Analytical Placement of Hypergraphs — I

Andrew Kennings and Igor Markov

Abstract

Global placement of hypergraphs is critical to modern physical implementation methodology for large timing-driven designs. Placement results are typically evaluated in terms of the half-perimeter wirelengths of hyperedges in the original circuit hypergraph. However, existing analytical placers rely on a heuristic transformation into a graph to enable classical linear and quadratic edgelength minimizations. As a result, optimizing a wrong objective leads to grossly suboptimal solutions, as confirmed by our experiments.

We present a novel approach to minimization of the half-perimeter hyperedge wirelength without graph models using provably good approximations of multivariate piece-wise linear functions. Our algorithm is tested on industrial instances, particularly, in a top-down placement context.

1 Introduction

Analytical placers are increasingly important in physical design as process technology advances and design complexity increases. The fact that interconnect delays dominate device and cell delays in large netlists entails a global optimization setting that focuses on the area and performance costs of interconnect. Analytical placers find locations of modules (individual cells or macros) so as to minimize a wirelength estimate representing a cumulative measure of interconnect delay and utilization; some algorithms also minimize specific timing-critical paths. To be implementable, placement solutions must satisfy various combinatorial constraints, e.g., use only prescribed module locations, avoid module overlaps etc. These constraints are temporarily relaxed for analytical placement and are later taken care of by specialized code which may itself make repeated calls to analytical placement. For example, a placement with excessive module overlaps or overutilization of routing resources can be spread out using *min-cut partitioning* [4, 21, 23, 22, 11], *transportation* [12, 26] or *force-directed techniques* [6].

Circuits are represented by weighted hypergraphs $G_H(V_H, E_H)$ with vertices $V_H = \{v_1, v_2, \cdots, v_n\}$ corresponding to modules, and hyperedges $E_H = \{e_1, e_2, \cdots, e_m\}$ corresponding to signal nets. Vertex weights correspond to module areas, while hyperedge weights correspond to criticalities and/or multiplicities. Vertices are either *fixed* or *free*. Hyperedge $e_i \in E$ connects $p_i \geq 2$ vertices and each vertex $v_j \in V$ is incident to $d_j \geq 0$ hyperedges. p_i and d_j are respectively called the vertex and hyperedge *degrees*, and are typically very small.[1] We say that v_j has d_j pins, and e_i has p_i pins, for a total of $P = \sum_{j=1}^{m} d_j = \sum_{i=1}^{n} p_i$ pins in the hypergraph. Module placements in x and y directions are captured by the *placement vectors* $\mathbf{x} = (x_1, \ldots, y_n)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_n)$.

Placement results are typically evaluated using the half-perimeter wirelength (HPWL) of the circuit hypergraph. Although direct minimization of HPWL has been attempted via linear programming [9, 27], this approach has proven too computationally intensive due to the sizes of the resulting linear programs. Thus, it is typical to transform the circuit hypergraph into a graph prior to solving, or even formulating, any optimization problems. Each hyperedge is modeled by a *star* [13, 19] or a *clique* [10, 24, 23] of edges. A commonly used *star model* adds a new *center* vertex and represents the original net by edges connecting the center to previously existing vertices. The *clique model* connects all pairs of vertices incident to the original hyperedge by edges of equal weight.

[1] In other words, circuit hypergraphs are *sparse*: the number of hyperedges varies from 0.8x to 1.5x of the number of free vertices and half the number of terminals since each module typically has only one or two outputs, each of which represents the source of a new signal net. In current applications, circuit hypergraphs may have up to one million device-, gate- and cell-level modules (instances of sizes 1,000 through 1,000,000 are equally important), yet average vertex and hyperedge degrees typically between 3 and 5; 50-80% of hyperedges have degree 2. Few vertices have degrees much higher than the average; these may correspond to large macro blocks. Very few hyperedges are extremely large, with degrees reaching the hundreds or thousands; these represent clock, reset, test and other global nets.

Analytical placers based on the graph transformation have proven quite popular. Early algorithms [24, 23] based on squared wirelength objectives were attractive since they relied on solving a single system of linear equations to obtain a global module placement. However, the squared wirelength objective tends to overemphasize the minimization of long wires at the expense of short wires; this increases the demand on routing resources, thereby leading to a poorer layout. Mahmoud et al. [16] compared linear and squared wirelength objectives for analog placement with the conclusion that the linear wirelength objective is superior. Hence, recent analytical placers [20, 1] rely on a linear wirelength objective that is optimized, e.g., by iterated approximations with quadratic wirelength objectives [20].

In this paper, we revisit the direct minimization of half-perimeter wirelength, by working with the circuit hypergraph without relying on graph models of hyperedges. Our strategy is based on the observation that the HPWL is a convex, but not everywhere differentiable function with singularities arising from "min" and "max" functions. We advance the *function smoothing* techniques, recently proposed in [3] for various VLSI layout problems, to handle HPWL.

The remainder of this paper is structured as follows. Section 2 demonstrates the difficulties inherent in the direct minimization of HPWL due to its non-differentiability. Reductions to graphs and previous work on analytical placement are described. Section 3 discusses important practical considerations, in particular, the utility of preconditioners based on direct solvers and the importance of the Newton-type methods for linear wirelength minimization. In Section 4 we propose novel piece-wise linear multivariate regularizations that result in approximations of HPWL with arbitrarily small relative error. This enables graph-free HPWL minimization via unconstrained smooth and convex optimization. Section 5 describes our analytical placement engine and presents numerical results from testcases from the industry in the context of top-down placement. Section 6 concludes with the directions of ongoing work.

