97-49

FASTER MINIMIZATION OF LINEAR WIRELENGTH FOR GLOBAL PLACEMENT

Charles J. Alpert[4], *Tony F. Chan*[2], *Dennis J.-H. Huang*[1], *Andrew B. Kahng*[1,3,*], *Igor L. Markov*[2], *Pep Mulet*[2], *Kenneth Yan*[1]

[1]UCLA Computer Science Department, Los Angeles, CA 90095-1596
[2]UCLA Mathematics Department, Los Angeles, CA 90095-1555
[3]Cadence Design Systems, Inc., San Jose, CA 95134
[4]IBM Austin Research Laboratory, Austin, TX 78758
{jenhsin,abk,kyan}@cs.ucla.edu, {chan,imarkov,mulet}@math.ucla.edu, alpert@austin.ibm.com

ABSTRACT

A linear wirelength objective more effectively captures timing, congestion, and other global placement considerations than a squared wirelength objective. The GORDIAN-L cell placement tool [16] minimizes linear wirelength by first approximating the linear wirelength objective by a modified squared wirelength objective, then executing the following loop – (1) minimize the current objective to yield some approximate solution, and (2) use the resulting solution to construct a more accurate objective – until the solution converges. In this paper, we first show that the GORDIAN-L loop can be viewed as a special case of a new algorithm that generalizes a 1937 iteration due to Weiszfeld [19]. Specifically, we formulate the Weiszfeld iteration using a *regularization parameter* to control the tradeoff between convergence and solution accuracy; the GORDIAN-L iteration is equivalent to setting this regularization parameter to zero. Other novel numerical methods described in the paper, the *Primal Newton* iteration and the *Primal-Dual Newton* iteration, further improve upon the linearly convergent Weiszfeld iteration. Our *Primal-Dual Newton* iteration stably attains quadratic convergence, making it a superior choice for implementing a placer such as GORDIAN-L, or for any linear wirelength optimization.

1. INTRODUCTION

The heart of the "quadratic placement" technique [2, 18]) lies in solving for one-dimensional placements that minimize squared wirelength. Two such placements in the x- and y-directions will induce a two-dimensional "global placement" which, due to the objective function and continuous formulation, will have most cells clumped in the center of the layout region. Quadratic placers vary mostly in how they map a global placement to a feasible cell placement (i.e., with non-overlapped cells in legal locations). Min-cut partitioning and center-of-gravity constraints are among the means of gradually "spreading out" the global placement during the course of this mapping.

*This work was supported by a grant from Cadence Design Systems, where ABK is currently Visiting Scientist (on sabbatical leave from UCLA).

The squared wirelength objective is applied only because it allows the one-dimensional placement problem to be reduced to the solution of a system of linear equations. However, this objective tends to overpenalize long wires and underpenalize short wires. Thus, a strongly connected cluster may be spread out over the placement, increasing wiring congestion for the router. This reduces the routing resource flexibility needed to satisfy timing and signal integrity constraints.

Mahmoud et al. [11] compared the linear and squared wirelength objectives for analog placement and concluded that the linear wirelength objective is superior. Works such as [14] have further shown that a linear wirelength objective can be used to form one-dimensional placements that directly yield effective bipartitioning solutions. In 1991, Sigl et al. [9, 16] proposed GORDIAN-L, an improved version of GORDIAN which optimizes the linear wirelength objective. Since the linear wirelength objective cannot be addressed directly by numerical methods, GORDIAN-L approximates the linear objective by a quadratic objective, then executes the following loop – (1) minimize the current objective to yield some approximate solution, and (2) use this solution to find a better quadratic approximation of the linear objective – until the solution converges. GORDIAN-L achieves solutions with up to 20% less area than GORDIAN while significantly reducing routing density and total minimum spanning tree cost [16]; it has been used widely in industry for both ASIC and structured-custom layout (e.g., Motorola PrediX floorplanner, Siemens LINPLACE placer, etc.). The GORDIAN-L improvement comes at the price of significantly increased CPU cost ([16] reports a factor of five increase over GORDIAN). To achieve reduced CPU cost, or improve solution accuracy within given CPU cost bounds, we have developed alternative numerical methods for linear wirelength minimization. We make the following contributions.

- We place the GORDIAN-L approach in the context of a new generalization, derived below, of a 1937 algorithm by Weiszfeld [19]. In its modern form, the Weiszfeld algorithm allows imposition of arbitrary linear constraints on the cell coordinates, including center of gravity constraints. Our derivation uses a β-regularization technique to allow a tradeoff between runtime and precision; GORDIAN-L corresponds to the special case of $\beta = 0$. The Weiszfeld approach has linear convergence as opposed to the superlinear (and sometimes quadratic) convergence of more modern numerical approaches.
- We next explore alternatives to the Weiszfeld (GORDIAN-L) solver. Specifically, we develop a *Pri-*

mal Newton iteration (an inexact Newton approach) and from this basis develop a *Primal-Dual Newton* iteration that stably attains quadratic convergence.[1] We highlight relevant mathematical theory and discuss implementation issues.

- Extensive experiments show that Primal-Dual Newton converges significantly faster than the Weiszfeld (GORDIAN-L) solver over a range of instance complexities and β-regularization regimes. It is straightforward to integrate the Primal-Dual Newton iteration into existing implementations.

2. PRELIMINARIES

A quadratic placer takes a netlist hypergraph as input and produces a placement of the cells. To apply existing numerical optimizations the netlist must first be transformed into a graph.

Definition: An *undirected weighted graph* $G(V, E)$ consists of a set of vertices $V = \{v_1, v_2, \ldots v_n\}$ and a set of edges $E = \{e_1, e_2, \ldots, e_m\}$ where each edge is an unordered pair of vertices. A weight function $w : E \rightarrow \Re^+$ assigns a nonnegative weight $w(e)$ to each edge in e.

To convert the netlist into a graph, the authors of [9] use a *star model* wherein a new "net node" is created for each net in the netlist, and an edge is added between the net node and each cell connected to the net. Placing the net node at the center of its incident cells (as GORDIAN assumes) makes the star model equivalent to a *clique model* which introduces an edge of weight $\frac{2}{p}$ between every pair of cells incident to a given p-pin net. The total squared wirelength will be the same for any placement under either the star or clique models.

Definition: The $n \times n$ *adjacency matrix* $\mathbf{A} = (a_{ij})$ for the graph G has entry $a_{ij} = w(v_i, v_j)$ if $(v_i, v_j) \in E$ and $a_{ij} = 0$ otherwise.

Definition: The $n \times n$ *Laplacian matrix* $\mathbf{Q} = (q_{ij})$ of A has entry q_{ij} equal to $-a_{ij}$ if $i \neq j$ and entry q_{ii} equal to $\sum_{j=1}^{n} a_{ij}$, i.e., q_{ii} is the *degree* of vertex v_i.

Definition: The n-dimensional *placement vector* $\mathbf{x} = (x_i)$ corresponds to the physical locations of cells $v_1, \ldots, v_n$ on the real line, i.e., x_i is the coordinate of vertex v_i.

