Placement Initialization via Sequential Subspace Optimization with Sphere Constraints

Pengwen Chen
pengwen@email.nchu.edu.tw
National Chung Hsing University

Chung-Kuan Cheng
ckcheng@eng.ucsd.edu
University of California San Diego

Albert Chern
alchern@eng.ucsd.edu
University of California San Diego

Chester Holtz*
chholtz@eng.ucsd.edu
University of California San Diego

Aoxi Li
aoli@ucsd.edu
University of California San Diego

Yucheng Wang
yuw132@ucsd.edu
University of California San Diego

ABSTRACT

State-of-the-art analytical placement algorithms for VLSI designs rely on solving nonlinear programs to minimize wirelength and cell congestion. As a consequence, the quality of solutions produced using these algorithms crucially depends on the initial cell coordinates. In this work, we reduce the problem of finding wirelength-minimal initial layouts subject to density and fixed-macro constraints to a Quadratically Constrained Quadratic Program (QCQP). We additionally propose an efficient sequential quadratic programming algorithm to recover a block-globally optimal solution and a subspace method to reduce the complexity of problem. We extend our formulation to facilitate direct minimization of the Half-Perimeter Wirelength (HPWL) by showing that a corresponding solution can be derived by solving a sequence of reweighted quadratic programs. Critically, our method is *parameter-free*, i.e. involves no hyperparameters to tune. We demonstrate that incorporating initial layouts produced by our algorithm with a global analytical placer results in improvements of up to 4.76% in post-detailed-placement wirelength on the ISPD'05 benchmark suite. Our code is available on github[1].

CCS CONCEPTS

• **Hardware → Very large scale integration design**.

KEYWORDS

global placement, VLSI, optimization

ACM Reference Format:
Pengwen Chen, Chung-Kuan Cheng, Albert Chern, Chester Holtz, Aoxi Li, and Yucheng Wang. 2023. Placement Initialization via Sequential Subspace Optimization with Sphere Constraints. In *Proceedings of the 2023 International Symposium on Physical Design (ISPD '23), March 26–29, 2023, Virtual Event, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3569052.3571877

*Corresponding author

[1]https://github.com/cholt95/laplacian-eigenmaps-revisited

ISPD '23, March 26–29, 2023, Virtual Event, USA

ACM ISBN 978-1-4503-9978-4/23/03.
https://doi.org/10.1145/3569052.3571877

1 INTRODUCTION

Given a circuit and a region, the placement problem is to assign each circuit module to a specific location in the region. Most state-of-the-art layout algorithms for large-scale VLSI placement rely on solving non-linear problems using iterative first-order optimization algorithms [7, 16–19]. As a consequence, there are typically few guarantees regarding the convergence of these methods to optimal, or even good, coordinate assignments in a limited time frame and initialization of the variables plays a critical role [18]. Despite the importance of initialization, existing methods for placement initialization are primarily based on naive heuristics—including minimizing wirelength without second-order constraints [16, 18, 19], uniformly assigning the cell coordinates to the origin, or assigning coordinates to small random values [12, 17].

In this work, we address the following question:

> *Is it possible to improve upon random initialization for large-scale placement engines?*

We investigate a novel fixed node-aware formulation and describe an efficient algorithm to solve it. More concretely, we formulate initialization as a Quadratically Constrained Quadratic Optimization Problem (QCQP) with sphere constraints. Our formulation is aware of fixed nodes via a decomposition of the netlist-graph. Although the QCQP is non-convex, we propose an algorithm that can recover local and block-globally optimal (under certain assumptions) solutions. We validate our technique by demonstrating scalability and convergence to superior post-detailed placement solutions compared to min-wirelength and random initializations using an open source placement flow [12, 17]. Furthermore, we propose a statistical test to quantify the preservation of local structures derived from the initialization through global placement.

1.1 Contributions

Our contributions are summarized below.

(1) We introduce a novel formulation of global placement initialization as a sphere-constrained quadratic programming problem, an extension of a classic Rayleigh Quotient problem [13] and devise a novel algorithm to solve it.
(2) We propose a way to exploit the structure of the QCQP to improve the efficiency of optimization by iteratively solving the problem in a sequence of carefully chosen subspaces.
(3) We adapt our approach via iterative reweighting to facilitate direct minimization of Half-Perimeter Wirelength (HPWL).

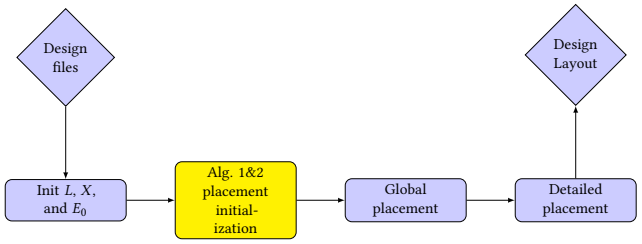


Figure 1: Placement flow. Our proposed method is a "placement initialization" stage, highlighted in yellow.

(4) We perform a comparison between various initialization schemes for analytic placement with fixed macros.

In Sec. 2, we introduce the global placement problem and a general framework for generating initial layouts (denoted in yellow in Fig. 1). In Sec. 3, we introduce our technique for solving the proposed QCQP and an extension (iterative reweighting) to facilitate direct minimization of HPWL (denoted in Fig. 3). In Sec. 4, we validate our approach on a standard set of benchmarks. In Sec. 5, we conclude and highlight potential avenues for future research.

2 PRELIMINARIES

Number of components	$n, n_{\text{free}}, n_{\text{fixed}} \in \mathbb{R}_+$
Placement coordinates	$x, y \in \mathbb{R}^n$
Adjacency, Degree, & Laplacian matrices	$A, D, L \in \mathbb{R}^{n\times n}$
Linear offset terms	$b, d \in \mathbb{R}^n, E_0 = [b : d] \in \mathbb{R}^{n\times 2}$
Cell volumes	$v \in \mathbb{R}^n_+, G = \text{diag}(v) \in \mathbb{R}^{n\times n}_+$
Cell area constraints	$c_i, i \in \{1, 2, 3, 4, 5\} \in \mathbb{R}_+$
Lagrange multipliers	$\Lambda \in \mathbb{R}^{2\times 2}$
Newton update direction	$Z \in \mathbb{R}^{n\times 2}$

Figure 2: Notation

Let $x, y \in \mathbb{R}^n$ be vectors corresponding to the coordinates of n components such that the i-th component has coordinates encoded in the i-th row of $[x : y]$; $[x : y]_i$. We aim to assign coordinates so that the resulting layout has small cumulative wirelength.