2 Review of analytical placement

2.1 Hypergraph placement

Let C_n be the set of indices of hypergraph vertices incident to net $e_n \in E_H$. The x-direction HPWL estimate is given by

$$HPWL_x(\mathbf{x}) = \sum_{e_n \in E_H} \left(\max_{i \in C_n}\{x_i\} - \min_{i \in C_n}\{x_i\} \right) = \sum_{e_n \in E_H} \max_{i,j \in C_n} |x_i - x_j|. \tag{1}$$

HPWL is a *convex* function of $\mathbf{x}$ since $|x_i - x_j|$ is convex for all i, j. However, HPWL is not *strictly convex* and most often has uncountably many minimizers. HPWL minimization requires fixed vertices, with at least two different locations (otherwise placing all vertices to the same location will achieve an optimal solution with wirelength 0). Fixed vertices in circuit hypergraph are conveniently provided by I/O pads and external pins. The non-differentiability of the min and max functions disables classic smooth minimization techniques such as the Newton's method. It can be shown that any single iteration of *the steepest descent* will end up in a solution where gradient is not well-defined and the new steepest descent direction is not easily available.[2]

In [27], the HPWL estimate is converted into an equivalent linear program (LP) by adding, for each net e_j, upper and lower bound variables u_j and v_j. The cost of the net is measured as the difference between the two, and in an optimal solution each u_j and v_j will be equal to the rightmost and leftmost module locations of net e_j. Each variable comes with p_i inequality constraints that restrict $u_j(v_j)$ to be larger (smaller) than the locations of every module incident to the net. Thus, for a circuit with n nets and m modules, the linear program will have $m + 2n$ variables and $2P$ constraints. While LP will return optimal placements, the large instance sizes effectively preclude any application to fast placement of large hypergraphs.

[2] An even bigger problem is demonstrated by a 3-clique of free vertices that is connected to a fixed vertex by one edge. As soon as all free vertices are located at the same point, no movement of a single vertex can improve wirelength while moving all three toward the fixed vertex will lead to the optimal placement. This shows that computing partials, even one-sided, may not give the steepest descent direction.

2.2 Reduction to graphs

Given the lack of relevant placement techniques for hypergraphs, heuristic modeling of circuit *hypergraphs* by *graphs* is common. In such models, every hyperedge is represented by a group of equally weighted edges. The *unoriented star model* adds a new *center* vertex and represents the original net by edges connecting the center to previously existing vertices (modules) [13, 19]. The *clique model* (see [2, 10, 14] for a review) connects all pairs of vertices (modules) incident to the original hyperedge by edges of non-unit weight.[3] Clique models of large hyperedges become prohibitively expensive due to the quadratic edge count. Therefore, large hyperedges are typically modeled by stars or dropped completely.

Wirelength estimates for individual edges of a graph are weighted by edge weights and added up to produce a total wirelength estimate. For an edge that connects modules with abscissae x_1 and x_2, the most popular x-wirelength estimates are (a) *linear* (Manhattan) $|x_1 - x_2|$ and (b) *square* (Euclidean) $(x_1 - x_2)^2$; the y-wirelength is computed in the same way and added to the x-wirelength. While both functions are *convex*, only the square wirelength is *strictly convex* for connected graphs, which guarantees a unique minimizer.

Minimization of squared (quadratic) wirelength

$$\min_{\mathbf{x}}\{\textstyle\sum_{i>j} a_{ij}(x_i - x_j)^2 : \mathbf{Hx} = \mathbf{b}\} \tag{2}$$

($\mathbf{H}$ represents various linear constraints) is the easiest because, without going into further detail (see [24, 23]), the unique solution to this optimization problem can be obtained by solving a single system of linear equations, either positive-definite or symmetric-indefinite depending on the approach. High-quality implementations of linear system solvers are freely available and their performance is good enough for most VLSI placement applications. Unfortunately, the squared wirelength objective tends to provide lower-quality placements.

Linear wirelength minimization has been proposed and achieves better placements while still using the above reduction of circuit hypergraph to a graph. Linear wirelength minimization is formulated as

$$\min_{\mathbf{x}}\{\textstyle\sum_{i>j} a_{ij}|x_i - x_j| : \mathbf{Hx} = \mathbf{b}\} \tag{3}$$

and is not amenable to Newton-type methods since it is neither differentiable nor strictly convex. It can be solved by GORDIAN-L [20] using iterated quadratic minimizations

$$\min_{\mathbf{x}^\nu}\{\textstyle\sum_{i>j} \frac{a_{ij}}{|x_i^{\nu-1} - x_j^{\nu-1}|}(x_i^\nu - x_j^\nu)^2 : \mathbf{Hx}^\nu = \mathbf{b}\} \tag{4}$$

where $\mathbf{x}^{\nu-1}$ and $\mathbf{x}^\nu$ denote the vectors of vertex positions at iterations $\nu - 1$ and ν. A quadratic objective is used to avoid the non-differentiability of the objective of (3), but the coefficients of the objective are updated at each iteration to approximate the linear wirelength estimate.

As an alternative, the following *regularization* of (3) has been proposed in [1]

$$\min_{\mathbf{x}}\{\textstyle\sum_{i>j} a_{ij}\sqrt{(x_i - x_j)^2 + \beta} : \mathbf{Hx} = \mathbf{b}\} \tag{5}$$

and offers two solution approaches: with a linearly-convergent fixed-point method following Eckardt's [7, 8] generalization of the Weiszfeld algorithm [28], and with a novel primal-dual Newton method having quadratic convergence. Numerical testing in [1] illustrates the tradeoffs in values of $\beta > 0$ versus time and difficulty, and the GORDIAN-L heuristic is interpreted as a special case $\beta = 0$ of a fixed-point method that has proven linear convergence for $\beta > 0$. This can be seen by computing the partials of the objective in (5), setting $\beta = 0$ and comparing to the partials of (4).

