MAPLE: Multilevel Adaptive PLacEment for Mixed-Size Designs

Myung-Chul Kim[†‡], Natarajan Viswanathan[‡], Charles J. Alpert[‡], Igor L. Markov[†], Shyam Ramji[§]
†University of Michigan, EECS Department, Ann Arbor, MI 48109
‡IBM Corporation, Austin, TX 78758 / §IBM Corporation, Hopewell Junction, NY 12533
mckima@umich.edu, nviswan@us.ibm.com, alpert@us.ibm.com, imarkov@eecs.umich.edu, ramji@us.ibm.com

ABSTRACT

We propose a new multilevel framework for large-scale placement called MAPLE that respects utilization constraints, handles movable macros and guides the transition between global and detailed placement. In this framework, optimization is adaptive to current placement conditions through a new density metric. As a baseline, we leverage a recently developed flat quadratic optimization that is comparable to prior multilevel frameworks in quality and runtime. A novel component called Progressive Local Refinement (ProLR) helps mitigate disruptions in wirelength that we observed in leading placers. Our placer MAPLE outperforms published empirical results — RQL, SimPL, mPL6, NTUPlace3, FastPlace3, Kraftwerk and APlace3 — across the ISPD 2005 and ISPD 2006 benchmarks, in terms of official metrics of the respective contests.

Categories and Subject Descriptors

B.7.2 [**Hardware, Integrated Circuits**]: Design Aids—*Placement and routing*

General Terms

Algorithms, Design, Performance

1. INTRODUCTION

Large-scale placement remains one of the most influential optimizations in interconnect-driven physical design and physical synthesis [3]. Despite the long history of research, three ISPD contests on placement have shown that recent algorithms achieve sizable gains over prior state of art [22]. The ISPD 2011 routability-driven placement contest [30] has demonstrated that the choice of the wirelength-driven global placement engine is paramount even in multi-objective placement — two of the top three teams relied on the high-quality SimPL framework [18], including the contest winners, who reimplemented SimPL without having access to the original source code [12]. Yet, no placer dominated across the entire benchmark set, indicating possible improvements. Such improvements are described in this paper, although our work is orthogonal to and compatible with the innovations developed for the ISPD 2011 contest [12, 13, 17].

In this work, we develop MAPLE — a multilevel force-directed placement algorithm that pioneers key algorithmic components and a more effective way of combining individual components into a reliable multi-objective optimization. MAPLE generates the coarsest-level placement by a variant of the SimPL algorithm [18] but also employs multilevel extensions reinforced by our new Progressive Local Refinement (ProLR).[1] This combination enhances trade-offs between wirelength and module density. Compared to recent literature, our implementation produces superior solution quality with reasonable runtimes.
The improvement on ISPD 2006 benchmarks is particularly encouraging because it demonstrates that MAPLE not only reduces the wirelength but also avoids highly concentrated placements, thus promoting routability and providing greater flexibility for timing optimization transforms. Note that the original SimPL algorithm was not evaluated with utilization constraints of the ISPD 2006 benchmark suite and could not handle movable macros present in those benchmarks. At a more conceptual level, our work explores limits to optimization imposed by noise inherent in analytic placement algorithms. After studying sources of this noise, we develop techniques to avoid noise or suppress it, which consistently improve end results beyond the best reported in the literature.

Our key contributions include:

- A study of obstacles to extending analytic placement with multilevel techniques. We observe that straightforward extensions cause disruptions between successive optimizations during global placement.
- A key insight to combine unclustering with two-tier Progressive Local Refinement (ProLR) so as to ensure graceful transitions between optimizations at different cluster levels. Optimization adapts to current wirelength/density trade-offs, which we track by a newly developed metric — ABU_γ.
- A placement algorithm (MAPLE) that relies on SimPL iterations, but augments them with two-level clustering and ProLR. MAPLE guides the transition from global to detailed placement to avoid unnecessary disruptions. This guidance allows MAPLE to derive the final placement from the lower- rather than the upper-bound placement as in the original SimPL, enhancing solution quality.
- Extensions of the MAPLE algorithm to handle movable macros. This includes extending the SimPL algorithm and dealing with macros during refinement.
- Empirical evaluation against best published results on ISPD 2005 and ISPD 2006 benchmarks using official metrics. MAPLE consistently outperforms all leading-edge placers described in the literature.

[1]The implementation used in this work was written from scratch.

The remainder of this paper is structured as follows. Section 2 presents background and prior art. Section 3 analyzes disruptions during multilevel placement optimization that undermine solution quality. In Sections 4 and 5, we present the MAPLE algorithm and specific techniques to ensure graceful transitions between successive optimizations. Section 6 describes extensions of the MAPLE algorithm to handle movable macros. Section 7 empirically validates our ideas and algorithms. Section 8 concludes our paper.

2. BACKGROUND AND PRIOR ART

Given a netlist $\mathcal{N} = (E, V)$ with nets E and nodes (cells) V, *global placement* seeks node locations (x_i, y_i) such that the area of nodes within any rectangular region does not exceed the area of (cell sites in) that region. Some locations of cells may be given initially and fixed. The interconnect objective optimized by global placement is the Half-Perimeter WireLength (HPWL). For node locations $\vec{x} = \{x_i\}$ and $\vec{y} = \{y_i\}$, HPWL$_{\mathcal{N}}(\vec{x}, \vec{y})$= HPWL$_{\mathcal{N}}(\vec{x})$+HPWL$_{\mathcal{N}}(\vec{y})$, where

$$HPWL_{\mathcal{N}}(\vec{x}) = \Sigma_{e \in E}[\max_{i \in e} x_i - \min_{i \in e} x_i] \qquad (1)$$

A consistent 2% HPWL improvement is considered significant and can affect routability, timing and power. For optimization, HPWL can be approximated by differentiable functions [7, 10, 16].

Quadratic optimization represents the netlist by a weighted graph $\mathcal{G} = (E_{\mathcal{G}}, V)$, using the star, clique or Bound2Bound net model [26]. Here we denote vertices by V and edges by $E_{\mathcal{G}}$. Edge weights $w_{ij} > 0$ for all edges $e_{ij} \in E_{\mathcal{G}}$. The *quadratic objective* $\Phi_{\mathcal{G}}$ is defined as

$$\Phi_{\mathcal{G}}(\vec{x}, \vec{y}) = \Sigma_{i,j} w_{i,j}[(x_i - x_j)^2 + (y_i - y_j)^2] \qquad (2)$$

$$\Phi_{\mathcal{G}}(\vec{x}, \vec{y}) = \frac{1}{2}\vec{x}^T Q_x \vec{x} + \vec{c}_x^T \vec{x} + \frac{1}{2}\vec{y}^T Q_y \vec{y} + \vec{c}_y^T \vec{y} + \text{const} \qquad (3)$$

The connectivity matrix Q_x captures connections between pairs of movable vertices, while vector $\vec{c}_x$ captures connections between movable and fixed vertices. Since Q_x is positive semi-definite, $\Phi_{\mathcal{G}}(\vec{x})$ is a convex function with a unique minimum, which can be found by solving the system of linear equations $Q_x \vec{x} = -\vec{c}_x$ using preconditioned Conjugate Gradient (CG) as in FastPlace, RQL and SimPL.