2.1 Global analytical placement

Conventional global placement strategies minimize wirelength subject to density constraints. Density constraints are usually integrated into the objective to yield an unconstrained relaxation [7, 17]:

$$\min_{x,y}(\textstyle\sum_{e\in\mathcal{E}} Wl(e; x, y) + \lambda\mathcal{D}(x, y)) \tag{1}$$

where $\mathcal{E}$ denotes a set of given nets and $Wl(\cdot;\cdot)$ is a function that takes a net instance e as input and returns the cumulative wirelength and $\mathcal{D}(\cdot)$ is a density penalty. In the context of VLSI placement, the wirelength of a net is commonly modelled with its Half-Perimeter Wirelength (HPWL) or a smooth alternative and $\mathcal{D}$ is a smooth density penalty [18].

A typical approach is to represent individual nets as rectangles and to minimize the sum-perimeters over all nets. Repulsion is often applied between overlapping nodes to reduce density. For example, [17] adopt the smooth and differentiable weighted-average wirelength (WL) model for the wirelength cost [15]. The horizontal net-wirelength for net e is given by

$$Wl_x^{(e)} = \frac{\sum_{i\in e} x_i \exp\left(\frac{x_i}{c}\right)}{\sum_{i\in e} \exp\left(\frac{x_i}{c}\right)} - \frac{\sum_{i\in e} x_i \exp\left(-\frac{x_i}{c}\right)}{\sum_{i\in e} \exp\left(-\frac{x_i}{c}\right)}$$

where c is a parameter that controls the smoothness and approximation error with respect to the HPWL of net e (i.e. $|x_i - x_j|$ for two-pin net $e = (i, j)$). The wirelength of e is:

$$Wl(e; x, y) = Wl_x^{(e)} + Wl_y^{(e)}$$

To model the density term, the placement area is divided into B bins, and the placer seeks to equalize the overlap at each bin via an analogy to an electrostatic system, with cells being modeled as charges, density penalty modeled as potential energy, and density gradient modeled as the electric field.

Overlap constraints are satisfied over the placement process by gradually increasing λ, usually at the cost of increased wirelength. Current state-of-the-art VLSI placement algorithms [7, 17, 18] solve Problem 1 in this manner.

2.2 QCQP-based layouts

In this section, we describe a basic formulation for spectral layouts for global pre-placements, or initializations. Additionally, we more generally motivate our adoption of the QCQP framework for global placement initialization.

2.2.1 Formulation. The VLSI placement problem is reduced to a graph layout problem by first collapsing the netlist hypergraph to a component graph via various models (e.g. clique, star, etc.) [23]. A matrix-representation of the graph connectivity—the *graph Laplacian* is then derived. The solution to the associated eigenvalue problem approximates the solution to the sparsest cut problem [2, 14], and clusters arising out of the vertex-projection into the space spanned by the first nontrivial eigenvalues correspond highly connected components of the graph.

More concretely, we solve a variant of the following problem where x and y are cell coordinates, c_i are constants, v is a vector of cell areas, and L is the graph Laplacian; $L = D - A$, where A is a (weighted) adjacency matrix, and D is the associated degree matrix.

$$\begin{aligned} \min_{x,y}\ & x^\top Lx + y^\top Ly \quad \text{s.t. } v^\top x = 0, \qquad v^\top y = 0, \\ & x^\top Gx = c_1, \qquad y^\top Gy = c_2, \qquad x^\top Gy = c_3 \end{aligned} \tag{2}$$

Typically, $G = \text{diag}(v)$. In general, one can recover a reduction to the case $G = I$ via the normalization $[x, y] \leftarrow G^{1/2}[x, y]$, $L \leftarrow G^{-1/2}LG^{-1/2}$ and $[v, b, d] \leftarrow G^{-1/2}[v, b, d]$. Intuitively, the objective is to minimize the weighted squared wirelength of a 2D placement. The linear constraints characterize an origin (i.e. remove translational invariance) and the quadratic constraints spread the layout evenly over the x and y axes (i.e. ensure that the embedding has nonzero constant variance).

2.2.2 Fixed node constraints. Many layouts involve constraints on a subset of the cells—typically large macros and primary input/output pads. We show how such fixed node constraints naturally lead to a decomposition of the x, y and L terms in Eq. 2. We denote the coordinates of the fixed nodes x_1, y_1. Likewise, let the movable nodes be x_2, y_2 Then, we can express x, y, and L and the parameters

v and G in terms of these indices: $L = \begin{bmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{bmatrix}$, with $x_1 \in \mathbb{R}^{n_{\text{fixed}}}$, $x_2 \in \mathbb{R}^{n_{\text{free}}}$, $L_{22} \in \mathbb{R}^{n_{\text{free}} \times n_{\text{free}}}$, and $L_{12} \in \mathbb{R}^{n_{\text{free}} \times n_{\text{fixed}}}$. $x = [x_1, x_2]^\top$ (likewise for y). By considering fixed-node terms (i.e. x_1 and y_1) as constants, Problem 2 may be re-written (ignoring constants):

$$\begin{aligned} \min_{x_2, y_2} & x_2^\top L_{22} x_2 + y_2^\top L_{22} y_2 + \quad 2b^\top x_2 + 2d^\top y_2 \\ & \text{s.t. } v_2^\top x_2 = c_1', \qquad v_2^\top y_2 = c_2', \\ & x_2^\top x_2 = c_3', \qquad y_2^\top y_2 = c_4', \qquad x_2^\top y_2 = c_5' \end{aligned} \tag{3}$$

with $b = L_{12} x_1$, $d = L_{12} y_1$. In Section 3.1, we derive c_i', $i = 1, 2, 3, 4, 5$.

2.2.3 Motivation and high-level flow. Our motivation for expressing initializations to Prob. 1 with Prob. 2—i.e. as a QCQP with sphere constraints—is derived from two observations assuming graph-models of netlists: (1.) if the $Wl(\cdot)$ corresponds to the *squared* wirelength, its minimization is equivalent to minimizing a quadratic form defined on a graph Laplacian. And if the $Wl(\cdot)$ corresponds to the *half-perimeter* wirelength, its minimization may be expressed as a sequence of quadratic problems of the same form as Prob. 2. For example, we describe such a method in Sec. 3. (2.) a quadratic equality—a *sphere* —constraint implies constant variance and satisfaction of density-constraints assuming a uniform grid.

We highlight the high-level flow of our framework in Fig. 3:

(1) **Eigenvector initialization:** The eigenvectors of L_{22}, which correspond to the minimum-squared wirelength solution are computed (Sec. 3.1, Eq. 4).
(2) **Eigenvector projection:** These eigenvectors are projected to satisfy the linear and quadratic constraints (Sec. 3.1, Prop. 2).
(3) **Eigenvector rotation:** An orthogonal transform is applied to the projected eigenvectors to minimize the distance between free and fixed components (Sec. 3.1, Prop. 3).
(4) **Sequential subspace method (SSM):** From these coordinates, an iterative projected-subspace-descent algorithm is applied which results in convergence to a local / block-globally optimal solution (Sec. 3.2, Sec. 3.3).
(5) **Iterative net reweighting:** During iterative descent, L is adjusted (*reweighted*) in order to find a min-HPWL coordinate assignment (Sec. 3.4).

