Analytical Engines Are Unnecessary in Top-Down Partitioning-Based Placement*

C. J. Alpert‡, A. E. Caldwell, T. F. Chan†, D. J.-H. Huang*,

A. B. Kahng, I. L. Markov and M. S. Moroz§

UCLA Computer Science Dept., Los Angeles, CA 90095-1596

† UCLA Mathematics Dept., Los Angeles, CA 90095-1555

‡ IBM Austin Research Laboratory, Austin, TX 78758

* Avant! Corporation, 46871 Bayside Parkway, Fremont CA 94538

§ UCLA Anderson Graduate School of Management, Los Angeles, CA 90095

Abstract

The top-down "quadratic placement" methodology is rooted in such works as [36] [9] [32] and is reputedly the basis of commercial and in-house VLSI placement tools. This methodology iterates between two basic steps: solving sparse systems of linear equations to achieve a continuous placement solution, and "legalization" of the placement by transportation or partitioning. Our work, which extends [5], studies implementation choices and underlying motivations for the quadratic placement methodology. We first recall some observations from [5], e.g., that (i) Krylov subspace engines for solving sparse linear systems

*This work was supported by a grant from Cadence Design Systems, Inc., and is an extension of work reported in Proc. ACM/IEEE Design Automation Conference, 1997 [5]. Principal author, to whom correspondence should be addressed: Igor Markov, UCLA CS, imarkov@cs.ucla.edu. See `http://vlsicad.cs.ucla.edu` for more information.

are more effective than traditional successive over-relaxation (SOR) engines [33] and (ii) that *correlation convergence* criteria can maintain solution quality while using substantially fewer solver iterations. The focus of our investigation is the coupling of numerical solvers to iterative partitioners that is a hallmark of the quadratic placement methodology. We provide evidence that this coupling may have historically been motivated by the pre-1990's weakness of min-cut partitioners, i.e., numerical engines were *needed* to provide helpful hints to weak min-cut partitioners. In particular, we show that a modern multilevel FM implementation [2] derives no benefit from such coupling. We also show that most insights obtained from study of individual min-cut partitioning instances (within the top-down placement) also hold within the overall context of a complete top-down placer implementation.

Keywords: quadratic placement, multi-level min-cut partitioning, hierarchical top-down placement.

1 Introduction

In the physical implementation of deep-submicron ICs, placement solution quality is a major determinant of whether timing correctness and routing completion will be achieved. The first-order objective is to place connected modules close together, to reduce total routing and lower bounds on signal delay. This implies a minimum wirelength based placement objective. Because there are many layout iterations (including those between placement and global/detailed routing, performance optimizations, technology mapping and logic synthesis) and because fast (constructive) placement estimation is needed within the floorplanner, an ideal placement tool will offer fast, consistent, and high-quality results. Due to its speed, "global" perspective, and ability to address wirelength-based objectives, the *quadratic placement* methodology (cf. such antecedents as [36] [14] [9] [33]) has been widely adopted in industry.

In this work, we revisit the quadratic placement methodology and develop insights into its effective implementation, particularly in light of recent algorithmic developments for partitioning. Our paper is organized as follows. After defining notation, Section 2 synthesizes a generic model of the quadratic placement methodology. The key elements of our model are: (i) top-down hierarchical placement, and (ii) use of a sparse linear systems solver, coupled with a min-cut iterative (FM-type) partitioner, to obtain any given partition within the top-down placement process. Section 3 discusses effective implementation of the quadratic placement methodology. We briefly review previous results [5] that suggest use of Krylov-subspace solvers, along with *correlation convergence* criteria within the solvers, to improve efficiency. We then focus on the coupling between the linear system solver and the min-cut partitioning step as the key to implementing quadratic placement: issues include the type of wirelength-based objective addressed by the solver, how the solver result is used by the partitioner, and the type of partitioning engine employed. In Section 4, we use a high-quality placement testbed to experimentally assess our various hypotheses. Our most significant results show that the solver-partitioner coupling may have historically been motivated by the pre-1990's weakness of min-cut partitioners, i.e., numerical engines were *needed* to provide helpful hints to weak min-cut partitioners. In particular, we show that a modern multilevel FM implementation [2] derives no benefit from such coupling. We also contrast the abilities of linear-wirelength and squared-wirelength solver objectives to drive partitioners to good solutions. Finally, we show that insights obtained from study of individual min-cut partitioning instances (within top-down placement) apply to the overall top-down placement results as well. We conclude the paper in Section 5 with a list of ongoing research directions, and some comments on the relevance of the quadratic placement methodology to future design methodology requirements.