FastPlace-Global [28] is a force-directed quadratic placer with two-level Best-choice clustering [2]. It relies on a hybrid (star-clique) net model[2] and employs *cell shifting* to spread the modules during the early stages of placement flow. The *Iterative Local Refinement (ILR)* technique is applied after quadratic optimization to reduce HPWL and spread the modules (see Section 5). **RQL [29]** extends FastPlace-Global by limiting spreading forces (*force-vector modulation*). **FastPlace-DP [24]** is a wirelength-driven detailed placer based on (i) single segment cell clustering, (ii) global cell swapping, (iii) vertical cell swapping, and (iv) local reordering. **SimPL [18]** is a flat, force-directed global placer. It maintains a lower-bound and an upper-bound placement and progressively narrows the displacement between the two. The final solution is derived from the upper-bound placement when the two bounds converge. The upper-bound placement is generated by *lookahead legalization (LAL)*, which is based on top-down geometric partitioning and non-linear scaling. Applying the upper-bound placement as fixed-points, the lower-bound placement is generated by minimizing the quadratic objective using the CG method. Unlike FastPlace-Global and RQL, the SimPL algorithm relies on the Bound2Bound net model [26].

[2]The numerical equivalence of the clique model and the star model with a star node was pointed out in [20] and proven in [19].

3. ANALYSIS OF DISRUPTIONS DURING ANALYTIC OPTIMIZATION

State-of-the-art algorithms for placement integrate multiple optimization steps, which sometimes target different objectives. Poor coordination between successive steps may cause radical changes in intermediate placements. These changes become disruptive when they reverse improvement obtained by previous steps, increasing overall runtime and undermining final solution quality. We now investigate the sources of disruptive changes between successive stages of analytic placement.

Unclustering. In multilevel global placement algorithms, placement iterations after unclustering often include changes to the optimization objective as well as the netlist. This may abruptly increase wirelength as illustrated in [15, Figure 4] for APlace. The authors state that "*Clustering helps to spread cells more quickly, but wirelength is impaired during cell expansion. It is clearly seen from the figures that when wirelength weight is decreased and the conjugate gradient optimizer restarts, discrepancy drops sharply and wirelength is often increased at first and then refined during the optimization*". However, in contrast to our observation in Section 5, the authors claim that when both discrepancy (overflow) and wirelength change slowly, they obtained a near stable suboptimal solution, in which additional iterations did not further reduce discrepancy and wirelength without a major change to the parameters.

Transition to the HPWL objective. FastPlace [28] and RQL [29] use ILR iterations to recover HPWL after quadratic optimization and before detailed placement. ILR iterations include bin resizing over wide ranges to allow large moves across the placement region [22, Chapter 8]. Moreover, each bin maintains a bin-specific utilization weight $0 \leq \theta \leq 1$, which changes depending upon the current bin's utilization. As history accumulates on dense bins over iterations, ILR increasingly penalizes such bins and allows abrupt moves to decrease local density (Figure 1). The density metric ABU_{10} is defined in Section 4.2.

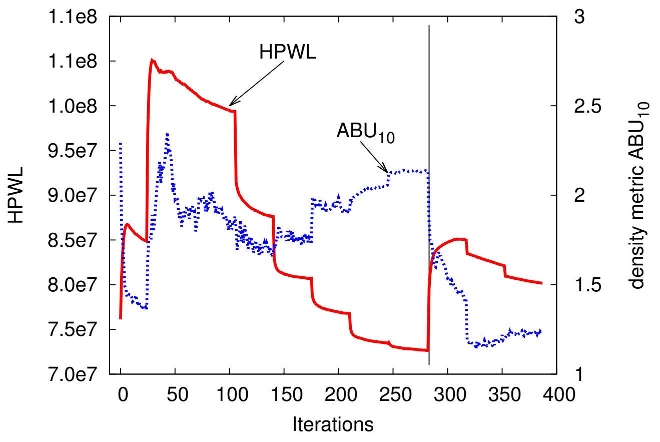

Figure 1: Progressions of wirelength and the density metric ABU_{10} over ILR iterations on ADAPTEC1. Unclustering is marked with a vertical line. ILR disruptively improves ABU_{10} and increases the wirelength. Each ILR iteration traverses all movable modules once.

Hand-off to detailed placement. Recall that the SimPL algorithm maintains two placements throughout its iterations, and legalization is invoked on the *upper-bound* placement, when the lower- and upper-bound placements are reasonably close. The lower-bound placement within SimPL is analogous to module locations maintained by other algorithms. Instead of using the upper-bound, invoking (full) legalization on the lower-bound placement should be potentially better in preserving wirelength optimized by the linear system solver. However, these placements typically exceed target utilization and undergo significant changes during full legalization (Figure 2). Despite local improvement in wirelength during detailed placement, such abrupt changes are detrimental to solution quality in terms of wirelength, routing congestion and timing.

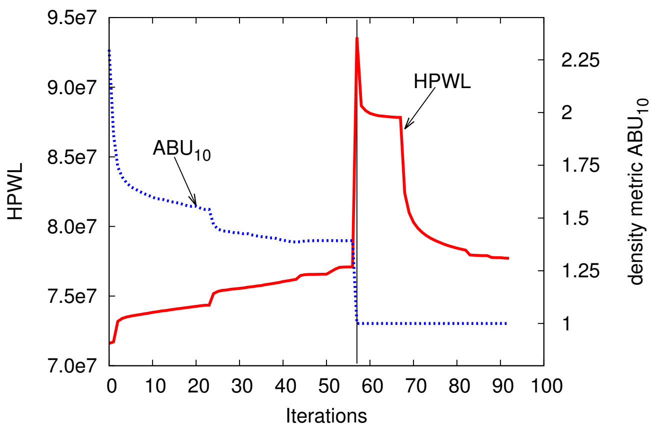

Figure 2: Progressions of wirelength and the density metric ABU_{10} over FastPlace-DP iterations on ADAPTEC1. The start of detailed placement is marked with a vertical line. Placements with high utilization undergo significant changes as full legalization completes.