In the following section, we describe a sequential subspace method for solving Prob. 3. We then show that one can easily adapt this method to facilitate direct minimization of HPWL.

3 CONSTRAINED SPECTRAL LAYOUTS

In this section, we describe a method to compute the spectrum of the matrix L_{22}. The eigenvectors corresponding to the smallest nontrivial eigenvalues are then projected and transformed to be used as a candidate solution to Prob. 3 and iteratively improved.

3.1 Eigenvector method and projection

We start by re-writing the objective defined in Eq. 3 (for brevity, writing L_{22} as "L" and v_2 as v). Let $X_2 = [x_2, y_2]$ and $E_0 = [b, d] \in \mathbb{R}^{n_{\text{free}} \times 2}$ and $X_1 = [x_1, y_1] \in \mathbb{R}^{n_{\text{fixed}} \times 2}$.

Let $[c_1', c_2']^\top = -r$, where $r := (v_1^\top X_1)^\top$. To eliminate the linear constraint $v^\top X_2 = -r^\top$, we introduce two adjustments: first, let $(X)_i = (X_2)_i + \frac{1}{w} r^\top$ denote a row-wise centering transformation

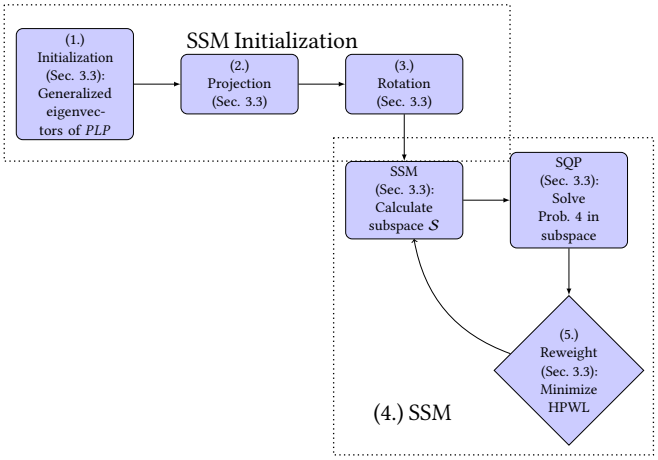


Figure 3: QCQP placement initialization with reweighting.

with respect to the fixed nodes, where w is a scale factor proportional to v_2. This yields the constraint $v^\top X = 0$ and implies the quadratic constraint

$$C = \begin{bmatrix} c_3' & c_5' \\ c_5' & c_4' \end{bmatrix} = \begin{bmatrix} c_1 & c_3 \\ c_3 & c_2 \end{bmatrix} - X_1^\top X_1 - \frac{1}{w} r r^\top.$$

Second, assuming that v is normalized to be a unit vector, let $P = I - vv^\top$ be the projection onto the subspace orthogonal to vector $v \in \mathbb{R}^{n_{\text{free}}}$, i.e., $v^\top (PX) = [0, 0]$. Without loss of generality, replacing E_0 with $P(E_0 - L \frac{1}{n_{\text{free}}} \mathbf{1} r^\top)$, we have $v^\top E_0 = [0, 0]$.

$$\min_X \{F(X) = \text{tr}(X^\top (PLPX + 2E_0))\} \tag{4}$$

subject to $X^\top X = C$, We first address this problem in three stages: (1.) by first solving the canonical eigenvalue problem $\min_X \text{tr}(X^\top PLPX)$ subject to the constraint $X^\top X = I$, (2.) invoking a projection to resolve the second order constraint $X^\top X = C$, (3.) appropriately transforming the solution so that $2X^\top PE_0$ is reduced.

Stage 1. First, the eigenvectors of PLP corresponding to the two smallest nontrivial eigenvalues are computed. While L may be extremely sparse, facilitating the application of sparse eigenvector algorithms, PLP may be prohibitively dense. To address this, we adapt the *Rayleigh Quotient Iteration* method which exclusively relies on matrix-vector multiplications, and the computation of $L^{-1}u$ which can be done efficiently using iterative methods (e.g. Conjugate Gradient). At a high level, the method proceeds by repeating the following updates on vectors u:

$$u_{k-1} = Pr_{k-1} / ||Pr_{k-1}|| \tag{5}$$
$$\text{Solve } Pr_k \text{ from } PLPr_k = u_{k-1} \tag{6}$$

To solve for r_k in the second part, we state the following result, which eliminates the need to compute $(PLP)^{-1}$:

Proposition 1 (Psudo-inverse of $PL^{-1}P$). *Let $P = I - vv^\top$. Let $r_k = \{PL^{-1}P\}^\dagger u_{k-1}$. Then, derive projection of r_k, i.e.*

$$Pr_k = \{PL^{-1}P\}^\dagger u_{k-1} = (I - \frac{\bar{v}^\top v}{v^\top \bar{v}}) L^{-1} u_{k-1}, \quad \bar{v} = L^{-1} v$$

Once the smallest eigenvector w is obtained, we can proceed to compute the eigenvector of PLP corresponding to the subsequent eigenvalue using the same approach with a minor adjustment: replacing $P = I - vv^\top$ with $P = I - vv^\top - ww^\top$.

Stage 2. Given a set of candidate coordinates, we apply the projection $[]_+$ to resolve the quadratic constraints according to the following proposition:

Proposition 2 (Projection). *Let X_1 be an intermediate solution and $C_1 := X_1^\top X_1$ and $C \succ 0$.*

$$\begin{aligned}\textit{The projection of } X_1,\ [X_1]_+ := \arg\min_X \{F(X) &= ||X - X_1||_F^2 \\ &= tr(C) + tr(C_1) - 2\max\langle X, X_1\rangle\}\end{aligned}$$

s.t. $X^\top X = C$. Take the Singular Value Decomposition (SVD) of $C^{1/2}C_1^{1/2}$, $U\Sigma V^\top = C^{1/2}C_1^{1/2}$.
Then the minimizer $X = [X_1]_+$ is given by

$$X = X_1 C_1^{-1/2} U V^\top C^{1/2} \tag{7}$$

Stage 3. We apply an orthogonal transformation (i.e. a rotation / reflection) which preserves the eigenvector structure while minimizing the euclidean distance between fixed pins and free cells.