2.3 β-regularization

Define[4] β-regularization of $f(x) = |x|$ as $f_\beta(x) = \sqrt{x^2 + \beta}$, which is a twice differentiable strictly convex function (thus amenable to Newton-type methods) that overestimates $f(x)$ at most by $\sqrt{\beta}$. Similar techniques apply to

[3] A shortcut computation is available when the clique model is used with squared wirelength [20]. Given a hyperedge of degree p, the squared wirelength of a clique with uniform edge weight $\frac{2}{p}$ equals the squared wirelength of a star whose center vertex is placed at the center of gravity of its p neighboring vertices. In this context, the center of the star is not an independently placed vertex; this is in contrast to the general star model.

[4] See [3] for more general definitions.

arbitrary convex univariate piece-wise linear functions [3], in particular, as $\beta \to 0$, the unique minimizer of the regularized function approaches minimizers of the original function.

β-regularization extends to a wider class of possibly multivariate convex piece-wise differentiable functions with cusps due to an absolute value or more general case analysis in their symbolic representation. Substituting a symbolic fragment, e.g., $|x|$, with a regularization leads to a smooth function and allows to regularize multivariate functions.

The convexity properties and the limit behavior of the fragment regularizations often extend to the resulting function through sums, products etc. In particular, β-regularized linear wirelength is twice differentiable and, for connected graphs, strictly convex (hence a unique minimizer).

Example: The function $\max\{a,b\} = \frac{a+b+|a-b|}{2}$ can be regularized with $\frac{a+b+\sqrt{|a-b|^2+\beta}}{2}$, and $\min\{a,b\} = \frac{a+b-|a-b|}{2}$ with $\frac{a+b-\sqrt{|a-b|^2+\beta}}{2}$.

In particular, for $f(t) = \max\{0, (t-t_0)\}$, $f_\beta(t) = \frac{1}{2}((t-t_0) + \sqrt{(t-t_0)^2+\beta})$, which allows to represent timing-critical path constraints by adding *penalty* terms to the objective function that turn to zero when the constraints are satisfied and grow rapidly with violations.

3 Practical considerations and pitfalls

While the utility of analytical placers has been empirically established, several inherent subtleties can potentially impair implementations of novel algorithms (or even well-established ones), particularly given new application contexts. One notable problem is the deterioration of convergence which results in iterations with no improvement and wasted CPU time. In this section, we provide two constructions that motivate (a) the utility of preconditioners based on exact (versus iterative) solvers, and (b) the superiority of Newton-type methods to the steepest descent method for linear wirelength minimization. The latter, in particular, shows the benefit of optimizing twice differentiable functions with "second-order" information available with the Hessian.

3.1 Chain graphs

Consider a chain graph with unit weight edges, n free vertices and two end vertices fixed at χ_0 and $\chi_1 \neq \chi_0$. All free vertices are initially placed at χ_0. While the initial solution is optimal for linear wirelength (any solution with all free nodes placed between χ_0 and χ_1 in the chain order is optimal), the unique minimizer for the squared wirelength and for β- regularized linear wirelength has the free nodes distributed uniformly between χ_0 and χ_1 in the chain order. Indeed, equivalent unconstrained formulations are given by minimization of

$$f_q(\mathbf{x}) = (\chi_0 - x_1)^2 + (x_1 - x_2)^2 + \ldots + (x_n - \chi_1)^2$$

and

$$f_\beta(\mathbf{x}) = \sqrt{(\chi_0 - x_1)^2 + \beta} + \sqrt{(x_1 - x_2)^2 + \beta} + \ldots + \sqrt{(x_n - \chi_1)^2 + \beta}$$

The j-th partials are given by $\frac{\partial f_q}{\partial x_j} = 4x_j - 2x_{j-1} - 2x_{j+1}$ and $\frac{\partial f_\beta}{\partial x_j} = \frac{2(x_j - x_{j-1})}{\sqrt{(x_j - x_{j-1})^2+\beta}} - \frac{2(x_{j+1} - x_j)}{\sqrt{(x_{j+1} - x_j)^2+\beta}}$. Since $f_q(\mathbf{x})$ and $f_\beta(\mathbf{x})$ are differentiable and *strictly convex*, the necessary and sufficient condition for $\mathbf{x}$ to be the minimum is for it to also zero all partials. Since the above partials turn to zero when $x_j = (x_{j+1} + x_{j-1})/2$, the unique minimizer of each function will space $\{x_j\}$ evenly between χ_0 and χ_1 according to the chain order.

No matter what a particular iterative method does, most existing algorithms are *local* in the sense that they look at the neighbors of each vertex to find locations that improve wirelength of incident edges (or edges incident to neighbors, etc.). Since all free vertices are initially placed at χ_0, all of them except one have all neighbors placed at the same location. Therefore, the first iteration of a typical iterative placer, being concerned only with immediate neighborhoods, is only capable of moving one vertex; the k^{th} iteration will move no more than k vertices. The cost of one iteration (i.e., the discovery of the vertices to be moved) is linear in most placers, implying an overall quadratic complexity that is most likely unacceptable.[5] During quadratic minimization, successive iterations are made inside the linear solver, e.g., using successive overrelaxation. One particular reason for the above problem

[5] While theoretic complexities of some popular methods may be even higher, "early stops" after only a few initial iterations make them practical. However, with the chain example no early stop is clearly possible.

to not be critical is that iterative linear solvers often use preconditioners based on algorithms for exact solvers, e.g., LU-factorization, which makes them insensitive to the problem discussed.

