ComPLx: A Competitive Primal-dual Lagrange Optimization for Global Placement

Myung-Chul Kim and Igor L. Markov
University of Michigan, EECS Department, Ann Arbor, MI 48109-2121
mckima@umich.edu, imarkov@eecs.umich.edu

ABSTRACT

We develop a projected-subgradient primal-dual Lagrange optimization for global placement, that can be instantiated with a variety of interconnect models. It decomposes the original non-convex problem into "more convex" sub-problems. It generalizes the recent SimPL, SimPLR and Ripple algorithms and extends them. Empirically, ComPLx outperforms all published placers in runtime and performance on ISPD 2005 and 2006 benchmarks.

Categories and Subject Descriptors

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

General Terms

Algorithms, Design, Performance

Keywords

Algorithms, optimization, physical design, placement

1. INTRODUCTION

The success of global placement determines all aspects of modern IC layout and physical synthesis [5] because it controls the amount of interconnect, which increasingly dominates on-chip resources and circuit performance [22]. However, the diverse algorithmic challenges posed by global placement and its complexity continue to surprise researchers [26]. Current algorithms still lag behind manual layout on circuits with structured components [34], do not always scale to extremely large circuits and are inconsistent in their handling of objective functions and various constraints. Analysis and comparisons of placement algorithms have been mostly empirical [26], with little formal justification.

A recent approach to global placement promises to support a variety of discrete and continuous constraints and was extended to handle routability-driven placement. Represented by the SimPL [23] and SimPLR [24] algorithms, this approach consistently outperforms previous state of the art in speed and solution quality, is amenable to thread-level and instruction-level parallelism, requires only a modest amount of code, and was successfully re-implemented by independent researchers (Ripple [18]). SimPL was extended to power-aware placement with integrated clock-network synthesis in [25]. However, a convincing mathematical foundation for this empirical success was lacking. While the SimPL approach is based on quadratic placement, the significance of this connection has remained unclear *vis-à-vis* techniques based on the log-sum-exp interconnect model [29].
Our contributions can be summarized as follows

- A projected subgradient primal-dual Lagrange optimization (ComPLx) for global placement compatible with a variety of interconnect models, including linearized quadratic, log-sum-exp, etc.
- Convergence analysis and ensuing enhancements.
- Casting existing algorithms SimPL [23], SimPLR [24] and Ripple [18] as special cases of ComPLx. In particular, ComPLx inherits their competitiveness and lends them mathematical substantiation.
- Algorithmic extensions for mixed-size, as well as timing and power-driven placement.
- Empirical validation of the theoretical framework underlying ComPLx. On ISPD 2005 benchmarks, ComPLx is 10% faster than FastPlace [32] (including detailed placement runtime). It outperforms SimPL and RQL (the best published placers) by 1%. On ISPD 2006 benchmarks, ComPLx outperforms the leading placer RQL [33] by 1% in terms of scaled HPWL while running 2.5× faster.

In the remainder of the paper, Section 2 reviews necessary background. Section 3 introduces our primal-dual Lagrangian relaxation ComPLx, whose convergence is discussed in Section 4. Section 5 points out that the SimPL, SimPLR and Ripple algorithms are special cases of ComPLx. It then extends ComPLx to mixed-size and timing-driven placement. Section 6 presents empirical studies with improvements over these algorithms. Conclusions are given in Section 7. Comparisons to other primal-dual Lagrange optimizations are discussed in Section S4.

2. BACKGROUND

Global placement [22] of a netlist $\mathcal{N} = (E, V)$ with nets E and n nodes (cells) V seeks a set of planar node locations $(\vec{x}, \vec{y}) \in [x_{min}, x_{max}]^n \times [y_{min}, y_{max}]^n$ that minimize

DAC 2012, June 3-7, 2012, San Francisco, California, USA.

the weighted Half-Perimeter WireLength (wHPWL). For locations $\vec{x} = \{x_i\}$, $\vec{y} = \{y_i\}$ and net weights $\vec{w} = \{w_i\}$, $\text{wHPWL}_{\mathcal{N}}(\vec{x}, \vec{y}) = \text{wHPWL}_{\mathcal{N}}(\vec{x}) + \text{wHPWL}_{\mathcal{N}}(\vec{y})$, where

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

This piecewise-linear function lends itself to linear programming (LP) and min-cost max-flows, but these techniques have been successful only for smaller netlists. In large-scale placement, HPWL is approximated by convex twice-differentiable functions $\Phi(\vec{x}, \vec{y})$ and optimized numerically by linear or nonlinear Conjugate Gradient.

Quadratic approximations are used in many placers

$$\Phi_Q(\vec{x}, \vec{y}) = \vec{x}^T Q_x \vec{x} + \vec{f}_x \vec{x} + \vec{y}^T Q_y \vec{y} + \vec{f}_y \vec{y} \quad (2)$$

with matrices Q_x, Q_y derived from the netlist and vectors $\vec{f}_x, \vec{f}_y$ that reflect connections to fixed objects. When sufficiently many nodes in a connected netlist are fixed, Φ_Q is strictly convex and can be optimized quickly. To approximate the HPWL by quadratic functions, one uses a *linearization* technique [30], adjusting the approximations at every global placement iteration. In particular, single-edge terms of the form $w_{ij}(x_i - x_j)^2$ are changed to $\frac{w_{ij}(x_i - x_j)^2}{|x'_i - x'_j| + \varepsilon}$ where the primed values are constants based on *the result of the last iteration* (a.k.a. the last *iterate*). Multipin nets are decomposed into sets of edges using stars, cliques or the Bound2Bound model [31]. Other differentiable approximations to the HPWL objective are outlined in supplementary material (Section S1).

Constraints in placement include *legality*, *target utilization*, *routability*, *resource-type* constraints, etc.

$$(\vec{x}, \vec{y}) \in \mathcal{C}$$

which prohibit multiple pairs (x_i, y_i) from concentrating in small regions. The demands for physical on-chip resources (gate area or number of routes in a region) must not exceed available supplies/design constraint (area for placing logic gates, target utilization, number of routing tracks) [22]. This is typically expressed by inequalities, e.g., allowing at most $C_{j,k}$ placeable objects in grid-cell (j, k). These inequalities are easy to satisfy when no optimization is performed. Unlike Φ, the constraints are noncovex, as illustrated by constraints on locations of two non-overlapping rectangles. Another type of constraints — routability of modern IC layouts — is NP-hard to evaluate with sufficient accuracy [22]. Some layout regions may be blocked by fixed obstacles and unavailable to (x_i, y_i), leading to discrete choices, such as placing an object on one side of an obstacle. This inhibits smooth convex optimization and, historically, motivated specialized global-placement techniques tailored to variant objective functions and constraints [26].

3. A PRIMAL-DUAL LAGRANGE METHOD

We propose a general method for handling constraints in global placement with a variety of possible interconnect models, and show how to decompose the original non-convex problem into "more convex" sub-problems.