Proposition 3 (Orthogonal transform of X). *Assume $C = I$. Note the first term of F satisfies the invariance $tr(X^\top PLPX) = tr(\tilde{X}^\top PLP\tilde{X})$, where $\tilde{X} = XQ$ for any orthogonal $Q \in \mathbb{R}^{2\times 2}$. X is a local minimizer if $-X^\top E_0 \succcurlyeq 0$ and symmetric. Take the SVD of $X^\top E_0 = U_E D_E V_E^\top$. Let $Q = -U_E V_E^\top$. Then, $tr((XQ)^\top E_0) = -tr(D_E) \le 0$.*

3.2 Sequential Quadratic Programming method

In this section, we introduce SQP, a key component of SSM. The framework of SQP is applied to iteratively compute search directions to improve the projected and transformed eigenvectors with respect to the quadratic objective while maintaining satisfication of all constraints (Problem 4). We define the Lagrangian of Problem 4 by introducing multipliers $\Lambda \in \mathbb{R}^{2\times 2}$.

$$\mathcal{L}(X, \Lambda) = \langle X, PLPX + 2E_0\rangle + \langle \Lambda, X^\top X - C\rangle \tag{8}$$

The derivative of the Lagrangian characterizes the first order conditions (FOC) satisfied by an optimal X:

$$PLPX = -E_0 - X\Lambda, \quad X^\top X = C \tag{9}$$

To find a solution, we derive Newton directions Δ and Z associated with Λ and X. Following the principal of SQP, $X \leftarrow [X + \alpha Z]_+$ and $\Lambda \leftarrow \Lambda + \alpha\Delta$ according to the linearization of the FOC:

$$\begin{aligned}&(PLPZ + Z\Lambda) + X\Delta = E := -E_0 - (PLPX + X\Lambda)\\ &X^\top Z = 0\end{aligned}$$

Proposition 4 (Newton Direction of the Lagrangian Eq. 8). *Assume Λ symmetric and $PZ = Z$, i.e., $v^\top Z = v^\top X = [0, 0]$. The solution (Z, Δ) is*

$$(\Delta)_j = (X^T(L + W_jI)^{-1}X)^{-1}X^T(L + W_jI)^{-1}E_j \tag{10}$$

$$Z_j = P(L + W_jI)^{-1}P\{-X(\Delta)_j + E_j\} \tag{11}$$

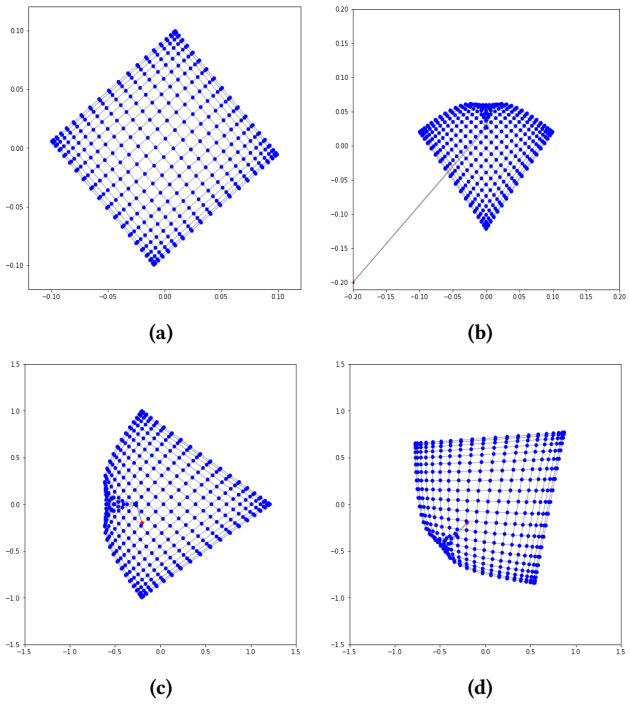


Figure 4: Eigenvector method and projection. (a): Eigenvectors of full Laplacian L (b): Eigenvectors of reduced Laplacian L, ignorant of fixed node (denoted in red) (c): Projected eigenvectors of L (Prop. 1) (note the axis scale). (d): Orthogonal transform applied to projected eigenvectors (Prop. 2).

where W is given by the eigenvector decomposition of Λ: $\Lambda = UWU^{-1}$, $W = diag(W_1, W_2)$ for two eigenvalues W_1, W_2 of Λ.

Algorithm 1 SQP Update

Input: Partial Laplacian L, linear objective term E_0, intermediate solution X, intermediate Lagrangian multipliers Λ
Output: $j-th$ columns of Newton updates—Δ_j, Z_j

function SQP(L, Λ, E_0, X, v)
  $W \leftarrow$ eigvals(Λ)
  $LPX \leftarrow L(X - v(v^\top X))$
  $PLPX \leftarrow LPX - v(v^\top LPX)$
  $E \leftarrow -E_0 - (LPX + XL)$
  $L_{W_j} \leftarrow L + W_jI$
  $\Delta_j \leftarrow (X^\top L_{W_j}^{-1}X)^{-1}L_{W_j}^{-1}E$ ▷ Eq. 10
  $T \leftarrow -X\Delta_j + E$
  $RHS \leftarrow T - (Tv)v^\top$
  $Z_j \leftarrow L_{W_j}^{-1}RHS - v(v^\top L_{W_j}^{-1}RHS)$ ▷ Eq. 11
  return Z_j, Δ_j
end function

Applying the projection operation $[]_+$ onto the manifold $X^\top X = C$, we generate $\{X = X_k, k = 1, 2, 3, \ldots\}$ $X_{k+1} = [X_k + \alpha Z]_+$ where α is chosen to decrease the cost.

Alg. 1 presents the detailed steps involved in the computation of the Newton directions, i.e. Eqs. 10, & 11 defined in Prop. 4. In Sec. 3.5, we refer to Alg. 1 in the context of evaluating computational cost.

3.3 Sequential subspace optimization

We introduce a Sequential Subspace Method (SSM) in Alg. 2 to address the scalability of SQP. Inspired by the 1-dimensional algorithm of [13], instead of solving Problem 4 directly, we instead solve a sequence of quadratic programs in subspaces of much smaller dimension relative to the size of the graph.