3.2 Cliques and clusters

The "second-order" information provided by the Hessian in Newton-type methods in *linear wirelength minimization* appears critical to iteration count and run time. A very simple example, which can be found embedded in typical circuit hypergraphs, shows that the Newton method can achieve an optimal placement in *one* iteration, while the steepest descent method is slowed by the "lack of look ahead" and moves vertices in numerous small-stepped iterations.[6]

Consider a clique of $n \geq 3$ free vertices, initially all located at $x = \chi_0$, with all edges of unit weight. Connect a single vertex fixed at $\chi_1 \neq \chi_0$ to one of the free vertices (whose location is represented by x_1) by an edge of unit weight. The unique placement minimizing regularized (and original too) linear wirelength has all free vertices located at χ_1.

An equivalent unconstrained problem minimizes

$$\sqrt{(\chi_1 - x_1)^2 + \beta} + \sum_{i=1}^{n} \sum_{j>i} \sqrt{(x_j - x_i)^2 + \beta} \tag{6}$$

whose gradient is given by

$$\frac{\partial f}{\partial \mathbf{x}} = -\frac{(\chi_1 - x_1)}{\sqrt{(\chi_1 - x_1)^2 + \beta}} \mathbf{b}_1 + \sum_{i=1}^{n} \sum_{j>i} \frac{(x_j - x_i)}{\sqrt{(x_j - x_i)^2 + \beta}} \mathbf{b}_{ji}$$

where $\mathbf{b}_{ji} = [\ 0\ \ \ldots\ \ 1\ \ 0\ \ \ldots\ \ -1\ \ 0\ \ \ldots\ \ 0\ ]^T$ and $\mathbf{b}_1 = [\ 1\ \ 0\ \ \ldots\ \ 0\ ]^T$. "1" and "-1" occupy the j-th and i-th entries in $\mathbf{b}_{ji}$ respectively.

Due to the terms $(x_j - x_i)$ in the numerators of the gradient components, all components except for the first will zero out due to all free vertices in the clique having identical initial locations. Therefore the steepest descent method will first move only one vertex by a very small step closer to the fixed vertex. At the next iteration the remainder of the clique will move by an even smaller step, and this 2-step pattern will continue until the clique is close enough to the fixed vertex. Clearly, this will take at least a linear (in terms of $|\chi_0 - \chi_1|$) number of iterations.

By contrast, the Newton method will converge in just one iteration by using the "second-order" information available in the Hessian:

$$\frac{\partial^2 f}{\partial \mathbf{x}^2} = \left[\frac{1}{[(\chi_1 - x_1)^2 + \beta]^{1/2}} - \frac{(\chi_1 - x_1)^2}{[(\chi_1 - x_1)^2 + \beta]^{3/2}} \right] \mathbf{b}_1 \mathbf{b}_1^T$$
$$+ \sum_{i=1}^{n} \sum_{j>i} \left[\frac{1}{[(x_j - x_i)^2 + \beta]^{1/2}} - \frac{(x_j - x_i)^2}{[(x_j - x_i)^2 + \beta]^{3/2}} \right] \mathbf{b}_{ji} \mathbf{b}_{ji}^T$$

Given the above initial placement, a number of terms in the Hessian will disappear – the coefficients for every term in the double summation will become $1/\sqrt{\beta}$.

Let the coefficient of the $\mathbf{b}_1 \mathbf{b}_1^T$ term be denoted by σ, and let $1/\sqrt{\beta}$ be denoted by ω. The Hessian written in matrix form is given by:

$$\begin{bmatrix} \sigma + (n-1)\omega & -\omega & -\omega & \cdots & -\omega \\ -\omega & (n-1)\omega & -\omega & \cdots & -\omega \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ -\omega & -\omega & \cdots & -\omega & (n-1)\omega \end{bmatrix} \tag{7}$$

The Hessian is positive definite and the right-hand side is not zero, implying a unique non-zero solution. It is easy to show that all the components of $\delta \mathbf{x}$ are equal to each other. All vertices in the clique will move in unison to the optimal location in one iteration using a reasonable line search method.

This phenomenon can be replicated with sufficiently dense clusters instead of cliques, and multiple instances are typically encountered in large circuit hypergraphs. While the *initial* solution may be different from what we considered, iterations of analytical placers bring cliques and tightly connected clusters closer together. Therefore, even after several large-stepped iterations, convergence of steepest descent may deteriorate.

[6] The comparison that we now make is different from traditional comparison of *linear convergence* of the steepest descent versus *quadratic convergence* of Newton-type methods.

4 Regularizations of HPWL

Motivated by the convexity of HPWL, we seek an efficient means of minimizing it without graph models. To enable traditional smooth and strictly convex minimization techniques, we *regularize* HPWL with arbitrarily small relative error, providing a unique minimizer and second-order information useable by efficient Newton-type methods.

4.1 Sequential β-regularization

(5) applies β-regularization [1, 3] to approximate graph edgelength by a smooth strictly convex function, e.g., in the context of graph models of circuit hypergraphs. The approximation overestimates the original function by at most $\sqrt{\beta}$ and therefore accurately approximates HPWL for 2-pin nets.

The difficulty in HPWL approximation is for hyperedges of degree ≥ 3, where HPWL is defined using maximum and minimum of more than two arguments. To resolve this difficulty, we rewrite those terms as sequences of nested two-variable maximum and minimum functions then regularize $\max\{a,b\} = \frac{a+b+|a-b|}{2}$ with $\frac{a+b+\sqrt{|a-b|^2+\beta}}{2}$ and similarly for min.[7] Unfortunately, this regularization of HPWL is quite difficult to compute efficiently. Moreover, it is not a symmetric function of vertex locations if hyperedge degree ≥ 3, leading to a bias in search directions produced by gradients and Hessians which will slow convergence with respect to true HPWL. A simpler regularization is available, symmetric and computationally competitive.