A Lagrangian relaxation of global placement can be constructed if constraints are specified as *equalities* $\Pi(\vec{x}, \vec{y}) = 0$. Since supply-demand *inequalities* are usually given instead, the more general Karush-Kuhn-Tucker conditions may at first seem more relevant.[1] However, working with supply-demand inequalities directly is difficult because they are specified algorithmically, not as closed-form expressions in $(\vec{x}, \vec{y})$. Without derivatives, one resorts to *subgradient optimization* [6], while the nature of the constraints calls for approximation. Placement techniques based on non-convex optimization [20, 9, 12] fit demand distribution to smooth functions using *kernel-density estimation*, and this facilitates gradient estimation. Each such step is laborious, and many steps may be required because, after moving in the gradient direction, one may need to "make turns" (as illustrated by moving around a rectangular obstacle). The reliance on local subgradient information in [20, 12] is common in analytical placement and with possible exceptions of Kraftwerk [31] and mPL6 [9] which estimate subgradients by solving second-order linear elliptic PDEs with global supply-demand information. Solutions of these PDEs can be written as convolutions of the density function with a fixed Green's function $G(s, t)$ (dependent on boundary conditions), which sometimes vanishes away from $s = t$. Further, local subgradient computations leave undefined the trade-off between demand-distribution subgradients and the gradients of the objective function. This *force modulation* problem was articulated in [33], but addressed there with *ad hoc* thresholding.

In contrast to other methods, our subgradients point to a closest $\mathcal{C}$-feasible solution, and their magnitude is modulated by respective distance. Thus, we define $\Pi_{\mathcal{C}}(\vec{x}, \vec{y})$ as the L_1-distance from $(\vec{x}, \vec{y})$ to a closest $\mathcal{C}$-feasible solution.

$$\Pi_{\mathcal{C}}(\vec{x}, \vec{y}) = \min_{(\vec{x}_*, \vec{y}_*) \in \mathcal{C}} ||(\vec{x}, \vec{y}) - (\vec{x}_*, \vec{y}_*)||_1$$

$$= \min_{(\vec{x}_*, \vec{y}_*)} \left(||\vec{x} - \vec{x}_*||_1 + ||\vec{y} - \vec{y}_*||_1 \right) \quad (3)$$

Clearly, $\Pi_{\mathcal{C}}(\vec{x}, \vec{y}) = 0 \Leftrightarrow (\vec{x}, \vec{y}) \in \mathcal{C}$. Therefore, in addition to primary variables $(\vec{x}, \vec{y})$, we introduce one dual variable (multiplier) $\lambda \geq 0$, and establish the following Lagrangian

$$\mathcal{L}_{\Phi,\mathcal{C}}(\vec{x}, \vec{y}, \lambda) = \Phi(\vec{x}, \vec{y}) + \lambda \, \Pi(\vec{x}, \vec{y}) \quad (4)$$

We use L_1-norms so that costs and penalties are expressed in meters and can be compared. Hence, λ is dimensionless.

Primal-dual Lagrangian relaxation [3] alternates minimization over the primal variables with maximization over the dual variable(s). $\min \Phi(\vec{x}, \vec{y})$ subject to $(\vec{x}, \vec{y}) \in \mathcal{C}$ can be found by *sequential unconstrained optimization*

$$\max_{\lambda} \min_{(\vec{x}, \vec{y})} \mathcal{L}_{\Phi,\mathcal{C}}(\vec{x}, \vec{y}, \lambda) \quad (5)$$

Starting with $\lambda_0 = 0$, the first primal iterate is produced by minimization of $\Phi(\vec{x}, \vec{y})$ (using quadratic optimization or non-linear Conjugate Gradient, depending on the function). At subsequent iterations, primal optimization must also account for the penalty term. A straightforward argument by contradiction shows that for $\lambda_k < \lambda_{k+1}$

$$\min_{(\vec{x}, \vec{y})} \mathcal{L}_{\Phi,\mathcal{C}}(\vec{x}, \vec{y}, \lambda_k) \leq \min_{(\vec{x}, \vec{y})} \mathcal{L}_{\Phi,\mathcal{C}}(\vec{x}, \vec{y}, \lambda_{k+1}) \quad (6)$$

As λ increases, so does the sensitivity of $\mathcal{L}_{\Phi,\mathcal{C}}$ to Π. Therefore, the minimization of $\mathcal{L}_{\Phi,\mathcal{C}}$ affects the Π term more, and this term decreases. However, since the minimized value of

[1] Inequalities can also be converted into equations by adding slack variables, but we avoid this common technique, to limit computational complexity.

$\mathcal{L}_{\Phi,\mathcal{C}}$ increases (per Formula 6), Φ must increase. Eventually, $(\vec{x},\vec{y})$ become $\mathcal{C}$-feasible (or very close to), making the Lagrangian insensitive to λ and indicating that an optimum is near. The following weak duality bounds hold for any $\mathcal{C}$-feasible solution $(\vec{x}^\circ,\vec{y}^\circ)$ and any iterate $(\vec{x},\vec{y})$ after primal optimization.

$$\Phi(\vec{x},\vec{y}) \le \mathcal{L}_{\Phi,\mathcal{C}}(\vec{x},\vec{y},\lambda) \le \mathcal{L}_{\Phi,\mathcal{C}}(\vec{x}^\circ,\vec{y}^\circ,\lambda) = \Phi(\vec{x}^\circ,\vec{y}^\circ) \quad (7)$$

The first $\le$ is due to $\lambda \ge 0$ in Formula 4. The second $\le$ is due to $(\vec{x},\vec{y})$ being argmin from Formula 5 and the third $=$ is due to $\Pi(\vec{x}^\circ,\vec{y}^\circ) = 0$ ($\mathcal{C}$-feasible). The second inequality is strict unless $(\vec{x},\vec{y})$ is $\mathcal{C}$-infeasible, hence $\Phi(\vec{x},\vec{y}) < \Phi(\vec{x}^\circ,\vec{y}^\circ)$. The *duality gap* is

$$\Delta_\Phi = \Phi(\vec{x}^\circ,\vec{y}^\circ) - \Phi(\vec{x},\vec{y}) \quad (8)$$

minimized over best available *primal feasible* $(\vec{x}^\circ,\vec{y}^\circ)$ and $(\vec{x},\vec{y})$ at a given point during optimization.

Approximating the penalty term allows us to replace the nonconvex Lagrangian by a convex one. Here we use the *feasibility projection*

$$P_\mathcal{C}(\vec{x},\vec{y}) = \text{argmin}_{(\vec{x}_*,\vec{y}_*)\in\mathcal{C}} ||(\vec{x},\vec{y}) - (\vec{x}_*,\vec{y}_*)||_1 \quad (9)$$

that finds a closest $\mathcal{C}$-feasible approximation (performs *pseudo-legalization*) of $(\vec{x},\vec{y})$.[2] Since $P_\mathcal{C}(\vec{x}',\vec{y}')$ is $\mathcal{C}$-feasible, $\Phi(\vec{x},\vec{y}) \le \Phi\big(P_\mathcal{C}(\vec{x}',\vec{y}')\big)$ by Inequalities 7. Given that Φ is continuous, $||(\vec{x},\vec{y}) - P_\mathcal{C}(\vec{x},\vec{y})||_1 \to 0$ would necessitate $\Phi\big(P_\mathcal{C}(\vec{x},\vec{y})\big) - \Phi(\vec{x},\vec{y}) \to 0$. Hence, $\Phi\big(P_\mathcal{C}(\vec{x},\vec{y})\big)$ must generally decrease, providing upper bounds on final placement cost.