Despite the sparsity of L, repeatedly computing inverse-vector products involving $L + W_j I$ in Eq. 10 and Eq. 11 may computationally bottleneck the proposed method for large benchmarks. SSM proceeds by iterating between the following three steps:

(1) Compute the Newton direction $Z = SQP(L, \Lambda, E_0, X)$ using Eq. 11 and Alg.1, line 5. Let V be the orthogonal matrix consisting of columns in S (Alg.2, lines 6 and 7), where
$$S = span(PX, Z, v, LPX + E_0).$$
(2) SSM generates an approximation of (X, Λ) and an approximation of the smallest pair of eigenvalues σ /eigenvectors v of L in the subspace S,
$$[X, \Lambda, v, \sigma] = SSM(L, E_0, S)$$
consider the approximation $X = V\tilde{X}$ for some $\tilde{X}$. Compute
$$\min_X F_S := \min_{\tilde{X}} F(\tilde{X}; B, V^\top E_0)$$
(3) The terms L and E_0 are reweighted according to Sec. 3.4 such that the objective remains a tight upper-bound on the HPWL (Alg.1, line 10).

It is interesting to note the connection with graph coarsening methods. The orthogonal matrix V can be interpreted as a graph coarsening transform, and its inverse as a graph lifting transform—by reducing the size of the graph, we achieve significant improvements in scalability without sacrificing solution quality. Future work may investigate this alternative interpretation of SSM.

Algorithm 2 Sequential Subspace Minimization

Input: Partial Laplacian matrix L, unit vector v maximum iterations n
Output: Placement coordinates X

1: **function** SSM(A, v)
2: $L \leftarrow D - A$ ▷ Compute the graph Laplacian
3: Initialize X to $[U_1 : U_2]$, where U_i is the eigenvector of PLP corresponding to the i-th *smallest* nonzero eigenvalue (Sec 3.1).
4: **while** $t < n$ **do**
5: $Z \leftarrow SQP(L, \Lambda, E_0, X, v)$ ▷ Compute Z using Eq. 11 & Alg. 2
6: $\mathcal{S} \leftarrow span(X, Z, v, \Lambda X + E_0)$
7: $V \leftarrow QR(col(S))$
8: $B \leftarrow V^\top L V$
9: $\hat{X} \leftarrow \min_X F(\hat{X}; B, V^\top E_0)$ ▷ Solve Problem 4 in the subspace
10: $L, E_0 \leftarrow reweight(X)$
11: $t \leftarrow t + 1$
12: **end while**
13: **return** $V^\top \hat{X}$ ▷ Return lifted coordinates
14: **end function**

3.4 Minimization of HPWL via re-weighting

In this section, we show how our method may be adapted to facilitate direct minimization of HPWL. A similar method was adopted by the GORDIAN-L cell placement tool [1]. Inspired by asymptotically optimal algorithms for lasso-type regression problems [4, 5, 8, 24], we solve an equivalent ℓ_1 minimization problem by solving a sequence of re-weighted ℓ_2 minimization problems. In particular, we propose an analogous algorithm for the 2-dimensional case. Note that we now consider the following problem:

$$\sum_{i,j \in \mathcal{E}} w_{ij}(|x_i - x_j| + |y_i - y_j|) \tag{12}$$

Informally, the objective is upper bounded by the expression

$$\min_{u_{i,j}>0} \max_{v_{i,j}>0} \left\{ \sum_{i,j \in \mathcal{E}} (u_{i,j}|x_i - x_j|^2 + \frac{1}{u_{i,j}} + v_{i,j}|y_i - y_j|^2 + \frac{1}{v_{i,j}}) \right\}$$

Crucially, the equality holds if and only if $u_{i,j} = |x_i - x_j|^{-1}$ and $v_{i,j} = |y_i - y_j|^{-1}$ and implies a strategy for solving Prob. 12 that involves Prob. 4 as a sub-problem:

(1) For each $u > 0, v > 0$, solve Prob. 12 with respect to x, y.
(2) For each x, y, solve Prob. 12 with respect to u, v.

$$u_{i,j} = |x_i - x_j|^{-1}, \quad v_{i,j} = |y_i - y_j|^{-1}$$

In practice, we alter the above algorithm in two ways: (1.) following [1], a small adjustment to the denominator of each weight for normalization and to address numerical instability in the situation where two nodes overlap—e.g., $u_{i,j} = 1/(\mathcal{W}\sqrt{(x_i - x_j)^2 + \beta})$, where $\mathcal{W}$ is the width of the placement area (2.) instead of solving Prob. 12 (step (1.)) to convergence, we perform incremental 1-step updates—i.e., we perform re-weighting *each* iteration of SSM and compute the subsequent subspace with respect to the new reweighted matrix L and associated E_0. While the concept of iterative re-weighting for optimization has most commonly been applied to ℓ_1 and ℓ_∞ minimization problems, the framework is quite general and a similar procedure motivates minimization of other kinds of norms-based objectives. Future work includes investigating the efficacy of this reweighting scheme for alternative norm-minimization problems (e.g. robust p-norm minimization) in the context of layout.

3.5 Complexity analysis of QCQP initialization

In this section, we discuss the computational cost of our QCQP-based method, which is dominated by the SQP routine to compute the Newton directions. We claim the complexity of the QCQP placement initialization is $O(\#\text{iterations} \times T_{\text{matrix}})$, where #iterations is the number of SSM iterations, and T_{matrix} is the complexity of each call to a sparse matrix (i.e. Laplacian-like) solver. Although fast, nearly linear-time solvers exist for solving Laplacian-like systems [22], we adopt the Jacobi-preconditioned conjugate gradient method due to its simplicity and efficacy in practice.

Generic quadratic programs are NP-hard [21], i.e. it takes super-polynomial time to solve QPs optimally. In the convex case, there are polynomial time interior point algorithms [9]. Also, there are approximation algorithms that return local solutions of nonconvex QPs in polynomial time [13]. Our method falls into the category of algorithms that guarantee local, or block-globally optimal solutions.

In particular, although the objective of our algorithm satisfies the conditions for convexity, the addition of quadratic equality constraints—*sphere* constraints—introduces a violation of the conditions necessary for convexity. In practice, we find that our method

is typically stable to perturbations of the initialization as long as (1.) the layout is feasible and (2.) there are a sufficient number of fixed pins (i.e. the norm of E is sufficiently large).

3.5.1 Computation of the descent direction Z. In Sec 3.3, we express the Newton direction Z as the solution to the system characterized by the linearization of the first order optimality conditions. Namely, within each iteration of our procedure, we compute a set of Lagrangian multipliers as well as their update directions and the update directions for X as defined in Eq. 10 and Eq. 11.

In Alg. 1, we present the detailed steps of our implementation. The computations in lines 3-6 and 8-9 primarily involve vector-vector and matrix-vector multiplications. Exploiting the sparsity of L, both multiplications can be done in $O(n)$. To compute the inverses in line 7, we first compute $L_{W_j}^{-1}E$ by solving the linear system $L_{W_j}b = E$ for b. We solve $X^\top L_{W_j}^{-1}X\delta_j = b$ for δ_j. Using conjugate gradient, with an appropriate preconditioner K, the computation of δ_j up to a residual ϵ can be done in $O(n\sqrt{\kappa(KL)}\log(1/\epsilon))$ time. The computation of Z_j can be done in the same way.

