Analytical Minimization of Half-Perimeter Wirelength

Andrew Kennings and Igor Markov*
Dept. Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, N2L 3G1
UCLA Computer Science., Los Angeles, CA 90095-1596
akenning@odysseus.uwaterloo.ca, imarkov@cs.ucla.edu

Abstract

Global placement of hypergraphs is critical in the top-down placement of large timing-driven designs [10, 16]. Placement quality is evaluated in terms of the half-perimeter wirelength (HPWL) of hyperedges in the original circuit hypergraph provided timing constraints are met. Analytical placers are instrumental in handling non-linear timing models [9, 3], but have two important drawbacks: (a) corresponding optimization algorithms are typically slower than top-down methods driven by multi-level mincut partitioning [2], and (b) hyperedges must be represented with net models [10, 17, 15, 8, 21, 20] which imply a mismatch of objective functions, with the alternative of computationally expensive linear programming (LP) [25, 16].

By comparing to optimal solutions produced by linear programming, we show that net models lead to solution quality loss. To address this problem, we present the first analytical algorithm that does not require net models and permits a direct inclusion of non-linear delay terms [3]; this allows to avoid expensive *linearization* of delay terms in [16]. Our numerical engine utilizes well-known quadratically convergent Newton-type methods [5, 11] for speed; it produces solutions within 12% of the LP optimum. Empirical results are for industrial placement instances.

1 Introduction

Analytical placers are increasingly important in physical design as process technology advances and design complexity increases. They locate modules (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. 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* [20, 19, 9], *transportation* [23] or *force-directed techniques* [4].

In top-down placement, the layout area is recursively split into blocks; cells inside each block are dealt with under the assumption that all other cells are fixed. Analytical placers do not handle well large macros with non-trivial geometries [24] and are difficult to apply when insufficiently many terminals lead to numerical degeneracy. However, analytical placers may be helpful before or instead of min-cut partitioning steps at middle and lower levels of top-down placement when terminal counts increase due to terminal propagations from other placement blocks and if timing constraints are considered. Analytical placers are also useful for quick delay budgeting before non-overlapping and routable placements are available [16]. Therefore, min-cut partitioning and analytical placement are not direct competitors, but rather complement each other when timing considerations are important.

Analytical placers typically transform a circuit hypergraph into a graph prior to solving any optimization problems. Each hyperedge is modeled by a *star* [17, 15] or a *clique* [8, 21, 20] of edges. Early algorithms [21, 20] used squared wirelength objectives since global module placements required the solution to a single system of equations. 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 [12]. Recent analytical placers [17, 1] rely on a linear wirelength objective that is optimized, e.g., by iterated approximations with quadratic wirelength objectives [17].

Placement qualities are typically evaluated using the half-perimeter wirelength (HPWL) of the circuit hyperedges, however this objective cannot be directly pursued after nets are converted into stars or cliques. In this paper, we seek to eliminate the divergence from the minimization of HPWL in analytical placers.

HPWL minimization is possible via linear programming (LP) [7, 25], but is too computationally intensive due to the sizes of the resulting linear programs. Moreover, inclusion of non-linear terms is not straightforward. [16] addresses convex delay terms and describes a general *linearization* procedure that constructs piece-wise linear approximations, but finds it is too costly to solve directly as the complexity of linearizations increases with required precision.[1] An optimal algorithm for HPWL minimization based on classic max-flow computation was proposed by Hur and Lillis [6]. This technique, however, makes the inclusion of non-linear timing constraints [3] even harder. We thus seek a convex nonlinear approximation to enable easy inclusion of delay terms into

*Research at UCLA was supported by a grant from Cadence Design Systems, Inc.

[1] Their faster algorithm for a particular delay budgeting problem is rather difficult to implement as it relies on min-cost flows and graph-based simplex methods.

analytical placement formulations.