After finding $\mathcal{C}$-feasible *anchor locations* $(\vec{x}^\circ,\vec{y}^\circ) = P_\mathcal{C}(\vec{x},\vec{y})$, we establish the simplified Lagrangian

$$\mathcal{L}^\circ_\Phi(\vec{x},\vec{y},\lambda) = \Phi(\vec{x},\vec{y}) + \lambda||(\vec{x},\vec{y}) - (\vec{x}^\circ,\vec{y}^\circ)||_1 \quad (10)$$

To minimize it with respect to fixed $(\vec{x}^\circ,\vec{y}^\circ)$ and λ, the L_1-term can be approximated by the same type of function as Φ (see Section 5). Thus, for quadratic Φ, the optimality condition $\nabla\mathcal{L}^\circ_\Phi(\vec{x},\vec{y},\lambda) = 0$ turns into a system of linear equations. For other functional forms, such as the log-sum-exp expressions, one can minimize $\mathcal{L}^\circ_\Phi(\vec{x},\vec{y},\lambda)$ using the nonlinear Conjugate Gradient method or other known alternatives.[3] In addition to being (strictly) convex, $\mathcal{L}^\circ_\Phi(\vec{x},\vec{y},\lambda)$ is usually separable into its x and y components which can be optimized independently. One can verify Inequalities 6 and 7 for $\mathcal{L}^\circ_\Phi(\vec{x},\vec{y},\lambda)$ subject to $(\vec{x}^\circ,\vec{y}^\circ) = P_\mathcal{C}(\vec{x},\vec{y})$.

The ComPLx framework re-solves $\nabla\mathcal{L}^\circ_\Phi(\vec{x},\vec{y},\lambda) = 0$ and $P_\mathcal{C}(\vec{x},\vec{y}) = 0$ until convergence. The result of global placement can be read from the last iterate $(\vec{x},\vec{y})$ or the last $\mathcal{C}$-feasible iterate $(\vec{x}^\circ,\vec{y}^\circ)$ as discussed in Section 4.

4. CONVERGENCE ANALYSIS

It is sufficient for $P_\mathcal{C}$ to find a $\mathcal{C}$-feasible solution that is *reasonably close*, rather than closest, to a given $(\vec{x},\vec{y})$.[4] Such *approximate projected subgradient* methods are relatively recent in the operations-research literature [16, Section 1] but are proven to converge as long as $P_\mathcal{C}$ does not increase the distance to the set $\mathcal{C}$ and typically reduces it during iterations [7, Sections 2 and 3]. In particular, $P_\mathcal{C}$ should return its input when the input is $\mathcal{C}$-feasible. Convergence can be improved if $P_\mathcal{C}$ exhibits reasonable *fidelity* with respect to the exact feasibility projection, and is *self-consistent*

$$||(\vec{x},\vec{y}) - P_\mathcal{C}(\vec{x},\vec{y})||_1 > ||(\vec{x}',\vec{y}') - P_\mathcal{C}(\vec{x},\vec{y})||_1 \;\Rightarrow$$
$$||(\vec{x},\vec{y}) - P_\mathcal{C}(\vec{x}',\vec{y}')||_1 > ||(\vec{x}',\vec{y}') - P_\mathcal{C}(\vec{x}',\vec{y}')||_1 \quad (11)$$

In other words, if $(\vec{x}',\vec{y}')$ is closer to $P_\mathcal{C}(\vec{x},\vec{y})$ than $(\vec{x},\vec{y})$, then it should also be closer to $P_\mathcal{C}(\vec{x}',\vec{y}')$. The ComPLx implementation of $P_\mathcal{C}$ reviewed in Section 5 handles both standard cells and macros. It is self-consistent through almost all iterations, as shown in Section S2. Figure 1 illustrates changes in $\mathcal{L}_k$, Φ_k, and Π_k over ComPLx iterations on BIGBLUE4. The same trends show on all other benchmarks, validating the discussion in Section 3.

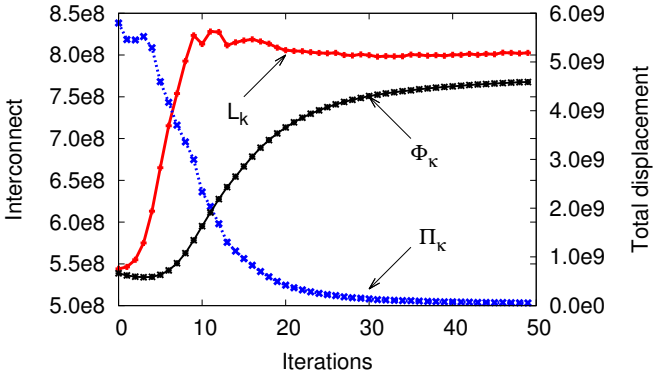

Figure 1: Progressions of $\mathcal{L}$ (the total Lagrangian), Φ (netlist interconnect), and Π (L_1-distance to legal) over ComPLx iterations on BIGBLUE4. $\mathcal{L}$ increases steeply in the early placement iterations, as λ increases. Π decreases while Φ gradually increases.

Global placement iterations stop when a $\mathcal{C}$-feasible value is reached, which must happen when λ exceeds its optimal value. But the resulting solution may be far from optimal. To avoid this, we propose to improve the efficiency of the first few iterations, since the first iterates are crucial to the overall success (given that we are solving a nonconvex problem overall). The earliest non-zero value of λ must be sufficiently small so that $\Phi(\vec{x},\vec{y}) \gg \lambda\Pi(\vec{x},\vec{y})$, to make sure that $\mathcal{L}_{\Phi,\mathcal{C}}(\vec{x},\vec{y},\lambda)$ is dominated by the convex *cost* term rather than the penalty term. Hence, we initially select $\lambda_1 = \Phi/100\Pi$. This calculation is supported by the fact that Π and Φ are expressed in the same units (meters). To *avoid* premature progress, a maximum increase in λ can be imposed, say 100% per iteration.

$$\lambda_{k+1} = \min\{2\lambda_k, \lambda_k + (\Pi_{k+1}/\Pi_k)h\} \quad (12)$$

where h is a scaling constant. λ increases proportionally to Π changes to ensure that Π decreases by a sufficient amount, as Φ increases. Considering the number of iterations until λ reaches its optimal value, there is no explicit dependency on the number of variables. In practice, the maximal λ values and the iteration count do not grow with the size of the problem instance as shown in Section S3.

[2]One can additionally require breaking ties toward smaller values of Φ (or even some trade-off with Φ), but this does not seem necessary for practical success (Section 6).
[3]Techniques such as Newton's method that approximate the objective f by quadratic functions based on Hessian(f) essentially perform sequential quadratic optimization.
[4]Section S2 points out that this is a "more convex" problem.

Convergence criteria can be defined in terms of $r(\vec{x},\vec{y}) = ||(\vec{x},\vec{y}) - P_{\mathcal{C}}(\vec{x},\vec{y})||_1$, rather than $\mathcal{L}$ — when the placement is close to $\mathcal{C}$-feasible, a detailed placer can produce optimized site-aligned legal locations. Given that pseudo-legalization $(\vec{x}^\circ,\vec{y}^\circ) = P_{\mathcal{C}}(\vec{x},\vec{y})$ is performed at every iteration, one can run detailed placement on $(\vec{x}^\circ,\vec{y}^\circ)$ rather than on $(\vec{x},\vec{y})$. This would allow an even more aggressive convergence criterion in terms of the duality gap $\Delta_\Phi = \Phi(\vec{x}^\circ,\vec{y}^\circ) - \Phi(\vec{x},\vec{y})$.[5] To substantiate this idea, we observe that performing detailed placement on a feasible solution $(\vec{x}^\circ,\vec{y}^\circ)$ should *not* increase costs (rather the opposite), whereas performing detailed placement on $(\vec{x},\vec{y})$ is likely to (as observed in practice). This observation upper-bounds the difference in final costs between these two scenarios by Δ_Φ.