In other words, the computation of the columns of the Lagrangian multipliers $(\Delta)_j$ can be decomposed into (1.) the computation of $L + W_jI$ twice in $O(n)$ time (2.) its inverse three times (once for each column of X and once for E_j) via conjugate gradient, again in $O(n\sqrt{\kappa(KL)}\log(1/\epsilon))$ time (3.) two left-multiplications by $X^\top$ in $O(2n)$ time. To compute the columns of the Newton direction; Z_j, first note that P can be re-written as $I - vv^\top$. Then, $(W_jI)^{-1}P = (W_jI)^{-1} - ((W_jI)^{-1}v)v^\top$.

In summary, the complexity of our method is dominated by the computation of the SQP newton direction Z in line 5 of Alg. 2, due to the necessity of computing three unique inverse-vector products involving the Laplacian (lines 7 and 10 of Alg. 1).

4 EXPERIMENTS

In this section we describe a set of comprehensive experiments on eight VLSI testcases from the ISPD'05 contest suite [20]. Summary statistics of the testcases are presented in Table 1. Our numerical experiments are aimed at establishing the efficacy of our method with respect to post-detailed placement wirelength. We leverage the DREAMPlace [17] placement engine and substitute the heuristic initialization schemes with our proposed method.

Table 1: Design characteristics. n_{free} =#Free cells and n_{fixed} =#Fixed pins. Max Deg, Avg Deg correspond to characteristics of the graph-models of the design netlists.

Design	#Free cells	#Fixed pins	#Nets	Max Deg	Avg Deg
adaptec1	211k	29k	221k	340	4.2
adaptec2	255k	21k	266k	153	3.9
adaptec3	452k	25k	467k	82	4.0
adaptec4	496k	29k	516k	171	3.7
bigblue1	278k	11k	284k	74	4.1
bigblue2	558k	141k	577k	260	3.5
bigblue3	558k	37k	1123k	91	3.4
bigblue4	2177k	170k	2230k	129	3.7

4.1 Experimental Setup

4.1.1 Algorithm parameters. To produce graph-layouts of IC netlists we adopt a hybrid net model [23]—a combination of the clique and star models. Each net is converted to a star or clique-graph depending on the size of the net—i.e. nets with three or fewer pins are modeled as cliques and nets with four or more pins are modeled as stars, with an associated free *pseudo-pin* variable introduced. To determine v, we first consider the surface area of cells (i.e. $v_i = w_i \times h_i$, where w_i and h_i is the width and height of cell i), scaled such that the distribution is centered about 1. v is then normalized. The c_i are determined according to the free layout space.

4.1.2 Implementation details. We implemented our algorithms in Python using the JAX framework [3] on a GCP c2-standard-8 machine with 8 virtual CPUs, 32 GB of memory, and a single Nvidia Tesla K80 GPU. In particular, we exploit JAX's capability to vectorize batched computation and compilation to XLA via the *jit* decorator. XLA facilitates hardware acceleration and the entire framework (initialization, global placement, detailed placement / legalization) may exploit GPU and multi-GPU-based parallelism without returning to a Python interpreter.

4.2 Results

4.2.1 Numerical results. We applied the proposed method to eight benchmarks from the ISPD'05 contest suite [20] and measured the cumulative HPWL post-detailed placement. Numerical results are provided in Table 2. We find that origin initializations consistently under-perform the other three methods, and that random and min-wirelength exhibit comparable results. However, initialization using the vanilla projected eigenvectors of the reduced Laplacian [6] result in superior HPWL—improvement between 1.0% and 3.0% compared to the random and min-wirelength heuristics. Larger gains are achieved when the initialization corresponds to the solution to Prob. 3 using SSM without reweighting—between 1.58% and 3.96%. Additionally, improvements in global placement runtime correlate with better initialization. We provide the global placement (DREAMPlace) runtime in Table 2. The GP runtime ranges from 62.42s to 1293.10s for the Projected Eigenvectors + SSM method, which is comparable to or less than the other methods.

4.2.2 Reweighted SSM iterations and runtime. In Table 3, we demonstrate that the directly minimizing HPWL via reweighting yields still further improvements—between 1.68% and 4.76% compared to random and min wirelength initializations. We note that reweighting methods are typically slow to converge [10]. As a consequence, instead of running our algorithm to convergence, we set a hard maximum limit of 100 reweighting / SSM steps. We additionally observe a mean per-iteration wall-time of $26.34 - 322.32$ and a significant ($\rho = 0.99$, $p = 1.1e-7$) linear correlation with the number of free cells. We plot this trend in Fig. 5b. It is likely that further gains could be achieved with a direct method for HPWL minimization.

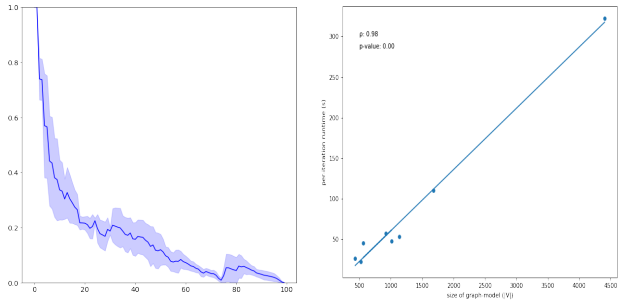
While the per-iteration runtime of our method is nontrivial, we highlight three key points: (1.) the experiments imply that the proposed QCQP formulation and method can consistently improve placement quality. This evidence incentivises future work to enhance the efficiency of these algorithms—particularly Laplacian solvers to drastically speed up turnaround time, (2.) few iterations

Table 2: Post-detailed place metrics. We report cumulative HPWL and runtime of global and detailed placement and legalization using various initializations. We report the percent improvement over random init. in parenthesis. The best result is bolded.

