflation technique to mitigate global routing congestion during global placement. Following the strategy of recent leading works [9] [10], we further include a post-placement optimization by [24]. By integrating all our innovations to improve routability, our placer delivers solution quality in terms of scaled HPWL that substantially improves over previous leading academic placers for the DAC-2012 and ICCAD-2012 benchmark suites.[3]

The remainder of this paper is organized as follows. Section II briefly states the fundamental placement problem formulation. Section III introduces our constraint-oriented local-density function that enables local smoothing. Section IV describes our improved dynamic step size adaptation. Section V describes our methodology to improve routability, with congestion estimation by a global router and a cell inflation technique. Section VI presents our experimental setups and results. Section VII concludes the paper.

II. Placement Overview

Placement seeks to determine the location of instances (e.g., standard cells and macros) while addressing optimization objectives such as HPWL, routed wirelength, timing, power, routability, etc. A placement solution is represented as $\mathbf{v} = (\mathbf{x},\ \mathbf{y})^T = (x_1,\ x_2,\ \cdots,\ x_n;\ y_1,\ y_2,\ \cdots,\ y_n)^T$, where $(x_i,\ y_i)$ is the physical location (of the origin, with orientation) of the i^{th} instance. We follow the basic notations in [29] and formulate the placement objective function as shown in Equation (1).

$$\min_{\mathbf{v}} f(\mathbf{v}) = W(\mathbf{v}) + \lambda D(\mathbf{v}) \qquad (1)$$

The wirelength objective $W(\mathbf{v})$ is the HPWL of the design modeled with a weighted-average (WA) smoothing technique [12], while the density cost function $D(\mathbf{v})$ addresses instance overlap via an electrostatic analogy [29]. During nonlinear optimization, a density penalty factor λ is gradually increased to reduce overlap, at the cost of increased wirelength.

III. Constraint-Oriented Local-Density Function

We now describe our improvement of the previous electrostatics-based density formulation in [27] [28] [29].

A. Necessity of Local Density Function

Scaling the density penalty factor in the placement objective function is critical since this directly impacts the placement solution quality. Like many other analytical placers [6] [11], the previous ePlace implementations of [27] [28] [29] use the

[1]Density overflow for a given placement bin is defined as the total area of instances inside the placement bin, minus the placement bin area.

[2]As we describe in detail below, a trial placement procedure is performed initially to capture *transition points* on the HPWL curve; these transition points inform the step size adaptation.

[3]We use these benchmarks instead of the ISPD-2014 and ISPD-2015 benchmarks since our focus is on mitigating congestion reported by the global router, along the lines of well-addressed, industry-formulated routability-driven global placement contests [39] [40]. From a practical IC implementation flow (i.e., turnaround time) standpoint, global placement-based mitigation of (global) routing congestion remains crucially important. In Section VI-C below, a brief comparison between RePlAce and a leading-edge commercial placer suggests that remaining "gaps" between academic research and industry practice are potentially tractable in today's university research context.

density penalty factor λ, applied equally to every placement grid or bin in each iteration, to balance wirelength and density costs. However, in the present work, we make the motivating observation that globally applying the density penalty factor (i.e., applying a global Lagrangian multiplier) sacrifices wirelength in less-overlapped bins to resolve more-overlapped bins, resulting in unnecessary wirelength increase.

With this in mind, we propose a new *local-density* function that comprehends locally overflowed (with respect to area resources) bins; this enables *constraint-oriented local smoothing* at a per-bin granularity. To help effective removal of overlaps among placement instances, our local-density function (per each bin) provides more repulsive forces[4] for overflowed bins, on the basis of the global-smooth density distribution obtained by global-density function. This is beneficial since the local-density function on top of the global-density function effectively helps us restrain suboptimal wirelength increase caused by global-density penalty factor λ. In this section, we use the term *global* when we refer to a density penalty that is applied equally throughout the layout, and *local* when we refer to penalty factors or coefficients that are separately defined and used at a per-bin granularity. We denote our new local-density function as $D^{local}(\mathbf{v})$, and the previous global-density function of [27] [28] [29] as $D^{global}(\mathbf{v})$.

With respect to Equation (1), we begin with $D(\mathbf{v}) = D^{global}(\mathbf{v})$, and then add a further term $D^{local}(\mathbf{v})$ to introduce the local smoothing to our placement engine. The local density function incorporates two innovations. First, we formulate a *constraint-oriented local-density penalty factor* ν_j (i.e., local Lagrangian multiplier) per bin to spread cells in highly overflowed regions by increased repulsive forces (Section III-B). Second, we apply a *constraint-oriented local-density coefficient* Δ_i per instance i to the local-density function (Section III-C). Δ_i helps the instance i maintain a certain amount of repulsive force induced from overflowed bins, even as the instance i escapes from (i.e., is no longer contained in) those overflowed bins.

B. Constraint-Oriented Local-Density Penalty for Each Bin b_j

To enable the constraint-oriented local smoothing, we introduce a local-density penalty factor ν_j (i.e., local Lagrangian multiplier) per each bin b_j based on demands for area resources. We formulate the local-density penalty factor ν_j as

$$\nu_j = e^{\alpha \cdot (BinDemand_j - BinCapacity_j)}. \tag{2}$$

In Equation (2), $BinCapacity_j$ and $BinDemand_j$ respectively denote the area of bin b_j and the total area of cells intersecting b_j. We define b_j's *overflow* as $(BinDemand_j - BinCapacity_j)$. The bin b_j is *overflowed* if $BinDemand_j - BinCapacity_j > 0$. α is a coefficient to weight the local-density cost function as detailed in Equations (5) and (6) below; α starts at a very small value, e.g., $1e$-12 (empirically determined), and gradually increases through the Nesterov's optimization. When a bin b_j is overflowed, ν_j has exponentially larger value, generating larger repulsive force. Thus, cells in b_j experience

[4]We note that in our electrostatic analogy, the gradient of the cost function is the repulsive force from electric charges. We use "force" to refer to this repulsive force due to electric charges.

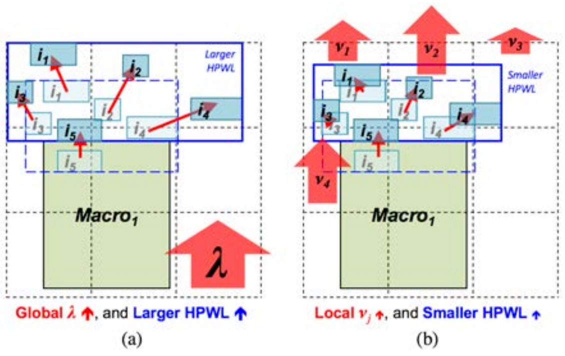
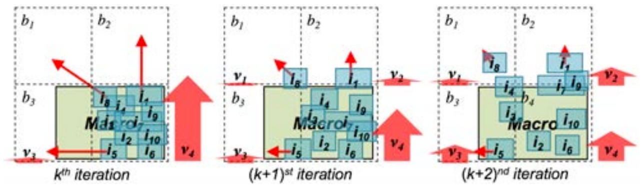
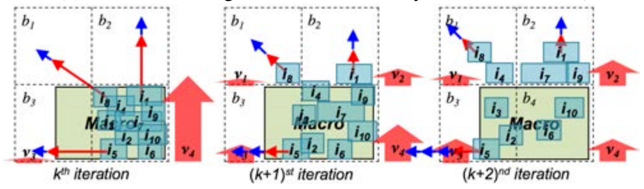


Fig. 1: Density forces with (a) global density-penalty factor λ, and (b) constraint-oriented local-density penalty factor ν_j per each bin. (Bin boundaries are indicated by black dotted lines. Standard-cell instances are labeled $i_1, \cdots, i_5$.) In (a), the global λ is applied equally to all cells in the layout, helping to remove overlap between i_5 and $Macro_1$. In (b), the local Lagrangian multiplier ν_j is applied, so that HPWL increases (from the dotted to the solid blue rectangle) less than in (a).



(a) Local smoothing without local-density cost coefficient Δ_i.



(b) Local smoothing with local-density cost coefficient Δ_i.

Fig. 2: Local smoothing methods (a) without and (b) with local-density cost coefficient Δ_i. Figures are ordered from left to right by iteration indices. Red arrows depict the repulsive force component induced from the ν_j. Blue arrows depict the repulsive force component induced from the local-density cost coefficient Δ_i. Figure (b) shows the effect of a larger force to spread cells from the overflowed bin b_4.

larger force to be spread toward not-overflowed bins. When b_j is not overflowed, cells in b_j experience small force. The local-density penalty factor ν_j is especially beneficial early in the placement procedure since it guides cells to quickly find their directions of movement.[5]

Figure 1 illustrates the benefit of the constraint-oriented local-density penalty factor ν_j. Each i represents a placement instance (cell) belonging to the same (5-pin) net, and red arrows indicate the repulsive force to spread cells. The faint outlines show the previous locations of cells. To remove the overlap between i_5 and $Macro_1$, in (a) the global density function $D^{global}(\mathbf{v})$ is applied to all cells, scaled uniformly by the global density penalty factor λ, even though most of the affected cells are not in the overflowed bins. The repulsive forces induced by $D^{global}(\mathbf{v})$ and λ cause a large

[5]Initial placement typically seeks only to minimize wirelength, which results in a number of highly overflowed bins.

HPWL increment, depicted by the transition between dotted and solid blue rectangles. By contrast, in (b) the constraint-oriented local-density penalty factor ν_j separately scales the magnitude of the repulsive force per each bin based on that bin's overflow, i.e., bins b_2 and b_4 have relatively larger ν_j to resolve overlaps. In this way, to achieve the same cell spreading, the version with our local-density function could use a smaller global Lagrangian multiplier (bins b_1 and b_3 experience smaller density force than in Figure 1(a)), resulting in a smaller HPWL increment.

C. Local-Density Cost Coefficient Δ_i per Each Cell i

With constraint-oriented local smoothing, per Equation (2), cell i from an overflowed bin immediately loses much of its repulsive force component after it escapes the overflowed bin. Without sufficient repulsive force induced from the global density function, the movement of cell i slows down if the adjacent bin to which it moves is not overflowed. Furthermore, the cell i can return to its previous, (formerly) overflowed bin as a consequence of wirelength-induced attractive force, causing undesired oscillation. In such a scenario, instances (cells) are not effectively spread to resolve the cell overlapping. To effectively achieve local smoothing via a local density penalty factor without globally sacrificing HPWL, we propose a new mechanism to maintain the repulsive forces generated by the constraint-oriented local-density penalty factor ν_j of the overflowed bin b_j.