The contributions of our work are

- we show that the use of "second order" information provided by the Hessian is critical for efficient analytical algorithms, while it is not used by linearly convergent algorithms like GORDIAN-L [17].
- we propose a new [smooth] regularization of the [piecewise linear] half-perimeter wirelength function with provable properties.
- we propose the first analytical algorithm for half-perimeter wirelength minimization that bypasses traditional graph models of multi-pin nets and is naturally amenable to non-linear timing terms [3]. Thus the expensive *linearization* of delay in [16] can be avoided.
- we empirically confirm our approach by producing solutions within 12% of optimum, outperforming graph-based wirelength minimizations.

Our proposed nonlinear and convex approximation to HPWL is based on the observation that HPWL of multi-pin nets is a convex, but not everywhere differentiable function with singularities arising from "max" functions. Based on this observation, we extend the recently proposed *function smoothing techniques* in [3] for various VLSI layout problems to make them applicable to HPWL.

Section 2 demonstrates the difficulty of the direct minimization of HPWL due to its non-differentiability and reviews previous work. Section 3 demonstrates the importance of Newton-type methods. Section 4 proposed regularizations that approximate HPWL with arbitrarily small relative error and enable graph-free HPWL minimization via unconstrained smooth and convex optimization. Section 5 presents numerical results for industrial testcases in the context of top-down placement. Conclusions are given in Section 6.

2 Review of analytical placement

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 $e_k \in E$ connects $p_k \geq 2$ vertices and each vertex $v_i \in V$ is incident to $d_i \geq 0$ hyperedges. p_k and d_i are respectively called the vertex and hyperedge *degrees*, and are typically very small.We say that v_i has d_i pins, and e_k has p_k pins, for a total of $P = \sum_{k=1}^{m} d_k = \sum_{i=1}^{n} p_i$ 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 $\mathbf{y} = (y_1, \ldots, y_n)$.

2.1 Hypergraph placement

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

$$HPWL_x(\mathbf{x}) = \sum_{e_n \in E_H} \max_{i,j \in C_k} |x_i - x_j|. \qquad (1)$$

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 provided by I/O pads and external pins. The non-differentiability of the max function disables classic smooth minimization techniques such as Newton 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 [25], the HPWL estimate is converted into an equivalent linear program (LP) by adding, for each net e_k, upper and lower bound variables U_k and L_k. The cost of the net is measured as the difference between the two, and in an optimal solution each U_k and L_k will be equal to the rightmost and leftmost module locations of net e_k. Each variable comes with p_i inequality constraints that restrict $U_j(L_j)$ to be larger (smaller) than the locations of every module incident to the net. Thus, n nets and m modules are represented by $m + 2n$ variables and $2P$ constraints. While LP returns optimal placements, the large instance sizes effectively preclude any application to fast placement of large hypergraphs.

2.2 Reduction to graphs

Circuit hypergraphs are typically transformed into graphs in which each 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) [17, 15]. The *clique model* (e.g., [8, 10]) 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 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* when all graph vertices are are reachable from fixed vertices, which guarantees a unique minimizer.

Minimization of squared (quadratic) wirelength

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

($\mathbf{H}$ represents various linear constraints) is the easiest because (see [21, 20]) the unique minimizer is obtained by solving a single system of linear equations, either positive-definite or symmetric-indefinite depending on the approach. While good public-domain implementations of linear system solvers are available, the squared wirelength objective tends to provide lower-quality placements; a comparison by

[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.

[3] A shortcut [17] computes the squared wirelength of a p-clique with uniform edge weights $\frac{2}{p}$ as the squared wirelength of a star whose center vertex is placed at the center of gravity of its p vertices. In contrast to the general star model, the center of the star is not an independently placed vertex.

Mahmoud et al. [12] concludes that the linear wirelength objective is superior.