Strategies for mitigating disruptions. Disruptions during analytic optimization can be mitigated by ensuring gradual transitions between successive optimizations. With this in mind, we develop a new use of placement metrics to make these transitions more adaptive to the actual module distribution and interconnect characteristics. **(1)** the overall placement flow is modified at the points where the objective function abruptly changes, as identified in the above analysis — before/after unclustering, and before detailed placement. We introduce a new intermediate stage that optimizes a linear combination of the preceding and succeeding objective functions, while gradually modifying parameters to ensure smooth transition between the objectives. **(2)** At each substage, we seek near-monotone improvement of either wirelength or module density in a predictable manner without disrupting the other objective. **(3)** Specifically, each intermediate stage prohibits abrupt cell movement and significant changes in key objective functions. Small moves are encouraged instead, as this smoothens changes in wirelength and module density. **(4)** Weighting is adaptively updated according to a new placement metric. These ideas are developed in Progressive Local Refinement (ProLR) in Section 5.

4. MULTILEVEL ADAPTIVE PLACEMENT

We developed our global placement algorithm to address or circumvent the pitfalls in prior art discussed above. This technique consists of three phases: clustering, top-level (coarsest-level) placement iterations, and Progressive Local Refinement (ProLR) used in conjunction with unclustering (Algorithm 1). We apply Best-choice clustering [2] until the number of clusters is reduced to half the size of the flat netlist. Top-level placement iterations perform quadratic optimization on a coarsened netlist and globally regulate module densities over the placement region while moderating wirelengh increase. We adopt a variant of the SimPL algorithm [18] for this phase. The ProLR technique discussed in Section 5 improves both wirelength and module density before/after unclustering. Section 7.3 gives an outlook for using more than 2 levels of clustering.

Algorithm 1 Multilevel Adaptive PlacEment (MAPLE)

```
1:  Phase 0: Clustering of Standard Cells
2:    $N_0$ = number_of_modules in flat netlist
3:    while number_of_clusters $> N_0$ / 2.0 do
4:      cluster netlist using the Best-choice clustering algorithm
5:    end while
6:
7:  Phase 1: Top-level Placement Iterations (SimPL extended)
8:    initial HPWL optimization
9:    while $ABU_{10}$ of lower-bound placement $>$ threshold do
10:     transform the lower-bound placement into an upper-bound
—         placement by Extended Lookahead Legalization (E-LAL)
11:     fix movable macros upon stabilization (Section 6)
12:     update pseudopin locations and pseudonet weights
—         in the linear system [18]
13:     solve the updated linear system using
—         the preconditioned CG method
14:   end while
15:
16: Phase 2: Refinement for Mixed-size Netlists
17:   determine parameters for ProLR
18:   perform ProLR-w and ProLR-d optimizations
19:   legalize and fix all movable macros      // the end of Phase2a
20:   while number_of_modules $< N_0$ do
21:     uncluster the netlist
22:     place unclustered cells side by side
23:   end while
24:   recalculate parameters for ProLR
25:   perform ProLR-w and ProLR-d              // the end of Phase2b
```

4.1 Top-level placement iterations

Top-level placement for the coarsest netlist is performed by the SimPL force-directed placement. It generates lower- and upper-bound placements at each iteration and reduces the displacement gap between the two upon convergence. In contrast to the original SimPL algorithm, MAPLE chooses the last lower-bound placement as a final solution of quadratic placement iterations. This choice is based on our observation that our implementation of SimPL in MAPLE does not completely close the gap between lower and upper bounds. Also, given that *lookahead legalization* [18] is unaware of wirelength objectives, the upper-bound placements are likely to suffer suboptimality. On ISPD 2005 benchmarks, MAPLE typically exhibits a gap of 5.63% to 13.89% between lower and upper bounds at its final iterations. However, even with superior wirelength, lower-bound placements typically exhibit worse module density than upper-bound placements. To address this challenge, we improve lower-bound placements using local-search techniques, as described in Section 5.

4.2 A placement density metric - ABU_γ

We now explore density metrics during global placement, which provide insights into the quality of module spreading in intermediate placements and estimate wirelength impact of legality enforcement. Based on such a metric, the global placer can adaptively adjust its parameters depending on how concentrated the placement is, as described in Section 5.[3] To this end, we propose a new den-

[3]Little is published on density metrics for global placement. Metrics based on *averaged overflow* (including *scaled-overflow per bin* in the ISPD 2006 contest) often fail to capture uneven module distribution. The *maximum utilization* metric leads to pessimistic estimation in the presence of many fixed modules.

sity metric, ABU_γ — average bin utilization of the top γ% densest bins excluding bins fully occupied by fixed macros. Given that the top γ% densest bin are averaged,[4] this metric reflects the non-uniformity of module distribution (Figures 1 and 2). Compared to overflow-based metrics, ABU_γ provides a more intuitive, cross-design perspective into the quality of module spreading.[5] Monitoring density along with wirelength during placement enables comparisons of different parameter settings and even different placers (Figure 3). Such comparisons speed up algorithm development.

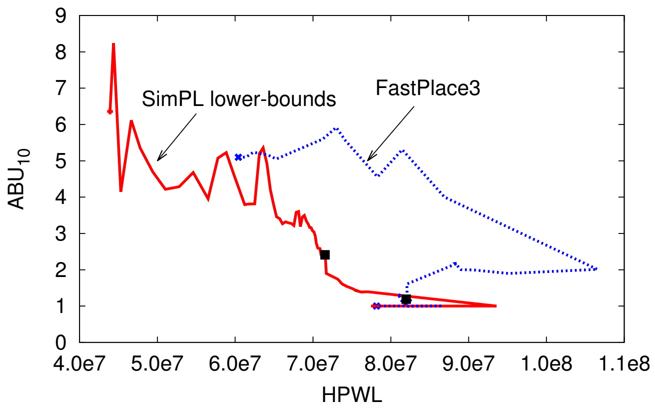

Figure 3: Progression of the density metric ABU_{10} versus wirelength, comparing SimPL lower-bounds (w/ FastPlace-DP) and FastPlace3 on ADAPTEC1. Steeper slope and datapoints closer to the origin indicate better trade-offs. Each square box indicates the beginning of detailed placement.

5. A METHODOLOGY FOR GRACEFUL OPTIMIZATION IN PLACEMENT

After quadratic optimization, placements typically exceed the target utilization in many regions, and their HPWL can be improved without increasing max module density. Furthermore, unclustering traditionally counts on subsequent quadratic placement and can be simple-minded in placing modules within clusters. MAPLE improves this situation by using ProLR — a two-tier technique to reduce wirelength and max module density. ProLR adopts single iterations of ILR [28, 29] — ***L**ocal **R**efinement (LR)* — as a baseline and a vehicle for placement modification. While ILR tends to be disruptive, ProLR promotes gradual transitions via (1) limited bin resizing, (2) *Explicit Bin-Blocking (EBB)*, (3) careful scheduling of utilization weights (θ) between wirelength and module density, and (4) optimizing one objective at a time, while limiting changes to other objectives; such optimizations are alternated.