Equation (3) gives the *constraint-oriented* local-density cost coefficient, Δ_i, per each cell i (c_i, $c_i \in b_j$).[6] Δ_i is used to multiplicatively scale the local density function $D^{local}(\mathbf{v})$.

$$\Delta_i^{iter+1} = \Delta_i^{iter} + \beta \cdot \frac{\max(Overflow_j, 0)}{\sum_i A_i} \tag{3}$$

Here, A_i is the area of cell i (c_i) and $iter$ is the index of the iteration. We initialize $\Delta_i = 0$. With respect to the current overflow of bin b_j, Δ_i accumulates positive overflow normalized by the total cell area in the design (i.e., $\sum_i A_i$). β is a coefficient to balance between global-density and local-density induced forces. β initially takes on a very small value, e.g., $1e$-13 (empirically determined), and gradually increases through the Nesterov's optimization.[7] Using $\max(Overflow_j, 0)$ instead of a constant encourages movement of only cells in overflowed bins.[8] Multiplicatively scaling the local-density function by Δ_i prevents cell i from losing its local-density penalty factor induced from the repulsive force, even as cell i moves out of the overflowed bin b_j. (See Equation (9) and Algorithm 1, which we now describe.)

Figure 2 illustrates the advantage of multiplying by the constraint-oriented local-density cost coefficient Δ_i. Bin b_4 is overflowed, while the other bins are not overflowed. Red arrows depict the repulsive force component induced from the local-density penalty factor ν_j. In Figure 2(a), without Δ_i, the magnitude of force rapidly decreases after a cell i escapes from the overflowed bin b_4, resulting in slower movements. In Figure 2(b), the local-density cost coefficient Δ_i compensates the loss of the repulsive force component induced from the overflowed bins. Blue arrows in Figure 2(b) indicate the repulsive force components induced from the local-density cost coefficient Δ_i in each iteration.

[6]We use the convention $c_i \in b_j$ to indicate that cell i intersects bin b_j.

[7]$\beta^{iter+1} = cof \times \beta^{iter}$ where cof is the step size defined in Section IV-A. Overall, since β increases at the same rate as λ, Δ_i still increases with increasing #iterations.

[8]Replacing $\max(Overflow_j, 0)$ in Equation (3) with 1/5/10% constant values, the resulting HPWLs are degraded by 0.38/0.40/0.49% on average.

TABLE I: Notations.

Term	Description
i	Index of the i^{th} placement instance (cell), $i = 1, ..., n$
c_i	i^{th} placement instance (cell), $i = 1, ..., n$
b_j	j^{th} bin in the placement region
B	A set of placement bins
q_i	Electric charge of the i^{th} placement instance
ϕ_j	Electric potential at bin j
ϕ_i	Electric potential at the location of i^{th} placement instance
$\mathbf{E}$	Gradient of the potential ϕ, i.e., electric field
$\mathbf{E}_j$	Electric field at bin j
A_{ij}	Overlap area between i^{th} placement instance and bin b_j
λ	Lagrangian multiplier for global density cost function
ν_j	Constraint-oriented local-density penalty factor of bin b_j
Δ_i	Local-density cost coefficient of the i^{th} placement instance

D. Formulation: Local Density Function and Gradient

Table I summarizes our notations.[9] Equation (4) shows the *global density function* introduced in [29],

$$D^{global}(\mathbf{v}) = \sum_i D_i^{global}(\mathbf{v}) = \sum_i q_i \phi_i(\mathbf{v}), \tag{4}$$

where q_i is the electric charge and ϕ_i is the potential of cell i. $D^{global}(\mathbf{v})$ is equal to $D(\mathbf{v})$ in Equation (1).

To achieve local smoothing, we add the *local-density function* $D^{local}(\mathbf{v})$ as a further term to our placement objective function. Using the constraint-oriented local-density penalty factor ν_j, we formulate the new local-density function $D^{local}(\mathbf{v})$ as

$$D^{local}(\mathbf{v}) = \sum_{b_j \in B} \nu_j D_j^{local}(\mathbf{v}) \ = \sum_{b_j \in B} \nu_j \Big(\sum_{c_i \in b_j} A_{ij} q_i \phi_j \Big), \tag{5}$$

where A_{ij} is the overlapped area between cell i and bin b_j, q_i is again the electric charge of cell i, and ϕ_j is the local electric potential of bin b_j.

Our new placement objective function $f(\mathbf{v})$ incorporates the global density function $D^{global}(\mathbf{v})$ of Equation (4) and the local density function $D^{local}(\mathbf{v})$:

$$\min_{\mathbf{v}} f(\mathbf{v}) = W(\mathbf{v}) + \lambda D^{global}(\mathbf{v}) + D^{local}(\mathbf{v}) \tag{6}$$

By differentiating Equation (6), we obtain the force to spread instances (cells and macros). Equation (7) gives the gradient of the global density function $D^{global}(\mathbf{v})$ described in [29]. Equation (8) gives the gradient of the local density function $D^{local}(\mathbf{v})$

$$\frac{\partial D^{global}(\mathbf{v})}{\partial x_i} = q_i \frac{\partial \phi_i}{\partial x_i} = q_i \mathbf{E}_i(\mathbf{v}), \tag{7}$$

[9]ϕ_j and $\mathbf{E}_j$ are the electric potential and field at bin j, respectively. They are calculated using the existing charge density (from current placement of all instances) by FFT. Due to the discrete nature of FFT, the electric potential and field has a discrete value at per-bin granularity. Thus, $\mathbf{E}_i$ for instance i is $\mathbf{E}_j$ where instance i is at bin j.

Algorithm 1 Local-Density Cost Function

```
1: Procedure $LocalDensityCostFunction(c_i)$
2: Initialize $\Delta_i \leftarrow 0$;
3: for all $b_j$ $(b_j \cap c_i \neq 0)$ do
4:     $overflow_j \leftarrow BinDemand_j - BinCapacity_j$;
5:     if $overflow_j > 0$ then
6:         $\Delta_i$ += $\beta \times \frac{overflow_j}{total\ cell\ area}$;
7:     end if
8:     $LDCost_i$ += $\Delta_i \times Gradient_j(c_i \in b_j)$;
9: end for
10: return $Gradient_i$;
11: Procedure $Gradient_j(c_i \in b_j)$
12: calculate $Gradient_j$ [29];
13: return $Gradient_j$;
```

$$\begin{aligned}\frac{\partial D^{local}(\mathbf{v})}{\partial x_i} &= \sum_{b_j \in B} \alpha \frac{\partial BinDemand_j}{\partial x_i} \nu_j \sum_{k \neq i, k \in b_j} A_{kj} q_k \phi_j \\ &+ \sum_{b_j \in B} \nu_j \left(\frac{\partial A_{ij}}{\partial x_i} q_i \phi_j + \sum_{k \neq i, k \in b_j} A_{kj} q_k \mathbf{E}_j \right),\end{aligned} \tag{8}$$

where $\mathbf{E}$ is the electric field.

$Cost_i$ in Equation (9) determines the cell i's movement, comprehending both the global density distribution and the local overflowed region.

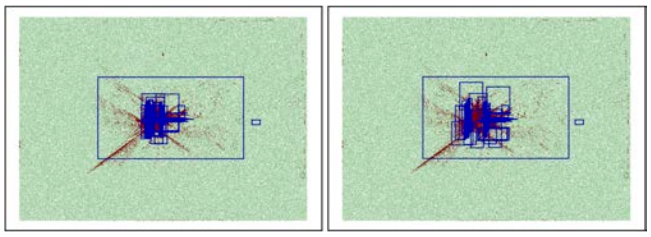
$$Cost_i = \frac{\partial W(\mathbf{v})}{\partial x_i} + \lambda \frac{\partial D^{global}(\mathbf{v})}{\partial x_i} + \Delta_i \frac{\partial D^{local}(\mathbf{v})}{\partial x_i} \tag{9}$$

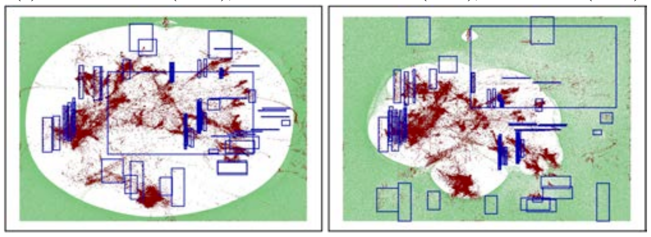
Algorithm 1 describes the procedure to obtain the local-density cost, $LDCost_i$. Line 2 initializes $\Delta_i = 0$. We store the electric potential information at a per-bin granularity, so we first calculate $Gradient_j$ to compute $LDCost_i$ (Line 8 and Lines 11-13). Based on cell/charge/energy distribution of the previous iteration, we calculate $Gradient_j$ by using FFT [29] (Lines 11-13). In Lines 5-7, we add the normalized $overflow_j$ of the bin b_j $(c_i \in b_j)$ to Δ_i when the bin b_j is overflowed. In Line 8, we obtain $LDCost_i$ after multiplying by Δ_i.

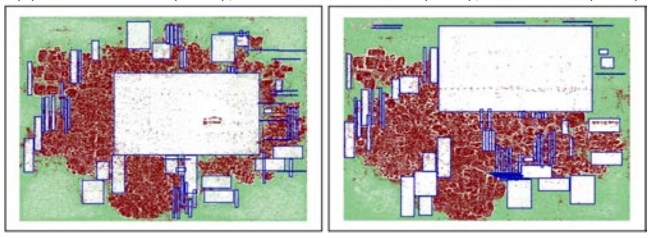
E. Additional Details and Illustration

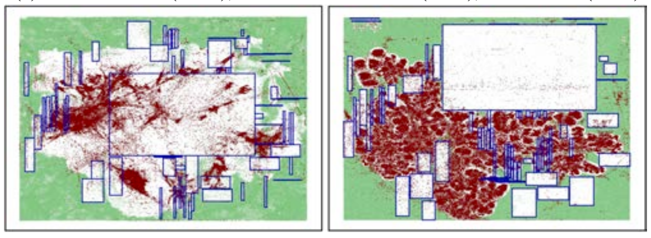
Figures 3(a)-(e) contrast the mixed-size global placement for *NEWBLUE1* [43], by ePlace-MS [29] (left-side images) and the constraint-oriented local-density function equipped RePlAce-ld (right-side images).[10] Red and green dots represent standard cells and filler cells, respectively. Blue rectangles are movable macros. RePlAce-ld encourages faster cell spreading in overflowed regions by virtue of the repulsive force induced from the local density function. Figures 3(a)-(c) show the mixed-size global placement stage (mGP) with movable standard cells and macros. The figures show how ePlace-MS [29] fails to move the largest macro due to lack of force, while RePlAce-ld moves the largest macro toward the boundary of the layout. After mGP, we execute a simulated annealing-based (SA-based) macro legalization stage (mLG) to fix all macro locations. The SA-based algorithm randomly picks a macro and randomly determines its movement vector within the search range, considering both wirelength and macro overlapping removal. The search range and the temperature (probability to accept the move) are adjusted to balance quality and efficiency [29]. Then, a standard-cell only global placement stage (cGP, inheriting instance locations from mLG with macros fixed) is called to recover solution quality lost during macro legalization. Figure 3(d) shows that moving the largest macro to the boundary of the layout helps to improve the solution quality by providing more space in the center of the layout region for the standard-cell placement. Moreover, Figure 3(b) (left) shows that ePlace-MS applies only one large center-oriented force to filler cells. By contrast, RePlAce-ld applies multiple overflowed-region-induced force components to filler cells, as a result of the local-density function. Figure 3(e) shows the final legalized layouts. HPWL values for ePlace-MS and RePlAce-ld are 6.39×10^7 and 5.60×10^7, respectively.