5. SPECIAL CASES AND EXTENSIONS

We now point out that the SimPL [23], SimPLR [24] and Ripple [18] algorithms are special cases of the proposed primal-dual Lagrangian relaxation. They implement Φ as a quadratic approximation Φ_Q of HPWL, adjusted at every iteration through the linearized Bound2Bound net model [31]. Linearization is also applied to represent the L_1-norm in the penalty term Π. To model this term, each movable object is connected to its anchor location by a *pseudonet*, contributing $w_i(x_i - x_i^\circ)^2$ to the overall objective (and a similar y-term), where $w_i = \frac{\lambda}{|x_i - x_i^\circ| + \varepsilon}$ is based on the last iterate. $\varepsilon > 0$ is used to bound the denominator away from zero and make the objective function strictly convex. In SimPL and SimPLR, ε is calculated as 1.5 times row height.[6] This matches Formula 10 if the L_1-distance term is approximated by a linearized quadratic function (Section 2). SimPL, SimPLR and Ripple maintain a lower and an upper-bound placement at each iteration, and these placement satisfy conditions in Formula 7 as seen in [23, Figure 6], [24, Figure 4].

The SimPL [23], SimPLR [24] and Ripple [18] algorithms differ in how they define and implement the feasibility projection $P_{\mathcal{C}}$. In practice, to identify overfilled bins with respect to a *target utilization/density limit* $0 < \gamma \leq 1$ [23, Section 4], a uniform grid is superimposed over the entire layout. Then the feasibility projection seeks to satisfy the given target utilization/density limit within each grid-cell. To this end, the SimPL $P_{\mathcal{C}}$ first localizes the changes in $(\vec{x},\vec{y})$ to the smallest rectangular grid-cell sub-arrays that satisfy a given target utilization/density limit, and then processes each region by a top-down geometric-partitioning framework. SimPL alternates (i) piecewise-linear scaling in x and y directions with (ii) spreading locations in each dimension to even out density, while preserving the relative order (determined by sorting). This pseudo-legalization is discussed in more detail in Section S2 and can be seen as solving a convex problem in terms of (always-positive) *distances between neighboring x locations* (y locations). As a runtime trade-off, SimPL gradually increases the accuracy of $P_{\mathcal{C}}$ as the grid-cell size decreases, and we use this feature in Section 6 to show that $P_{\mathcal{C}}$ does not need to be implemented precisely. SimPLR and Ripple generally follow the SimPL techniques, but are concerned with routability in addition to HPWL. Therefore, they estimate congestion after placement

[5] As Φ is Lipschitz, $r(\vec{x},\vec{y}) \to 0$ implies $\Delta_\Phi \to 0$.

[6] In [30], a lower bound on the distance between two modules is defined as the average module width.

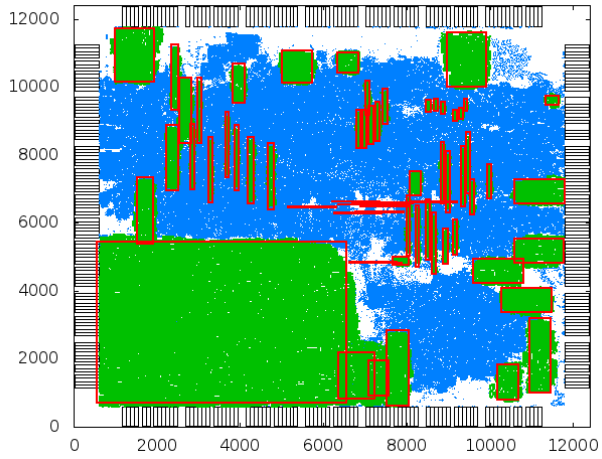

Figure 2: Macro shredding for feasibility projection $P_{\mathcal{C}}$ on NEWBLUE1 (an intermediate placement). Red boxes show the locations of macro cells at the centers of gravity of constituent cells (shown as green dots). Standard cells are shown as blue dots.

iterations (SimPLR calls a global router, whereas Ripple estimates congestion directly) and modify $P_{\mathcal{C}}$ to produce low-congestion placements. SimPLR preprocesses $P_{\mathcal{C}}$ by temporarily increasing the dimensions of some movable objects, so as to enhance geometric separation between them. Ripple distinguishes congestion maps for horizontal and vertical wiring, and scales minimal-sized rectangular regions differently each direction. Despite the technical differences, all these variants compute $P_{\mathcal{C}}$ by a series of convex optimizations. The use of feasibility projections is not only common between SimPL [23], SimPLR [24] and Ripple [18], but also distinguishes them from other placement algorithms. This is why SimPL, SimPLR and Ripple are particularly good at handling nonlinear, nonconvex layout constraints, such as numerous fixed obstacles present in modern SoC layouts.

Mixed-size placement requires careful accounting for pin offsets during quadratic optimization (since pin-offsets can be large in macros), as well as an approximate feasibility projection $P_{\mathcal{C}}$ which can handle macros and standard cells. We have therefore revised and extended the *macro shredding technique* from [2]. Macro cells are divided into equal-sized cells (2×2 standard-cell height), but unlike prior work, ComPLx does not connect constituent cells (shreds) with fake nets and thus does not modify the linear systems it solves. The conventional $P_{\mathcal{C}}$ [23] is applied to the shreds, after which the action of $P_{\mathcal{C}}$ on the original macro is interpolated by averaging the displacement of the shreds. Given that the conventional $P_{\mathcal{C}}$ [23] mostly preserves the relative placement of cells and is approximately locally isometric, the arrays of shreds are transformed into shapes similar to arrays, as seen in Figure 2. As $P_{\mathcal{C}}$ seeks to satisfy the given target utilization ($0 < \gamma < 1$), additional whitespace is inserted among constituent cells. Then the bounding box of projected locations of shreds outgrows the original macro cell, creating a halo around the macro, where other cells cannot be placed. To compensate, we multiply the widths and heights of constituent cells by $\sqrt{\gamma}$. Stabilizing macro positions early is important, as they greatly impact adja-

Benchmarks		Best published as of 02/27/2011	ComPLx					
			Finest Grid		P_C+=FastPlace-DP		Default Config.	
size (# of modules)		HPWL (placer)	HPWL	Runtime	HPWL	Runtime	HPWL	Runtime
adaptec1	211K	**77.82 (RQL**	78.95	3.86	78.39	93.52	**77.75**	**3.09**
adaptec2	255K	**88.51 (RQL)**	89.81	5.23	91.09	162.07	**88.76**	**4.31**
adaptec3	452K	207.67 (SimPL)	207.07	11.54	**203.70**	323.04	206.57	**10.75**
adaptec4	496K	186.80 (SimPL)	185.13	10.90	188.37	252.40	**184.07**	**9.57**
bigblue1	278K	94.98 (RQL)	96.15	7.22	**94.78**	140.27	95.30	**7.00**
bigblue2	558K	**145.47 (SimPL)**	**144.59**	9.91	146.32	200.56	145.87	**8.53**
bigblue3	1.10M	**323.09 (RQL)**	352.33	32.52	327.49	629.23	330.74	**24.80**
bigblue4	2.18M	797.66 (RQL)	**787.11**	47.87	792.26	961.25	789.45	**41.89**
Geomean		1.00×	1.01×	1.16×	1.00×	26.56×	1.00×	1.00×

Table 1: Legal HPWL (×10e6) and total runtime (in min.) comparison on ISPD 2005 benchmarks. Each run uses a single thread on a 2.8GHz workstation. Best-published numbers are annotated with the placers that produced them – SimPL [23] or RQL [33]. mPL6 and NTUPlace3 were also considered in this comparison. We regenerated placements of SimPL without a cell-orientation optimization.

cent standard cells. To accelerate the convergence of macro cells and decrease their displacement during legalization, we extend Formulae 4 and 10 with separate, larger λ values for each macro, computed as the default λ times the ratio of the size of macro cell to the average standard-cell size. As seen in Figure 2, our mixed-size feasibility projection P_C may leave small overlaps between macros. Rather than force complete legalization, we let multiple global placement iterations (including P_C) gradually decrease these overlaps. We observe that as P_C displaces cells and macros *less*, the changes in the shapes of shredded macros also *decrease*, and this *increases* the precision of legalization for macros during P_C. Even if slight overlaps remain at the end of global placement, they can be fixed by the detailed placer without undermining the overall performance. While less sophisticated than algorithms in [10, 11, 35], our mixed-size approximate feasibility projection P_C is easy to implement and produces very good results, motivating additional studies.