Bin sizing. ILR and ProLR use regular bin structures and greedily move modules between adjacent bins based on Formula 4. Unlike in ILR, the bins in ProLR are small and remain unchanged during each invocation of LR. Each bin is 5 times the average movable-module area (bins shrink after unclustering). This restricts moves in ProLR.

Explicit Bin-Blocking (EBB) makes local-refinement moves less disruptive. The technique consists of two components: EBB^+ and EBB^-. EBB^+ stops the inflow of modules to some bins (when such moves are expected to be harmful), while EBB^- stops the outflow of modules from some bins and encourages the inflow of modules into these bins. Therefore, EBB^+ is applied to a handful of bins to limit density, while EBB^- is applied to a larger set of bins to attract modules from remaining bins (the density of these bins may decrease).

Joint optimization of density and wirelength. Local refinement moves individual modules based on the linear combination of improvements in HPWL and density.

$$\text{Score}(m) = \alpha \cdot \Delta_{HPWL} + \beta \cdot \theta \cdot \Delta_{density} \tag{4}$$

where θ is the *utilization weight*, and α and β are normalizing coefficients [22, Chapter 8]. In FastPlace and RQL, bin-specific θ_b values are managed after they are reset to values $0.4 \leq \theta \leq 0.6$ when ILR iterations start at each level.

Existing move-based algorithms for optimizing (i) max density and (ii) HPWL use effective techniques for finding highest-gain moves. Yet, no known algorithms are currently known for directly finding the best moves with respect to Formula 4. ProLR inspects best moves for each objective and select those that do not harm the other objective. ProLR performs two simpler optimizations ProLR-w and ProLR-d, which optimize wirelength and module density, respectively. To smoothen placement changes, utilization weight (θ) starts from a small value $\theta_w^0 = 0.1$ for ProLR-w with a coarsened netlist, and θ_{step}^0 is found via a monotonic function

$$\theta_{step}^0 = f(\Upsilon_{target} - \Upsilon_{design}) \tag{5}$$

When the difference between *design utilization* (Υ_{design}) and *target utilization* (Υ_{target}) is small, placement iterations should aggressively reduce density, which is achieved by using a large θ_{step}^0 (greater emphasis on spreading in LR). On the other hand, a wider gap between the two justifies a greater weight for wirelength, and the best wirelength is often achieved by using a small θ_{step}^0 (greater emphasis on wirelength in LR). Details can be found in the Appendix. The utilization weight for ProLR-w with a flat netlist, θ_w^1 is determined as $\theta_w^1 = \theta_d^{M-1}$ where M is the number of ProLR-d invocations performed for the coarsened netlist. The θ_d^k values in the k-th invocation of ProLR-d are determined by

$$\theta_{step}^k = \theta_{step}^{k-1} \cdot \left(1 + \frac{ABU_{10}}{100\Upsilon_{target}}\right) \tag{6}$$

$$\theta_d^k = \theta_w^{k/M} + \theta_{step}^k \quad \forall k \in \{0, M\} \tag{7}$$

$$\theta_d^k = \theta_d^{k-1} + \theta_{step}^k \quad \forall k \notin \{0, M\} \tag{8}$$

ProLR-w improves placement wirelength while maintaining the initial module density distribution. As ProLR-w begins, bin-specific θ_b are reset to θ_w^0 for the clustered netlist and to θ_w^1 for the flat netlist. These values are updated throughout the LR iterations of ProLR-w. Given that ProLR-w maintains θ_b over the entire 300 LR iterations, it closely resembles the use of ILR in FastPlace [28]. However, ProLR-w prohibits abrupt cell movement and significant changes in placement by (1) EBB^+ for bins whose utilization exceeds ABU_{10} and (2) keeping small bin sizes. ProLR-w terminates when ABU_{10} of the current placement exceeds the initial ABU_{10}. Otherwise, ProLR-w continues until there is no improvement in wirelength.

ProLR-d reduces module density of a given placement while keeping wirelength low. The changes in wirelength and density are nearly monotonic. Unlike ProLR-w, ProLR-d consists of up to 15 LR iterations, and bin-specific θ are reset to θ_d^k of each ProLR-d invocation. ProLR-d initially rejects abrupt moves that greatly impact wirelength, and increasing θ_d^k progressively puts a greater emphasis on spreading over multiple invocations. In contrast to ProLR-w,

[4]In our experiments $\gamma = 10\%$ and the equal-sized square bins in the grid have 6 standard-cell heights on the side.

[5]Empirical validation of the ABU_γ metric is not reported due to page limitations.

EBB^- is applied to bins with below-target utilization, attracting modules to sparse bins. We repeat ProLR-d up to 12 times until ABU_{10} stabilizes.

Refinement. When a cluster is broken down, constituent modules are placed side by side. The placement is refined by ProLR.[6] Note in Figures 1 and 2 that during disruptions, wirelength increases sharply and density decreases. Therefore, we schedule ProLR-d before the disruption and ProLR-w after the disruption. Figure 4 shows that this schedule smoothens disruptions in both objectives.

Hand-off to detailed placement. Preprocessing lower-bound placements by ProLR gives better trade-offs between wirelength and density than passing either upper-bound or lower-bound placements to detailed placement algorithms as in original SimPL [18].

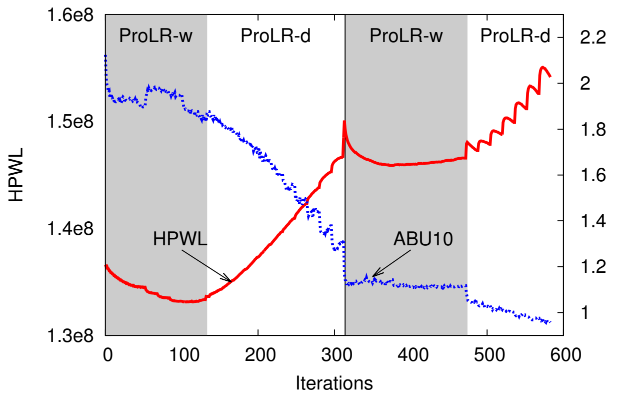

Figure 4: Progressions of wirelength and the density metric ABU_{10} over ProLR iterations (BIGBLUE2). Unclustering is marked with a vertical line. ProLR alternates ProLR-w (shaded) and ProLR-d phases.