[10]We follow the same mixed-size placement procedure and naming convention as detailed in [29].


(a) Iteration: 250 (mGP), HPWL= 2.07×10^7 (LHS), 2.14×10^7 (RHS).


(b) Iteration: 400 (mGP), HPWL= 4.51×10^7 (LHS), 4.32×10^7 (RHS).


(c) Iteration: 600 (mGP), HPWL= 6.55×10^7 (LHS), 5.32×10^7 (RHS).


(d) Iteration: 30 (cGP), HPWL= 5.18×10^7 (LHS), 5.96×10^7 (RHS).

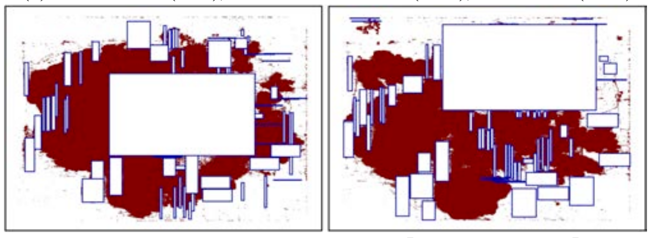

(e) Final legalized layout, HPWL= 6.39×10^7 (LHS), 5.60×10^7 (RHS).

Fig. 3: Placement of *NEWBLUE1* [43]: left hand side (LHS) images are from ePlace-MS [29], and right hand side (RHS) images are from RePlAce-ld. The target density is set to 100%.

IV. Improved Dynamic Step Size Adaptation

In this section, we describe our *improved dynamic step size adaptation*. How to decide the step size is a well-studied topic in the global placement literature [5] [6]. In general, constant large step size leads to faster convergence, but can sacrifice solution quality (Figure 4(a)). Constant small step size generates an accurate solution according to the given formulation, but with large runtime (Figure 4(b)). Neither strategy accounts for the shape characteristics (e.g., instantaneous slope at different iterations) of the *HPWL curve*[11] shown in the figure. Figure 4(c) conceptually illustrates our improved dynamic step size adaptation that comprehends the characteristics of the HPWL curve. To effectively allocate optimization effort, we dynamically adjust both the step size and the step size scaling range during global placement, while previous approaches [5] [6] [27] [28] [29] adjust step size within a constant step size range.

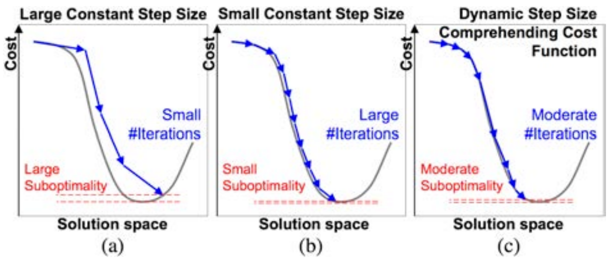


Fig. 4: An illustration conceptually showing the benefit of dynamic step size adaptation. Cost is composite of wirelength and density. (a) Constant large step size obtains a result quickly but with large suboptimality. (b) Constant small step size may generate a near-optimal result but with many iterations. (c) Our improved dynamic step size adaptation starts and ends with small step sizes, and dynamically scales the step size to efficiently allocate the optimization effort.

A. Improved Dynamic Step Size Adaptation

The density penalty factor λ (in Equation (1)) is used to resolve instance overlaps. In previous ePlace implementations [27] [28] [29], the λ scaling tries to maintain a constant HPWL increment[12] across iterations. Algorithm 2 summarizes the ePlace λ scaling methodology, whereby step size (i.e., cof in Algorithm 2) is dynamically scaled to maintain a constant HPWL increment of $\Delta HPWL_{ref}$ (Line 4 in Algorithm 2). In our implementation, we use the same $\Delta HPWL_{ref}$ as in ePlace-MS [29]. Based on HPWL increment per iteration, cof varies within a predefined *step size scale* [cof_{min}, cof_{max}], cof_{max} and cof_{min} respectively indicate the maximum and the minimum step sizes. A larger (resp. smaller) HPWL increment corresponds to a smaller (resp. larger) cof. The previous work of ePlace-MS [29] fixes $cof_{min} = 0.95$ and $cof_{max} = 1.05$.

In RePlAce, to efficiently allocate the optimization effort we propose an *improved dynamic step size adaptation* strategy that dynamically adjusts the step size cof and the maximum step size cof_{max} (i.e., the range of step size scaling) with respect to the HPWL increment per each iteration. Figure 5 shows the HPWL curve from the testcase *ADAPTEC1* [43] across iterations in the placement procedure. The slope of the HPWL curve can change at each iteration, and changes rapidly near the star symbols. In our experience, all observed HPWL curves from a wide range of testcases have very strong commonality, with trajectory shapes as shown in Figure 5 and two classes of extreme points. We define two classes of *transition points* based on the HPWL curve's instantaneous slope-change rate: (i) *2nd-order transition point* (TP^2), and (ii) *1st-order transition point* (TP^1).

On the HPWL curve from the placement procedure,[13] the TP^2 points are defined to be the two points with largest absolute instantaneous rate of slope change (red and blue stars in Figure 5). Two TP^2 points divide the HPWL curve into three *phases* (blue, green, and yellow regions in Figure 5), and each phase has one TP^1 point. The TP^1 point within a given phase is determined as follows. (i) Within each of the 1st and 3rd phases, the TP^1 point has the same instantaneous slope as the line segment (purple solid line segment in Figure 5) drawn between the two extreme (i.e., leftmost and rightmost) points of the HPWL curve within the phase. (ii) Within the 2nd

Algorithm 2 λ Scaling

```
1: Procedure λ_Scaling()
2: $\lambda \leftarrow (\sum_i gradient\_wirelength)/(\sum_i gradient\_potential)$;
3: for $k = 0$ to $last\_iteration$ do
4:     $p \leftarrow (HPWL_k - HPWL_{k-1}) / \Delta HPWL_{ref}$;
5:     if $p < 0$ then
6:         $cof \leftarrow cof_{max}$;
7:     else
8:         $cof \leftarrow \max(cof_{min}, pow(cof_{max}, 1-p))$;
9:     end if
10:    $\lambda \leftarrow \lambda \times cof$;
11: end for
```

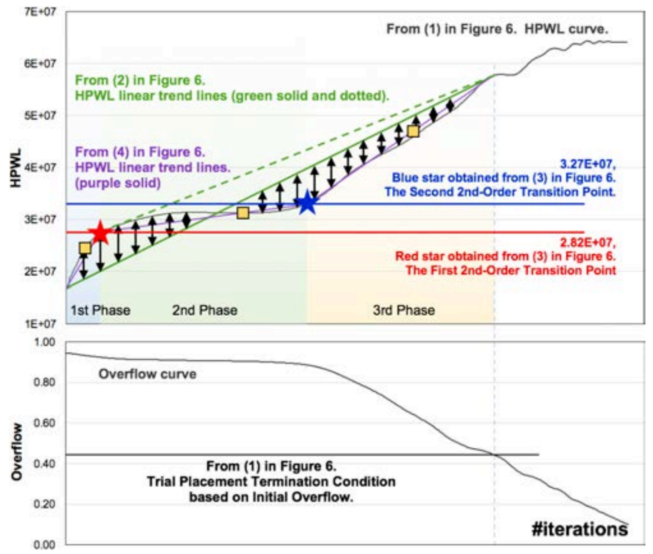


Fig. 5: HPWL curve of *ADAPTEC1* from the trial placement procedure and the estimated transition points (TP^2=red/blue stars, TP^1=yellow squares).

[11]We define the *HPWL curve* as the plot of HPWL values versus iterations in the global placement procedure.

[12]A "constant cell displacement" results in failure to converge for at least four of 16 (ISPD-2005 and ISPD-2006) testcases; we therefore believe that this is an enabling difference in RePlAce.

[13]We perform a *trial placement* ahead of the actual placement procedure to capture transition points. We observe that the HPWL on transition points from the trial placement are close to those from the actual placement procedure (i.e., $< 5\%$). The trial placement procedure is described in Section IV-B.

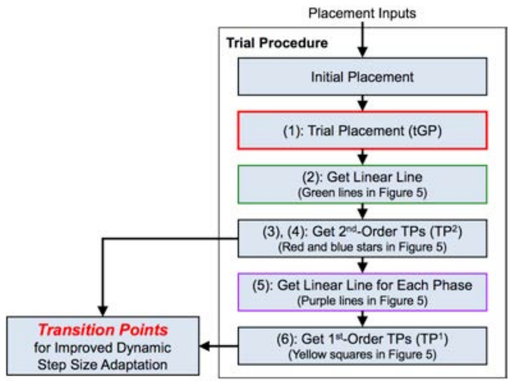

Placement Inputs
Trial Procedure
Initial Placement
(1): Trial Placement (tGP)
(2): Get Linear Line
(Green lines in Figure 5)
(3), (4): Get 2nd-Order TPs (TP2)
(Red and blue stars in Figure 5)
(5): Get Linear Line for Each Phase
(Purple lines in Figure 5)
(6): Get 1st-Order TPs (TP1)
(Yellow squares in Figure 5)
Transition Points
for Improved Dynamic
Step Size Adaptation

Fig. 6: Flowchart of our trial placement procedure. The red rectangle indicates nonlinear optimization using Nesterov's method. The actual placement procedure follows this trial placement procedure.