Extensions for timing- and power-driven placement traditionally rely on net weights computed from activity factors and timing slacks [22, Chapter 8]. Net-weighting schemes in the literature include rigorous, provably convergent methods [8]. Since our mathematical formulation for *global placement* in Section 3 accounts for net weights in Φ, existing techniques [8] and their provable properties apply directly. An extension of SimPL with power-driven net weights is reported in [25]. However, we observe that *the impact of the feasibility projection and detailed placement* suggests revising the penalty term in the Lagrangian. Minimizing L_1-distance to $\mathcal{C}$ may leave some cells far from their legal positions, forcing P_C or the detailed placer to displace them. This may stretch out incident nets, which is undesirable for timing- and power-critical standard cells. Hence, in the simplified Lagrangian of Formula 10 we weigh the penalty term by timing/power criticality and replace

$$\lambda||(\vec{x},\vec{y})-(\vec{x}^\circ,\vec{y}^\circ)||_1 \quad \text{by} \quad \lambda(\vec{\gamma}\cdot|(\vec{x},\vec{y})-(\vec{x}^\circ,\vec{y}^\circ)|) \qquad (13)$$

where $|\ |$ represents the vector of pointwise distances and $\vec{\gamma}$ represents the vector of cell-criticalities. Initially, $\vec{\gamma}$ is populated with switching activity factors (no cells are critical). When static timing analysis, performed between placement iterations, indicates that cell i lies on a critical path (violates a timing constraint), the cell's criticality must be increased $\gamma_i = \gamma_i(1+\delta)$, (along with the weights of critical nets in Φ).

6. EMPIRICAL VALIDATION

Our implementation of ComPLx inherits the performance and runtime advantages of SimPL [23]. Experiments ran on a 2.8GHz Intel Core-i7 860 Linux server with 8GB RAM, using one CPU core. Detailed placement was done by FastPlace-DP [28]. All settings were the same for all benchmarks.

Given that quadratic optimization in SimPL and ComPLx is optimal, we tried to improve the feasibility projection P_C. One such attempt used the finest grid during all global placement iterations. In a second attempt, we post-processed the result of P_C by the detailed placer [28] at each iteration. Our data in Table 1 show only a marginal improvement, but at a runtime cost. *Vice versa*, coarsening the grid speeds up P_C without undermining solution quality. Thus, no interconnect optimization during P_C is required. While surprising, this is consistent with the discussion in Section 4 and can be explained by the known convergence properties of Primal-dual Lagrange optimization [6, 3]. In practice this decreases the risk of incorrect implementation. On ISPD 2005 benchmarks, ComPLx outperforms SimPL, sometimes by a small amount, sometimes significantly. The similarities are not surprising because ComPLx generalizes SimPL. The improvements are due to the refined convergence criterion (Section 4) and improved scheduling of λ (λ corresponds to the pseudonet weight in [23]). ComPLx outperforms SimPL and RQL (the best published placers) by 1%. ComPLx produces best results on more benchmarks than any prior placer, while running 10% faster than FastPlace (including FastPlace-DP runtime in both cases).

Table 2 covers ISPD 2006 benchmarks, which include density constraints and movable macros, not handled by SimPL. ComPLx outperforms the best-published placer RQL [33] by 1% in terms of scaled HPWL (the official contest metric). ComPLx is about 12% faster than FastPlace (including FastPlace-DP runtime in both cases), as well as 6.88× and 8.47× faster than NTUPlace3 and mPL6, respectively. RQL is 3.1× faster than mPL6 [33], hence > 2.5× as slow as ComPLx (including detailed placement by FastPlace-DP).

The SimPL placer (generalized in this work) was extended to a routability-driven placer SimPLR in [24] with strong results on ISPD 2011 benchmarks. Section S5 demonstrates that ComPLx naturally supports region constraints (Figure 4). Section S6 illustrates timing-driven placement.

Benchmarks (Υ_{target})	NTUPL3 [12]	MPL6 [9]	RQL [33]	COMPLX
ADAPTEC5 (0.5)	451.22 (21.0)	**431.27 (1.09)**	443.28 (9.25)	432.60 (3.09)
NEWBLUE1 (0.8)	**62.65 (1.09)**	68.08 (0.14)	64.43 (0.34)	64.71 (0.18)
NEWBLUE2 (0.9)	205.45 (2.53)	201.85 (1.52)	199.60 (1.45)	**197.24 (1.04)**
NEWBLUE3 (0.8)	277.87 (0.00)	284.11 (0.59)	**269.33 (0.07)**	272.87 (0.69)
NEWBLUE4 (0.5)	306.56 (13.1)	**300.58 (1.63)**	308.75 (15.2)	306.00 (3.00)
NEWBLUE5 (0.5)	**509.71 (9.56)**	537.14 (1.42)	537.49 (13.6)	540.29 (2.58)
NEWBLUE6 (0.8)	520.31 (8.40)	522.54 (1.40)	515.69 (4.33)	**501.90 (1.06)**
NEWBLUE7 (0.8)	1109.6 (5.32)	1084.4 (1.14)	1057.8 (2.57)	**1042.2 (1.27)**
Geomean	**1.01×** (2.40)	**1.03×** (1.22)	**1.01×** (2.30)	**1.00×** (1.61)

Table 2: Comparison of scaled HPWL (×10e6) on ISPD 2006 benchmarks. Overflow penalties are reported in parentheses. RQL results are from [33].

7. CONCLUSIONS

We developed a global placement algorithm ComPLx based on *subgradient projected primal-dual Lagrange optimization*. In its basic form, it consists of (*i*) interconnect optimization, (*ii*) a feasibility projection P_C that represents placement constraints, (*iii*) a penalty term that includes the Lagrange multiplier λ. Our extensions for mixed-size placement handle macros through the feasibility projection P_C and establish a separate, larger λ parameter for each macro. Timing-driven extensions track separate λ for timing-critical cells and increase λ based on criticality (slack). Our baseline algorithm generalizes recent SimPL [23], SimPLR [24] and Ripple [18] algorithms and inherits their empirical success. *Vice versa*, ComPLx provides mathematical substantiation and convergence analysis for SimPL, SimPL and Ripple, suggesting improvements and algorithmic extensions.

A key difference from most prior analytical frameworks is in the spreading mechanism — rather than estimate *density gradients based on local information*,[7] we use a global feasibility projection P_C. Consequently, the handling of *region*, *alignment* and other types of constraints requires only the modification of the feasibility projection (Section S5). Avoiding local gradients also improves runtime (compared to APlace and NTUPlace3), and so does our avoidance of optimization by local search (compared to FastPlace and RQL). The tradeoff between spreading and interconnect optimization is controlled by Lagrange multipliers λ.