6. PLACING MACRO BLOCKS

In placers based on nonconvex optimization, the handling of pre-placed macro blocks requires dedicated techniques (sigmoid functions, level smoothing, etc). In MAPLE, the handling of pre-placed macro blocks is inherited from the SimPL algorithm [18] and LR. To handle movable macros, we extend *lookahead legalization (LAL)* of SimPL, and call the resulting step *E-LAL*. With E-LAL, upper-bound placements are generated in two steps: macro positions are determined first, followed by standard-cell placement [23]. As in original SimPL, roughly legalized placements generated by E-LAL produce fixed pseudopins for subsequent quadratic optimization. Movable macros are legalized by a variant of the *cell shifting* algorithm in FastPlace2 [27]. Our variant uses larger regular bins at 6 times the row height, and employs a 3×3 Laplacian [28] to smoothen bin utilization. A broader view of utilization allows E-LAL to move macros further than FastPlace-Global can and find an almost-legal placement. In the early top-level placement iterations, MAPLE simultaneously places movable macros and standard cells. Upon stabilization (when the gap between the upper- and lower-bounds reduces below 50% from the gap at the 10^{th} iteration), we fix only movable macros with heights $> 2\times$ the row height. Further iterations optimize locations of standard and double-height cells (Figure 5). Recent macro placement literature [8, 11] points out that naive force-directed methods do not reliably find overlap-free placements and that a poor macro placement may cause large overlaps and substantial disruption when removing those overlaps. To address this problem, unlike other force-directed placers, MAPLE fixes macro positions from the upper-bound placement, which tend to have little overlap among macros (Figure 5). Local refinement (LR) moves double-height and standard cells. For double-height cells, bin-specific θ_b and the utilization weights are averaged over all relevant bins. Following the contest protocol, flipping and rotation of macro blocks were disallowed in this work. While macro placement [8, 11, 23] is not a *primary* focus of this work, our techniques produce competitive results on ISPD 2006 benchmarks. Ongoing work indicates that our algorithms for mixed-size placement can be improved further.

[6]Unclustering is followed by *interpolation* in [6, 9] to improve ordering, but ProLR explicitly optimizes HPWL and module density.

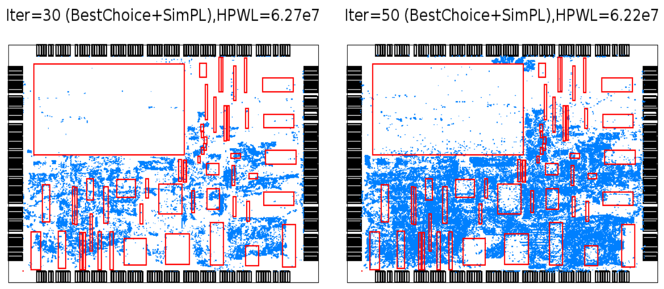
Iter=30 (BestChoice+SimPL),HPWL=6.27e7
Iter=50 (BestChoice+SimPL),HPWL=6.22e7

Figure 5: Macro placement on NEWBLUE1. (left) Macros are fixed at top-level placement iteration 30. (right) Further iterations optimize cell locations.

7. EMPIRICAL VALIDATION

The MAPLE algorithm is implemented in C/C++ within an industry infrastructure for placement optimization, including a variant of FastPlace-DP [24] for final legalization and detailed placement. We compared MAPLE to other state-of-the-art academic and industry placers on the ISPD 2005 and ISPD 2006 placement contest benchmark suites. For placers available to us, benchmark runs were performed on an Intel Core i7 860 Linux workstation running at 2.8GHz with 8GB RAM, using only one CPU core. For other placers (marked with asterisks), results were quoted from respective publications. To ensure the reproducibility of our empirical results, Formula 9 reports specific constants used in our experiments. All benchmarks were placed with identical parameter settings. HPWL of solutions produced by each placer was computed by the GSRC Bookshelf Evaluator [1].

7.1 ProLR versus ILR

Figure 6 illustrates the use of ProLR and ILR in MAPLE through snapshots of placements at different phases of Algorithm 1, starting with identical placements at Phase1. The use of ILR in Phase2a relocates many cells over great distances across fixed macros, as seen in the upper left regions of ILR plots on the left. These moves decrease maximal density, but change the placement abruptly and increase HPWL. After Phase2b, the difference in HPWL between ILR and ProLR decreases, but ILR results remain inferior. One can also see that ILR placements on the left are more clustered than the ProLR placements on the right *and* deviate more from the top-level placements. Table 1 compares MAPLE with ProLR to MAPLE with ILR on ISPD 2005 benchmarks in terms of final HPWL. The results confirm the superiority of ProLR. On the two largest benchmarks — BIGBLUE3 and BIGBLUE4, ProLR was on average, $1.5\times$ slower than ILR.

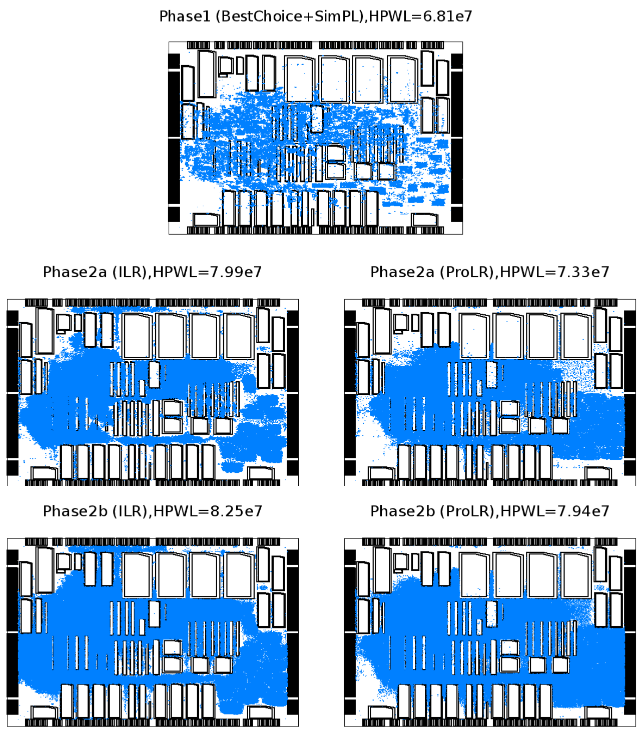
Phase1 (BestChoice+SimPL),HPWL=6.81e7
Phase2a (ILR),HPWL=7.99e7
Phase2a (ProLR),HPWL=7.33e7
Phase2b (ILR),HPWL=8.25e7
Phase2b (ProLR),HPWL=7.94e7

Figure 6: Snapshots of global placement (ADAPTEC1) after each phase of Algorithm 1 for MAPLE with ILR (left) and MAPLE with ProLR (right). Phase1 is top-level placement (BestChoice+SimPL). Phase2a and Phase2b perform LR placement of the coarsened and flat netlist, respectively.

7.2 Comparisons on ISPD 2005 testcases

As shown in Table 3, MAPLE found placements with the lowest HPWL for seven out of eight circuits in the ISPD 2005 benchmarks (no parameter tuning to specific benchmarks was employed). On average, MAPLE improves wirelength by 9.50%, 6.24%, 6.53%, 7.10%, 8.06%, 4.72%, 2.73% and 2.09% versus APlace2 [16], NTUPlace3 (V7.05.30) [10], FastPlace3 [28], Kraftwerk2 [26], mFAR [14], mPL6 [7], SimPL [18] and RQL [29], respectively.