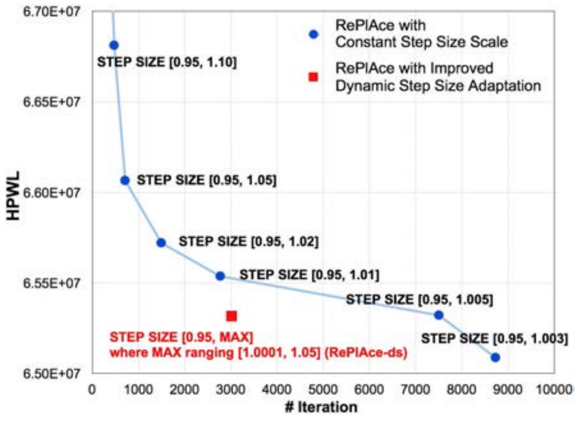


Fig. 7: Solution qualities achieved by RePlAce with constant step size scale and by RePlAce-ds' improved dynamic step size adaptation strategy, on the *ADAPTEC1* [43] testcase. RePlAce-ds achieves a dominating runtime and solution quality (red square). We note that RePlAce with constant step size scale (blue line) does not converge with smaller step sizes (i.e., $[0.95, 1.002]$ or below).

phase, the TP^1 point corresponds to the intersection between the HPWL curve and the purple solid line segment. The TP^1 points are shown as yellow squares in Figure 5.

We observe that transition points are important to obtaining improved solution quality because (i) instances tend to alter their moving directions and to settle their locations near the TP^2s, which determines the overall solution quality; and (ii) instances move actively toward their final locations near the TP^1s, which provides an opportunity to save runtime. We allocate optimization effort based on these observations, applying the smallest step size at the TP^2s and the largest step size at the TP^1s. We achieve this dynamical adaptation of the step size scale by controlling the maximum step size cof_{max}. Our empirically determined step size scale ranges from [0.95, 1.0001] to [0.95, 1.05].

$$cof_{max} = min_{cof_{max}} + cof_{range} \times \frac{|HPWL_{current} - HPWL_{TP^2}|}{|HPWL_{TP^1} - HPWL_{TP^2}|} \tag{10}$$

With the given $min_{cof_{max}}$, cof_{range}, and the current iteration's HPWL, we compute maximum step size using Equation (10),

Algorithm 3 Finding Transition Points

```
1: Procedure Trial()
2: Trial placement();
3: Connect initial and final points on HPWL curve using a linear line;
4: Calculate HPWL differences between HPWL curve and linear line for each tGP
   iteration;
5: Get the 2nd-order transition points ($TP^2$);
6: Divide the HPWL curve into three phases by $TP^2$;
7: Connect initial and final points on HPWL curve separately for each phase;
8: Calculate HPWL differences between HPWL curve and linear line for each phase
   for each tGP iteration;
9: Get the 1st-order transition points ($TP^1$);
10: return $TP^2$ and $TP^1$;
```

which achieves dynamic control of the step size scale. In our implementation, $min_{cof_{max}}$ is 1.0001 in the first phase, 1.001 in the second phase, and 1.005 in the third phase. We empirically determine cof_{range} as 0.0009, 0.024 and 0.045 for the three phases, respectively. We allocate more optimization effort at the beginning of the nonlinear optimization based on our observations that the solution quality achieved in early iterations is critical to the final placement solution. Figure 7 shows the tradeoff between solution quality (HPWL) and runtime (#iterations) with various *constant* step sizes. Our improved dynamic step size adaptation achieves a superior result (red square) in terms of both HPWL and runtime.

B. Trial Global Placement

To find the 1st-order (TP^1) and the 2nd-order (TP^2) transition points on the HPWL curve, we perform a *trial placement procedure* that includes a trial global placement (tGP). We terminate tGP when the density overflow is $\leq \frac{\tau_{init}}{2.5}$.[14] Figure 6 describes our trial procedure. Inspired by [36], the transition points are determined as follows. (1) In Line 2 of Algorithm 3, we first perform tGP and obtain the HPWL curve, as shown in Figure 5. (2) In Line 3, we connect the initial (start of tGP) and final (end of tGP) HPWL points on the HPWL curve by a *primary* line segment (green solid line segment in Figure 5). In Line 4, for each tGP iteration (along the x-axis in Figure 5), we calculate the absolute HPWL differences between the HPWL curve and the primary line segment (black arrows in Figure 5). (3) In Line 5, we pick the point with the largest absolute difference as one of the TP^2 points (red star in Figure 5). (4) To find the other TP^2 point, we connect the existing TP^2 point to the initial and final HPWL points, respectively, using two *secondary* line segments (green dotted line segments in Figure 5). The point (blue star in Figure 5) with the largest absolute HPWL difference between the HPWL curve and the two secondary line segments is determined as the other TP^2 point. (5) In Line 6, we divide the HPWL curve into three phases based on the two TP^2 points. In Line 7, we repeat (2) separately for the HPWL curve of each phase. (6) In Lines 8-9, we repeat (3) to find a TP^1 point within each of the 1st and 3rd phases. The TP^1 point in the 2nd phase is the point of intersection where the HPWL curve transitions from above to below the purple line segment.

[14]Overall density overflow τ is defined as the sum of the density overflow for all placement bins over the total cell area. $0 \leq \tau \leq 1$. The initial density overflow τ_{init} is the density overflow obtained from a wirelength-only optimization before our nonlinear optimization. We empirically determine the constant as 2.5, which provides better results in terms of tradeoff between HPWL and the number of iterations.

V. Routability-Driven Placement

To produce routable placement has recently being recognized as being of critical importance. This is reflected by many routability-driven placement contests [3] [38] [39] [40] [45] and placers [7] [8] [10] [13] [21] [23]. In this section, we describe how global routability-driven placement is achieved in RePlAce, including (i) capacity and blockage calculation; (ii) demand calculation; (iii) cell inflation technique; and (iv) overall flow. Notations are described in Table II.

TABLE II: Notations for routability-driven placement. We use the default length unit in the DAC-2012 and ICCAD-2012 benchmark suites.

Term	Description
blk	blocked routing capacity per tile edge
cap	routing capacity (by length unit) per tile edge
$demand$	routing demand per tile edge
$infl\_ratio$	cell inflation ratio
ml	metal layer index
$pincnt$	pin count per tile
$pitch_{ml}$	metal pitch (by length unit) on layer ml
rt	global routing tile index
$usage$	# tracks used per tile edge

A. Capacity and Blockage Calculation

According to DAC-2012 [39] and ICCAD-2012 [40] contest and benchmark suite descriptions, global routing is performed using global routing tiles, with back-end-of-line capacity (i.e., tile width or height) and blockage (in the unit of tile dimension) defined for each layer in the official benchmark inputs. We follow the definitions from [39] [40], and associate a capacity value (cap) to each edge of the global routing tile for each metal layer. As an example, for a unidirectional horizontal routing layer, the left and right tile edges have a capacity value directly from the benchmark specification, while the upper and lower tile edges have a capacity of zero. We also associate a routing blockage (blk) to each edge of a global routing tile *for each metal layer*. The blockage is defined as the total blocked capacity (total blocked tile width or height), as illustrated in Figure 8.

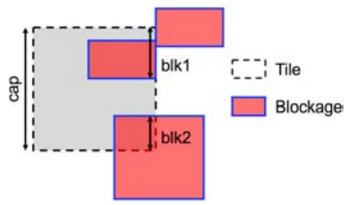


Fig. 8: Illustration of blockage calculation. For the vertical edge on the right, $blk = blk1 + blk2$. Note the union of blocked capacity for the upper two blockages.

Algorithm 4 Inflation Ratio Adjustment

```
1: Procedure $AdjustInflationRatio$
2: $total\_inflated\_area \leftarrow GetTotalInflatedArea()$;
3: while $total\_inflated\_area \geq max\_inflated\_area$ do
4:     $infl\_ratio_0 \leftarrow GetInflationRatioForLeastCongestedTile()$;
5:     for all $tiles\ (rt)$ do
6:         $infl\_ratio_{rt} \leftarrow \frac{infl\_ratio_{rt}}{infl\_ratio_0}$;
7:     end for
8:     $total\_inflated\_area \leftarrow UpdateTotalInflatedArea()$;
9: end while
```

B. Demand Calculation

During global placement, we invoke the official router *NCTU-GR* [48] from the DAC-2012 and ICCAD-2012 contests to obtain the routing demand. As described in [39], the global router reports cross-tile routing segments, so that tile edge-based routing usage (#tracks used) can be obtained.

For each layer, we multiply the usage ($usage$) with metal pitch to obtain the routing demand. Additionally, for metal 2 and below, we further consider the pin blockage effect. Here, we calculate the total number of pins ($pincnt$) within a tile and use γ_{pin} as a pin blockage factor. We describe the demand calculation in Equation (11):

$$demand_{e,ml} = (usage_{e,ml} + \gamma_{pin} \cdot pincnt) \cdot pitch_{ml}, \qquad (11)$$

where the subscript e indicates one of the four edges of a given global routing tile, and the subscript ml indicates a specific metal layer. Thus, we calculate the demand for all four tile edges and for all metal layers. We use the same pin blockage factor as specified in each of the benchmark suites, i.e., $\gamma_{pin} = 0$ for DAC-2012 benchmark and $\gamma_{pin} = 0.05$ for ICCAD-2012 benchmark.

C. Cell Inflation

To resolve congestion, we inflate cells within tiles for which the demand is larger than the corresponding capacity. Since we calculate the capacity and demand per tile edge on each layer, there are multiple (capacity, demand) pairs. The inflation ratio for each cell is calculated as the maximum demand over capacity ratio (i.e., Equation (12)). Cell width and height are increased according to the square root of the inflation ratio for each direction. Based on empirical studies, we enable superlinear cell inflation with $\gamma_{super} = 2.33$ and we bound the maximum inflation ratio to be 2.5. In this way, we achieve metal layer-aware inflation, rather than relying only on sums of capacities and demands over all metal layers [10] [23].

$$infl\_ratio = \max_{all\ \ e,ml}\left(\left(\frac{demand_{e,ml} + blk_{e,ml}}{cap_{e,ml}}\right)^{\gamma_{super}}, 2.5\right) \qquad (12)$$

To avoid the total inflated area exceeding available whitespace, we adopt the dynamic inflation ratio adjustment methodology from [9]. Algorithm 4 describes the inflation ratio adjustment. We first calculate the total inflated area according to the initial inflation ratio. If the total inflated area exceeds a predefined maximum value, $max\_inflated\_area$, we divide the horizontal inflation ratio for each tile by the inflation ratio of the least-congested tile that has a ratio greater than one, and recalculate the total inflated area. We repeat the above procedure until the total inflated area becomes smaller than the predefined maximum value. We describe $max\_inflated\_area$ in Section V-D.