A key difference from analytical placement based on nonconvex optimization [20, 9, 12] is the emphasis on decomposing the original problem into a series of convex optimizations, which enables duality and accelerates convergence. Unlike prior works limited to a single interconnect model, our technique can be used with *quadratic*, *log-sum-exp* and other models (Section S1). The closest published primal-dual Lagrangian optimizations are discussed in Section S4.

[7] mPL6 and Kraftwerk2 are the only competitive prior placers with a global view of supply-demand trade-offs. We are exploring theoretical comparisons to them in ongoing work.

8. REFERENCES

[1] S. N. Adya, I. L. Markov, P. G. Villarrubia, "On Whitespace and Stability in Physical Synthesis," *Integration, the VLSI Journal* vol. 39/4, 2006, pp. 340-362.

[2] S. N. Adya, I. L. Markov, "Combinatorial techniques for Mixed-size Placement," *ACM Trans. Design Autom. Electr. Syst.* 10(1), 2005, pp. 58-90.

[3] R. K. Ahuja, T. L. Magnati, J. B. Orlin, "Network Flows: Theory, Algorithms, and Applications," *Prentice Hall* 1993.

[4] C. J. Alpert, T. F. Chan, A. B. Kahng, I. L. Markov, P. Mulet, "Faster Minimization of Linear Wirelength for Global Placement," *IEEE Trans. on CAD of Integrated Circuits and Systems* 17(1), 1998, pp. 3-13.

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

[6] D. P. Bertsekas, "Nonlinear Programming," 2nd ed., *Athena Scientific* 1999.

[7] S. Boyd, L. Xiao, A. Mutapcic, "Subgradient Methods," Notes for EE392o, *Stanford University* 2003. `http://www.stanford.edu/class/ee392o/subgrad_method.pdf`

[8] T. F. Chan, J. Cong, E. Radke, "A Rigorous Framework for Convergent Net-weighting Schemes in Timing-driven Placement," *ICCAD* 2009, pp. 288-294.

[9] T. F. Chan, J. Cong, J. Shinnerl, K. Sze, M. Xie, "mPL6: Enhanced Multilevel Mixed-Size Placement," *ISPD* 2006, pp. 212-214.

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

[11] T.-C. Chen et al.,"MP-trees: A Packing-based Macro Placement Algorithm for Mixed-size Designs," *IEEE TCAD* 27(9) 2008, pp. 1621-1634.

[12] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, Y.-W. Chang, "NTUPlace3: An Analytical Placer for Large-Scale Mixed-Size Designs With Preplaced Blocks and Density Constraints," *IEEE TCAD* 27(7) 2008, pp.1228-1240.

[13] Y.-L. Chuang et al., "Design-hierarchy Aware Mixed-size Placement for Routability Optimization," *ICCAD* 2010, pp. 663-668.

[14] J. Cong, M. Romesis, J. Shinnerl, "Robust Mixed-Size Placement Under Tight White-Space Constraints," *ICCAD* 2005, pp. 165-172.

[15] P. E. Gill, D. P. Robinson, "A Primal-Dual Augmented Lagrangian," *Computational Optimization and Applications* 2010, DOI: 10.1007/s10589-010-9339-1.

[16] K. C. Kiwiel, T. Larsson, P. O. Lindberg,"Lagrangian Relaxation via Ballstep Subgradient Methods,"*Mathematics of Operations Research* 32(3), 2007, pp. 669-686. `http://mor.journal.informs.org/content/32/3/669`

[17] C. Li, C.-K. Koh, "Recursive Function Smoothing of Half-Perimeter Wirelength for Analytical Placement," *ISQED* 2007, pp. 829-834.

[18] X. He, T. Huang, L. Xiao, H. Tian, G. Cui, E. F. Y Young, "Ripple: An Effective Routability-Driven Placer by Iterative Cell Movement," *ICCAD* 2011, pp. 74-79.

[19] M.-K. Hsu, Y.-W. Chang, V. Balabanov, "TSV-aware Analytical Placement for 3D IC Designs," *DAC*'11, pp. 664-669.

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

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

[22] A. B. Kahng, J. Lienig, I. L. Markov, J. Hu, "VLSI Physical Design: from Graph Partitioning to Timing Closure," Springer 2011, 312 pages.

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

[24] M.-C. Kim, J. Hu, D.-J. Lee, I. L. Markov, "A SimPLR method for Routability-driven Placement" *ICCAD* 2011.

[25] D.-J. Lee, I. L. Markov, "Obstacle-Aware Clock-tree Shaping During Placement," *ISPD* 2011, pp. 123-130.

[26] G.-J. Nam, J. Cong, "Modern Circuit Placement: Best Practices and Results," *Springer* 2007.
[27] A. N. Ng et al., "Solving Hard Instances of Floorplacement," *ISPD 2006*, pp. 78-85.
[28] M. Pan, N. Viswanathan, C. Chu, "An Efficient & Effective Detailed Placement Algorithm," *ICCAD* 2005, pp. 48-55.
[29] A. E. Ruehli, P. K. Wolff, and G. Goertzel, "Analytical Power/Timing Optimization Technique for Digital Systems," *DAC* 1977, pp. 142-146.
[30] G.Sigl, K.Doll, F.Johannes,"Analytical Placement:A Linear or a Quadratic Objective Function?" *DAC*'91, pp. 427-432.
[31] 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.
[32] N. Viswanathan, M. Pan, C. Chu, "FastPlace 3.0: A Fast Multilevel Quadratic Placement Algorithm with Placement Congestion Control," *ASPDAC* 2007, pp. 135-140.
[33] N. Viswanathan et al., "RQL: Global Placement via Relaxed Quadratic Spreading and Linearization," *DAC* 2007, pp. 453-458.
[34] S. I. Ward et al., "Quantifying Academic Placer Performance on Custom Designs, " *ISPD* 2011, pp. 91-98.
[35] J. Z. Yan et al., "Handling Complexities in Modern Large-scale Mixed-size Circuit Designs," *DAC* 2009.

S1. APPROXIMATIONS OF INTERCONNECT OBJECTIVE FUNCTIONS

Alternatives to quadratic approximations to the HPWL objective (Section 2) include the β-regularization [4]

$$\sqrt{(x_i - x_j)^2 - \beta} \rightarrow |x_i - x_j|, \quad \beta \rightarrow 0,$$

the p, β-regularization for a net $e \in \mathcal{N}$, with $p \rightarrow \infty$ [21]

$$\Sigma_{i,j \in e}(|x_i - x_j|^p + \beta)^{1/p} \rightarrow \max_{i,j \in e} |x_i - x_j| = [\max_{i \in e} x_i - \min_{i \in e} x_i]$$

and the log-sum-exp technique with $\gamma \rightarrow 0$ [29]

$$\gamma \log \Sigma_{k \in e} \big(\exp(x_k/\gamma) + \exp(-x_k/\gamma) \big) \rightarrow [\max_{i \in e} x_i - \min_{i \in e} x_i]$$

Other such techniques are surveyed and compared in [17, 19]. Any one of these approximations can be used in ComPLx.

S2. ADDITIONAL DISCUSSION OF THE FEASIBILITY PROJECTION in ComPLx

The feasibility projection $P_{\mathcal{C}}$ is defined in this work for a variety of placement constraints $\mathcal{C}$ and illustrated for (a) density constraints, (b) region constraints. The former is related to look-ahead legalization (LAL) in the original SimPL placer [23] and routability-driven variants in SimPLR and Ripple.