Linear wirelength minimization also relies on the above reduction of circuit hypergraph to a graph

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

Being neither differentiable nor strictly convex, it is not amenable to Newton-type methods. GORDIAN-L [17] minimization heuristic uses iterated quadratic minimizations

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

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 a_{ij} are updated at each iteration to approximate the linear wirelength. As an alternative, the *regularization* of (3)

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

was proposed in [1] with two solution methodologies: a linearly-convergent fixed point method and a novel primal-dual Newton method with quadratic convergence. Testing in [1] illustrated tradeoffs in values of $\beta > 0$ versus time and difficulty. The interpretation of the GORDIAN-L heuristic as a special case of $\beta = 0$ of a fixed point method was also provided. The techniques of β-**regularization** have been further extended in [3].

3 Importance of Newton-type methods

While the utility of analytical placers has been empirically established, inherent subtleties can impair implementations, particularly in new application contexts. One notable problem is the potential to ignore "second-order" information available during linear wirelength minimization with twice differentiable approximations. Newton-type methods that use second derivatives *converge quadratically*[14], while steepest-descent and fixed-point methods, including GORDIAN-L, are *converge linearly* [1]. However, iterations of linearly convergent methods are cheaper, and the comparison is not straightforward if the number of iterations is not big [2]. Below we demonstrate that the difference in convergence of steepest-descent and Newton-type wirelength minimizations goes beyond the asymptotics.

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

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

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 which, for the above initial placement of the clique, is given by

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

(several terms disappear for the described initial placement) where $\omega = 1/\sqrt{\beta}$ and $\sigma > 0$. 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.

4 Multivariate regularization for HPWL

Motivated by the convexity of HPWL, we seek a technique for its minimization via smooth convex Newton-type methods without graph approximations.

For two-pin hyperedges, the β-regularization [1, 3] can be used to approximate edgelength by a smooth convex function; such an approximation overestimates the original function by at most $\sqrt{\beta}$ and used in (5) to approximate graph edgelengths. In order to apply it to hyperedges of degree ≥ 3 and the HPWL, one can nest maximum functions, e.g., for a 3-pin net, $\max\{|x_1 - x_2|, |x_1 - x_3|, |x_2 - x_3|\}$ can be rewritten as $\max\{|x_1 - x_2|, \max\{|x_1 - x_3|, \max\{|x_2 - x_3|\}\}\}$, and recursively apply the following regularization:

Example [3]: 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 $\frac{a+b-\sqrt{|a-b|^2+\beta}}{2}$.

Unfortunately, the derivatives of the resulting regularization are difficult to compute. Moreover, the regularization is not symmetric and will send line search in wrong search directions.

A simple and computationally efficient alternative is available. **First**, note that we only consider $\max\{\}$ of non-negative numbers, **second**, $\| (a_1, a_2, \ldots, a_k) \|_\infty = \max\{|a_1|, |a_2|, \ldots, |a_k|\}$, **third**, the L_p-norms

$\| \quad (a_1, a_2, \ldots, a_k) \quad \|_p = (\sum_{j=1}^{k} |a_j|^p)^{1/p}$ converge to the L_∞-norm $\| \cdot \|_\infty$ 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" $\| (a_1, a_2) \|_\infty = \max(a_1, a_2) = 1$.

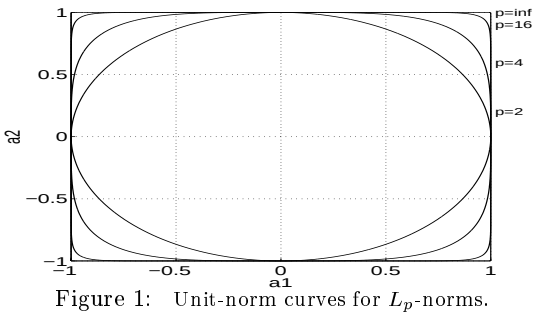

Figure 1: Unit-norm curves for L_p-norms.