D. Overall Flow

Figure 9 shows the overall flow of our global routability-driven placement. When the wirelength-driven global placement reaches 20% density overflow, we invoke the global router *NCTU-GR* [48] to obtain our internal routing congestion (RC) evaluation. We then perform routability optimization

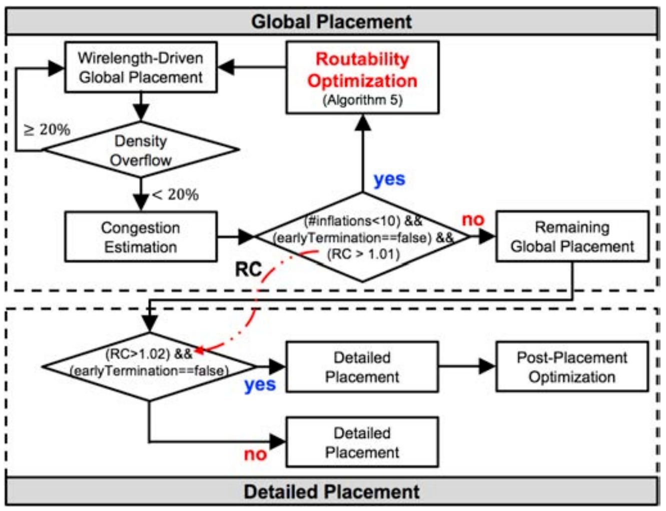

Global Placement
Wirelength-Driven Global Placement
Routability Optimization (Algorithm 5)
≥ 20%
Density Overflow
< 20%
yes
Congestion Estimation
(#inflations<10) && (earlyTermination==false) && (RC > 1.01)
no
Remaining Global Placement
RC
(RC>1.02) && (earlyTermination==false)
yes
Detailed Placement
Post-Placement Optimization
no
Detailed Placement
Detailed Placement

Fig. 9: Overall flowchart of our routability-driven placement.

using our cell inflation technique (see discussion below), then feed the new cell sizes, die utilization, and density penalty factor back to the global placement engine. We choose to perform routability optimization only if we meet all three of the following conditions: (i) the latest congestion estimation indicates a less than 1% routing overflow ($RC < 1.01$); (ii) fewer than 10 rounds of cell inflation have been performed; and (iii) the binary indicator *earlyTermination* is false. Otherwise, we execute the remaining global placement procedure and pass the placement solution on to the detailed placer. The binary *earlyTermination* flag is used to skip the following rounds of cell inflation once the design is considered difficult to improve further (see discussion below). Since our work solely focuses on global placement, we use *NTUplace3* [6] as the detailed placer. We then perform a post-placement optimization [24] if we believe there can be further benefits. In our implementation, we go through post-placement optimization only when both of the following conditions are met: (i) there is still more than 2% routing overflow ($RC > 1.02$); and (ii) there is still room for improvement (*earlyTermination* is false).

Results reported below (in Section VI-C) show that improvements over previous works can be attributable to RePlAce global placement as opposed to the use of *NTUplace3* (or, even, *NTUplace4h* [13]).

The routability optimization process is described in Algorithm 5. In Line 2, we first calculate the inflation ratio from congestion estimation, as described in Sections V-A and V-B. Then, in Line 3, we adjust the inflation ratio according to Algorithm 4. In Line 4, we perform cell inflation to improve routability. In Line 5, we roll back the density penalty factor λ by 100 Nesterov's optimization iterations [29] to encourage replacement of cells based on the new cell sizes. Line 6 reflects

Algorithm 5 Routability Optimization

```
1: Procedure RouteOpt
2: $CalculateInflationRatio()$;
3: $AdjustInflationRatio()$;
4: $InflateCell()$;
5: $\lambda_{curr\_iter} \leftarrow \lambda_{curr\_iter-100}$;
6: $AdjustUtilization()$;
7: $UpdateEarlyTerminationIndicator()$;
```

that the overall die utilization should be adjusted because the equivalent total cell area becomes larger due to inflation. We adjust the die utilization based on the ratio of equivalent total cell area over die area, as given in Equation (13). In each routability optimization, we limit $max\_inflated\_area$ to be 10% of the total whitespace area, so that the total utilization is constrained to remain less than 100%. In Line 7, we update the binary indicator *earlyTermination*. The indicator remains false until the minimum RC value thus far has not improved by 0.008 over the last four consecutive rounds of cell inflation.

$$util = \frac{curr\_cell\_area + total\_inflated\_area}{die\_area} \tag{13}$$

Figure 10 shows snapshots of congestion maps during our routability optimization procedure for *SUPERBLUE12*. The figure shows how the global placement process effectively reduces the congestion (i.e., hotspots) indicated by red regions.

VI. Experiments

In this section, we describe our experimental setups and results. We implement RePlAce in C++ and perform experiments in single-thread mode using a 2.6GHz Intel Xeon server. Our implementation has no benchmark-specific code or tuning: a single binary produces all results, with command-line options as we describe below. Experiments are performed on three types of well-studied academic benchmarks: (i) ISPD-2005 [32] and ISPD-2006 [33] benchmark suites for standard cell placement; (ii) the large-scale modern mixed-size (MMS) [43] benchmark suite for mixed-size placement; and (iii) DAC-2012 [39] and ICCAD-2012 [40] benchmark suites for global routability-driven placement. RePlAce functionalities and corresponding suffixes (command-line options) are summarized in Table III. We briefly give some insight into the remaining gaps between academic and real-world placers and testcases, by comparing final-routed wirelength for real standard-cell placements obtained by RePlAce-r and a leading-edge commercial place-and-route tool in a foundry 28LP technology. In all of our experimental results tables, bold numbers indicate the best HPWL (sHPWL) for each testcase.

TABLE III: RePlAce functionalities and the corresponding suffixes (command-line options) that produce the results reported below.

Benchmark Type	-ld	-ds	-ldds	-r
ISPD-2005 and ISPD-2006 benchmarks		●		
MMS benchmarks	●	●	●	
DAC-2012 and ICCAD-2012 benchmarks				●

A. Standard Cell Placement

For standard cell placement, we validate RePlAce using the ISPD-2005 [32] and ISPD-2006 [33] benchmark suites, whose parameters are summarized in Table IV. We employ *NTUplace3* [6] as our detailed placer. Experimental results are summarized in Table V. For testcases with a specified target density, we report the scaled HPWL using the official evaluation scripts [33].

Table V compares RePlAce-ds to the best known results [28] [46] across ISPD-2005 and ISPD-2006 benchmark suites. We observe that RePlAce-ds achieves (new) best known results

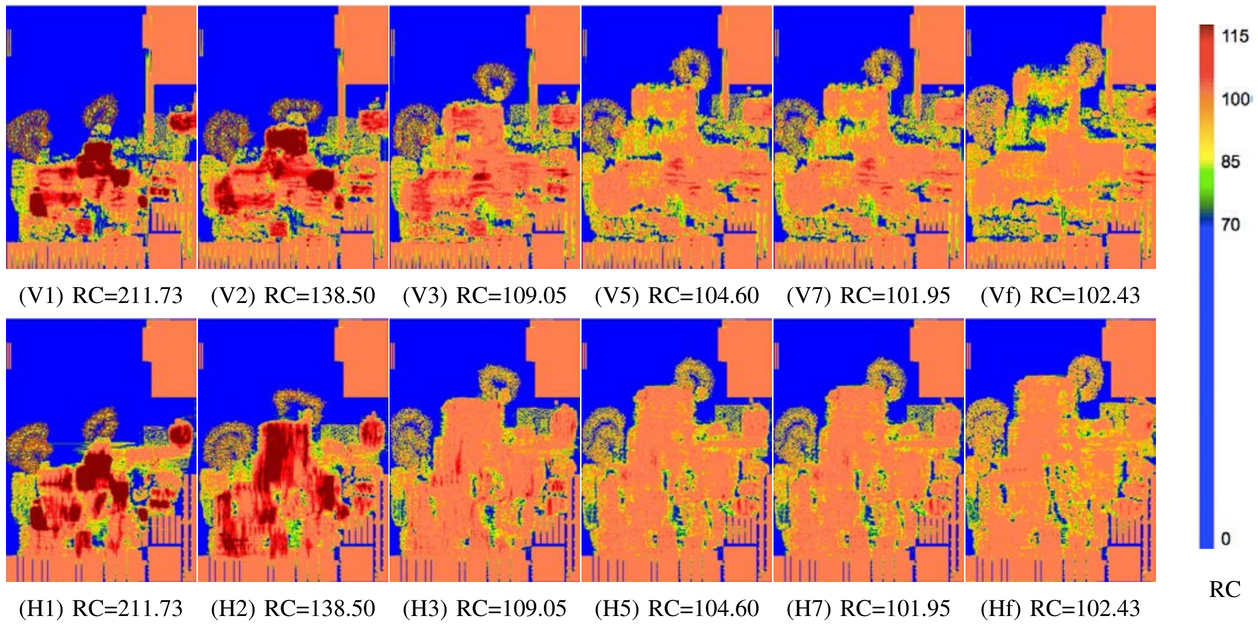


Fig. 10: Global routing overflow (*SUPERBLUE12*) during routability-driven global placement procedure. RC = Routing Congestion reported by the DAC-2012 official evaluation script. (Hk) and (Vk) show horizontal and vertical congestion before the k^{th} cell inflation; (Hf) and (Vf) show final horizontal and vertical routing congestion after detailed placement.

for 15 out of 16 testcases. Overall, RePlAce-ds achieves up to 4.00% HPWL reduction (*ADAPTEC2*) and 2.00% HPWL reduction on average, when compared to the previous best known results.

B. Mixed-Size Placement

We demonstrate the benefit from our constraint-oriented local density function using the large-scale modern mixed-size (MMS) placement benchmarks [43]. Parameters of the benchmarks are summarized in Table IV. The MMS benchmarks embody the same designs as the ISPD-2005 and ISPD-2006 benchmarks, except that some macros are movable. We use *NTUplace3* [6] as our detailed placer. Experimental results are summarized in Table V. For testcases with a specified target density, we report scaled HPWL using the official evaluation scripts [43].

Table V compares RePlAce mixed-size placement results with best known previous results [29]. We apply three different schemes: (1) local density function equipped RePlAce-ld,[15] (2) dynamic step size adaptation equipped RePlAce-ds, and (3) combined RePlAce-ldds. Runtimes of (2) and (3) include the trial procedure's runtimes. From the bottom row of the table, we see that RePlAce-ld reduces HPWL by approximately 2.25% on average compared to best known results.[16] Compared to best known results, RePlAce-ld produces shorter HPWL for 15 out of 16 testcases. In addition, RePlAce-ldds shows further improvement of solution quality with the addition of dynamic step size adaptation. Albeit with increased runtime, RePlAce-ldds appears to effectively invest its effort (i.e., iterations, runtime), and achieve the best solution quality on average, by incorporating both the -ld and -ds techniques. Together, RePlAce-ds and RePlAce-ldds produce the best solution quality for 15 out of 16 testcases. As our results show merits to both constraint-oriented local smoothing and dynamic step size adaptation, as well as their explicit combination, we leave to future practitioners the question of how to best apply and orchestrate these techniques.