2 A Synthesis of the Quadratic Placement Methodology

2.1 Notation and Definitions

A gate-level netlist is represented for placement by a weighted hypergraph. The n vertices correspond to modules, with vertex weights representing module areas or routing requirements. Hyperedges correspond to signal nets, with hyperedge weights representing criticalities and/or multiplicities. The two-dimensional layout region is represented as an array of legal placement locations. Placement seeks to assign all modules of the design to legal locations, such that no modules overlap and chip timing and routability are optimized. The placement problem is a type of NP-hard quadratic assignment.

The numerical techniques used within the quadratic placement methodology apply only to graphs (hypergraphs with all hyperedge sizes equal to 2). Therefore, we must assume some transformation of hypergraphs to graphs via a *net model.* Throughout the discussion and experiments reported below, we use the standard clique model for nets of degree 10 or less, and the directed star model for nets of degree larger than 10, to preserve sparsity. For a given multipin signal net with k pins, the graph edges that represent the net may be constructed in several ways, e.g., a directed star model of $k-1$ edges, an unoriented star model, or an unoriented clique model of $\frac{k(k-1)}{2}$ edges (see [3] for a review). The resulting weighted graph representation $G = (V, E)$ of the circuit topology has edge weights a_{ij} derived by "superposing" all derived edges in the obvious manner. The standard undirected clique model [25] assigns all clique edges weight $\frac{1}{k-1}$.[1]

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

[1]Some other clique models have been proposed, e.g., the model of [33] assigns all clique edges weight $\frac{8}{k^3}$. We have obtained similar results for the clique model of [33] as well as for a directed star model.

Certain vertices are *fixed*, typically due to pre-placement of I/O pads or the inducing of terminals around a block's periphery during top-down placement. All other vertices are *movable*. The *one-dimensional placement problem* seeks to place *movable* vertices onto the real line so as to minimize an objective function that depends on the edge weights and the vertex coordinates. The n-dimensional *placement vector* $\mathbf{x} = (x_i)$ denotes the physical locations of modules $v_1, \ldots, v_n$ on the real line, i.e., x_i is the coordinate of vertex v_i. Let c be the number of movable modules and let $f = n - c$ be the number of fixed ("pad") modules. Without loss of generality, the c movable modules are $v_1, \ldots, v_c$ and the f fixed modules are $v_{c+1}, \ldots, v_n$. The modules can always be permuted prior to optimization to ensure this condition is satisfied.

Squared Wirelength Formulation: Minimize the objective

$$\Phi_q(\mathbf{x}) = \sum_{i>j} a_{ij}(x_i - x_j)^2 \quad \text{such that} \quad x_{c+1}, \ldots, x_n \ \text{ are fixed.}$$

This function can also be written as $\Phi_q(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x}$.

We are interested in quadratic placers that solve the two-dimensional placement problem with a top-down approach, i.e., one-dimensional placement in the horizontal direction is used to divide the netlist into left and right halves, after which one-dimensional placement in the vertical direction is used to subdivide the netlist into quarters, etc.[2]

[2] "Decomposition" of the two-dimensional placement problem into independent one-dimensional placement problems is used in the quadratic placement methodology to yield smaller linear systems. Notice that for the quadratic objective, the Euclidean problem decomposes cleanly into coordinates (by Pythagoras' theorem) while the Manhattan problem does not. (Hence, we presumably are minimizing squared Euclidean wirelength.) For the linear objective, the Euclidean problem does not decompose while the Manhattan objective does.

2.2 Essential Structure of a Quadratic Placer

We now state the essential components of the quadratic placement methodology. Our primary goal is to establish the coupling between (i) numerical methods for sparse linear systems and (ii) min-cut optimizations or other means of "spreading", or "legalizing", a continuous placement solution. We illustrate our discussion by referring to the PROUD algorithm of Tsay et al. [32] [33].