Fact 1 For $p > 1$ and $\mathbf{a} = (a_1, a_2, \ldots, a_k)$

(a) $\| \mathbf{a} \|_\infty \leq \| \mathbf{a} \|_{p+1} \leq \| \mathbf{a} \|_p \leq k^{1/p} \| \mathbf{a} \|_\infty$

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

Proof We first establish the inequalities in **(a)**.

$\max\{|a_1|, \cdots, |a_k|\} = (\max\{|a_1|^p, \cdots, |a_k|^p\})^{1/p} \leq (|a_1|^p + \cdots + |a_k|^p)^{1/p}$ implies $\| \mathbf{a} \|_\infty \leq \| \mathbf{a} \|_p$ for any $p \geq 1$.

$\| \mathbf{a} \|_p \geq \| \mathbf{a} \|_{p+1}$ can be shown by induction from 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 binomial 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 canceled out), which follows from $(x^p + y^p) \geq \max\{x^p, y^p\} \geq x^i y^{p-i}$. This concludes the proof of **(a)**.

The differentiability in **(b)** can be proven by taking partials, e.g., $\frac{\partial f}{\partial a_i} = p a_i^{p-1} (\sum_{j=1}^{k} |a_j|^p)^{\frac{1}{p}-1}$. 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 in **(b)** can be proven by comparing the Jacobian to zero. □

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.[4]

The multiplicative upper bound of $(\frac{m(m-1)}{2})^{1/p}$ on the overestimation for HPWL of one m-pin net, as given by Fact 1(**a**), 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.[5]

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

Table 1: Single net HPWL overestimation by p-regularization.

As follows from Fact 1, 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

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

is a smooth and strictly convex[6] upper bound on exact HPWL with arbitrary small relative error of approximation as $p \to \infty$ and $\beta \to 0$. Figure 2 illustrates the combined p- and β-regularization for HPWL. We note that restricting p to powers of 2 allows for particularly effective computations.

Our regularization subsumes the one in [3] with an important addition of $p \to \infty$ (p was defined differently in [3] and set to 2 for all applications). We set $\beta = (\beta_0 M)^p$ (cf. [3]) where M is the maximal distance between fixed terminals and β_0 is instance-independent.

5 Experimental validation

To compare to optimal solutions found via linear programming, our tests do not include *vertex spreading*, such as equality constraints in GORDIAN/GORDIAN-L or top-down framework in PROUD. We expect that improvements in the fundamental analytical engines translate to complete algorithms.

For our proposed HPWL minimization, both the objective and the gradient can be computed analytically, but the Hessian computations are hard and time-consuming. Given the crucial nature of second order information, we have implemented the limited memory quasi-Newton method in [11, 13] which uses limited memory BFGS updates to approximate the Hessian.[7] Our implementation maintains storage for seven past iterations. Iterations continue until (i)

[4] Nets that entail prohibitively many $(\frac{m(m-1)}{2})$ terms can be handled via the L_p-norm taken over edges of a star model with the center located at the center of gravity of the net's pins.

[5] 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.

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

[7] Notably, the BFGS (Broyden-Fletcher-Goldfarb-Shanno) updates are closely related to the DFS (Davidon-Fletcher-Powell) updates [5], except for a slightly different inverse Hessian approximation. BFGS updates typically perform better and are preferred for practical applications.