Table 2 compares the runtime of MAPLE with mPL6, APlace2, NTUPlace3, FastPlace3 and SimPL. On average, MAPLE is 1.13×, 2.68× faster than mPL6, APlace2, and 2.32×, 6.25×, and 7.14× slower than NTUPlace3, FastPlace3 and SimPL, resp. On BIGBLUE4, **top-level placement iterations** consume 26.3% of total runtime: 64.1% is in CG, and 18.3% in building sparse matrices for CG. **ProLR iterations** consume 65.4% split almost evenly between ProLR-w and ProLR-d. **Best-choice clustering and unclustering** consume 0.2% of the runtime. **Detailed placement** takes 5.5%.

Ckts	MAPLE w/ ILR	MAPLE w/ ProLR	Improv.
AD1	77.41	76.36	1.37%
AD2	89.07	86.95	2.38%
AD3	210.13	209.78	0.17%
AD4	190.07	179.91	5.35%
BB1	95.25	93.74	1.59%
BB2	149.84	144.55	3.53%
BB3	345.20	323.05	6.42%
BB4	792.20	775.71	2.08%
Avg	**1.03×**	**1.00×**	**2.86%**

Table 1: HPWL (×10e6) produced by ProLR and ILR on ISPD 2005 benchmarks "ADAPTEC (AD)" and "BIGBLUE (BB)".

7.3 Runtime considerations

As MAPLE is currently slower than some of its competitors, we note that industry implementations like ours tend to be handicapped (versus standalone academic implementations) by the use of a multipurpose design database. Because such a database stores information unnecessary to placement, the decreased cache locality increases runtime. Other relevant legacy infrastructures in our database include netlist-query support for accurate timing analysis and physical synthesis. In contrast to academic placers, our industry-strength implementation can work with a netlist that is dynamically changed during physical synthesis.

Unlike the original SimPL, our implementation does not use SSE instructions and is almost twice as slow (so far, we focused on solution quality and not runtime). Also, ProLR should parallelize well on multicore CPUs. Another consideration deals with the role of placement in physical synthesis, where it is invoked several times [3]. Fast execution is particularly important for early runs that estimate interconnect before netlist optimization. The top-level placement step from MAPLE produces good estimates because the final placement result does not look very different (Figure 6). Top-level placement consumes only $25 - 30\%$ of MAPLE runtime and can be accelerated as outlined above. As timing analysis and optimizations dominate the runtime of physical synthesis, greater effort in placement can be justified by improved results.

Runtime can sometimes be reduced by *deeper clustering* (more levels). To estimate its potential impact in MAPLE, we note that top-level placement takes 26% and ProLR takes 65% of MAPLE runtime on BIGBLUE4 (195.52 min. / 91% total). ProLR runtime is split 1:2 between the coarse and flat netlists. For *three levels* of clustering, top-level placement will take 13%, and ProLR will take 11% + 22% + 43% = 76% runtime. The total (191.23 min. / 89%) is only a 2% reduction versus *two levels*.

7.4 Comparisons on ISPD 2006 testcases

We compared MAPLE to other state-of-the-art academic and industry placers on the ISPD 2006 benchmark suite. Table 4 reports scaled HPWL and overflow penalty for several placers. Following the contest protocol, scaled HPWL is calculated as $HPWL \cdot (1 + 0.01 \cdot overflow\_penalty)$. On average, MAPLE achieved 11.28%, 5.59%, 13.58%, 6.63%, 11.57%, 4.37%, 3.13% scaled HPWL improvements versus APlace3 [22], NTUPlace3 (V7.05.30) [10], FastPlace3 [28], Kraftwerk2 [26], mFAR [14], mPL6 [7], and RQL [29], respectively. MAPLE obtains the best scaled HPWL results on seven out of eight circuits. Furthermore, compared to the other two best-performing placers on the benchmarks — RQL and NTUPlace3, MAPLE achieves lower overflow penalty on average. Thus, MAPLE not only reduces the wirelength but also avoids highly concentrated placements. Recall that the original implementation of SimPL [18] does not support density constraints of ISPD 2006 benchmarks and does not perform mixed-size placement.

Ckts	AP2	NTU3	mPL6	FP3	SimPL	MP
AD1	46.29	7.92	21.45	2.36	2.48	17.48
AD2	65.49	7.28	21.87	3.58	3.46	24.30
AD3	144.27	14.98	67.14	7.56	6.43	47.34
AD4	158.30	15.47	57.70	6.69	5.44	44.32
BB1	56.68	12.67	24.56	3.67	3.53	24.31
BB2	110.96	25.18	65.44	6.51	6.36	43.96
BB3	233.70	49.70	88.87	19.85	13.25	94.36
BB4	516.37	109.82	199.74	32.27	29.50	214.86
Avg	2.68×	0.43×	1.13×	0.16×	0.14×	1.00×

Table 2: Runtime comparison (minutes) on ISPD 2005 benchmarks for APlace2 (AP2), NTUPlace3 (NTU3), mPL6, FastPlace3 (FP3), SimPL and MAPLE (MP).

Benchmarks	APLACE2 [16]	NTUPLACE3 [10]	FASTPLACE3 [28]	KRAFTWERK2* [26]	mFAR* [22]	mPL6 [7]	SIMPL [18]	RQL* [29]	MAPLE
ADAPTEC1	78.35	81.82	78.66	82.43	82.50	77.93	78.58	77.82	**76.36**
ADAPTEC2	95.70	88.79	94.06	92.85	92.79	92.04	91.24	88.51	**86.95**
ADAPTEC3	218.52	214.83	214.13	227.22	217.56	214.16	**208.90**	210.96	209.78
ADAPTEC4	209.28	195.93	197.50	199.43	197.90	193.89	185.39	188.86	**179.91**
BIGBLUE1	100.02	98.41	96.67	97.67	98.80	96.80	97.54	94.98	**93.74**
BIGBLUE2	153.75	151.55	155.74	154.74	160.40	152.34	145.28	150.03	**144.55**
BIGBLUE3	411.59	360.66	365.16	343.32	368.70	344.10	340.24	323.09	**323.05**
BIGBLUE4	871.29	866.43	836.20	852.40	865.40	829.44	801.35	797.66	**775.71**
Geomean	**1.10**×	**1.07**×	**1.07**×	**1.08**×	**1.09**×	**1.05**×	**1.03**×	**1.02**×	**1.00**×

Table 3: Legal HPWL (×10e6) comparison on the ISPD 2005 benchmark suite. The previous best wirelengths are marked with gray. The placers marked by asterisks were unavailable to us in binary, and we reproduce HPWL from respective publications.