GORDIAN [9] uses a squared wirelength objective:
Squared Wirelength Formulation: Minimize

$$\Phi_Q(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{d}^\mathbf{T}\mathbf{x} = \sum_{i,j=1}^{n} a_{ij}(x_i - x_j)^2 + \mathbf{d}^\mathbf{T}\mathbf{x}$$

$$\text{s.t. } \mathbf{H}\mathbf{x} = \mathbf{b} \tag{1}$$

Here, $\mathbf{H}$ is a $q \times n$ *constraint matrix* that represents q center of gravity constraints (a special case is that of fixed (pad) locations). Vector $\mathbf{b}$ gives the coordinates of the q centers of gravity. For v_j belonging to the i-th group of vertices, the (i, j) entry of $\mathbf{H_x}$ is set to be $\frac{1}{n_i}$, where n_i is the total number of vertices in the group. The optional linear term $\mathbf{d}^\mathbf{T}\mathbf{x}$ represents connections of cells to fixed I/O pads. The vector $\mathbf{d}$ can also capture pin offsets [16].

Definition: The $m \times n$ *incidence matrix* $\mathbf{C} = (c_{ki})$ for G represents the relationship between edges and vertices of G. For each edge $e_k = (v_i, v_j) \in E$, $c_{ki} = w(e_k)$ and $c_{kj} = -w(e_k)$; the orientation of edges (signs of c_{ki} and c_{kj}) can be arbitrary. All other entries of C are zero.

Note that there are $2m$ non-zero entries in $\mathbf{C}$ and that each row sum is zero. The GORDIAN-L linear wirelength objective is as follows:
Linear Wirelength Formulation: Minimize

$$\Phi_L(\mathbf{x}) = \| \mathbf{C}\mathbf{x} \|_1 = \sum_{i,j=1}^{n} a_{ij}|x_i - x_j| \quad \text{s.t.} \quad \mathbf{H}\mathbf{x} = \mathbf{b} \tag{2}$$

A term $\mathbf{d}^\mathbf{T}\mathbf{x}$ can again be added to incorporate pin offsets and connections to pads.

3. GORDIAN AND GORDIAN-L

GORDIAN [9] obtains a placement by repeatedly optimizing Φ_Q, alternating between the horizontal and vertical directions. Constraints for the optimizations are respectively given by $\mathbf{H_x x} = \mathbf{b_x}$ and $\mathbf{H_y y} = \mathbf{b_y}$, corresponding to the x and y directions as explained above.[2].

The algorithm must ensure that at each iteration $\mathbf{H_x}$ and $\mathbf{H_y}$ have maximal rank and constrain each cell by exactly one center of gravity. This implies that there is exactly one nonzero element in each column of $\mathbf{H_x}$ and $\mathbf{H_y}$.

GORDIAN begins with each cell attracted to the same center of gravity located in the center of the layout. During an iteration, for each center of gravity we consider the group of cells (if of size ≥ 2, i.e., if the constraint is non-trivial) attracted to it. The corresponding region is cut in two by a vertical or horizontal line passing through the center of gravity, and two new groups of cells with respective new centers of gravity replace the old group. This leads to a final solution where cells do not overlap.

Figure 1 summarizes the flow of the GORDIAN algorithm. The algorithm takes as input a graph G obtained by applying the $\frac{2}{p}$-clique model to a circuit netlist; it outputs the coordinates of the placed cells. The **repeat** loop in Steps 2-8 continues as long as the number of cells is bigger then the number of center of gravity constraints. At each iteration, Step 3 minimizes a quadratic objective function which is derived below. The resulting placement is used to refine the center of gravity constraints, yielding left and right constraints and larger linear systems (Steps 5 and 6). This process is then repeated in the y direction (Step 7), with each non-trivial constraint refined into top and bottom constraints. In each iteration, the number of subregions can quadruple, so the number of iterations through the **repeat** loop in Step 2 is $O(\log n)$.

We find the unique minimizer $\mathbf{x}$ for $\Phi_Q(\mathbf{x})$ defined in Equation (1) by solving the possibly underdetermined constraint system $\mathbf{H}\mathbf{x} = \mathbf{b}$, passing to an unconstrained formulation and finally solving a quadratic programming problem as follows. Matrix $\mathbf{H}$ is diagonalized by a permutation of columns[3] into $[\mathbf{H}_d \mid \mathbf{H}_i]$ ($\mathbf{H}_d$ is $q \times q$-diagonal; $\mathbf{H}_i$ has size $q \times (n - q)$). Correspondingly, the placement vector $\mathbf{x}$ splits into $n - q$ independent variables $\mathbf{x}_i$ and q dependent variables $\mathbf{x}_d$, so that we can rewrite $\mathbf{H}\mathbf{x} = \mathbf{b}$ as $[\mathbf{H}_d \mid \mathbf{H}_i] \begin{bmatrix} \mathbf{x}_d \\ \mathbf{x}_i \end{bmatrix} = \mathbf{b}$ or $\mathbf{H}_d\mathbf{x}_d + \mathbf{H}_i\mathbf{x}_i = \mathbf{b}$.

[1] This should not be confused with the "primal-dual" approach for timing-driven placement in [7], but similar variational ideas are used.

[2] To handle non-unit areas $a(v_i)$ for each vertex v_i, entry ij is changed to $\frac{a(v_i)}{A_j}$, where A_j is the sum of the areas of all vertices with center of gravity b_j.

[3] This is possible since every column has exactly one nonzero element and constraints are non-degenerate.

GORDIAN Placement Algorithm

Input: Graph $G(V, E)$ representing a circuit netlist, and its Laplacian $\mathbf{Q}$; Offset vectors $\mathbf{d}_x$, $\mathbf{d}_y$

Output: Vectors $\mathbf{x}, \mathbf{y}$ denoting the vertex coordinates

Variables: Constraint systems $\mathbf{H_x x} = \mathbf{b_x}, \mathbf{H_y y} = \mathbf{b_y}$ of increasing size representing current set of center of gravity constraints

1. Set $\mathbf{b_x}$ and $\mathbf{b_y}$ to 1-dim vectors c_x and c_y where (c_x, c_y) is the center of the layout
   Set up the objectives
   $\Phi_Q(\mathbf{x}) = \mathbf{x^T Q x} + \mathbf{d_x^T x}$ and $\Phi_Q(\mathbf{y}) = \mathbf{y^T Q y} + \mathbf{d_y^T y}$
2. **repeat** (Steps 3-7)
3.   Minimize $\Phi_Q(\mathbf{x})$ s.t. $\mathbf{H_x x} = \mathbf{b_x}$
4.   **for** *each non-trivial constraint* **do** (Steps 5-6)
5.   Replace with two new constraints: the cells to the left from the center are attracted to the center of the left half of the region. Similarly, for those cells to the right from the center. Update $\mathbf{H_x}, \mathbf{H_y}$
6.   Replace the center of gravity b_i with centers of gravity of two new groups (update $\mathbf{b_x}$ and $\mathbf{b_y}$)
7.   Repeat Steps 3-6 for y instead of x (in Step 5 use *top/bottom* instead of *left/right*)
8. **until** no two cells share the same center of gravity
9. **return** $\mathbf{x}$, $\mathbf{y}$

Figure 1. The GORDIAN algorithm.