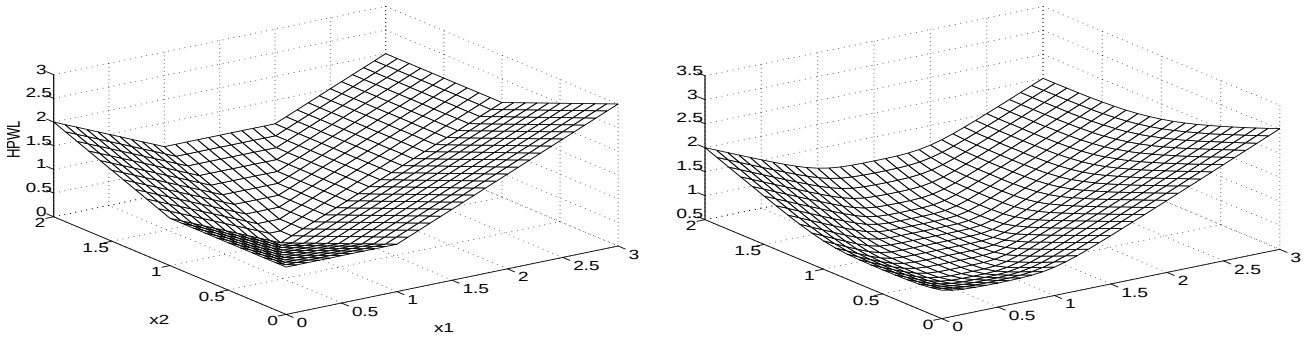

Figure 2: HPWL(left) for a 3-pin net — $\max\{|x_1 - x_2|, |x_1 - 1|, |x_2 - 1|\}$ over $(x_1, x_2) \in [0,3] \times [0,2]$. The p-regularization (right) is $(|x_1 - x_2|^p + |x_1 - 1|^p + |x_2 - 1|^p + \beta)^{1/p}$. Here $p = 8$ and $\beta = 1.0e6$.

a prescribed iteration limit (100) OR (ii) the improvement in the objective function is below a prescribed threshold OR (iii) the gradient norm falls below a prescribed threshold. We use line search developed by Jorge J. Moré and David J. Thuente for the MINPACK project in 1983. Although not as accurate as a binary convex line search, the Moré-Thuente is several times faster and has a better cost-performance ratio.

We use five testcases from industry (see Table 2) that are either original ("top-level") placement instances or have arisen on further levels of top-down placement.[8] Test3 through Test5 correspond to placement blocks at various levels in top-down placement of industrial designs of sizes up to 69K cells. Applying analytical placement on higher levels is not practical as insufficient number of fixed vertices leads to degeneracy of the analytical placement model in Section 2. We use the proposed approximation of HPWL

Instance	Modules		Nets	Design
	Fixed	Free		size
test1	76	200	242	276
test2	545	2686	2840	3.2K
test3	2155	6739	7330	12K
test4	2191	3205	3835	12K
test5	6545	17380	20902	69K

Table 2: Testcase parameters. Test3,4,5 are top-down placement blocks sized at 1/2, 1/4 and 1/4 of their complete designs.

in an algorithm called BoxPlace, which we implemented in C++ using the SunPro CC4.2 compiler (`-O5`) and Solaris 2.6 operating system. During experiments, parameters were set $p = 16$ and $\beta_0 = 0.01$.

Table 3 shows that BoxPlace produces better placements than graph-based algorithms and is faster. Its global convergence was tested by running from multiple random initial starting points. Alternatively, we started with a "one-point" solution placing all vertices into one location. This yielded much better *initial* wirelength and faster convergence to a solution comparable to those achieved in the randomized experiment.

Table 4 gives optimal placement costs and run times required to achieve those using `CPLEX 6.5.1` and academic implementations: `LPSolve 2.3` with enhancements by David Warme and a network-flow-based optimal algorithm by Hur and Lillis [6].[9] Optimal costs provide a baseline for comparing heuristics. We observe that BoxPlace achieves placement quality within 12% of optimal. On larger instances, BoxPlace runs substantially faster than LP-based implementations and the Hur-Lillis algorithm. We also ran a multi-level FM (MLFM) partitioner on the same circuits, and one start was at least five times faster than BoxPlace. Unlike those methods, our solver can naturally accommodate additional non-linear terms in the objective function as long as they are convex and differentiable (or can be regularized).

6 Conclusions

We analyzed fundamental placement algorithms that are used in common placement tools. 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 suboptimal solutions as illustrated by Table 3.