Benchmarks (Υ_{target})	APLACE3* [22]	NTUPLACE3 [10]	FASTPLACE3 [28]	KRAFTWERK2* [26]	mFAR* [22]	mPL6 [7]	RQL* [29]	MAPLE
ADAPTEC5 (0.5)	520.97 (15.9)	430.73 (12.2)	541.22 (36.5)	449.84 (3.69)	476.28 (6.21)	431.27 (1.09)	443.28 (9.25)	**407.33 (4.76)**
NEWBLUE1 (0.8)	73.31 (0.14)	**62.39 (0.76)**	76.56 (1.02)	65.95 (0.05)	77.54 (0.23)	68.08 (0.14)	64.43 (0.34)	69.25 (1.05)
NEWBLUE2 (0.9)	198.24 (0.42)	211.77 (3.21)	240.56 (1.97)	206.53 (1.28)	212.90 (0.59)	201.85 (1.52)	199.60 (1.45)	**191.66 (1.01)**
NEWBLUE3 (0.8)	273.64 (0.00)	280.19 (0.01)	301.72 (0.78)	279.58 (0.38)	303.91 (0.11)	284.11 (0.59)	269.33 (0.07)	**268.07 (0.77)**
NEWBLUE4 (0.5)	384.12 (1.74)	302.25 (9.22)	306.07 (7.74)	309.44 (1.71)	324.40 (5.42)	300.58 (1.63)	308.75 (15.2)	**282.49 (5.86)**
NEWBLUE5 (0.5)	613.86 (12.5)	547.20 (20.82)	633.72 (28.31)	563.15 (2.69)	601.27 (5.92)	537.14 (1.42)	537.49 (13.6)	**515.04 (4.05)**
NEWBLUE6 (0.8)	522.73 (0.03)	518.25 (6.08)	531.56 (1.26)	537.59 (1.70)	535.96 (1.63)	522.54 (1.40)	515.69 (4.33)	**494.82 (1.08)**
NEWBLUE7 (0.8)	1098.9 (0.06)	1114.2 (5.19)	1116.7 (1.33)	1162.1 (3.15)	1153.8 (1.58)	1084.4 (1.14)	1057.8 (2.57)	**1032.6 (1.70)**
Geomean	**1.13** × **(0.32)**	**1.04** × **(2.55)**	**1.16** × **(3.47)**	**1.07** × **(1.09)**	**1.13** × **(1.29)**	**1.06** × **(1.22)**	**1.03** × **(2.30)**	**1.00** × **(1.90)**

Table 4: Comparison of scaled HPWL (×10e6) which includes overflow penalty w.r.t the given target utilization on the ISPD 2006 benchmark suite. Overflow penalty values computed by the contest script are reported in parentheses. The placers marked by asterisks were unavailable to us in binary, and we reproduce results from respective publications. This hinders runtime comparisons.

8. CONCLUSIONS AND FUTURE WORK

The significance of large-scale placement in IC physical design is well-documented in recent literature [3] and is continuing to grow with the amount of on-chip random logic and current trends in interconnect scaling. Placement algorithms in the industry and academia were initially developed with the HPWL objective in mind [22] and later extended [3] to account for other objectives and concerns [12, 13, 17]. Despite known pitfalls, the HPWL objective appears to be a good performance predictor for various extensions of core placement algorithms. Focusing on the HPWL objective and module density, our research (i) contributes the discovery of essential deficiencies in prior techniques and (ii) advances the state of the art by developing algorithms that improve the quality of benchmark layouts beyond all published results. A full list of our contributions can be found in Section 1. For results on the ISPD 2011 routability-driven placement contest benchmark suite, see our related publication [17].

8.1 Perspectives

Our results bear some relevance to three recurring themes in physical design and physical synthesis. **One** is *the comparisons and trade-offs between linear and quadratic wirelength functions*. Since the 1960s, it was known that quadratic optimization was computationally efficient, but did not adequately track the demand for routing resources, which is much closer to the HPWL objective and its weighted variants [4]. Seminal work by Sigl, Doll and Johannes in the early 1990s developed a *linearization* technique that represents the linear wirelength objective on graphs by a dynamically-weighted quadratic objective [25]. However, the modeling of multi-pin nets remained inaccurate, and the research community has largely replaced quadratic optimization by much more cumbersome and slow non-convex optimization techniques ten years later [7, 10, 16]. In the mid-2000s, Spindler and Johannes developed the Bound2Bound model [26], which considerably improved the modeling accuracy for multi-pin nets in quadratic placement by employing a dynamic (placement-dependent) graph topology. With additional improvements to flat quadratic placement, this technique has recently outperformed prior art in both runtime and quality of results, both in terms of HPWL and in routability-driven placement [12, 17, 18]. This development raised several key research questions:

- Is there a tangible gap between the Bound2Bound model and the HPWL objective in practice ?
- Can global quadratic optimization with the Bound2Bound model be effectively improved on multi-million gate netlists (with respect to HPWL) ?
- Is multilevel placement optimization compatible with Bound2Bound and competitive in performance ?

Our work answers these three questions in the affirmative. The gap between Bound2Bound and HPWL is illustrated by the SimPL line in Figure 3 — note the *return* to smaller HPWL when detailed placement is invoked. Global quadratic placement of multi-million gate netlists can be improved by using the ProLR technique proposed in Section 5. MAPLE demonstrates that multilevel placement is compatible with the Bound2Bound model and is competitive with state of the art, as long as abrupt changes to placement are avoided before/after clustering. However, Section 7.3 shows that only two levels of clustering are useful for current benchmarks. Larger netlists may justify deeper clustering.

The second theme addressed in our work is relatively new to physical design, but no less fundamental — *methodology for module spreading and handling of whitespace*. These considerations are essential not only to global placement, but also to buffer insertion, gate sizing and other physical synthesis transformations, as well as to congestion-driven placement. Until the late 1990s, whitespace was rare in IC layouts, but now can reach over 60% by area [22]. We develop efficient techniques for spreading modules during placement, while satisfying density constraints and optimizing HPWL beyond the accuracy of the Bound2Bound model.

The third fundamental theme explored in our work has not received as much recognition, but may deserve it — we study *the composition of multiple optimizations into a high-precision, reliable multi-objective optimization process*. Our key discovery is that transitions between multiple objective functions and optimization techniques in placement often lead to major disruptions. In particular, adding netlist clustering or ILR to the SimPL algorithm for quadratic placement with the Bound2Bound model does not directly improve quality of results because the disruptions overshadow the benefits of such integration. To this end, we developed new techniques, such as two-tier Progressive Local Refinement (ProLR), to facilitate graceful transitions between multiple optimizations. In placement, these techniques are applied before and after unclustering, during the transition from a quadratic objective to HPWL, and before detailed placement. Many more applications exist in physical synthesis.