Like other works, PROUD considers the squared wirelength objective $\Phi_q(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x}$. An unconstrained formulation is obtained by optimizing the objective function $\Phi_q(\mathbf{x})$ for c movable modules while satisfying f fixed pad constraints, but without discrete slot constraints. (The term *slot constraint*, originated by Cheng and Kuh [9], refers to the fact that a legal placement must locate modules within the two-dimensional array of allowed locations (slots). E.g., the first-order slot constraint forces the sum of module coordinates to equal the sum of slot coordinates.) The objective function can be written as

$$\begin{aligned}\Phi_q(\mathbf{x}) &= \frac{1}{2}\begin{bmatrix}\mathbf{x}_c & \mathbf{x}_f\end{bmatrix}\begin{bmatrix}\mathbf{Q}_{cc} & \mathbf{Q}_{cf}\\ \mathbf{Q}_{fc} & \mathbf{Q}_{ff}\end{bmatrix}\begin{bmatrix}\mathbf{x}_c\\ \mathbf{x}_f\end{bmatrix}\\ &= \frac{1}{2}(\mathbf{x}_c^T\mathbf{Q}_{cc}\mathbf{x}_c + \mathbf{x}_c^T\mathbf{Q}_{cf}\mathbf{x}_f + \mathbf{x}_f^T\mathbf{Q}_{fc}\mathbf{x}_c + \mathbf{x}_f^T\mathbf{Q}_{ff}\mathbf{x}_f)\end{aligned}$$

where $\mathbf{x}_f$ denotes the vector of *fixed* module positions and $\mathbf{x}_c$ denotes the vector of *movable* module positions; the Laplacian $\mathbf{Q}$ is partitioned into four corresponding parts $\mathbf{Q}_{cc}$, $\mathbf{Q}_{cf}$, $\mathbf{Q}_{fc}$ and $\mathbf{Q}_{ff}$ with $\mathbf{Q}_{cf}^T = \mathbf{Q}_{fc}$.

In this formulation, the optimal positions of all movable modules are inside the convex hull of the fixed module locations [33]. Hence, we can consider the minimization problem for $\Phi_q(\mathbf{x})$ over this convex hull. Since $\Phi_q(\mathbf{x})$ is a strictly convex smooth function over a compact set (in c-dimensional Euclidean space), the unique minimum objective function value is attained at the extremal or a boundary point; the nature of the problem implies that it will be at the extremal point. To find the zero of the derivative of the objective function $\Phi_q(\mathbf{x})$, we solve the $c \times c$ linear system

$$\nabla \Phi_q(\mathbf{x}) = \mathbf{Q}_{cc}\mathbf{x}_c + \mathbf{Q}_{cf}\mathbf{x}_f = 0$$

which can be rewritten as

$$\mathbf{Q}_{cc}\mathbf{x}_c = -\mathbf{Q}_{cf}\mathbf{x}_f \tag{1}$$

This development is similar to that of other "force-directed" or "resistive network" analogies (see, e.g., [36] [27] [14] [9]). The essential tradeoff is the relaxation of discrete slot constraints, along with the changing of the "true" linear wirelength objective into a squared wirelength objective, to obtain a continuous quadratic objective for which a global minimum can be found. The typical resulting "global placement" concentrates modules in the center of the layout region. Hence, the key issue is how the "global placement" (actually, a "continuous solution obtained using an incorrect objective") should be "spread" or "legalized" into a solution to the original discrete problem.

Two approaches have been used to obtain a feasible placement from a "global placement". The first approach is based on *assignment*, either in one step (to the entire two-dimensional array of slots) or in two steps (to rows, and then to slots within rows) [14]. The second and more widely-used approach is based on *partitioning*: the global placement result is used to derive a horizontal or vertical cut in the layout, and the continuous squared-wirelength optimization is recursively applied to the resulting subproblems (see [36] [27] [9] [24]). The main difficulty is making partitioning decisions on the extremely overlapped modules in the middle of the layout. The obvious median-based partitioning (find the median module and use it as a "splitter") is sensitive to numerical convergence criteria. Thus, FM-type iterative improvement strategies ([22] [13]; see [3] as well as the discussion of Section 3 for a review) are commonly used to refine the resulting partitioning (see, e.g., [24]). Since the typical objective for iterative improvement partitioning is some form of minimum weighted cut,[3] the quadratic placement methodology can be seen to be quite similar in structure to top-down min-cut placers, with initial cuts induced from (one-dimensional) placements under the squared-wirelength objective.[4]