4.2 Multivariate p-regularization

In what follows we will use the well-known fact that L_p-norms converge to the L_∞-norm as $p \to \infty$.

This is illustrated in Figure 1, where unit-norm curves ("unit circles") for L_p-norms on the plane are seen converging to the the "unit circle" $\| (a_1, a_2) \|_\infty = \max(a_1, a_2) = 1$.

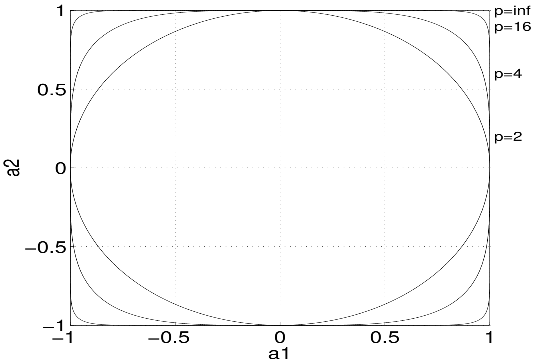

Figure 1: Unit-norm curves for L_p-norms.

For $p > 1$ and k-dimensional vector $\mathbf{a} = (a_1, a_2, \ldots, a_k)$, let $\| \mathbf{a} \|_p = (\sum_{j=1}^{k} |a_j|^p)^{1/p}$ and recall

Fact 1

(a) $\| \mathbf{a} \|_p \geq \| \mathbf{a} \|_{p+1} \geq \| \mathbf{a} \|_\infty$

(b) $\lim_{p\to\infty} \| \mathbf{a} \|_p = \| \mathbf{a} \|_\infty = \max_j |a_j|$, in particular, $\| \mathbf{a} \|_\infty \leq \| \mathbf{a} \|_p \leq k^{1/p} \| \mathbf{a} \|_\infty$

(c) $\varrho(\mathbf{a}) = \| \mathbf{a} \|_p$ is strictly convex and infinitely differentiable except at $\mathbf{a} = \mathbf{0}$

Proof $\max\{|a_1|, \cdots, |a_k|\} = (\max\{|a_1|^p, \cdots, |a_k|^p\})^{1/p} \leq (|a_1|^p + \cdots + |a_k|^p)^{1/p} \leq (k \max\{|a_1|^p, \cdots, |a_k|^p\})^{1/p} = k^{1/p} \max\{|a_1|, \cdots, |a_k|\}$ implies $\| \mathbf{a} \|_p \geq \| \mathbf{a} \|_\infty$ and proves **(b)**. $\| \mathbf{a} \|_p \geq \| \mathbf{a} \|_{p+1}$ can be reduced by induction to the case $k = 2$. Assume positive a_i, x and y. The induction step can now be accomplished by introducing $d_1 = (a_2^p + \ldots + a_k^p)^{1/p}$ and $d_2 = (a_2^{p+1} + \ldots + a_k^{p+1})^{1/(p+1)}$ for which $d_1 \geq d_2$ holds by induction hypothesis. Then $(a_1^p + a_2^p + \ldots + a_k^p)^{1/p} \geq (a_1^p + d_1^p)^{1/p} \geq (a_1^p + d_2^p)^{1/p} \geq (a_1^{p+1} + d_2^{p+1})^{1/(p+1)} \geq (a_1^{p+1} + a_2^{p+1} + \ldots + a_k^{p+1})^{1/(p+1)}$, the middle inequality being case $k = 2$: $(x^p + y^p)^{1/p} \geq (x^{p+1} + y^{p+1})^{1/(p+1)}$. Equivalently $(x^p + y^p)^{p+1} = (x^p + y^p)(x^p + y^p)^p \geq (x^{p+1} + y^{p+1})^p$, where both sides can be interpreted as binomials with equal number of terms. It now suffices to prove that the inequality holds term by term $(x^p + y^p)(x^p)^i (y^p)^{p-i} \geq (x^{p+1})^i (y^{p+1})^{p-i}$ (equal constants cancelled out), which follows from $(x^p + y^p) \geq \max\{x^p, y^p\} \geq x^i y^{p-i}$.

The differentiability can be proven by taking partials, e.g., $\frac{\partial f}{\partial a_i} = p a_i^{p-1} (\sum_{j=1}^{k} |a_j|^p)^{\frac{1}{p}-1}$. Namely, all n-th partials will be polynomials of $a_i, i = 1..k$ and $(\sum_{j=1}^{k} |a_j|^p)^{\frac{1}{p}-l}, l = 1..n$. The latter have a pole at $\mathbf{a} = 0$ and are differentiable elsewhere since $\frac{1}{p} - l < 0$. Strict convexity can be deduced by considering second partials. □

[7] E.g., for a 3-pin net, $\max\{|x_1 - x_2|, |x_1 - x_3|, |x_2 - x_3|\}$ can be rewritten as $\max\{|x_1 - x_2|, max\{|x_1 - x_3|, \max\{|x_2 - x_3|\}\}\}$ to which β-regularization can be applied sequentially.