Inverting the diagonal matrix, we get

$$\mathbf{x}_d = -\mathbf{H}_d^{-1}\mathbf{H}_i\mathbf{x}_i + \mathbf{H}_d^{-1}\mathbf{b}$$

This allows us to express $\mathbf{x}$ as

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_d \\ \mathbf{x}_i \end{bmatrix} = \begin{bmatrix} -\mathbf{H}_d^{-1}\mathbf{H}_i \\ \mathbf{I} \end{bmatrix} \mathbf{x}_i + \begin{bmatrix} \mathbf{H}_d^{-1}\mathbf{b} \\ \mathbf{0} \end{bmatrix}$$

or as $\mathbf{x} = \mathbf{Z}\mathbf{x}_i + \zeta$ with obvious notation for $\mathbf{Z}$ and ζ. We combine this formula with (1) to reduce the dimension of the unknown minimizer and obtain an unconstrained formulation

$$\Phi_Q(\mathbf{x}) = \frac{1}{2}\mathbf{x}_i^T\mathbf{Z}^T\mathbf{Q}\mathbf{Z}\mathbf{x}_i + (\mathbf{Q}\zeta + \mathbf{d})^T\mathbf{Z}^T\mathbf{x}_i + C$$

where C represents all constant terms. As $\Phi_Q(\mathbf{x})$ depends on $\mathbf{x}_i$ only, we introduce $\Psi_Q(\mathbf{x}_i) = \Phi_Q(\mathbf{x})$, so that

$$\Psi_Q(\mathbf{x}_i) = \frac{1}{2}\mathbf{x}_i^T\mathbf{Z}^T\mathbf{Q}\mathbf{Z}\mathbf{x}_i + \mathbf{c_0}^T\mathbf{Z}^T\mathbf{x}_i + C$$

where $\mathbf{c_0} = \mathbf{Q}\zeta + \mathbf{d}$. We see now that $\Psi(\mathbf{x}_i)$ gives an $(n-q)$-dimensional unconstrained quadratic programming problem. To determine its optimal solution, the gradient $\nabla\Psi(\mathbf{x}_i)$ is set to zero, yielding the $(n-q)\times(n-q)$ linear system

$$\mathbf{Z}^T\mathbf{Q}\mathbf{Z}\mathbf{x}_i = -\mathbf{c}$$

which can be efficiently solved with, e.g., conjugate gradient or another Krylov subspace solver [6]. Once the optimal value $\mathbf{x}_i$ is obtained, the optimal solution for $\mathbf{x}$ is given by $\mathbf{x} = \mathbf{Z}\mathbf{x}_i + \zeta$.

GORDIAN-L

Placement with minimum squared wirelength objective has an unique solution that can be found by solving the corresponding linear system. In contrast, placement with a minimum linear wirelength objective can have multiple optimal solutions. For example, a single movable cell connected to two fixed pads by edges of equal weight can be optimally placed anywhere between the two pads. In general, the set of optimal placements is closed and lies within the convex hull of fixed pads (see [18]). Direct minimization of a linear objective function can be achieved by linear programming, but this is usually computationally expensive.

Sigl et al. [16] minimize the linear wirelength objective $\Phi_L(\mathbf{x})$ by repeatedly applying the GORDIAN quadratic solver. They observe that the linear objective can be rewritten as

$$\Phi_L(\mathbf{x}) = \sum_{(i,j)\in E} a_{ij}|x_i - x_j| = \sum_{(i,j)\in E} \frac{a_{ij}(x_i - x_j)^2}{|x_i - x_j|}.$$

If $|x_i - x_j|$ were constant in the denominator of the last term, then a quadratic objective would be obtained and could be handled easily. The GORDIAN-L solver first solves the system $\Phi_Q(\mathbf{x})$ to obtain a reasonable approximation for each $|x_i - x_j|$ term. Call this solution $\mathbf{x}^0$. GORDIAN-L then derives successively improved solutions $\mathbf{x}^1, \mathbf{x}^2, \ldots$ until there is no significant difference between $\mathbf{x}^k$ and $\mathbf{x}^{k-1}$. From a given solution $\mathbf{x}^{k-1}$, the next solution $\mathbf{x}^k$ is obtained by minimizing

$$\Phi_L^k(\mathbf{x}^k) = \sum_{(i,j)\in E} \frac{a_{ij}(x_i^k - x_j^k)^2}{|x_i^{k-1} - x_j^{k-1}|} = \sum_{(i,j)\in E} g_{ij}^k \cdot (x_i^k - x_j^k)^2 \quad (3)$$

where $g_{ij}^k = \frac{a_{ij}}{|x_i^{k-1} - x_j^{k-1}|}$. Note that the coefficients g_{ij}^k are adjusted between iterations. The iterations terminate when the factors $(x_i^k - x_j^k)$ no longer change significantly.[4] Just as with $\Phi_Q(\mathbf{x})$, we can minimize $\Phi_L^k(\mathbf{x})$ in Equation (3) by applying a Krylov subspace solver. The GORDIAN algo-

GORDIAN-L Solver (new Step 3 for Figure 1)

Input: Adjacency matrix $\mathbf{A}$, constraint matrix $\mathbf{H_x}$ and vector $\mathbf{b_x}$.

Output: Solution $\mathbf{x}$ that optimizes $\Phi_L(\mathbf{x})$ such that $\mathbf{H_x} = \mathbf{b_x}$.

Variables: Intermediate solutions $\mathbf{x}^k$

1. Solve $\Phi(\mathbf{x}^{(0)})$ as in Step 3 of Figure 1. Set $k = 1$.
2. **do** (Steps 3-7)
3.   Update each edge weight g_{ij}^k to $\frac{a_{ij}}{|x_i^{k-1} - x_j^{k-1}|}$.
4.   Construct $\Phi_L^k(\mathbf{x}^k)$ from Equation (3).
5.   Minimize $\Phi_L^k(\mathbf{x}^k)$ s.t. $\mathbf{H_x x}^k = \mathbf{b_x}$.
6. $k = k + 1$.
7. **while** $\sum_{1\le i\le n} |x_i^k - x_i^{k-1}| > \epsilon$
8. **return** $\mathbf{x}^k$.

Figure 2. The GORDIAN-L solver.

[4] Some convergence criterion must be specified in any implementation. Unfortunately, we do not know convergence criterion used in GORDIAN-L, which makes CPU time comparisons impossible.

rithm can be transformed into GORDIAN-L by replacing Step 3 of Figure 1 with the solver shown in Figure 2.[5]

4. WEISZFELD METHOD

We now show that the GORDIAN-L solver is equivalent to a special case of what we call the "Weiszfeld algorithm". More precisely, we will describe, in modern terms, a method first suggested in 1937 by Weiszfeld [19] and later generalized by Miehle [12]. A contemporary exposition of this method, along with a proof of its global *linear*[6] convergence, can be found in [4, 5] (see also [10]). Technical differences between GORDIAN-L and our generalization of Weiszfeld (vis-a-vis what we call β-regularization) are discussed in the second subsection.