We summarize the essential structure of a quadratic placer as consisting of:

- a sparse linear systems solver;
- a min-cut iterative (FM-type) partitioner; and
- a top-down hierarchical min-cut framework wherein for any given partitioning instance the solver results are used to induce an initial solution for the iterative partitioner.

[3]Formally, denote the n modules of the netlist hypergraph $H(V,E)$ as $V = \{v_1, v_2, \ldots v_n\}$. A ***net*** $e \in E$ is a subset of V with size greater than one. A ***bipartitioning*** $P = \{X, Y\}$ is a pair of disjoint ***clusters*** (i.e., subsets of V) X and Y such that $X \cup Y = V$. The ***cut*** of a bipartitioning $P = \{X, Y\}$ is the number of nets which contain modules in both X and Y, i.e., $cut(P) = |\{e \mid e \cap X \neq \emptyset, e \cap Y \neq \emptyset\}|$. Let $A(v)$ denote the area of $v \in V$ and let $A(S) = \sum_{v \in S} A(v)$ denote the area of a subset $S \subseteq V$. Given a balance tolerance r, the ***min-cut bipartitioning problem*** seeks a solution $P = \{X, Y\}$ that minimizes $cut(P)$ subject to $\frac{A(V)(1-r)}{2} \leq A(X), A(Y) \leq \frac{A(V)(1+r)}{2}$.

[4]The PROUD placer [33] [32], which we have cited as a prototype for the quadratic placement methodology, is more careful than previous works in how it applies partitioning during the top-down min-cut process. Suppose that a vertical cut has been made along some centerline, so that the left and right halves must each be split by horizontal cuts. PROUD applies a "block Gauss-Seidel" analogy, as follows. Modules in the left half are ordered vertically, followed by terminal propagation (projection) to the centerline. The projected terminals then influence the vertical ordering of the modules in the right half. The modules of the right half are then fixed and projected to the centerline, where they influence a new vertical ordering of the left half. Eventually, both orderings converge and can be split to induce subproblems. In this way, PROUD affords more global interaction between siblings in the hierarchy (see also the "cycling and overlapping" metaheuristics discussed in [19]).

More generally, we recognize (i) that intermediate steps involving assignment or transportation may also be used to derive hierarchical subproblems from the initial global placement; (ii) that while top-down bipartitioning is the standard approach, hierarchical subproblems may also be derived by top-down quadrisection; (iii) that iterative partitioners can use more sophisticated (placement-based) objectives than the traditional weighted min-cut, and (iv) that any number of metaheuristics can be used within the general top-down framework to improve routability, timing, etc. (such recent works as [34] [19] exemplify such possibilities). Nonetheless, we have chosen to synthesize a quadratic placement methodology based on FM-type iterative bipartitioning: this is the cleanest (and most typical) framework for the solver-partitioner interaction in quadratic placement.

3 Effective Implementation of the Quadratic Placement Methodology

In this section, we list five major degrees of freedom in the implementation of the quadratic placement methodology. For two of these degrees of freedom – the use of modern Krylov-subspace solvers with preconditioners, and correlation convergence criteria – we refer the reader to earlier work of [5]. The remaining three degrees of freedom – linear-wirelength vs. squared-wirelength objectives, the solver-partitioner interface, and the use of modern (multilevel) FM partitioners – are the focus of detailed experiments in Section 4 below.

Of course, we realize that there are many other degrees of freedom in the implementation of a "quadratic placer", e.g., the number of partitioning trials made at a given level of the top-down placement, the integration of metaheuristics such as cycling and overlapping [19], etc. In our experience, tuning these degrees of freedom is an important activity, requiring substantial effort. However, we have chosen to concentrate on what we believe are the more fundamental issues for quadratic placement implementation. In particular, our line of investigation is motivated by the recent advances in algorithm technology for iterative partitioning. Our main question is: Do modern partitioners really require seeding from (one-dimensional placements computed by) numerical solvers? Our discussion sets the stage for experimental evidence showing that implementation choices for quadratic placement are dominated by the strengths of modern multilevel partitioners. We find that the "quadratic placement methodology" no longer benefits significantly from the use of linear systems solvers that minimize quadratic wirelength objectives.