Last, Figure 11 shows a breakdown of #iterations across the component global placement procedures for (a) RePlAce-ds with the ISPD-2005 and ISPD-2006 benchmark suites, and (b) RePlAce-ds and (c) RePlAce-ldds with the MMS benchmark suite, aggregated over all reported testcases. (Note that for any given testcase, runtime will be roughly proportional to #iterations.) In the figure, tGP indicates trial placement; mGP indicates macro and standard-cell placement; and cGP indicates standard-cell only placement. With dynamic step size adaptation, approximately 17-22% of iterations (i.e., of runtimes) are additionally consumed by the tGP procedure.

[15]We see little benefit by applying local density to testcases without large movable macros. Thus, "-ld" option is only applied to MMS benchmark suites.

[16]However, with local density function, we do not find a uniform trend with accelerated convergence rate for global placement. The local density calculation takes on average $\sim 1.8\%$ of the total runtime.

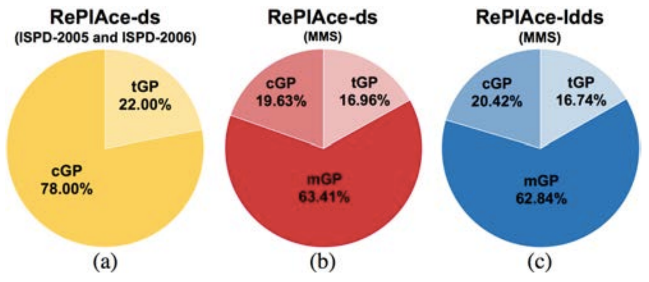


Fig. 11: Runtime breakdown (#iterations) aggregated over all reported testcases in the ISPD-2005 and ISPD-2006 benchmark suites for (a) RePlAce-ds, and in the MMS benchmark suite for (b) RePlAce-ds and (c) RePlAce-ldds. Here, tGP, mGP, and cGP respectively denote trial placement, macro and standard cell placement, and standard cell-only placement.

TABLE IV: Statistics for ISPD-2005 [32], ISPD-2006 [33], and MMS [43] benchmark suites.

Circuits	# Objects	# Standard Cells	# Nets	Target Density (%)	ISPD Macros		MMS Macros	
					#Movable	#Fixed	#Movable	#Fixed
ADAPTEC1	211447	210904	221142	100	0	543	63	480
ADAPTEC2	2255023	254457	266009	100	0	566	127	439
ADAPTEC3	2451650	450927	466758	100	0	723	58	665
ADAPTEC4	2496045	494716	515951	100	0	1329	69	1260
BIGBLUE1	2278164	277604	284479	100	0	560	32	528
BIGBLUE2	2557866	534782	577235	100	0	23084	959	22125
BIGBLUE3	21096812	1093034	1123170	100	2485	1293	2549	1229
BIGBLUE4	22177353	2169183	2229886	100	0	8170	199	7970
ADAPTEC5	2843128	842482	867798	50	0	646	76	570
NEWBLUE1	2330474	330137	338901	80	64	337	64	337
NEWBLUE2	2441516	436516	465219	90	3723	1277	3748	1252
NEWBLUE3	2494011	482833	552199	80	0	11178	51	11127
NEWBLUE4	2646139	642717	637051	50	0	3422	81	3341
NEWBLUE5	21233058	1228177	1284251	50	0	4881	91	4790
NEWBLUE6	21255039	1248150	1288443	80	0	6889	74	6815
NEWBLUE7	22507954	2481372	2636820	80	0	26582	161	26421

TABLE V: (Scaled†) HPWL ($\times 10^6$) and runtime (minutes) comparison of best known, RePlAce-ld, RePlAce-ds and RePlAce-ldds on ISPD and MMS benchmarks. Best known ISPD results are cited from Tables II and V of Nonsmooth placer [46] and Tables II and V of ePlace [28]. Best known MMS results are cited from Table II of ePlace-MS [29]. sHPWL is HPWL scaled by placement density overflow. Runtime with "*" indicates cited results, using Intel Core processors at 2.4GHz [46], or 2.67GHz [28]. Our runtime is reported using Intel Xeon processors at 2.6GHz. All cited and reported results use *NTUplace3* [6] as the detailed placer.

Circuits	ISPD-2005, ISPD-2006				MMS							
	Best known [28] [46]		RePlAce-ds		Best known [29]		RePlAce-ld		RePlAce-ds		RePlAce-ldds	
	HPWL	CPU*	HPWL	CPU	HPWL	CPU*	HPWL	CPU	HPWL	CPU	HPWL	CPU
ADAPTEC1	74.20	13.13	**73.01**	14.18	66.82	5.47	65.17	5.30	65.32	15.34	**64.98**	19.08
ADAPTEC2	84.84	4.90	**81.45**	25.75	76.74	7.58	72.75	7.68	71.68	20.43	**71.51**	25.08
ADAPTEC3	194.07	26.13	**190.45**	50.88	161.55	27.23	154.18	23.22	**151.34**	62.81	151.42	69.15
ADAPTEC4	174.11	27.85	**172.22**	87.55	145.89	56.40	142.05	32.91	**139.70**	96.71	140.57	117.35
BIGBLUE1	90.98	6.25	**89.05**	23.78	86.29	7.82	85.79	8.15	86.03	23.85	**85.04**	28.23
BIGBLUE2	141.83	10.50	**136.67**	48.19	130.06	13.70	**125.33**	16.29	125.84	47.95	125.49	53.46
BIGBLUE3	306.94	27.29	**298.61**	112.98	284.39	72.98	**270.17**	73.83	282.42	165.23	280.31	183.76
BIGBLUE4	742.45	145.00	**740.57**	337.23	656.68	204.15	653.24	162.34	650.09	317.67	**647.55**	363.32
ADAPTEC5†	**391.02**	83.65	391.24	74.92	**294.24**	46.07	303.36	35.13	301.78	81.83	302.53	92.53
NEWBLUE1†	59.26	14.00	**57.44**	27.56	60.43	11.70	58.63	9.90	57.75	27.56	**57.44**	31.67
NEWBLUE2†	182.42	20.01	**177.82**	32.56	159.11	51.12	152.32	15.44	152.34	51.31	**150.09**	58.29
NEWBLUE3†	264.48	33.15	**255.07**	68.62	269.47	36.30	258.53	20.97	**257.22**	57.19	257.67	65.76
NEWBLUE4†	269.58	56.26	**267.71**	58.33	226.29	28.27	**223.52**	26.08	224.02	59.92	223.62	68.24
NEWBLUE5†	492.62	54.83	**486.37**	118.19	392.77	55.47	390.14	75.81	388.74	151.55	**386.30**	167.19
NEWBLUE6†	464.36	116.70	**460.24**	118.45	409.28	69.65	408.89	84.04	407.04	168.23	**406.60**	184.09
NEWBLUE7†	978.07	246.00	**950.27**	335.19	889.18	392.02	**876.36**	172.58	877.83	277.77	880.67	326.00
Avg.	+0.00%	1.00×	-2.00%	1.78×	+0.00%	1.00×	-2.25%	0.72×	-2.43%	1.81×	-2.73%	2.09×

C. Routability-Driven Placement

We validate RePlAce global routability-driven placement using the DAC-2012 [39] and ICCAD-2012 [40] benchmark suites. We compare our placement solutions to those of all published results from leading-edge academic placers [7] [10] [13] [23]. Parameters of the DAC-2012 and ICCAD-2012 benchmark suites are summarized in Table VI. The DAC-2012 and ICCAD-2012 benchmark suites do not include movable macros, but contain *.shapes* and *.route* files that respectively describe the component shapes for non-rectangular nodes and routing-related information (e.g., the number of routing tracks per each metal layer, routing blockages, etc.).

Experimental results are summarized in Table VII and Table VIII. For all testcases, we set the target density as 90%.[17] We report scaled HPWL (sHPWL) and routing congestion (RC) as evaluated by the official global router *NCTU-GR* [25] [48] and contest evaluation scripts [39] [40]. In the DAC-2012 and ICCAD-2012 benchmark suites, routing congestion is described using the "ACE(X)" metric, which is the averaged congestion of the top X% tile edges. Then, peak weighted congestion (PWC) and Routing Congestion (RC) are obtained by Equations (14) and (15), where $k_i = 1$. Equation (16) defines the final evaluation metric, i.e., scaled HPWL. Each unit of RC in excess of 100 results in a 3% sHPWL penalty.

$$PWC = \frac{\Sigma_{i=0.5,1,2,5} k_i \cdot ACE(i)}{\Sigma k_i} \tag{14}$$

$$RC = \max(100, PWC) \tag{15}$$

$$sHPWL = HPWL \times (1 + 0.03 \times (RC - 100)) \tag{16}$$

Compared to all published results, RePlAce achieves smaller sHPWL for all testcases (10 and eight testcases from DAC-2012 and ICCAD-2012, respectively). RePlAce achieves on average 9.80% and 9.60% sHPWL reduction over best DAC-2012 and ICCAD-2012 contest results, respectively; and on average 8.50% to 9.59% scaled HPWL reduction over the leading previous academic placers.[18] In our flow as shown

[17]In our work, we apply the uniform 90% target density, based on experience, to balance wirelength and congestion from a initial global placement perspective, and we keep using the same value without per-benchmark tuning.

[18]To match the reporting convention in Table V, the improvement over previous best known DAC-2012 and ICCAD-2012 results (with those previous best known results set to be 1.00×) would be -7.93% and -6.66%, respectively.

in Figure 9, 16 out of 18 testcases automatically bypass the post-placement optimization stage, and only two testcases (*SUPERBLUE2* and *SUPERBLUE10*) invoke post-placement optimization, with sHPWL improved by 2.74% and 7.47% respectively. These indicate that our routability-driven global placement effectively reduces overall routing congestion, with an average RC value similar to all published leading-edge results.

To show the impact of the detailed placer, we have conducted additional experiments with the DAC-2012 and ICCAD-2012 benchmark suites by employing a *routability-driven* detailed placer, *NTUplace4h* [13], shown as “RePlAce-r alt” in Table VII and Table VIII. Interestingly, all 18 testcases end up with worse sHPWL than “RePlAce-r”, and 11 out of 18 testcases have worse routing congestion. However, “RePlAce-r alt” is still 7.09% to 8.47% better on average compared to the leading previous academic placers. This provides confirmation of the effectiveness of our routability optimization during the global placement.

D. State of Academic vs. Industry Placement

Finally, to help assess remaining gaps between our work and “the real world,” and to demonstrate tractability of such assessment, we apply RePlAce to standard-cell placement using a foundry 28LP 8-track cell library after applying format conversion scripts as in [18]. We place-and-route four design blocks (*JPEG*, *VGA*, *LEON3MP* and *NETCARD* from [34] [49]), with up to 300K instances.[19] In our experiments, RePlAce achieves 2.4% reduction of routed wirelength on average with similar number (<100) of DRVs, and consumes less than $2\times$ runtime compared to a commercial place-and-route tool. Our internal benchmarking studies indicate that in an ideal case, use of an 8-core configuration (i.e., 8-thread, to match the commercial tool) should speed up the cGP (and mGP) stage by up to 89%. With this speedup using 8 cores, the overall runtime (including the single-threaded detailed placement performed by *NTUplace3* [6]) would be reduced by up to 59.2%. We believe that these results show encouraging progress toward bridging remaining gaps between academic and real-world placement.