Design	Random		Min-wirelength		Projected Eigenvectors			Projected Eigenvectors + SSM		
	HPWL	GP runtime (s)	HPWL	GP runtime (s)	HPWL	GP runtime (s)	runtime (s)	HPWL	GP runtime (s)	runtime / iter. (s)
adaptec1	73.24	84.39	73.23	74.31	70.36 (3.9%)	63.86	93.6	**70.34** (3.96%)	62.42	26.34
adaptec2	82.51	189.46	82.24	172.91	81.68 (1.0%)	164.37	88.2	**81.21** (1.58%)	162.49	22.56
adaptec3	194.12	314.54	193.87	309.88	189.13 (2.5%)	313.29	181.2	**187.95** (3.18%)	314.01	57.78
adaptec4	174.43	371.72	174.16	354.16	171.73 (1.5%)	372.14	168.6	**171.62** (1.61%)	361.37	47.94
bigblue1	89.43	112.64	89.43	107.56	87.32 (2.3%)	94.11	124.2	**87.04** (2.67%)	94.23	45.71
bigblue2	136.69	387.94	136.69	361.75	132.49 (3.0%)	327.14	150.6	**131.37** (3.89%)	321.86	53.56
bigblue3	303.99	1064.63	303.99	1047.66	298.47 (1.8%)	847.03	369.0	**297.31** (2.20%)	849.23	110.63
bigblue4	743.75	1534.11	743.75	1500.70	726.71 (2.2%)	1372.49	1539.6	**724.78** (2.55%)	1293.10	322.32

Table 3: HPWL and structure-preservation test statistic for Prob. 3 (min-squared objective) and Prob. 3 (HPWL objective).

Design	Squared-wirelength		Direct HPWL	
	HPWL	z	HPWL	z
adaptec1	70.34 (3.96%)	0.131 ± 0.046	**70.12** (4.26%)	0.139 ± 0.052
adaptec2	81.21 (1.58%)	0.069 ± 0.031	**81.12** (1.68%)	0.073 ± 0.038
adaptec3	187.95 (3.18%)	0.072 ± 0.041	**186.61** (3.87%)	0.076 ± 0.043
adaptec4	171.62 (1.61%)	0.126 ± 0.057	**170.34** (2.34%)	0.131 ± 0.061
bigblue1	87.04 (2.67%)	0.063 ± 0.039	**85.72** (4.15%)	0.067 ± 0.041
bigblue2	131.37 (3.89%)	0.079 ± 0.037	**130.19** (4.76%)	0.081 ± 0.044
bigblue3	297.31 (2.2%)	0.074 ± 0.041	**296.04** (2.61%)	0.074 ± 0.043
bigblue4	724.78 (2.55%)	0.081 ± 0.053	**723.77** (2.69%)	0.081 ± 0.054

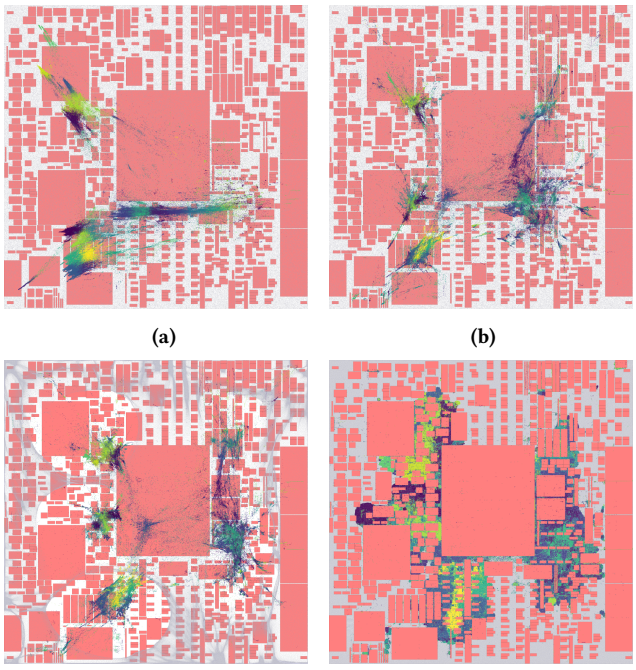


(a) (b)

Figure 5: Eigenvector method and projection. (a): Mean normalized decay in HPWL of adaptec cases. (b): Per-iteration turnaround (seconds) vs. dimension of L_{22}: # free cells + # nets in 10^3 unit.

are needed to significantly improve the post-detailed placement wirelength (as demonstrated in Fig. 5a), (3.) typical placement flows usually involve multiple runs of the global and detailed placement engine to validate different choices of hyperparameters, while our parameter-free initializations need only be computed once.

In Fig. 5a, we demonstrate that relatively few iterations are needed to improve the quality of post-detailed placement HPWL. For each testcase, we apply 100 iterations of SSM. Global and detailed placement is performed using each intermediate SSM iterate as the initialization. The HPWL of the post-detailed placement is measured and normalized to lie in the range $[0, 1]$. We plot the distribution of normalized post-detailed placement HPWL with the shaded region corresponding to 1 standard deviation in normalized HPWL. We observe that across all testcases, 60% of the improvement in post-detailed placement wirelength is achieved within the first $5 - 10$ iterations while roughly 80% of the improvement is achieved after the first $\sim$ 20 iterations. Additionally, we emphasize that our method is *parameter free* and yields the same solution across multiple runs. One may only need to generate a single initialization to validate multiple choices of global / detailed placement hyperparameters.



(a) (b) (c) (d)

Figure 6: Adaptec3 layout. (a): Projected eigenvectors for seed layout. Colors denote initial spatial partitions. (b—d) Intermediate DREAMPlace results. Note the preservation of cell groups (colors) through global placement.

4.2.3 Preservation of initial structure through global placement. In Fig 6, we plot intermediate iterations of the global placer, with colors corresponding to clusters of standard cells derived according to physical proximity via Euclidean k-means with $k = 10$. The

consistency of the colors (cluster) pre- and post-global placement serves demonstrate that the global placement algorithm preserves the global and local structure induced by the seed layout. Inspired by metrics proposed in Fogaça et al. [11] to evaluate the quality of a graph partitioning / clustering, we propose to evaluate this hypothesis by proposing a novel two-sample permutation test. We formulate the null (H_0) and alternative (H_a) hypotheses below:

H_0: *no effect of the initialization on the final layout*

H_a: *there is an effect*

Intuitively, under the null hypothesis, the cells component to any initial *spatial partitioning* (e.g. an arbitrary cell's neighbors) would separate during the global placement process, and a new partitioning *after* global placement would yield very different groups of cells. We consider a partitioning computed based on the initial layout—e.g. we apply Euclidean k-medoids[2] with $k = 100$. After global placement, we re-partition the final layout using k-means. For each centroid-cell c of an initial partition P_c, we find c's partition P'_c in the final layout. The statistic with respect to c is

$$z_c = \frac{|P_c \cap P'_c|}{|P_c| + |P'_c|} \tag{13}$$

We consider the mean over all c; $z = \frac{1}{k}\sum_{i\in[k]} z_{c_i}$, as the test statistic for a given initialization. Intuitively, the null-distribution is centered about zero (samples in the initial partition P_c characterized by c may end up arbitrarily far from c after global placement). Likewise, the "ideal" test-static corresponds to 0.5 ($P_c = P'_c$, partitions don't change after global placement). In Table 3, we report the z-scores associated with each design (since we find p-values are trivial). We simulate the null-distribution associated with each testcase 1000 times to compute the p-value p_{struct}, the percentage of simulations which result in a test statistic equal to or larger than proposed method's test statistic. We find significance at the 0.01-level for all designs, with the null-distribution close to zero (e.g. $\bar{z}_{\text{null}} = 0.00579$ with standard deviation $< 10^{-5}$ for adaptec3).