Comparision to prior work. Look-ahead legalization (LAL) was earlier used for macro placement in PolarBear [14] and SCAMPI [27]. In both cases, the main issue was the feasibility of macro packing within a given fixed outline. Both algorithms use top-down min-cut partitioning to minimize interconnect and need to check if each geometric partition is feasible. The result of this check is binary — a positive result for both partitions allows top-down partitioning to proceed, while a negative result for one of partitions triggers backtracking or end-case processing. The locations of macro blocks in a feasible placement are typically not used directly, therefore the LAL algorithm in PolarBear does not seek to optimize any objective. In contrast, SimPL does not deal with movable macros, and the main concern for LAL in SimPL is to minimize total displacement from an initial solution, which has not been considered in PolarBear and SCAMPI. Since relevant algorithms in PolarBear and SCAMPI do not work with an initial solution, they cannot be considered *feasibility projections*. In other words, $P_{\mathcal{C}}$ in ComPLx and LAL in SimPL differ from prior work in that they pursue different goals — finding a closest feasible solution, rather than check packing feasibility. They are used in a different context (analytic placement vs. top-down min-cut placement), employ entirely different algorithms, and their results are interpretted differently (as anchors that influence the next iteration of analytic global placement). Whereas PolarBear and SCAMPI did not anticipate primal-dual Lagrange optimization in placement, the feasibility projection in ComPLx is a key element of the proposed primal-dual Lagrange formulation.

ComPLx also differs from PolarBear and SCAMPI in how it handles movable macros. Whereas prior work seeks to ensure the feasibility of macro placements at every step and sometimes sacrifices interconnect optimization for this, ComPLx pursues a different strategy based on macro shredding — it allows for temporarily overlapping macro placements and focuses on interconnect optimization.

Implementation details and analysis of properties. As pointed out in Section 5, the feasibility projection in ComPLx *generalizes* look-ahead legalization (LAL) in SimPL, SimPLR and Ripple to deal with macros and broader placement constraints. Here we restructure LAL so as to check its convexity and self-consistency.

Whereas LAL was defined recursively in [23, Section 4], the top-level structure of $P_{\mathcal{C}}$ in our description is that of alternating horizontal and vertical *spreading* passes. Each pass operates over a slicing floorpan, which gets refined between the passes. Specifically, spreading occurs only inside the rooms of the floorplan. For example, at the very first iteration, there is only one room, and one-dimensional spreading evens out the density. To formalize the problem solved by one-dimensional spreading from [23, Section 4], we note that relative placements are preserved. This justifies a change of variables: initial cell locations x_i (or y_i) are sorted, and the new variables $\delta_i \geq 0$ represent distances between neighboring x (y) locations, subject to $\Sigma_i \delta_i \leq W_x$ (or W_y) for a $W_x \times W_y$ floorplan room (a convex constraint). This linear change of variables preserves the convexity of the optimization objective (L_1-distance from given locations). The density constraint requires that for (some m and) all k, $\Sigma_{i=k}^{k+m} \delta_i$ is sufficiently large (based on cell sizes). This constraint

$$\min_k \{\Sigma_{i=k}^{k+m} \delta_i - (1/\sqrt{\gamma}) \Sigma_{i=k}^{k+m} \text{width}(\text{cell}_i)\} \geq 0$$

is convex since the minimum of downward convex (linear) functions is also downward convex (the $\geq$ sign is important).

As pointed out in [23, Section 4], after one-dimensional spreading, the median location should divide cell area evenly. Since this equalizes average densities on both sides, this new median location indicates a fixed point of the spreading transform. Hence, the walls of the slicing floorplan built by alternating one-dimensional spreading steps represent fixed lines. As slicing floorplan is gradually refined, the displacement affected by later steps of $P_{\mathcal{C}}$ rapidly decreases.

Self-consistency (Formula 11). To establish the self-consistency of $P_{\mathcal{C}}$ it suffices to independently establish the self-consistency of each horizontal and vertical pass [23, Section 4] in each room of the floorplan (see below). While the overall algorithm described above using alternating passes differs from LAL in [23], the results produced are essentially the same. The argument for self-consistency remains valid when the algorithm is applied multiple times.

The self-consistency of our $P_{\mathcal{C}}$ seems related to convexity, but in this work we only test it empirically on ISPD 2005 and 2006 benchmarks. Since the self-consistency condition of Formula 11 is transitive, we checked it between every two consecutive ComPLx iterations. Our implementation of the approximate feasibility projection $P_{\mathcal{C}}$ was self-consistent 96.0% and inconsistent 0.6% of the time, while the sufficient condition $||(\vec{x},\vec{y}) - P_{\mathcal{C}}(\vec{x},\vec{y})||_1 > ||(\vec{x}',\vec{y}') - P_{\mathcal{C}}(\vec{x},\vec{y})||_1$ for successive iterations was not satisfied only 3.3% of the time. Thus, the approximate feasibility projection $P_{\mathcal{C}}$ used by our implementation is approximately self-consistent. The convergence plots in Figure 1 do not show any disruptions that one would expect with a seriously inconsistent $P_{\mathcal{C}}$. Inconsistencies mostly occur in the early global placement iterations (< 5) where projected feasible placements can differ significantly between consecutive iterations.

S3. THE SCALABILITY OF COMPLX

Figure 3 plots the final values of λ (solid red line) and the number of global placement iterations of ComPLx (dotted blue line with a greater range) on ISPD 2005 and 2006 benchmarks. The number of iterations correlates with the final λ value because each iteration increases λ by a limited amount until the final value is reached. All final values in our experiments are well below 1.0, and the iteration counts do not grow systematically with the size of the input. This phenomenon is consistent with the rapid convergence for which primal-dual Lagrange optimization is known. Given that ComPLx spends near-linear time $O(n(\log n)^p)$ per iteration [23], the overall runtime is near-linear as well. In comparison, the runtime of FastPlace is estimated as $\Theta(n^{1.38})$.

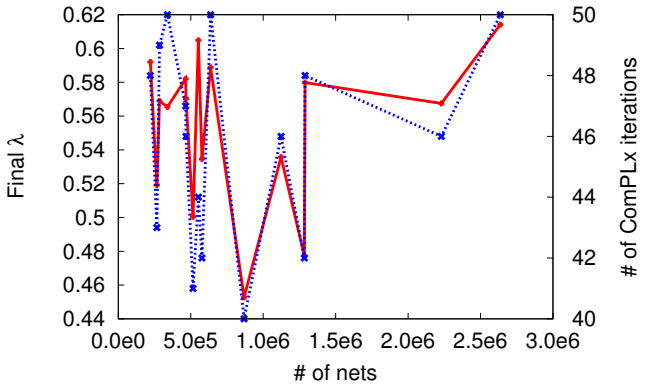

Figure 3: The final λ and total number of ComPLx iterations performed, against the number of nets.

S4. COMPARISONS TO RELATED PRIMAL-DUAL LAGRANGIAN OPTIMIZATIONS

Primal-dual optimization was used once in global placement in [4], where it was limited to explicit center-of-gravity (CoG) "spreading" constraints. These constraints appear in GORDIAN and GORDIAN-L algorithms [30], but not in modern placers — being convex and linear, they are insufficient to handle modern IC layouts (the 1997 implementation reported in [4] is not a full-fledged global placer). To deal with CoG constraints, [4] introduced slack variables, as is common in linearly-constrained primal-dual Lagrange optimization [15]. Instead, we deal with more general nonlinear, nonconvex constraints (such as fixed obstacles) by means of *approximate projected subgradient optimization*. Our primal-dual Lagrangian relaxation, in its basic form, requires only a single real-valued multiplier, making optimization very efficient. Unlike in [4, 15], we use the linear Conjugate Gradient method rather than the non-linear Newton's method.