4.1. Derivation of the Weiszfeld algorithm

We wish to minimize $f(\mathbf{x}) = \| \mathbf{Cx} \|_1$ subject to $\mathbf{Hx} = \mathbf{b}$. Observe that

$$f(\mathbf{x}) = \| \mathbf{Cx} \|_1 = \sum_{j=1}^{m} |\mathbf{C}_j\mathbf{x}| \approx \sum_{j=1}^{m} \sqrt{(\mathbf{Cx})_j^2 + \beta}$$

where $\beta > 0$ is a small constant. The purpose of β is to approximate the non-differentiable objective function by a smooth function. In order to write the derivative of $f(\mathbf{x})$ compactly, notice that[7]

$$\frac{d(|\mathbf{C}_j\mathbf{x}|^2)}{dx} = 2\mathbf{C}_j^T\mathbf{C}_j\mathbf{x} \tag{4}$$

Hence,

$$\nabla f(\mathbf{x}) = \sum_{j=1}^{m} \frac{\mathbf{C}_j^T\mathbf{C}_j\mathbf{x}}{\sqrt{(\mathbf{C}_j\mathbf{x})^2 + \beta}}$$

and the partial derivatives of the Lagrangian $L(\mathbf{x}, \lambda)$ are

$$\frac{\delta L}{\delta \mathbf{x}} = \sum_{j=1}^{m} \frac{\mathbf{C}_j^T\mathbf{C}_j\mathbf{x}}{\sqrt{(\mathbf{C}_j\mathbf{x})^2 + \beta}} + \lambda\mathbf{H}^T = 0 \tag{5}$$

$$\frac{\delta L}{\delta \lambda} = \mathbf{Hx} - \mathbf{b} = 0 \tag{6}$$

Thus, the original minimization problem has been transformed into two systems of equations which can be combined to yield the nonlinear system

$$\begin{bmatrix} \mathbf{B}(\mathbf{x}) & \mathbf{H}^T \\ \mathbf{H} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \end{bmatrix} \tag{7}$$

where $\mathbf{B}(\mathbf{x}) = \sum_{j=1}^{m} \frac{\mathbf{C}_j^T\mathbf{C}_j}{\sqrt{(\mathbf{C}_j\mathbf{x})^2 + \beta}}$.

[5] Note that GORDIAN-L [16] also includes an additional modification. Rather than subdivide each region into two subregions, GORDIAN-L subdivides each region into *three* subregions and then minimizes the objective Φ_L; the result is then used to subdivide the region into *five* subregions, and the minimization is performed again. The resulting solution is used as the output for Step 3 in Figure 1, and centers of gravity are assigned as before. This modification improves performance but increases the number of calls to the numerical solver.

[6] Choose a norm and let $\epsilon(k)$ denote the norm of the residual vector at the k-th iteration. Assume that $\log \frac{\epsilon(0)}{\epsilon(k)} \approx Bk^s$ for some constant B and $s \in \{1, 2\}$. We say that the convergence is *linear* when $s = 1$ and *quadratic* when $s = 2$.

[7] Here $\mathbf{C}_j^T\mathbf{C}_j\mathbf{x} = \mathbf{C}_j \otimes \mathbf{C}_j$ a.k.a. the Kronecker product.

To solve this system, we guess an initial approximation $\mathbf{x}^0$ and solve the system with $B(\mathbf{x}^1) = B(\mathbf{x}^0)$, where $\mathbf{x}^1$ is the next iterate. In general, we compute the Weiszfeld iterate[8] $\mathbf{x}^k$ from the previous value $\mathbf{x}^{k-1}$ by solving the linear system

$$\begin{bmatrix} \mathbf{B}(\mathbf{x}^{k-1}) & \mathbf{H}^T \\ \mathbf{H} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}^k \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \end{bmatrix} \tag{8}$$

which we call the *low-level* system. The iterations continue until a convergence criterion is met, e.g., one may look at some norm of the *residual vector* (the difference between the left hand side and the right hand side of Equation (7)). This is the main flow of the *Weiszfeld algorithm.*

Recall that the heart of the GORDIAN-L solver is its approximation of the linear wirelength objective by a quadratic objective:

$$\Phi_L(\mathbf{x}^k) = \sum_{(i,j)\in E} \frac{a_{ij}(x_i^k - x_j^k)^2}{|x_i^{k-1} - x_j^{k-1}|} \tag{9}$$

In other words, GORDIAN-L uses the same type of iteration as Weiszfeld. Equation (9) can be rewritten as

$$\Phi_L(\mathbf{x}^k) = \sum_{j=1}^{m} \frac{(\mathbf{C}_j\mathbf{x}^k)^2}{|\mathbf{C}_j\mathbf{x}^{k-1}|}$$

for which the Lagrangian is

$$L(\mathbf{x}^k, \lambda) = \sum_{i=1}^{m} \frac{(\mathbf{C}_j\mathbf{x}^k)^2}{|\mathbf{C}_j\mathbf{x}^{k-1}|} + \lambda^T(\mathbf{Hx}^k - \mathbf{b})$$

and

$$\frac{\delta L}{\delta \mathbf{x}^k} = \sum_{j=1}^{m} \frac{\mathbf{C}_j^T\mathbf{C}_j\mathbf{x}^k}{|\mathbf{C}_j\mathbf{x}^{k-1}|} + \lambda\mathbf{H}^T = 0 \tag{10}$$

Setting $\tilde{\mathbf{B}}(\mathbf{x}) = \frac{\mathbf{C}_j^T\mathbf{C}_j\mathbf{x}^k}{|\mathbf{C}_j\mathbf{x}^{k-1}|}$ and using Equation (6), we obtain a system very similar to Equation (8) except that $\mathbf{B}(\mathbf{x}^{k-1})$ is now replaced with $\tilde{\mathbf{B}}(\mathbf{x}^{k-1})$. The only difference between these two matrices is that the Weiszfeld algorithm approximates $|\mathbf{C}_j\mathbf{x}|$ with $\sqrt{(\mathbf{C}_j\mathbf{x})^2 + \beta}$. This is necessary to avoid numerical problems when $|x_i^{k-1} - x_j^{k-1}|$ becomes too small (cf. Step 3 of Figure 2). In GORDIAN-L (see [16]), if this term becomes smaller than the *minimal gate width*, it is replaced with this gate width. In summary, it is simply a matter of Weiszfeld and GORDIAN-L using different schemes (β-regularization versus minimal gate size, cf. Figure 3) in order to guarantee reasonable behavior of the solver at the cusps of the objective function. β-regularization is superior to using the minimal gate width in the sense that it is closer to the original objective function and its unique minimizer has convenient limit behavior when $\beta \to 0$ (see below).

[8] The mathematical idea behind solving Equation (7) is to build an iteration $\mathbf{x}^k \mapsto \mathbf{x}^{k-1}$ whose fixed point is the unknown solution. Hence, Weiszfeld can be classified as a fixed-point method.