VII. Conclusion

We have described RePlAce, a single engine for global placement that, without testcase-specific tuning, achieves significant improvements over best known HPWL results on standard-cell and mixed-size benchmark suites, as well as improvements over best known sHPWL on global routability-driven placement benchmark suites. We propose a new density function that comprehends local over-demand for area resources, leading to constraint-oriented local smoothing at a per-bin granularity. Our dynamic step size adaptation determines step size and allocates optimization effort to significantly improve solution quality without undue runtime impact. We achieve an average HPWL reduction of 2.00% over best known ISPD-2005 and ISPD-2006 benchmark results, and of 2.73% over best known MMS benchmark results. For routability-driven placement, we achieve better sHPWL on all testcases from the DAC-2012 and ICCAD-2012 benchmark suites, with on average 8.50% to 9.59% scaled HPWL reduction compared to the leading previous academic placers. To our knowledge, RePlAce is the first work to achieve overall superior solution quality across the ISPD-2005, ISPD-2006, MMS, DAC-2012 and ICCAD-2012 benchmarks with a single global placement engine. We leave to our future work several enhancements: (i) timing-driven global placement; (ii) parallel and/or distributed implementation for runtime improvement; (iii) a more systematic approach for dynamic step size adaptation with reduced runtime; (iv) integration of constraint-driven local smoothing with routability-driven placement; and (v) techniques to accelerate convergence.

[19]We push placement utilization up to the limit of a 2016 release of a leading commercial tool, i.e., until post-detailed routing design rule violations (DRVs) appear without exceeding 100 DRVs in the commercial tool. The number of post-route DRVs for the commercial tool (C), and RePlAce (R) are (C, R) = (5, 3), (40, 12), (79, 68), (34, 81) for *JPEG*, *VGA*, *LEON3MP* and *NETCARD*, respectively. Due to license restrictions, we are unable to more specifically identify the commercial tool.

Acknowledgments

RePlAce [50] source code is available at http://vlsicad.ucsd.edu/RePlAce/. We thank the authors of [30] for providing access to, and useful discussions of, their implementation. We also thank the authors of NTUplace3 [6] and NTUplace4h [13] for providing the executables that we use as our detailed placer.

References

[1] C. J. Alpert, Z. Li, M. D. Moffitt, G.-J. Nam, J. A. Roy and G. Tellez, ”What Makes a Design Difficult to Route”, *Proc. ISPD*, 2010, pp. 7-12.

[2] C. J. Alpert, Z. Li, G.-J. Nam, C. N. Sze, N. Viswanathan and S. I. Ward, “Placement: Hot or Not?”, *Proc. ICCAD*, 2012, pp. 283-290.

[3] I. S. Bustany, D. Chinnery, J. R. Shinnerl and V. Yutsis, “ISPD 2015 Benchmarks with Fence Regions and Routing Blockages for Detailed-Routing-Driven Placement”, *Proc. ISPD*, 2015, pp. 157-164.

[4] U. Brenner and A. Rohe, “An Effective Congestion-Driven Placement Framework”, *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 4, pp. 387-394, Apr. 2003.

[5] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen and Y.-W. Chang, “A High-Quality Mixed-Size Analytical Placer Considering Preplaced Blocks and Density Constraints”, *Proc. ICCAD*, 2006, pp. 187-192.

[6] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen and Y.-W. Chang, “NTUplace3: An Analytical Placer for Large-Scale Mixed-Size Designs with Preplaced Blocks and Density Constraints”, *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 7, pp. 1228-1240, Jul. 2008.

[7] J. Cong, G. Luo, K. Tsota and B. Xiao, “Optimizing Routability in Large-scale Mixed-size Placement”, *Proc. ASP-DAC*, 2013, pp. 441-446.

[8] N. K. Darav, A. Kennings, A. F. Tabrizi, D. Westwick and L. Behjat, “Eh?Placer: A High-Performance Modern Technology-Driven Placer”, *ACM Trans. on DAES* 21(3) (2016), article 37.

[9] X. He, T. Huang, L. Xiao, H. Tian and E. F. Y. Young, “Ripple: A Robust and Effective Routability-Driven Placer”, *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 10, pp. 1546-1556, Oct. 2013.

[10] X. He, Y. Wang, Y. Guo and E. F. Y. Young, “Ripple 2.0: Improved Movement of Cells in Routability-Driven Placement”, *ACM Trans. on DAES* 22(1) (2016), article 10.

[11] M.-K. Hsu and Y.-W. Chang, “Unified Analytical Global Placement for Large-Scale Mixed-Size Circuit Designs”, *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 9, pp. 1366-1378, Sep. 2012.

[12] M. K. Hsu, Y.-W. Chang and V. Balabanov, “TSV-aware Analytical Placement for 3D IC Designs”, *Proc. DAC*, 2011, pp. 664-669.

TABLE VI: Statistics for DAC-2012 [39] and ICCAD-2012 [40] benchmark suites.

Circuits	# Objects	# Movable Cells	# Terminal Nodes	# Nets	# Pins	Util (%)	Density (%)
SUPERBLUE1	849441	765102	52627	822744	2861188	69	35
SUPERBLUE2	1014029	921273	59312	990899	3228345	76	28
SUPERBLUE3	919911	833370	55033	898001	3110509	73	42
SUPERBLUE4	600220	521466	40550	567607	1884008	70	44
SUPERBLUE5	772457	677416	74365	786999	2500306	77	37
SUPERBLUE6	1014209	919093	65316	1006629	3401199	73	43
SUPERBLUE7	1364958	1271887	66995	1340418	4935083	76	58
SUPERBLUE9	846678	789064	37574	833808	2898853	73	47
SUPERBLUE10	1202665	1045874	96251	1158784	3894138	70	32
SUPERBLUE11	954686	859771	67303	935731	3071940	79	40
SUPERBLUE12	1293433	1278084	8953	1293436	4774069	56	44
SUPERBLUE14	634555	567840	44743	619815	2049691	72	50
SUPERBLUE16	698741	680450	419	697458	2280931	69	46
SUPERBLUE18	483452	442405	25063	468918	1864306	67	47
SUPERBLUE19	522775	506097	286	511685	1714351	78	49

TABLE VII: Scaled HPWL (sHPWL) ($\times 10^7$), routing congestion (RC) and CPU runtime (minutes) comparison of RePlAce to leading published results for DAC-2012 [39] global routability-driven placement. sHPWL is HPWL scaled with routing congestion penalty. (Each unit of RC in excess of 100 results in a 3% sHPWL penalty.) Published results are cited from best contest results at DAC-2012 [39], and newest versions of mPL [7]. Missing benchmark results are indicated by N/A. "RePlAce-r alt" uses *NTUplace4h* as its detailed placer. Global routing is performed by the official global router *NCTU-GR* [48], and sHPWL is evaluated by the official DAC-2012 contest evaluation scripts. To match the reporting convention in Table V, the improvement of RePlAce-r over previous best known DAC-2012 results (with those previous best known results set to be 1.00×) would be -7.93%. Runtime with "*" indicates cited results, using Intel Core [7] or Xeon [39] processors at 2.27GHz. Our runtime is reported using Intel Xeon processors at 2.6GHz. Only *SUPERBLUE2* invokes post-placement optimization, with sHPWL improvement of 2.74% (sHPWL from 62.58 to 60.87).

Circuits	best in contest [39]			mPL12 [7]			RePlAce-r			RePlAce-r alt		
	sHPWL	RC	CPU*	sHPWL	RC	CPU*	sHPWL	RC	CPU	sHPWL	RC	CPU
SUPERBLUE2	62.40	100.68	291	61.40	N/A	312	**60.87**	100.96	155	60.96	101.00	160
SUPERBLUE3	36.20	100.56	236	36.00	N/A	215	**30.68**	100.78	62	32.08	102.10	64
SUPERBLUE6	34.25	100.32	186	34.00	N/A	285	**31.20**	100.51	41	31.40	100.52	43
SUPERBLUE7	39.85	100.71	433	39.50	N/A	287	**37.28**	101.22	47	37.36	101.07	51
SUPERBLUE9	25.46	102.48	219	25.00	N/A	212	**21.28**	100.78	42	21.39	100.81	44
SUPERBLUE11	34.22	100.02	254	34.00	N/A	245	**33.69**	102.07	52	34.20	102.35	54
SUPERBLUE12	31.19	100.02	581	30.40	N/A	320	**26.52**	102.43	75	27.49	103.02	80
SUPERBLUE14	22.56	100.07	156	24.50	N/A	126	**21.21**	100.65	16	21.32	100.68	18
SUPERBLUE16	27.39	101.38	46	27.40	N/A	157	**25.27**	101.87	42	25.51	101.94	44
SUPERBLUE19	15.31	100.61	140	15.10	N/A	165	**14.27**	100.71	29	14.65	101.55	30
Avg.	+9.80%	0.995×	5.31×	+9.59%	N/A	5.00×	+0.00%	1.000×	1.00×	+1.54%	1.003×	1.05×

[13] M.-K. Hsu, Y.-F. Chen, C.-C. Huang, S. Chou, T.-H. Lin, T.-C. Chen and Y.-W. Chang, "NTUplace4h: A Novel Routability-Driven Placement Algorithm for Hierarchical Mixed-Size Circuit Designs", *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 12, pp. 1914-1927, Dec. 2014.

[14] C.-C. Huang, C.-H. Chiou, K.-H. Tseng and Y.-W. Chang, "Detailed-Routing-Driven Analytical Standard-Cell Placement", *Proc. ASP-DAC*, 2015, pp. 378-383.

[15] J. Hu, A. B. Kahng, S. Kang, M. Kim and I. Markov, "Sensitivity-Guided Metaheuristics for Accurate Discrete Gate Sizing", *Proc. ICCAD*, 2012, pp. 233-239.

[16] J. Hu, J. A. Roy and I. L. Markov, "Completing High-Quality Global Routes", *Proc. ISPD*, 2010, pp. 35-41.

[17] Z.-W. Jiang, B.-Y. Su and Y.-W. Chang, "Routability-Driven Analytical Placement by Net Overlapping Removal for Large-Scale Mixed-Size Designs", *Proc. DAC*, 2008, pp. 167-172.

[18] A. B. Kahng, H. Lee and J. Li, "Horizontal Benchmark Extension for Improved Assessment of Physical CAD Research", *Proc. GLSVLSI*, 2014, pp. 27-32.