5 CONCLUSION AND FUTURE WORK

We have presented a novel QCQP formulation to initialize global placement engines. Despite the nonconvexity of the constraints, we describe an algorithm to efficiently solve the problem and extend it to facilitate minimization of HPWL. In an extensive study on eight VLSI designs, we have demonstrated that our approach to initialization consistently outperforms relevant methods with respect to post-detailed placement layout quality. Furthermore, we have proposed a statistical test for initialization quality. Future work includes a more detailed analysis of the algorithm, exploration of formulations for partitioning and local congestion, improving the method for HPWL minimization, and improving runtime.

ACKNOWLEDGMENTS

We acknowledge support from NSF CCF-2110419 and the Ministry of Science and Technology, Taiwan 110-2115-M-005-007-MY3.

[2] k-means assigns centers to arbitrary coordinates, k-medoids assigns centers to cells.

REFERENCES

[1] C.J. Alpert, T.F. Chan, A.B. Kahng, I.L. Markov, and P. Mulet. 1998. Faster minimization of linear wirelength for global placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 17, 1 (1998), 3–13.

[2] C.J. Alpert and A.B. Kahng. 1996. Simple eigenvector-based circuit clustering can be effective. In *1996 IEEE International Symposium on Circuits and Systems. Circuits and Systems Connecting the World. ISCAS 96*, Vol. 4. 683–686.

[3] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. *JAX: composable transformations of Python+NumPy programs.* http://github.com/google/jax

[4] Emmanuel Candès, Michael Wakin, and Stephen Boyd. 2007. Enhancing Sparsity by Reweighted L1 Minimization. *J. of Fourier Analysis and Applications* 14 (2007), 877–905.

[5] Rick Chartrand and Wotao Yin. 2008. Iteratively reweighted algorithms for compressive sensing. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. 3869–3872. https://doi.org/10.1109/ICASSP.2008.4518498

[6] Pengwen Chen, Chung-Kuan Cheng, Albert Chern, Chester Holtz, Aoxi Li, and Yucheng Wang. 2022. Placement Initialization via a Projected Eigenvector Algorithm: Late Breaking Results. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*. 1398–1399.

[7] C. Cheng, A. B. Kahng, I. Kang, and L. Wang. 2018. RePlAce: Advancing Solution Quality and Routability Validation in Global Placement. *IEEE TCAD*.

[8] Ingrid Daubechies, Ronald DeVore, Massimo Fornasier, and C. Sinann Güntürk. 2010. Iteratively reweighted least squares minimization for sparse recovery. *Communications on Pure and Applied Mathematics* 63, 1 (2010), 1–38.

[9] I. I. Dikin. 1967. Iterative solution of problems of linear and quadratic programming. *Sov. Math., Dokl.* 8 (1967), 674–675.

[10] Alina Ene and Adrian Vladu. 2019. Improved Convergence for ℓ_∞ and ℓ_1 Regression via Iteratively Reweighted Least Squares. *arXiv* abs/1902.06391 (2019).

[11] Mateus Fogaça, Andrew B. Kahng, Ricardo Reis, and Lutong Wang. 2019. Finding Placement-Relevant Clusters with Fast Modularity-Based Clustering. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference* (Tokyo, Japan) *(ASPDAC)*. Association for Computing Machinery, 569–576.

[12] Jiaqi Gu, Zixuan Jiang, Yibo Lin, and David Z. Pan. 2020. DREAMPlace 3.0: Multi-Electrostatics Based Robust VLSI Placement with Region Constraints. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*.

[13] William W. Hager. 2001. Minimizing a Quadratic Over a Sphere. *SIAM J. Optim.* 12.

[14] Kenneth M. Hall. 1970. An r-Dimensional Quadratic Placement Algorithm. *Management Science* 17.

[15] Meng-Kai Hsu, Valeriy Balabanov, and Yao-Wen Chang. 2013. TSV-Aware Analytical Placement for 3-D IC Designs Based on a Novel Weighted-Average Wirelength Model. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, 4 (2013), 497–509. https://doi.org/10.1109/TCAD.2012.2226584

[16] Andrew B. Kahng, Sherief Reda, and Qinke Wang. 2005. APlace: A General Analytic Placement Framework. In *ISPD* (San Francisco, California, USA).

[17] Yibo Lin, Shounak Dhar, Wuxi Li, Haoxing Ren, Brucek Khailany, and David Z. Pan. 2019. DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement. In *DAC*.

[18] Jingwei Lu, Pengwen Chen, Chin-Chih Chang, Lu Sha, Dennis Jen-Hsin Huang, Chin-Chi Teng, and Chung-Kuan Cheng. 2015. ePlace: Electrostatics-Based Placement Using Fast Fourier Transform and Nesterov's Method. *ACM TODAES* 20.

[19] Jingwei Lu, Hao Zhuang, Pengwen Chen, Hongliang Chang, Chin-Chih Chang, Yiu-Chung Wong, Lu Sha, Dennis Huang, Yufeng Luo, Chin-Chi Teng, and Chung-Kuan Cheng. 2015. ePlace-MS: Electrostatics-Based Placement for Mixed-Size Circuits. *IEEE TCAD* 34, 5 (2015), 685–698.

[20] Gi-Joon Nam, Charles J. Alpert, Paul Villarrubia, Bruce Winter, and Mehmet Yildiz. 2005. The ISPD2005 Placement Contest and Benchmark Suite. In *ISPD*.

[21] Panos M. Pardalos and Stephen A. Vavasis. 1991. Quadratic programming with one negative eigenvalue is NP-hard. *J. of Global Optimization* 1 (1991), 15–22.

[22] Daniel A. Spielman and Shang-Hua Teng. 2014. Nearly Linear Time Algorithms for Preconditioning and Solving Symmetric, Diagonally Dominant Linear Systems. *SIAM J. Matrix Anal. Appl.* 35, 3 (jan 2014), 835–885.

[23] Natarajan Viswanathan and Chris Chu. 2004. FastPlace: Efficient analytical placement using cell shifting, iterative local refinement, and a hybrid net model. *IEEE TCAD*, 26–33.

[24] David Wipf and Srikantan Nagarajan. 2010. Iterative Reweighted ℓ_1 and ℓ_2 Methods for Finding Sparse Solutions. *IEEE J. of Selected Topics in Signal Processing* 4 (2010), 317–329.