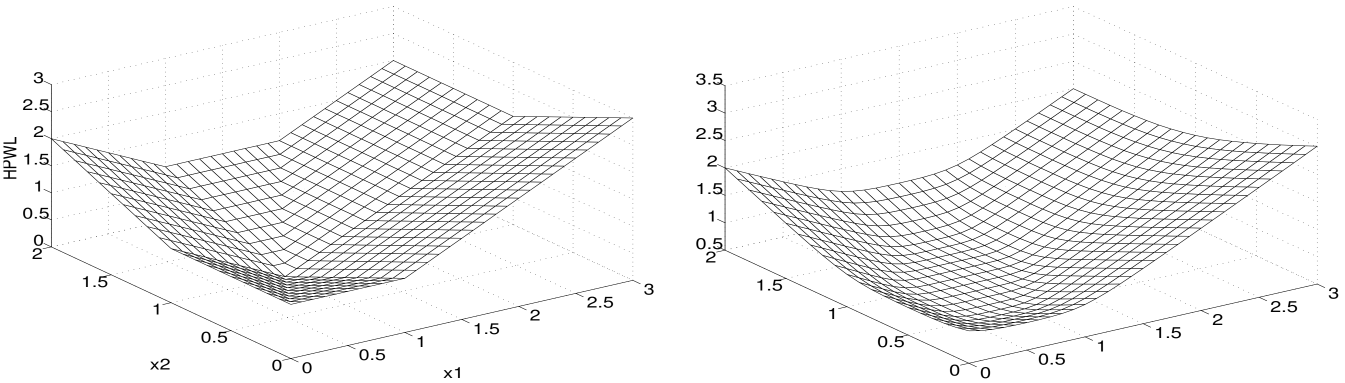

Figure 2: Illustration of HPWL(left) for a 3-pin net; $HPWL = \max\{x_1, x_2, 1\} - \min\{x_1, x_2, 1\} = \max\{|x_1 - x_2|, |x_1 - 1|, |x_2 - 1|\}$ over the intervals $x_1 \in [0,3]$ and $x_2 \in [0,2]$. The p-regularization (right) is $((|x_1 - x_2|^p + |x_1 - 1|^p + |x_2 - 1|^p + \beta)^{1/p}$. Here $p = 8$ and $\beta = 1.0e6$.

To approximate the HPWL of an m-pin net, we enumerate all $\frac{m(m-1)}{2}$ pairwise distances of the form $x_i - x_j$ and rewrite the HPWL as their L_∞-norm (see Equation 1), then approximate with L_p-norms.[8] The resulting function will be convex as composition of linear and convex functions.

The multiplicative upper bound of $(\frac{m(m-1)}{2})^{1/p}$ on overestimation for HPWL of one m-pin net, as given by Fact 1(**b**), appears very loose since all $\frac{m(m-1)}{2}$ distances cannot be equal unless being 0. Tighter bounds can be derived given that the L_∞-norm is applied only to vectors, whose coordinates are all pairwise distances between m points on the real line. Such tighter bounds for our regularization of the HPWL of small nets are given in Table 1.[9]

Net size	Upper bounds		Maximal relative overestimation					
m	loose	tight	p=2	p=4	p=8	p=16	p=32	p=64
3	$3^{1/p}$	$2^{1/p}$	41%	19%	9%	4%	2%	1%
4	$6^{1/p}$	$4^{1/p}$	100%	41%	19%	9%	4%	2%
5	$10^{1/p}$	$6^{1/p}$	145%	56%	25%	12%	6%	3%
6	$15^{1/p}$	$9^{1/p}$	200%	73%	32%	15%	7%	3%

Table 1: Upper bounds for HPWL overestimation by p-regularization for single nets.

As follows from Fact 1(**c**), overestimating $\max\{|a_1|, |a_2|, \cdots, |a_k|\}$ by its p-regularization $(|a_1|^p + |a_2|^p + \cdots + |a_k|^p)^{1/p}$ removes all nondifferentiabilities except for $\mathbf{a} = \mathbf{0}$. Additional overestimation by β-regularization $(|a_1|^p + |a_2|^p + \cdots + |a_k|^p + \beta)^{1/p}$ smoothens the function at $\mathbf{a} = \mathbf{0}$.

The resulting approximation of HPWL

$$HPWL(\mathbf{x}) \leq HPWL_{reg}(\mathbf{x}) = \sum_{e_n \in E_H} (\sum_{i,j}^{|C_n|} |x_i - x_j|^p + \beta)^{1/p} \tag{8}$$

is a smooth and strictly convex[10] upper bound on exact HPWL with arbitrary small relative error of approximation as $p \to \infty$ and $\beta \to 0$. An illustration of the combined p- and β-regularization for the HPWL example in Figure 2. Finally, we note that restricting p to powers of 2 allows for particularly effective computations.

[8] Nets with prohibitively big $\frac{m(m-1)}{2}$ can be ignored or handled with *sequential* β-regularization from subsection 4.1. Yet, a simpler approach appears to be as efficient: L_p-norm can be taken over edges of a star model with the center located at the barycenter of the net's pins.

[9] E.g., for a 3-pin net, the L_p-norm is $(|x_1 - x_2|^p + |x_1 - x_3|^p + |x_2 - x_3|^p)^{1/p}$. Clearly, the three terms cannot be equal (unless 0). Assume an arbitrary ordering $x_1 \leq x_2 \leq x_3$ and maximize the L_p-norm for fixed x_1 and x_3: the maximal overestimation $2^{1/p}$ is reached when x_2 is placed on top of either x_1 or x_3. A similar argument for 4-pin nets yields a tight bound of $4^{1/p}$ even though there are 6 terms involved in the L_p-norm for a 4-pin net.

[10] Strict convexity requires that all free vertices be reachable from fixed vertices. Otherwise there will be multiple optimal solutions, contradicting strict convexity.

4.3 Scale-independent β

To determine specific values of β to evaluate the regularization or its derivatives, [3] multiplied the p-th exponent of the maximal $|x|$ value associated to problem instances by an instance-independent value β_0.

However, with the newly proposed regularization, such a selection results in minimum function values dependent on p. Therefore, β is computed as the product of the p-th exponent of the maximal distance value with the p-th exponent of the instance-independent value β_0.