We proposed a fast analytical placement algorithm based on a new approximation of half-perimeter wirelength that has arbitrary small error. *This is the first analytical algorithm to minimize half-perimeter wirelength bypassing traditional net models.* Unlike previously known heuristics, it can accommodate convex non-linear delay terms and produces solutions within 12% of optimum. The advantage of our approach compared to [16] where delay terms are linearized to apply linear programming, is that the complexity of non-linear approximations does not grow when better precision is needed.

We have also pointed out that "second-order" information is important for handling the clique/clustered nature of circuit hypergraphs and encourages the use of Newton-type methods in conjunction with twice-differentiable approximations. To avoid the difficulties of computing Hessian information in the context of HPWL minimization, we have adapted a known limited memory quasi-Newton method which *implicitly* keeps track of second order information derived from gradient computations. Comparisons to the authors' implementation of Weiszfeld algorithm [1] confirms the superiority of quadratically convergent methods and goes well with comments in [24] that GORDIAN-L is rather slow. Our techniques are applicable to other objectives, e.g., to the piece-wise linear objective in [22].

[8] Several test circuits had disconnected cells not reachable from fixed terminals. To avoid degeneracy and subsequent breakdown of numerical solvers, 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.

[9] Compared to [25], our linear programs have 50% less additional variables for 2- and 3-pin nets as well as correspondingly fewer constraints for 2-pin nets.

HEURISTIC PLACEMENT ALGORITHMS FOR HALF-PERIMETER WIRELENGTH OBJECTIVE											
	BoxPlace from random			BoxPlace from "one-point"				Quadratic		Weiszfeld[1]	
	initial	final WL	CPU⏲	initial	final WL	+greed x2	CPU⏲	WL	CPU⏲	WL	CPU⏲
test1	5.73**e7**	6.38e6	0.4	6.72e6	6.39e6	**6.23e6**	0.2	7.66e6	0.11	6.72e6	0.15
test2	3.08**e8**	3.25e7	24.4	3.51e7	3.11e7	**3.01e7**	15.0	4.20e7	3.75	3.51e7	7.12
test3	6.27e6	2.36e6	28.5	2.69e6	2.32e6	**2.21e6**	13.8	2.75e6	19.9	2.67e6	22.4
test4	5.37e6	1.49e6	13.1	1.91e6	1.44e6	**1.38e6**	5.8	1.61e6	5.4	1.78e6	12.9
test5	3.75e7	2.56e7	45.1	1.50e7	1.40e7	**1.33e7**	33.8	1.50e7	72.1	1.47e7	87.6

Table 3: BoxPlace algorithm compared to graph-based algorithms by total HPWL (the sum of x- and y-values). Run times are in seconds on a `Sun Ultra-10`/300 MHz and averaged over $x-$ and $y-$ to represent expected runtime in top-down placement.

OPTIMAL PLACEMENT ALGORITHMS WITH HALF-PERIMETER WIRELENGTH									
	Linear Program			Optimal	`LPSolve2.3w`	`CPLEX 6.5.1` CPU⏲			Hur-Lillis [6]
	rows	cols	non0s	WL($x+y$)	CPU⏲	`primopt`	`netopt`	`tranopt`	CPU⏲
test1	1050	512	2250	5.93e6	1.2	0.54	0.34	0.32	X
test2	15602	6602	32500	2.75e7	6 min	31	9.11	10.74	X
test3	21276	9046	45563	1.98e6	8 hr	30 min	2 min	2.2 min	6.58
test4	43486	17598	93860	1.24e6	1 hr	72	18.44	21.97	48.82
test5	123648	47438	265750	1.19e7	>3days	12 hr	12 min	15 min	3.3min

Table 4: Optimal hypergraph placement implementations for the HPWL objective. Run times are averaged over the $x-$ and $y-$ directions and given in seconds unless indicated otherwise. `LPSolve3.2w` runs were performed on a Sun `Ultra-10`/300MHz, `CPLEX 6.5.1` — on an `IBM RS/6000 3CT` workstation, which measured $1.6-1.15$ times slower than the Sun `Ultra-10`. The Hur-Lillis algorithm [6] ran on a `Sun Ultra-1`/200Mhz that measured 1.4 slower than the `Ultra-10`. Optimal costs are sums of $x-$ and $y-$ components. `test1` and `test2` are only available in LEF/DEF format, thus we could not run Hur-Lillis on them. $x-$ and $y-$ linear programs have the same numbers of rows and columns, the numbers of nonzeros are averaged.