Recent *operations research* work (unrelated to EDA) by Kiwiel et al [16] discusses *approximate subgradient projected optimization*, focusing on step-size selection and convergence analysis. Unlike prior projected subgradient methods, the Lagrangian relaxation in [16] finds both primal and dual solutions. This is also a key feature of our methods. However, [16] is not solving global placement and lacks numerous domain-specific details we described. *Vice versa*, we are not using their hallmark *ballstep strategy* that bundles multiple subgradient iterations.

S5. HANDLING REGION CONSTRAINTS IN THE FEASIBILITY PROJECTION

Chip designers often impose a region constraint on a subset of cells to express logic hierarchy and clock domains, to keep clock sinks close to clock drivers, or to assist the placer in dealing with challenging critical paths. Traditional placers convert the hard region constraints to soft constraints, which can be addressed by heavily-weighted fake nets [1, Figure 5] or modification of the objective function [13]. While ComPLx supports such techniques, it also allows for a more straightforward and robust implementation of region constraints by enforcing them as part of the feasibility projection at every global placement iteration – each cell is *snapped to* the constraining region after feasibility projection for density constraints. Figure 4 illustrates the enforcement of

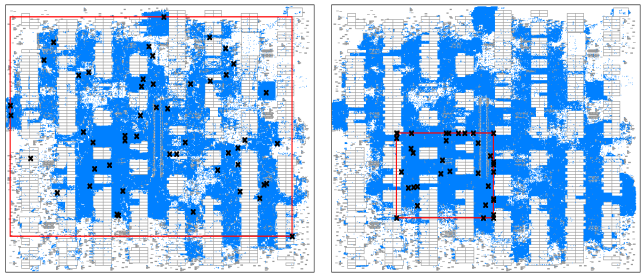

Figure 4: A hard region constraint imposed on 50 cells that were initially placed unconstrained (left). The resulting ComPLx placement (right) satisfies the constraint. HPWL drops from 143.55 to 142.70.

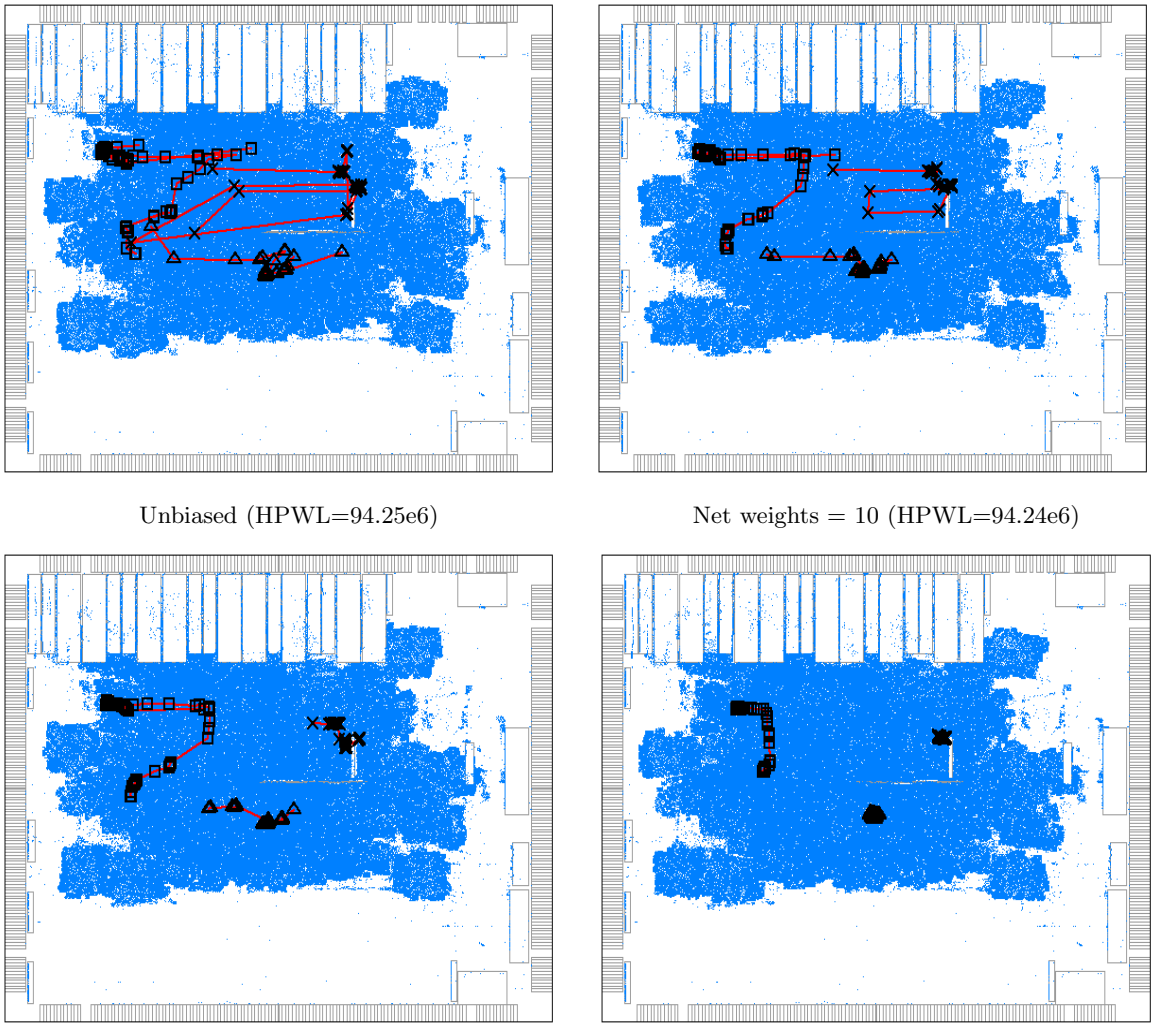

Figure 5: In a ComPLx placement of BIGBLUE1 (upper left), three critical signal paths between registers are chosen. Subsequent ComPLx runs are performed with progressively larger net weights on those paths, which straightens the paths and reduces their lengths. Legal HPWL values are reported in parentheses.

region constraints by "before" and "after" pictures. The locations from the modified feasibility projection are then used as anchors to influence the subsequent iteration of analytic global placement. Rather than degrade, HPWL actually improves — a surprising phenomenon often observed with industry placers.

S6. HANDLING TIMING-CRITICAL NETS

To demonstrate effective timing optimization, we show that timing-critical paths can be shortened and straightened by manipulating net weights without adverse effects on total HPWL. Working with the standard benchmark BIGBLUE1, we performed 30 global iterations to obtain an unbiased, stable intermediate placement that allowed us to estimate net lengths. We then selected several critical paths, increased the weights of nets comprising these paths, and ran our placer to completion in three configurations with different net weights. Figure 5 shows that the desired outcome was achieved. With sufficiently large net weights, selected paths notably shrunk. Given that only a small fraction of net weights were modified, the overall placement and its wirelength were largely unaffected. Essential for these results was our scheduling of λ in Formula 12. While not a full-fledged demonstration of timing-driven placement, this experiment confirms that our proposed core placement algorithm is capable of controlling critical paths without tangible overhead in HPWL. Specific formulas for provably-good timing-driven net weighting can be found in [8]. As a side-effect, this experiment also demonstrates the *stability* of ComPLx to small netlist changes, which is important in the context of physical synthesis [1].