5 Experimental results

5.1 Numerical solver

We tested the proposed approach with a modified Newton method for minimizing our HPWL approximation that is both smooth and strictly convex. Both the objective and the gradient can be computed analytically, but the Hessian computations are hard and time-consuming. Nevertheless, given the necessity of handling cliques and clusters of modules, the second-order information provided by the Hessian is critical for fast convergence as explained in Section 3.

To this end, we have implemented a limited memory quasi-Newton method which can be viewed as a means of extending the simple conjugate gradient method by using additional storage to accelerate convergence or as an implementation of a quasi-Newton method in which storage is restricted [15, 17]. Simply stated we use limited memory BFGS updates [18, 15]. Let the solution at iteration k be x_k and define $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$. At iteration k, a search direction is computed as $d_k = -H_k g_k$ where H_k is an inverse Hessian approximation. The iterate is updated as $x_{k+1} = x_k + \alpha_k d_k$ where α_k is determined from a line search to satisfy descent conditions. At each iteration, an approximation to the inverse of the Hessian is computed as $H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T$ ($\rho_k = 1/y_k^T s_k$ and $V_k = I - \rho y_k s_k^T$); H_k is not represented explicitly, but rather computed by storing $m \ll n$ pairs $\{s_k, y_k\}$ that defines H_k implicitly through the BFGS update formula. Since $m \ll n$, once the storage space is exhausted, the oldest pair $\{s_{k-m}, y_{k-m}\}$ is discarded to make room for the newest one $\{s_k, y_k\}$.

Our implementation maintains storage for seven past iterations. Iterations continue until (i) a prescribed iteration limit is exceeded OR (ii) the improvement in the objective function drops below a prescribed threshold (indicating little progression to optimality) OR (iii) the gradient norm falls below a prescribed threshold.

5.2 Results

Our experimental testbed includes five testcases provided by the industry (see Table 2) which are either original ("top-level") placement instances or have arisen on further levels of top-down placement. The proposed placement algorithm named BoxPlace-1 has been implemented in C++, compiled with the SunPro CC 4.2 compiler (optimization level `-O5`) and executed within Solaris 2.6 operating system on a Sun Ultra-1 workstation running at 140MHz. Our regularizations have been used with $p = 16$ and $\beta_0 = 0.01$.

Instance	Modules		Nets
	Fixed	Free	
test0	76	200	242
test1	545	2686	2840
test2	2155	6739	7330
test3	2191	3205	3835
test4	6545	17380	20902

Table 2: Testcase parameters

Several of our test circuits had disconnected (bonus) cells not reachable from fixed terminals. To prevent multiple optimal placement solutions and subsequent breakdown of numerical solver, we assured that only free vertices reachable from fixed vertices have been passed to the solver. Others have been placed in the center of the layout to minimize WL.

To test the global convergence of the proposed algorithm, we have produced placements from multiple random initial starting points. The iteration limit of the numerical solver was set to 200 in those runs to permit full progression. Alternatively, we started with a "one-point" solution placing all vertices into the same location. This yielded much better *initial* wirelength and faster convergence to a solution of essentially the same quality as in the randomized experiment. The same placement instances have been solved by minimizing quadratic wirelength. While quadratic minimization is certainly faster, the resulting linear wirelength was significantly worse than that produced by our analytical placer. Results presented in Table 3 include initial and final wirelengths observed in the three experiments as well as cumulative run times for x- and y-directions. Finally, we were able to find optimal placements for test0 and test1 in 23.2 sec and 64 min respectively by formulating these instances as linear

	BoxPlace-1 from random		... from "one-point"		Quadratic		Weiszfeld	
	ini/**final** WL	CPU⊕	ini/**final** WL	CPU⊕	WL	CPU⊕	WL	CPU⊕
test0	5.93e7/6.39e6	2.3	6.72e6/**6.32e6**	2.3	7.66e6	0.22	6.76e6	1.1
test1	3.07e8/3.13e7	111.4	3.51e7/**3.11e7**	58.6	4.20e7	7.5	3.88e7	24.4
test2	6.27e6/2.31e6	175.1	2.69e6/**2.31e6**	63.2	2.75e6	39.7	2.7e6	90.8
test3	5.37e6/1.45e6	88.7	1.91e6/**1.44e6**	32.3	1.61e6	10.8	1.96e6	42.4
test4	3.75e7/1.47e7	565.9	1.50e7/**1.34e7**	230.9	1.50e7	144.2	1.62e7	360.5

Table 3: Placement results for the BoxPlace-1 algorithm compared against quadratic minimization. Total HPWL (the sum of x- and y-values) is presented with initial/final values.

programs and solving with the public domain package LPSolve. The optimal HPWL values of $5.03e6$ and $2.75e6$ allow to estimate the suboptimality of solutions produced by fast placement algorithms. LPSolve was unable to solve larger linear programs in 3 days on an Ultra-1 workstation with sufficient memory.

During our experiments we have discovered that the applicability of analytical wirelength minimization in top-down placement is somewhat limited by the requirement of sufficiently many fixed vertices. Indeed, with too few fixed vertices, the "one-point" solution minimized wirelength since most nets were not incident to terminals and had zero wirelength. In circuits which are not pad driven, analytical wirelength minimization may not be able to separate cells until sufficiently many terminals are generated during several top-down levels. In such circumstances, min-cut partitioners apparently have little competition.