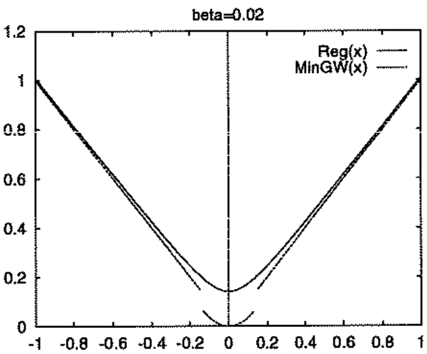

Figure 3. Comparing the Minimal Gate Width function (MinGW) with the β-regularization (Reg) in one dimension. The original objective is represented by $|x|$. Here $\beta = 0.02$ and the minimal gate width is set to $\sqrt{\beta} \approx 0.141$. Note that the discontinuous MinGW has continuous one-sided derivative (optimality conditions), as its branches are given by $\frac{x^2}{2\sqrt{\beta}}$ and $|x|$. By demanding continuity of the objective function and therefore dropping the factor of 1/2, we would make the optimality conditions discontinuous.

4.2. β-regularization.

In the Weiszfeld algorithm the objective function was β-regularized, in particular, to bound the denominator in Equation (5) away from zero. We now discuss some properties of the regularized objective function and its relation to the original objective function.

By changing all expressions of the form $|\cdot|$ in the original objective to $\sqrt{(\cdot)^2+\beta}$, the resulting objective becomes strictly convex and therefore has an unique global minimizer. As $\beta \to 0$, this minimizer approaches that of the original linear objective which, in turn, can have plenty of minimizers. For example, consider Φ_L for the instance of a single movable vertex connected to two fixed pads on opposite ends of the layout. One can see that Φ_L will have uncountably many optimal solutions, while the β-regularization will have but one (for $\beta > 0$).

Clearly, as β increases, the disparity between the regularized objective and the original objective increases as well (and the derived solutions will be further from optimal). On the other hand, if β is too small, the derivative of the objective function (which is used in variational methods) will behave badly near points where the original objective is not smooth: matrices in linear systems will become ill-conditioned and solving them will become computationally expensive, if impossible.

The first question now is: "How should β be expressed to have comparable effects for various unrelated placement problems?" In the original objective function, all expressions of form $|\cdot|$ are actually $|x_i - x_j|$. These expressions are upper bounded by L, the length of placement interval, which varies across different placement instances. If we set $\beta = \beta_r L^2$, where β_r is a small number, then $\sqrt{(x_i-x_j)^2+\beta} = L\sqrt{(\tilde{x}_i-\tilde{x}_j)^2+\beta_r}$ with $\tilde{x}_i$ and $\tilde{x}_j$ being on order of 10^0 for any placement problem. Our experiments show that in this way we obtain similar behavior for different placement problems if we use the same value of β_r — independent of problem size. We have worked with values ranging from 10^1 to 10^{-7}.

The second question — how to choose good values of β_r — is harder; the answer largely depends on how the solutions produced by the Weiszfeld method are used. One can repeatedly solve Weiszfeld for decreasing values of β and stop when the difference between successive placements is small; alternatively, one can stop when the objective function stabilizes. Both strategies can lead to premature stopping, and finding a good heuristic is an open question.

5. THE PRIMAL NEWTON METHOD

The Newton approach is often used as a base for developing more sophisticated methods with superlinear convergence(e.g. in [1, 10]). In this section, we develop what we call the *Primal Newton* method for minimizing the linear wirelength objective. Our main purpose is to introduce the reader to techniques that we will use in developing the *Primal-Dual Newton* method.[9] Primal-Dual will also be a Newton method, but with an additional set of dual variables. Because the Primal-Dual Newton method converges at least as fast as the Primal Newton method and is more stable (i.e., its region of convergence is strictly larger), we do not report experimental data for Primal Newton.

Consider minimizing $\sum_{j=1}^{m}\sqrt{(\mathbf{C}_j\mathbf{x})^2+\beta}$ such that $\mathbf{Hx}=\mathbf{b}$. As before, $\mathbf{C}_j \in R^n$ contains only two nonzero entries — plus and minus the i-th edge weight — at locations corresponding to the two vertices of the edge.
The Lagrangian for this problem is

$$L(\mathbf{x},\lambda) = \sum_{j=1}^{m}\sqrt{(\mathbf{C}_j\mathbf{x})^2+\beta} + \lambda^T(\mathbf{Hx}-\mathbf{b}) \tag{11}$$

Taking partial derivatives and using Equation (4), gives us

$$\frac{\delta L}{\delta x} = \sum_{j=1}^{m}\frac{\mathbf{C}_j^T\mathbf{C}_j\mathbf{x}}{\sqrt{(\mathbf{C}_j\mathbf{x})^2+\beta}} + \mathbf{H}^{\mathbf{T}}\lambda = 0 \tag{12}$$

$$\frac{\delta L}{\delta \lambda} = \mathbf{Hx}-\mathbf{b} = 0 \tag{13}$$

Applying the Newton method to this nonlinear system, we rewrite the system (invoking the fact that rows of matrix C are precisely $\mathbf{C}_j$ and utilizing some linear algebra) as follows:

$$\begin{bmatrix} \mathbf{M} & \mathbf{H^T} \\ \mathbf{H} & 0 \end{bmatrix}\begin{bmatrix} \delta\mathbf{x} \\ \delta\lambda \end{bmatrix} = -\begin{bmatrix} \mathbf{K}(\mathbf{x},\lambda) \\ \mathbf{Hx}-\mathbf{b} \end{bmatrix} \tag{14}$$

where

$$\mathbf{M}(\mathbf{x}) = \mathbf{C^T}\mathbf{E}(\mathbf{x})^{-1}\mathbf{F}(\mathbf{x})\mathbf{C} \tag{15}$$

and the following notations are used:

- $\mathbf{K}(\mathbf{x},\lambda) = \sum_{j=1}^{m}\frac{\mathbf{C}_i^T\mathbf{C}_i\underline{\mathbf{x}}}{\sqrt{(\mathbf{C}_i\mathbf{x}^k)^2+\beta}} + \mathbf{H}^T\lambda$
- $\eta_i = \sqrt{(\mathbf{C}_i\mathbf{x}^k)^2+\beta}$, with β defined as for the Weiszfeld algorithm
- $\mathbf{E}(\mathbf{x})$ is an $m \times m$ diagonal matrix with values in the i-th row equal to η_i
- $\mathbf{F}(\mathbf{x})$ is an $m \times m$ diagonal matrix with values in the i-th row equal to $(1 - \frac{(\mathbf{C}_i\mathbf{x}^k)^2}{\eta_i^2})$

[9] The key issue addressed by Primal-Dual Newton is global convergence, which Primal Newton lacks. No precise mathematical statement about global convergence of Primal-Dual Newton has been proven, but its reliable convergence properties have been observed in the literature (e.g. [1, 10]) and our experiments.

At the end of each iteration, we update $\mathbf{x}$ and λ as

$$\mathbf{x} = \mathbf{x} + \delta\mathbf{x}$$
$$\lambda = \lambda + \delta\lambda$$

Starting with initial values of $\mathbf{x}$ and λ, we compute corresponding values for $M(x)$ and $K(x, \lambda)$, then update $\mathbf{x}$ and λ by solving the system in (14). This is repeated until some convergence criterion is met. We call this particular implementation of the Newton method *Primal Newton.*

The Primal Newton method does not possess any kind of global convergence property. Local convergence takes place — a proof can be found in [13] — but we do not know of any estimates of the size of the local convergence region. One can use various globalization techniques (e.g., *line search* and *trust regions*) to guarantee convergence of the Primal Newton method everywhere. However, all of these globalization techniques are known to be inefficient in a number of applications[10] due to the small size of the region where Primal Newton converges quadratically. Modifications to a Newton method which allow it to achieve global convergence can be found in [10], where a corresponding theorem is proven and numerical results demonstrating advantages over the Weiszfeld method are shown. [10] also contains a discussion of *degeneracy* — a feature of some placement problems for which Newton-like methods are only linearly convergent.