[19] A. B. Kahng, J. Lienig, I. L. Markov and J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*, Springer, 2011.

[20] A. B. Kahng and Q. Wang, "Implementation and Extensibility of an Analytic Placer", *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 5, pp. 734-747, May 2005.

[21] M.-C. Kim, J. Hu, D. J. Lee and I. L. Markov, "A SimPLR method for Routability-Driven Placement", *Proc. ICCAD*, 2011, pp. 67-73.

[22] M.-C. Kim and I. L. Markov, "ComPLx: An Competitive Primal-dual Lagrange Optimization for Global Placement", *Proc. DAC*, 2012, pp. 747-752.

[23] T. Lin and C. Chu, "POLAR 2.0: An Effective Routability-Driven Placer", *Proc. DAC*, 2014, pp. 1-6.

[24] W.-H. Liu, C.-K. Koh and Y.-L. Li, "Optimization of Placement Solutions for Routability", *Proc. DAC*, 2013, pp. 1-9.

[25] W.-H. Liu, W.-C. Kao, Y.-L. Li and K.-Y. Chao, "Multi-Threaded Collision-Aware Global Routing with Bounded-Length Maze Routing", *Proc. DAC*, 2010, pp. 200-205.

[26] W.-H. Liu, W.-C. Kao, Y.-L. Li and K.-Y. Chao, "NCTU-GR 2.0: Multithreaded Collision-Aware Global Routing with Bounded-Length Maze Routing", *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 5, pp. 709-722, May 2013.

[27] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-H. Huang, C.-C. Teng and C.-K. Cheng, "ePlace: Electrostatics Based Placement Using Nesterov's Method", *Proc. DAC*, 2014, pp. 1-6.

[28] J. Lu, P. Chen, C.-C. Chang, L. Sha, D. J.-H. Huang, C.-C. Teng and C.-K. Cheng, "ePlace: Electrostatics-Based Placement Using Fast Fourier Transform and Nesterov's Method", *ACM Trans. on DAES* 20(2) (2015), article 17.

[29] J. Lu, H. Zhang, P. Chen H. Chang, C.-C. Chang, Y.-C. Wong, L. Sha, D. Huang, Y. Luo, C.-C. Teng and C.-K. Cheng, "ePlace-MS: Electrostatics-Based Placement for Mixed-Size Circuits", *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 5, pp. 685-698, May 2015.

[30] J. Lu, H. Zhuang, I. Kang and C.-K. Cheng, "ePlace-3D: Electrostatics based Placement for 3D-ICs", *Proc. ISPD*, 2016, pp. 11-18.

[31] I. L. Markov, J. Hu and M.-C. Kim, "Progress and Challenges in VLSI Placement Research", *Proc. IEEE* vol. 103, no. 11, pp. 1985-2003, Nov. 2015.

[32] G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter and M. Yildiz, "The ISPD 2005 Placement Contest and Benchmark Suite", *Proc. ISPD*, 2005, pp. 216-220.

TABLE VIII: Scaled HPWL (sHPWL) ($\times 10^7$), routing congestion (RC) and CPU runtime (minutes) comparison of RePlAce to leading published results for ICCAD-2012 [40] global routability-driven placement. sHPWL is HPWL scaled with routing congestion penalty. (Each unit of RC in excess of 100 results in a 3% sHPWL penalty.) Published results are cited from best contest results at ICCAD-2012 [40], and newest versions of POLAR 2.0 [23], NTUplace4h [13] and Ripple 2.0 [10] (listed in chronological order). “RePlAce-r alt” uses *NTUplace4h* as its detailed placer. Global routing is performed by the official global router *NCTU-GR* [48], and sHPWL is evaluated by the official ICCAD-2012 contest evaluation scripts. To match the reporting convention in Table V, the improvement of RePlAce-r over previous best known ICCAD-2012 results (with those previous best known results set to be 1.00×) would be -6.66%. Runtime with “*” indicates cited results, using Intel Xeon processors at 2.0GHz [13], 2.67GHz [23] [40], or 3.4GHz [10]. Our runtime is reported using Intel Xeon processors at 2.6GHz. Only *SUPERBLUE10* invokes post-placement optimization, with sHPWL improvement of 7.47% (sHPWL from 63.12 to 58.41).

Circuits	best in contest [40]			Polar 2.0 [23]			NTUplace4h [13]		
	sHPWL	RC	CPU*	sHPWL	RC	CPU*	sHPWL	RC	CPU*
SUPERBLUE1	27.89	100.97	39	28.20	101.15	27	28.13	101.15	84
SUPERBLUE3	34.39	100.56	45	33.30	101.06	29	34.59	101.06	92
SUPERBLUE4	22.69	101.32	143	22.40	100.96	21	23.05	100.96	65
SUPERBLUE5	34.86	100.38	180	35.10	100.70	18	35.56	100.70	86
SUPERBLUE7	41.37	100.71	250	40.70	100.82	31	39.82	100.82	166
SUPERBLUE10	61.11	101.91	439	62.10	101.11	49	61.67	101.11	153
SUPERBLUE16	28.33	101.55	100	27.20	101.30	17	27.94	101.30	63
SUPERBLUE18	17.09	103.15	77	16.90	101.47	21	16.36	101.47	55
Avg.	+9.60%	1.003×	2.63×	+8.50%	1.000×	0.50×	+9.06%	1.007×	1.81×

Circuits	Ripple 2.0 [10]			RePlAce-r			RePlAce-r alt		
	sHPWL	RC	CPU*	sHPWL	RC	CPU	sHPWL	RC	CPU
SUPERBLUE1	28.48	100.74	51	**25.89**	100.43	43	27.84	102.67	46
SUPERBLUE3	34.07	100.22	64	**30.78**	100.85	52	30.91	100.84	45
SUPERBLUE4	22.51	100.30	32	**20.94**	100.52	35	20.99	100.53	36
SUPERBLUE5	35.38	100.41	70	**33.37**	100.93	64	33.40	100.82	66
SUPERBLUE7	40.76	100.79	100	**37.10**	100.76	44	37.42	100.84	48
SUPERBLUE10	60.44	100.57	90	**58.41**	101.32	189	58.79	101.53	191
SUPERBLUE16	27.95	100.71	46	**25.46**	101.35	45	25.64	101.34	48
SUPERBLUE18	17.07	100.78	35	**14.60**	102.10	36	14.70	102.16	38
Avg.	+9.29%	0.995×	1.15×	+0.00%	1.000×	1.00×	+1.41%	1.003×	1.04×

[33] G.-J. Nam, “ISPD 2006 Placement Contest: Benchmark Suite and Results”, *Proc. ISPD*, 2006, pp. 167.

[34] M. M. Ozdal, C. Amin, A. Ayupov, S. Burns, G. Wilke and C. Zhuo, “The ISPD-2012 Discrete Cell Sizing Contest and benchmark suite”, *Proc. ISPD*, 2012, pp. 161-164.

[35] M. Pan and C. Chu, ”FastRoute: a Step to Integrate Global Routing into Placement”, *Proc. ICCAD*, 2006, pp. 464-471.

[36] V. Satopää, J. Albrecht, D. Irwin and B. Raghavan, “Finding a “Kneedle” and a Haystack: Detecting Knee Points in System Behavior”, *Proc. ICDCS*, 2011, pp. 166-171.

[37] P. Spindler and F. M. Johannes, “Fast and Accurate Routing Demand Estimation for Efficient Routability-Driven Placement”, *Proc. DATE*, 2007, pp. 1226-1231.

[38] N. Viswanathan, C. J. Alpert, C. N. Sze, Z. Li, G.-J. Nam and J. A. Roy, ”The ISPD-2011 Routability-Driven Placement Contest and Benchmark Suite”, *Proc. ISPD*, 2011, pp. 141-146.

[39] N. Viswanathan, C. J. Alpert, C. N. Sze, Z. Li and Y. Wei, “The DAC 2012 Routability-driven Placement Contest and Benchmark Suite”, *Proc. DAC*, 2012, pp. 774-782,

[40] N. Viswanathan, C. J. Alpert, C. N. Sze, Z. Li and Y. Wei, “ICCAD-2012 CAD Contest in Design Hierarchy Aware Routability-Driven Placement and Benchmark Suite”, *Proc. ICCAD*, 2012, pp. 345-348,

[41] M. Wang, X. Yang, K. Eguro and M. Sarrafzadeh, “Multicenter Congestion Estimation and Minimization During Placement”, *Proc. ISPD*, 2000, pp. 147-152.

[42] J. Westra, C. Bartels and P. Groeneveld, “Probabilistic Congestion Prediction”, *Proc. ISPD*, 2004, pp. 204-209.

[43] J. Z. Yan, N. Viswanathan and C. Chu, “Handling Complexities in Modern Large-Scale Mixed-Size Placement”, *Proc. DAC*, 2009, pp. 436-441.

[44] X. Yang, R. Kastner and M. Sarrafzadeh, “Congestion Estimation During Top-Down Placement”, *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 1, pp. 72-80, Jan. 2002.

[45] V. Yutsis, I. S. Bustany, D. Chinnery, J. R. Shinnerl and W.-H. Liu, “ISPD 2014 benchmarks with sub-45nm technology rules for detailed-routing-driven placement”, *Proc. ISPD*, 2014, pp. 161-168.

[46] W. Zhu, J. Chen, Z. Peng and G. Fan, “Nonsmooth Optimization Method for VLSI Global Placement”, *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 4, pp. 642-655, Apr. 2015.

[47] General Purpose FFT Package, http://www.kurims.kyoto-u.ac.jp/~ooura/fft.html.

[48] NCTU-GR, http://people.cs.nctu.edu.tw/~whliu/NCTU-GR.htm.

[49] OpenCores, http://opencores.org/projects/.

[50] RePlAce, http://vlsicad.ucsd.edu/RePlAce/.

Chung-Kuan Cheng, Photograph and biography not available at the time of publication.

Andrew B. Kahng, Photograph and biography not available at the time of publication.

Ilgweon Kang received the B.S. and M.S. degrees in electrical and electronic engineering from the Yonsei University, Seoul, Korea, in 2006 and 2008, respectively. He is currently pursuing the Ph.D. degree at the University of California at San Diego, La Jolla. He was with the Research and Development Division, SK Hynix, Icheon, Korea, from 2008 to 2012. His current research interests include VLSI layout design automation and optimization, e.g., IC floorplanning, placement, DFM methodologies, etc.

Lutong Wang received the B.S. degree in microelectronics from Tsinghua University, Beijing, China, in 2014 and the M.S. degree in electrical and computer engineering from the University of California at San Diego, La Jolla, in 2016. He is currently pursuing the Ph.D. degree at the University of California at San Diego, La Jolla. His research interests include physical design implementation and DFM methodologies.