8.2 Further directions for future work

Empirical results in Tables 3 and 4 indicate a trend — quadratic placers RQL, SimPL and MAPLE produce overall better solutions than placers APlace3, NTUPlace3 and mPL6 based on non-convex optimization, which also tend to be slower. This is due, in part, to the greater amount of recent research on quadratic placement, including the development of successful industry tools [5, 29]. Yet, many of our contributions, such as ProLR, can be adapted for use in non-convex placers. Whether this will make non-convex placers competitive again, remains a curious direction for future work.

The SimPL placer used by MAPLE was recently extended to routability-driven placement [17] and power-driven placement with integrated clock-network synthesis [21]. Precision-handling of net weights demonstrated in [21] enables timing optimization. Opportunities remain for improving mixed-size placement in MAPLE.

Appendix - Computation of Initial θ_{step}

To implement Formula 5, MAPLE uses a step function that distinguishes three different cases: (i) emphasis on wirelength optimization, (ii) no bias, and (iii) emphasis on spreading. Given that Υ_{design} is fixed, the step function only depends on Υ_{target}, which is typically chosen by the designer. Assuming fixed-outline placement ($\Upsilon_{target} \geq \Upsilon_{design}$),

$$\theta^0_{step} = \begin{cases} 0.0250, & \text{if } \Upsilon_{target} - \Upsilon_{design} \geq 0.5 \\ 0.0275, & \text{if } \Upsilon_{target} - \Upsilon_{design} \geq 0.05 \\ 0.0375, & \text{if } \Upsilon_{target} - \Upsilon_{design} < 0.05 \end{cases} \qquad (9)$$

9. REFERENCES

[1] S. N. Adya, I. L. Markov, "Executable Placement Utilities," `http://vlsicad.eecs.umich.edu/BK/PlaceUtils/`

[2] C. J. Alpert et al., "A Semi-persistent Clustering Technique for VLSI Circuit Placement," *ISPD* 2005, pp. 200-207.

[3] C. J. Alpert et al., "Techniques for Fast Physical Synthesis," *Proc. IEEE* 95(3), 2007, pp. 573-599.

[4] A. E. Caldwell et al., "On Wirelength Estimations for Row-based Placement," *TCAD* 18(9), 1999, pp. 1265-1278.

[5] U. Brenner, M. Struzyna, J. Vygen, "BonnPlace: Placement of Leading-Edge Chips by Advanced Combinatorial Algorithms," *IEEE TCAD* 27(9) 2008, pp.1607-20.

[6] T. F. Chan, J. Cong, K. Sze, "Multilevel Generalized Force-directed Method for Circuit Placement," *ISPD* 2005, pp. 185-192.

[7] T. F. Chan et al., "mPL6: Enhanced Multilevel Mixed-Size Placement," *ISPD* 2006, pp. 212-214.

[8] H.-C. Chen et al., "Constraint Graph-based Macro Placement for Modern Mixed-size Circuit Designs," *ICCAD* 2008, pp. 218-223.

[9] H. Chen et al., "An Algebraic Multigrid Solver for Analytical Placement with Layout Based Clustering," *DAC* 2003, pp. 794-799.

[10] T.-C. Chen et al.,"NTUPlace3: An Analytical Placer for Large-Scale Mixed-Size Designs With Preplaced Blocks and Density Constraints," *IEEE TCAD* 27(7) 2008, pp.1228-1240.

[11] T.-C. Chen et al.,"MP-trees: A Packing-based Macro Placement Algorithm for Mixed-size Designs," *TCAD* 27(9) 2008, pp. 657-662.

[12] X. He et al., "Ripple: An Effective Routability-Driven Placer by Iterative Cell Movement," *ICCAD* 2011, pp. 74-79.

[13] M.-K. Hsu et al., "Routability-Driven Analytical Placement for Mixed-Size Circuit Designs," *ICCAD* 2011, pp. 80-84.

[14] B. Hu, M. Marek-Sadowska, "mFAR: Fixed-Points-Addition-based VLSI Placement Algorithm," *ISPD* 2005, pp. 239-241.

[15] A. B. Kahng, Q. Wang, "Implementation and Extensibility of an Analytic Placer," *IEEE TCAD* 2005, pp. 734-747.

[16] A. B. Kahng, Q. Wang, "A Faster Implementation of APlace," *ISPD* 2006, pp. 218-220.

[17] M.-C. Kim, J. Hu, I. L. Markov, "A SimPLR Method for Routability-driven Placement," *ICCAD* 2011, pp. 67-73.

[18] M.-C. Kim, D.-J. Lee, I. L. Markov, "SimPL: An Effective Placement Algorithm," *IEEE TCAD* 31(1), 2012, pp. 50-60.

[19] A. A. Kennings, I. L. Markov, "Smoothening Max-terms and Analytical Minimization of Half Perimeter Wirelength," *VLSI Design* 14(3), 2002, pp. 229-237.

[20] J. J. Kleinhans et al., "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization," *IEEE TCAD* 10(3), 1991, pp. 356-365.

[21] D.-J. Lee, I. L. Markov, "Obstacle-aware Clock-tree Shaping during Placement" to appear in *IEEE TCAD* 31(2), 2012.

[22] G.-J. Nam, J. Cong, "Modern Circuit Placement: Best Practices and Results," *Springer*, 2007.

[23] A. N. Ng et al., "Solving Hard Instances of Floorplacement," *ISPD* 2006, pp. 170-177.

[24] M. Pan, N. Viswanathan, C. Chu, "An Efficient & Effective Detailed Placement Algorithm," *ICCAD* 2005, pp. 48-55.

[25] G. Sigl, K. Doll, F. M. Johannes,"Analytical Placement: A Linear or a Quadratic Objective Function?" *DAC* 1991, pp.427-432.

[26] P. Spindler, U. Schlichtmann, F. M. Johannes, "Kraftwerk2 - A Fast Force-Directed Quadratic Placement Approach Using an Accurate Net Model," *IEEE TCAD* 27(8) 2008, pp. 1398-1411.

[27] N. Viswanathan, M. Pan, C. Chu, "FastPlace2.0: An Efficient Analytical Placer for Fixed-mode Designs," *ASPDAC* 2006, pp. 195-200.

[28] N. Viswanathan, M. Pan, C. Chu, "FastPlace3.0: A Fast Multilevel Quadratic Placement Algorithm with Placement Congestion Control," *ASPDAC* 2007, pp. 135-140.

[29] N. Viswanathan et al., "RQL: Global Placement via Relaxed Quadratic Spreading and Linearization," *DAC* 2007, pp. 453-458.

[30] N. Viswanathan et al., "Routability-Driven Placement Contest and Benchmark Suite," *ISPD* 2011, pp. 141-146.