The various considerations related to top-level stopping criteria for Weiszfeld in the previous section do not carry over to the Newton method, since we are searching for $\mathbf{x}$ which cannot be characterized as satisfying a particular linear system. In other words, we do not have an analogue for the residual norm. However, convergence criteria in terms of successive iterates are easily defined since $\delta\mathbf{x}$ is the difference between successive iterates. Alternatively, various convergence criteria can be deduced from the observation that the *nonlinear residual* — the right hand side of (14) — goes to zero as the Newton method progresses.

6. THE PRIMAL-DUAL NEWTON METHOD

The idea of the primal-dual Newton approach was developed by Conn and Overton in [3] and has been recently used for a denoising application in image processing (see [1]). Numerical results suggest that the approach has fast convergence, stability and significant practical value.

Recall that the optimization problem we will solve is: find $\mathbf{x}$ which minimizes

$$f(\mathbf{x}) = \sum_{j=1}^{m} \sqrt{(\mathbf{C}_j\mathbf{x})^2 + \beta} \quad \text{s. t.} \quad \mathbf{Hx} = \mathbf{b} \tag{16}$$

where $\mathbf{C}_j \in R^n$ again contains only two nonzero entries — plus and minus the i-th edge weight — at locations corresponding to the two vertices of the edge. Let $s_j = \mathbf{C}_j\mathbf{x}$, Then we can rewrite (16) as: find $\mathbf{x}$ which minimizes

$$\sum_{j=1}^{m} \sqrt{s_j^2 + \beta} \quad \text{s. t.} \quad \mathbf{C}_j\mathbf{x} - s_j = 0 \ \text{ and } \ \mathbf{Hx} = \mathbf{b} \tag{17}$$

The Lagrangian for this problem is

$$L(\mathbf{x}, \mathbf{s}, \mathbf{z}, \lambda) = \sum_{j-1}^{m} \sqrt{s_j^2 + \beta} + \sum_{j-1}^{m} z_j(\mathbf{C}_j\mathbf{x} - s_j) + \lambda(\mathbf{Hx} - \mathbf{b})$$

where λ and $\mathbf{z}$ are the Lagrange multipliers for $\mathbf{x}$ and $\mathbf{s}$. The Karush-Kuhn-Tucker first order necessary conditions are

$$\frac{\partial L}{\partial \mathbf{x}} = \sum_{j=1}^{m} \mathbf{C}_j^T z_j + \mathbf{H}^{\mathbf{T}}\lambda = 0 \tag{18}$$

$$\frac{\partial L}{\partial s_j} = \frac{s_j}{\sqrt{s_j^2+\beta}} - z_j = 0, \qquad j = 1, ..., m \tag{19}$$

$$\frac{\partial L}{\partial z_j} = \mathbf{C}_j\mathbf{x} - s_j = 0, \qquad j = 1, ..., m \tag{20}$$

$$\frac{\partial L}{\partial \lambda} = \mathbf{Hx} - \mathbf{b} = 0 \tag{21}$$

Using (20) to eliminate s_j from (19) and rearranging slightly:

$$\sum_{j=1}^{m} \mathbf{C}_j^T z_j + \mathbf{H}^{\mathbf{T}}\lambda = 0 \tag{22}$$

$$\mathbf{C}_j\mathbf{x} - (\sqrt{(\mathbf{C}_j\mathbf{x})^2 + \beta})z_j = 0, \quad j = 1, ..., m \tag{23}$$

$$\mathbf{Hx} - \mathbf{b} = 0 \tag{24}$$

We can now apply Newton's method to this nonlinear system. Differentiating the left hand side of Equation (22) with respect to $\mathbf{z}$ and writing the result as a matrix, we get $\mathbf{C}^{\mathbf{T}}$ (because $\mathbf{C}^{\mathbf{T}}$ is composed of $\mathbf{C}_j^T$). Differentiating the left hand side of Equation (23) with respect to $\mathbf{x}$, we get (refer to (4))

$$\mathbf{C}_j - \frac{z_j(\mathbf{x}^{\mathbf{T}}\mathbf{C}_j^{\mathbf{T}})\mathbf{C}_j}{\sqrt{(\mathbf{C}_j\mathbf{x})^2 + \beta}}, j = 1, ..., m \tag{25}$$

which can be rewritten in matrix form as

$$\mathbf{I}(\mathbf{z}, \mathbf{x})\mathbf{C} \tag{26}$$

with $\mathbf{I}(\mathbf{z}, \mathbf{x}) = diag(1 - \frac{z_i(\mathbf{x}^{\mathbf{T}}\mathbf{C}_i^T)}{\eta_i})$, $\eta_j = \sqrt{(\mathbf{C}_j\mathbf{x})^2 + \beta}$. We introduce $\mathbf{E} = diag(\eta_i)$ to express in matrix form the derivative of the left hand side of Equation (23) with respect to $\mathbf{z}$.

Finally, Newton's method gives the following linear system[11] which we need to solve repeatedly:

$$\begin{bmatrix} \mathbf{C}^{\mathbf{T}} & 0 & \mathbf{H}^{\mathbf{T}} \\ \mathbf{E} & -\mathbf{I}(\mathbf{z}, \mathbf{x})\mathbf{C} & 0 \\ 0 & \mathbf{H} & 0 \end{bmatrix} \begin{bmatrix} \delta\mathbf{z} \\ \delta\mathbf{x} \\ \delta\lambda \end{bmatrix} = - \begin{bmatrix} \mathbf{C}^{\mathbf{T}}\mathbf{z} + \mathbf{H}^{\mathbf{T}}\lambda \\ \mathbf{Ez} - \mathbf{Cx} \\ \mathbf{Hx} - \mathbf{b} \end{bmatrix} \tag{27}$$

To reduce the dimension of this system, we eliminate $\delta\mathbf{z}$ by substituting its second equation

$$\delta\mathbf{z} = -\mathbf{z} + \mathbf{E}(\mathbf{x})^{-1}\mathbf{Cx} + \mathbf{E}(\mathbf{x})^{-1}\mathbf{I}(\mathbf{z}, \mathbf{x})\mathbf{C}\delta\mathbf{x} \tag{28}$$

into the first equation. After cancelation, we get

$$\mathbf{C}^{\mathbf{T}}\mathbf{E}^{-1}\mathbf{I}(\mathbf{z}, \mathbf{x})\mathbf{C}\delta\mathbf{x} + \mathbf{H}^{\mathbf{T}}\delta\lambda = -(\mathbf{C}^{\mathbf{T}}\mathbf{E}^{-1}\mathbf{Cx} + \mathbf{H}^{\mathbf{T}}\lambda)$$

and together with the third equation of (27) this makes

$$\begin{pmatrix} \mathbf{C}^{\mathbf{T}}\mathbf{E}(\mathbf{x})^{-1}\mathbf{I}(\mathbf{z}, \mathbf{x})\mathbf{C} & \mathbf{H}^{\mathbf{T}} \\ \mathbf{H} & 0 \end{pmatrix} \begin{pmatrix} \delta\mathbf{x} \\ \delta\lambda \end{pmatrix} = - \begin{pmatrix} \mathbf{K}(\mathbf{x}, \lambda) \\ \mathbf{Hx} - \mathbf{b} \end{pmatrix} \tag{29}$$

[10] Experience indicates that our application is one; image processing is another.