6 Conclusions and ongoing work

Quadratic placers and linear variants are popular primarily due to their ease of implementation and empirical successes. However, such methods are strictly indirect in their treatment of the minimization of half-perimeter wirelength and result in grossly suboptimal solutions, as evidenced by our results in Table 3. Graph-free minimization of HPWL has been difficult due to both the non-differentiable nature of the objective function and the lack of strict convexity. We have demonstrated the feasibility of approximating HPWL of a hyperedge with an arbitrarily close upper bound derived from a multivariate regularization of the max{} function; minimization of this upper bound provides a reliable means of minimizing HPWL without representing the circuit hypergraph with graphs. We have also pointed out that "second-order" information is important for handling the clique/clustered nature of circuit hypergraphs which encourages approximations by twice-differentiable functions and the use of Newton-type methods. To avoid the difficulties of computing Hessian information in the context of HPWL minimization, we have proposed a limited memory quasi-Newton method which *implicitly* keeps track of second order information derived from gradient/solution computations.

Comparisons to the Weiszfeld placement algorithm [1], which is representative of the GORDIAN-L algorithm [20], indicate that the proposed placer achieves superior half-perimeter wirelength while spending similar amount of time.[11] Numerical results demonstrate global convergence of our method and its ability to significantly improve upon results achieved by quadratic minimization.

The ongoing work includes speed improvements and embedding the proposed analytical placer into a running top-down placer. An extension called BoxPlace-2 is planned to reduce the gap between theoretically minimal half-perimeter wirelength (as produced by linear programming) and what can be achieved within an acceptable CPU time budget. We are also interested in detection of placement instances for which analytical placers are not useful due to insufficient fixed vertices.

[11] The implementation of Weiszfeld that we used has a small overhead to support additional features not used in this paper. While we do not claim strict run time comparisons, solution quality should not have been affected.

References

[1] C. J. Alpert, T. F. Chan, D. J. H. Huang, A. B. Kahng, I. L. Markov, P. Mulet and K. Yan, "Faster Minimization of Linear Wire Length for Global Placement", *Proc. ACM/IEEE Intl. Symp. on Physical Design*, 1997, pp. 4-11.

[2] C. J. Alpert and A. B. Kahng, "Recent Directions in Netlist Partitioning: A Survey", *Integration, the VLSI Journal*, 19(1-2), pp. 1-81, 1995.

[3] R. Baldick, A. Kahng, A. Kennings and I. Markov, "Function Smoothing with Applications to VLSI Layout", accepted to *ASP-DAC 99*

[4] M. A. Breuer, "Min-Cut Placement", *J. Design Automation and Fault-Tolerant Computing* 1(4) (1977), pp. 343-362.

[5] A. E. Caldwell, A. B. Kahng and I. L. Markov, "Relaxed Partitioning Balance Constraints in Top-Down Placement", *Proc. ASIC-98*

[6] H. Eisenmann, F. M. Johannes, "Generic global placement and floorplanning". *Proc. Design Automation Conference*, 1998. p. 269-74.

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

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

[9] M. A. B. Jackson and E. S. Kuh, "Performance-Driven Placement of Cell Based IC's", *Proc. ACM/IEEE Design Automation Conf.*, 1989, pp. 370-375.

[10] M. Hanan, P. K. Wolff, and B. J. Agule, "A Study of Placement Techniques." *J. Design Automation and Fault-Tolerant Computing*, vol. 2, 1978, pp. 28–61.

[11] D. J. Huang and A. B. Kahng, "Partitioning-based Standard-cell Global Placement with an Exact Objective", *Proc. ACM/IEEE Intl. Symp. on Physical Design*, 1997, pp. 18-25.

[12] K. M. Just, J. M. Kleinhans, and F. M. Johannes, "On the Relative Placement and the Transportation Problem for Standard-Cell Layout", *Proceedings of Design Automation Conference*, 1986, pp. 308-313.

[13] 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.

[14] T. Lengauer , *Combinatorial Algorithms for Integrated Circuit Layout*, John Wiley & Sons Ltd.

[15] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization", *Mathematical Programming* 45 (1989), pp. 503-528.

[16] 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 Trans. on Fundamentals of Electronics, Communications and Computer Sciences* 4(E77-A) (1994), pp. 710-725.

[17] J. Nocedal, "Large Scale Unconstrained Optimization", To appear *The State of the Art in Numerical Analysis*, Ed. A Watson and I. Duff, Oxford University Press (1996).

[18] D. G. Luenberger, "Linear and Nonlinear Programming", 2nd Edition, Addison Wesley, 1984.

[19] B. M. Riess and G. G. Ettelt, "Speed: Fast and Efficient Timing Driven Placement", *Proc. IEEE Intl. Symp. on Circuits and Systems*, 1995, pp. 377-380.

[20] G. Sigl, K. Doll and F. M. Johannes, "Analytical Placement: A Linear or Quadratic Objective Function?" *Proc. ACM/IEEE Design Automation Conf.*, 1991, pp. 57-62.

[21] P. R. Suaris and G. Kedem, "Quadrisection: A New Approach to Standard Cell Layout," *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 1987, pp. 474-477.

[22] Takahashi, K.; Nakajima, K.; Terai, M.; Sato, K., "Min-cut placement with global objective functions for large scale sea-of-gates arrays.", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, April 1995, vol.14, (no.4),pp.434-446.

[23] 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.

[24] R. S. Tsay, E. Kuh, and C. P. Hsu, "Proud: A Sea-Of-Gate Placement Algorithm", *IEEE Design & Test of Computers*,1988, pp 44-56.

[25] R. S. Tsay and J. Koehl, "An Analytical Net Weighting Approach for Performance Optimization in Circuit Placement", *Proc. ACM/IEEE Design Automation Conf.*, 1991, pp. 620-625.

[26] J. Vygen, "Algorithms for large-scale flat placement", *Proc. Design Automation Conference*, 1997, p. 746-51.

[27] B. X. Weis and D. A. Mlynski, "A new relative placement procedure based on MSST and linear programming. *Proc. IEEE Int. Symp. on Circuits and Systems*, 1987, pp. 564-567.

[28] 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.