Acknowledgements

We thank Prof. Andrew Kahng at UCLA for motivating this work and providing extensive comments, as well as Sung Hur and Prof. John Lillis at UIC for benchmarking their algorithm for us. Professors David Morton and Ross Baldick at UT Austin helped us with CPLEX.

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. ISPD '97*, pp. 4-11.

[2] C. J. Alpert, A. E. Caldwell, T. Chan, D. J.-H. Huang, A. B. Kahng, I. L. Markov and M. Moroz, "Analytical Engines Are Unnecessary in Top-down Partitioning-Based Placement" *VLSI Design* (1999), to appear.

[3] R. Baldick, A. Kahng, A. Kennings and I. Markov, "Function Smoothing with Applications to VLSI Layout", *Proc. ASP-DAC '99*, pp. 225-228.

[4] H. Eisenmann, F. M. Johannes, "Generic Global Placement and Floorplanning". *Proc. DAC '98*, pp. 269-274.

[5] R. Fletcher and M.J.D. Powell, "A Rapidly Convergent Descent Method for Minimization", Computer J. 6, 1963, pp. 163-168.

[6] S.W. Hur and J. Lillis, "Relaxation and Clustering in a Local Search Framework: Application to Linear Placement", *Proc. DAC '99*, pp. 360-366.

[7] M. A. B. Jackson and E. S. Kuh, "Performance-Driven Placement of Cell Based IC's", *Proc. DAC '89*, pp. 370-375.

[8] 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.

[9] T. Koide at al, "Par-POPINS: a Timing-driven Parallel Placement Method With the Elmore Delay Model For Row Based VLSIs", *Proc. ASP-DAC '97*, 1997. pp. 133-40.

[10] T. Lengauer , *Combinatorial Algorithms for Integrated Circuit Layout*, Wiley-Teubner, 1990.

[11] D. C. Liu and J. Nocedal, "On the Limited Memory BFGS Method For Large Scale Optimization", *Mathematical Programming* 45 (1989), pp. 503-528.

[12] 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.

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

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

[15] B. M. Riess and G. G. Ettelt, "Speed: Fast and Efficient Timing Driven Placement", *Proc. ISCAS '95*, pp. 377-380.

[16] M. Sarrafzadeh, D. Knol and G. Tellez, "Unification of Budgeting and Placement", In *Proc. DAC '97*, pp. 758-761.

[17] G. Sigl, K. Doll and F. M. Johannes, "Analytical Placement: A Linear or Quadratic Objective Function?" *Proc. DAC '91*, pp. 57-62.

[18] A. Srinivasan, K. Chaudhary and E. S. Kuh, "RITUAL: A Performance Driven Placement for Small-Cell ICs", *Proc. ICCAD '91*, pp. 48-51.

[19] 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*, vol. 14, no. 4, April 1995, pp. 434-446.

[20] 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.

[21] 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.

[22] Y.-W. Tsay, H.-P. Su, Y.-L. Lin, "An Improved Objective For Cell Placement", *Proc. ASP-DAC '97*, Japan, 1997, pp. 281-284.

[23] J. Vygen, "Algorithms For Large-scale Flat Placement", *Proc. DAC '97*, pp. 746-51.

[24] J. Vygen, *Personal Communication*, November, 1999.

[25] B. X. Weis and D. A. Mlynski, "A New Relative Placement Procedure Based on MSST and Linear Programming. *Proc. ISCAS '87*, pp. 564-567.