[11] Here, dual variable $\mathbf{z}$ and matrices $\mathbf{E}$, $\mathbf{I}$ are m-dimensional, while $\mathbf{x}, \mathbf{b}, \lambda$ and $\mathbf{H}$ are n-dimensional. $\mathbf{C}$ has size $m \times n$.

where[12]

$$\mathbf{K}(\mathbf{x},\lambda) = \mathbf{C^T E^{-1} C x} + \mathbf{H}^T\lambda = \sum_{j=1}^{m} \frac{\mathbf{C}_i^T \mathbf{C}_i \mathbf{x}}{\eta_i} + \mathbf{H}^T\lambda \quad (30)$$

In the overall algorithm, $\delta\mathbf{z}$ gives an *update direction* for $\mathbf{z}$, and we are free to use $\delta\mathbf{z}$ with any factor we want. However, as noted in [1, p.9], for the matrix in (29) to be nonsingular one requires $\| \mathbf{z}^k \|_\infty \leq 1$. To ensure this, iterates $\mathbf{z}^k$ of the *dual* variable are defined recursively with $\mathbf{z}^0 = \mathbf{0}$, and updates computed at each iteration by (28) and the *line search* formula $\mathbf{z}^{k+1} = \mathbf{z}^k + S\ \delta\mathbf{z}$, where

$$S = \min\{0.9 \sup\{S| \parallel \mathbf{z}^k + S\ \delta\mathbf{z} \parallel_\infty < 1\}, 1\} \qquad (31)$$

One computes the supremum by looping over coordinates and solving $-1 \leq \mathbf{z}^k + S\,\delta\mathbf{z} \leq 1$ for S. (In practice, $S \to 1$ as iterates converge, and we find that $S = 1$ for most iterates.) The variables $\mathbf{x}$ and λ are updated at each iteration using

$$\mathbf{x} = \mathbf{x} + \delta\mathbf{x}$$

$$\lambda = \lambda + \delta\lambda$$

Computationally, we deal with the system (29) just as with the Primal Newton method for linear objective in Section 5. To find an initial approximation close to the quadratic convergence region, one can solve a few linear systems as if using the Weiszfeld algorithm, then switch to Primal-Dual iterations[13].

The right hand side of (14) goes to zero as top-level iterates converge. This means that all convergence tests involving residual vectors should be formulated in terms of *relative tolerance* or should otherwise depend on the right hand side of the system. We have observed in our experiments that if for some reason (14) is not solved precisely enough, Newton top-level iterates can start to diverge.

The remarks given for the Primal Newton method above also apply to Primal-Dual Newton (see [1]); in particular, Primal-Dual Newton possesses quadratic convergence (see [8, 5.4.1]) and is preferable to the linearly convergent Weiszfeld algorithm. Primal-Dual Newton converges quadratically in strictly larger regions than Newton method and is only 30-50% more expensive in computation and memory per iteration than the Weiszfeld method.

7. EXPERIMENTAL VALIDATION

In this section, we review some aspects of our experimental testbed and substantiate the efficiency of Primal-Dual Newton in comparison to Weiszfeld.

Test Case	Pads	Cells	Nets
primary1	107	752	704
biomed	97	6417	6442
avq_small	64	21854	21884
golem3	2767	100281	144949

Figure 4. VLSI benchmark circuits used in comparing the Weiszfeld and Primal-Dual Newton methods.

7.1. Implementing the Low-Level Solver

We implemented the Weiszfeld and Primal-Dual Newton iterations within our own sparse-matrix testbed; this testbed is coupled to a design database and partitioning and layout tools, with interfaces via standard design interchange formats. Thus, we were able to verify our methods using standard benchmarks from the literature (ftp to cbl.ncsu.edu).

When implementing the Primal-Dual method, it is crucial to solve the linear system (29) precisely enough that the top-level iterates will converge. One finds that matrices arising in (29) are usually much denser and more ill-conditioned than in analogous systems arising from denoising problems in image processing or from numerical solution of partial differential equations. This makes it harder for any low-level solver to find sufficiently precise approximate solutions. To avoid undue loss of sparsity when $\mathcal{O}(p^2)$ edges are introduced for some very large p-pin net, we represent any large net with >100 pins by a random cycle through its cells[14].

Since our implementation is designed to accommodate examples of any size, we use iterative solvers, specifically, GMRES or BICGSTAB with ILU preconditioner[15]. Here, we must refer the reader to [8, Chap 6], where usage of iterative (inexact) solvers is considered with special regard to Newton methods. Our solver changes the values of relative tolerance according to the rule in (6.18) of [8], using parameters $\gamma = 0.5$ and $\eta_{Max} = 10^{-4}$ in that rule.

7.2. Convergence of Primal-Dual Newton and Weiszfeld Methods

We now give experimental evidence showing that the Primal-Dual Newton iteration achieves quadratic convergence. Figure 5 compares its convergence behavior with that of Weiszfeld algorithm on standard benchmarks (see Table 4) maintained by the CAD Benchmarking Laboratory. While our implementation is not yet optimized for speed, runtimes for the avq_small test case are still only on the order of 7 CPU seconds per Weiszfeld iteration on a 140 MHz Sun Ultra-1.[16]

In all of our tests, the residual norm tends to converge linearly in the beginning, although not always monotonically. However, when Primal-Dual iterates near the optimal solution, their residual norm converges *quadratically.* At the same time, the Weiszfeld method shows *linear* convergence

[12] The second equality in (30) relies on $\mathbf{Q} = -\mathbf{C}_0^{\mathbf{T}}\mathbf{W}\mathbf{C}_0$, which expresses the *Laplacian* $\mathbf{Q}$ in terms of the *pure* (i.e. having only 0, 1 and -1 entries) *incidence matrix* $\mathbf{C}$ and the *weight matrix* $\mathbf{W}$. In the simple case where the edge weights of the original graph are all 1, $\sum_{j=1}^{m} \frac{\mathbf{C}_i^T\mathbf{C}_i}{\eta_j}$ can be interpreted as the negative Laplacian of the graph with connectivity matrix $\mathbf{C}$ and edge weights given by η_j^{-1}. (Here, $\mathbf{E} = \mathbf{W}$.) The general case can be reduced to the simple case by writing $\mathbf{C} = \mathbf{W}\mathbf{C}_0$.

[13] This is needed only as a speedup. As seen from the experimental results below, Weiszfeld algorithm can find a rough approximation faster than Primal-Dual.

[14] Note that the graph representation of the netlist must be connected, e.g., when using an ILU preconditioner.