3.1 Use of Krylov-Subspace Solvers with Preconditioners

Recall that quadratic placement requires solving sparse linear systems with dimensions on the order of the number of movable modules in a given one-dimensional placement instance. The time complexity of an iterative solver depends on both the cost of a single iteration (which is constant during the solution of a given system) and the number of iterations needed until iterates adequately approximate the true solution. The theory of iterative methods shows that the number of iterations needed to obtain a good approximation in norm depends on the spectrum of the matrix involved [15]. Hence, the idea of a preconditioner – a way to transform the original system to an equivalent one with "improved" spectrum. Because most implementations of preconditioners entail additional per-iteration cost, one must carefully examine the overall efficiency of solver/preconditioner combinations on particular classes of instances: more expensive iterations must be balanced against the number of iterations saved.

The work of [5] experimentally compares iteration counts and CPU times for various combinations of solvers and preconditioners when solving typical systems arising within the quadratic placement methodology. These studies were motivated by the observation that works such as [33] use the 1950's-era SOR iteration to solve the linear system in Equation (1), despite many improved methods for solving sparse symmetric linear systems having been developed in recent years. (See [7] for pseudocodes of solvers and preconditioners, as well as a taxonomy of the types of systems to which these iterations apply; [15] gives theoretical analyses.[5]) The authors of [5] test comparable implementations of solver-preconditioner combinations using the PETSc library [6], and conclude that BiConjugate Gradient Stabilized (BiCGS) is among

[5] Briefly, iterative methods for solving large systems of linear equations can be classified as *stationary* or *non-stationary.* Stationary methods include Jacobi, Gauss-Seidel, Successive Over-relaxation (SOR) and Symmetric Successive Over-relaxation (SSOR). They are older, easier to implement and computationally cheaper per iteration. Non-stationary methods include Conjugate Gradient (CG), Generalized Minimal Residual (GMRes) and numerous variations. These are relatively newer and harder to implement, but afford much faster convergence. Additional computational expense per iteration is normally justified by much smaller numbers of iterations. Solvers which provide smooth convergence can be also used as preconditioners. Direct solvers present a different source of preconditioners for iterative methods, with examples being incomplete Cholesky (ICC), LU-factorization and incomplete LU-factorization (ILU).

the best solvers. Though it does not guarantee convergence, BiCGS is good even for degenerate (not necessarily symmetric) matrices and provides more robust convergence than conjugate gradient (CG). For preconditioners, Incomplete LU-factorization and the Successive Over-relaxation family (including SSOR) are particularly successful. In verifying the superiority of BiCGS, performance of SOR and SSOR was evaluated with the best value of ω parameter, which was determined to be $\omega = 1.95$ (for SOR/SSOR solvers) and $\omega = 1.0$ (for SOR/SSOR preconditioners) over a range of problem instances.

3.2 Linear-Wirelength vs. Squared-Wirelength Objectives

In Section 2.2 above, we saw that the continuous placement formulation with squared wirelength objective has a unique optimum solution that can be found by solving a sparse linear system. However, while being convenient to work with, the quadratic objective does not correspond to any intuitive physical quantity. Compared to the linear objective, the quadratic objective more heavily weights long connections and less heavily weights short connections. To see this, consider the example of two wires with respective lengths 2 and 10. According to the linear objective, the relative cost of the long versus the short wire is $10/2 = 5$, but according to the quadratic objective, the relative cost of the long versus the short wire is $100/4 = 25$ – i.e., the cost ratio of long to short wires is much higher under the quadratic objective.