[15] For better results with examples of small size (say, under 1000 cells), one can solve the linear systems (7) and (29) directly; this limit can be increased if matrices are sparser. Also note that matrices arising from (29) and (7) are always symmetric and semidefinite. Thus, other Krylov Subspace methods which can be used here are BICGSTAB, QMR, SYMMLQ, etc.

[16] Iterations can be sped up considerably if we relax accuracy requirements in the solver and preconditioner. In general, many control parameters allow tradeoffs between solution quality and runtime.

everywhere. We stop the top-level iterations when the nonlinear residual has decreased by a prescribed factor (10^{-13} in this experiment), or when the iteration count reaches 40. The β_r value we used was 10^{-4}.

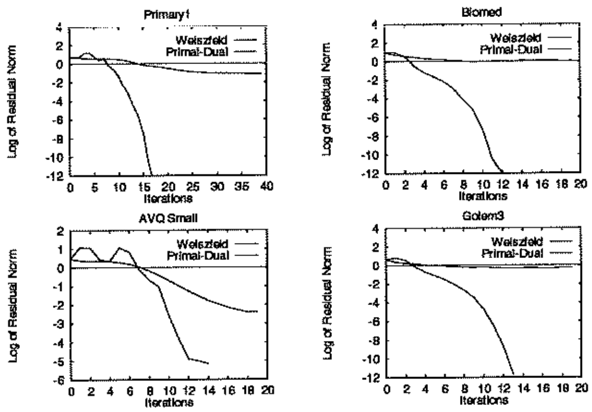

Figure 5. Comparison between convergence of Primal-Dual and Weiszfeld. We plot $\log_{10}$ of L_2 norm of the nonlinear residual against the number of top-level iterations.

We discovered (Figure 6) that more iterations are needed to reach the quadratic convergence region for smaller β_r values. However, the difference in convergence behavior between the two algorithms is more apparent for smaller β_r values.

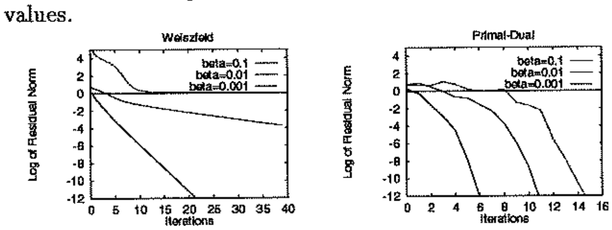

Figure 6. Dependence of convergence (Primary1 benchmark) on the value of β_r used ($10^{-1}, 10^{-2}$ and 10^{-3}). We plot $\log_{10}$ of L_2 norm of the nonlinear residual versus the number of top-level iterations.

8. CONCLUSIONS

As shown in the previous section, the Weiszfeld algorithm corresponding to GORDIAN-L is at best linearly convergent, while Primal-Dual Newton provides robust quadratic convergence.

We note that the original Weiszfeld formulation is in fact much weaker than what we have developed (in the original work, only one point is placed and there is no concept of β-regularization). Our generalization and underlying formulation have significant theoretical value in that they allow derivation of a large family of effective global optimization methods in an uniform mathematical setting. We have recently begun integration of the Primal-Dual Newton algorithm to address linear[17] wirelength minimization and a variety of alternate objectives within a standard-cell placement engine.

REFERENCES

[1] T. F. Chan, G. H. Golub and P. Mulet, "A Nonlinear Primal-Dual Method for TV-Based Image Restoration", In Proc. of ICAOS'96, 12 th Int'l Conf. on Analysis and Optimization of Systems: Images, Wavelets and PDEs, Paris, June 26-28, 1996, M. Berger et al (eds.), No. 219 in Lecture Notes in Control and Information Sciences, 1996, pp. 241-252.

[2] C. K. Cheng and E. S. Kuh, "Module Placement Based on Resistive Network Optimization", *IEEE Transactions on Computer-Aided Design*, CAD-3, 1984, pp. 218–225.

[3] A. Conn and M. Overton, "A Primal-Dual Interior Point Method for Minimizing a Sum of Euclidean Distances", Computer Science Department, New York University, *Technical Report*, 1995.

[4] U. Eckhardt, "On an Optimization Problem Related to Minimal Surfaces with Obstacles", in: R. Bulirsch, W. Oetti and J. Stoer editors; "Optimization and Optimal Control", *Lecture Notes in Mathematics* 477 Springer Verlag, 1975, pp. 95-101.

[5] U.Eckhardt, "Weber's Problem and Weiszfeld's Algorithm in General Spaces", *Mathematical Programming* 18, 1980, pp. 186-196.

[6] W. Hackbush, *Iterative Solution of Large Sparse Systems*, Springer Verlag, 1994.

[7] T. Hamada, C.-K. Cheng and P. M. Chau, "*Prime*: A Timing-Driven Placement Tool using A Piecewise Linear Resistive Network Approach", *Proc. 30th ACM/IEEE Design Automation Conference*, 1993, pp. 531-536.

[8] C. Kelley, "Iterative Methods for Linear and Nonlinear Equations", *Frontiers in Applied Mathematics*, vol. 16, SIAM, 1995.

[9] J. Kleinhans, G. Sigl, F. Johannes and K. Antreich, "GORDIAN:VLSI Placement by Quadratic Programming and Slicing Optimization." *IEEE Transactions on Computer-Aided Design.* 10 (3), March 1991, pp. 356-365.

[10] Y. Li, "A Newton Acceleration of the Weiszfeld Algorithm for Minimizing the Sum of Euclidean Distances", Cornell University, *Technical Report*, 1996.

[11] I.I. Mahmoud, K. Asakura, T. Nishibu and T. Ohtsuki, "Experimental Appraisal of Linear and Quadratic Objective Functions Effect on Force Directed Method for Analog Placement", *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E77-A (4), April 1994, pp. 719-725.

[12] E. Miehle, "Link-Length Minimization in Networks", *Operations Research*, 6, 1958, pp. 232-243.

[13] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, New York, Academic Press, 1970.

[14] B. M. Riess, K. Doll and F. M. Johannes, "Partitioning Very Large Circuits Using Analytical Placement Techniques", *Proc. 31st ACM/IEEE Design Automation Conference*, 1994, pp. 646–651.

[15] B. Riess and G. Ettelt, "SPEED: Fast and Efficient Timing Driven Placement", *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 1, 1995, pp. 377–380.

[16] G. Sigl, K. Doll and F. Johannes, "Analytical Placement: A Linear or a Quadratic Objective Function?" *Proc. 28th ACM/IEEE Design Automation Conference*, 1991, pp. 427-432.

[17] W. Swartz and C. Sechen. "Timing Driven Placement for Large Standard Cell Circuits", *Proc. 32nd ACM/IEEE Design Automation Conference*, 1995, pp. 211–215.

[18] R. S. Tsay, E. Kuh, "A Unified Approach to Partitioning and Placement", *IEEE Transactions on Circuits and Systems*, Vol.38, No.5, May 1991. pp. 521-633.

[19] E. Weiszfeld, "Sur le Point pour Lequel la Somme des Distances de n Points Données est Minimum." *Tôhoku Mathematics J.* 43, 1937, pp. 355-386.

[17] As well as for arbitrary real exponent $\alpha \geq 1$.