Several works have suggested that the following linear wirelength objective is superior for placement:

Linear Wirelength Formulation: Minimize the objective

$$\Phi_l(\mathbf{x}) = \sum_{i>j} a_{ij}|x_i - x_j| \quad \text{such that} \quad x_{c+1}, \ldots, x_n \quad \text{are fixed.}$$

Mahmoud et al. [26] and Sigl et al. [31] demonstrate the superiority of the linear wirelength objective for analog and row-based placement, and Riess et al. [30] show that a one-dimensional placement which minimizes linear wirelength can lead to excellent netlist bipartitioning results.

Optimizing the linear wirelength objective is less straightforward. For example, there can be multiple optimal solutions (consider a single movable module connected to two fixed pads by edges of equal weight – this module can be optimally placed anywhere between the two pads). The set of optimal placements is again closed and contained within the convex hull of fixed pad locations (see [33]). Thus, direct minimization of the linear wirelength objective can be achieved by linear programming, but this is usually computationally expensive. Consequently, most placers that address the linear wirelength objective find a solution by iteratively solving several quadratic formulations. We use the GORDIAN-L placer [31] to illustrate this technique. (Note that the set of constraints that GORDIAN-L can handle is more general than described here. In particular, GORDIAN-L can handle center-of gravity constraints whereby the coordinates for any subset of modules must be centered around a prescribed center. However, the technique described here for optimizing a linear objective by transforming it into a quadratic objective is independent of the types of constraints applied.)

The objective Φ_l can be rewritten as

$$\Phi_l(\mathbf{x}) = \sum_{i>j} a_{ij}|x_i - x_j| = \sum_{i>j} \frac{a_{ij}(x_i - x_j)^2}{|x_i - x_j|}.$$

If the $|x_i - x_j|$ term in the denominator were constant, then a quadratic objective would result which can be solved via the above technique. GORDIAN-L first solves the system $\Phi_q(\mathbf{x})$ to obtain a reasonable approximation for the $|x_i - x_j|$ terms. 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 $\mathbf{x}^{k-1}$ and $\mathbf{x}^k$. From a given solution $\mathbf{x}^{k-1}$, the next solution $\mathbf{x}^k$ is obtained by minimizing

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

where $g_{ij}^k = a_{ij}/|x_i^{k-1} - x_j^{k-1}|$. 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. (It turns out that this approach is actually a special case of a method due to Weiszfeld [35]; see [5] for a detailed discussion.) Below, we will experimentally study the effects of the choosing the linear-wirelength, vs. the squared-wirelength, objective for one-dimensional placement within the quadratic placement methodology. In particular, we will consider the effects on individual cutsizes as well as on total placement wirelength.

3.3 Correlation Convergence Criteria

Any iterative solver builds a sequence of iterates that converges to the solution $\mathbf{x}$ of Equation (1). In the quadratic placement methodology that we have described, the one-dimensional placement information is used to "seed", i.e., construct an initial solution for, the partitioner. How soon the iteration can be stopped will affect the CPU efficiency of the overall implementation. Typically, iterative solvers have stopping criteria, or *convergence tests*, that are based on some norm of the residual vector for an iterate[6], which is taken to represent error with respect to the true solution. In practice, most norms are equivalent, and various heuristics (check convergence every j iterations, check differences of iterates rather than residual vectors, etc.) can reduce the time spent on convergence tests.

Constructing an initial min-cut partitioning solution from a one-dimensional placement solution wastes information, particularly if nothing is retained but memberships of vertices in "left" and "right" initial

[6]When solving the system $\mathbf{Ax} = \mathbf{b}$, the residual vector for a given iterate $\mathbf{x^k}$ is $\mathbf{b} - \mathbf{Ax^k}$.

partitions. If the final iterate will be sorted and split to induce an initial solution for the min-cut partitioner, then the iteration should terminate as soon as further changes will be inessential to the partitioner.[7] Determination of "inessential" fundamentally depends on the strength and stability of the partitioner, as will be discussed in the next subsection. However, regardless of what partitioner is used, solver iterations should certainly stop when the left and right groups stabilize. The work of [5] proposed a number of *correlation convergence* criteria, based on permutations, that may be useful in efficiently measuring such stabilization. Instead of residual norms, correlation convergence criteria use correlations and rank correlations between successive iterates to compute their similarity. Convergence is detected when such a measure becomes sufficiently close to 1, and iterations are stopped. (Note the analogy to residual norms which are used in traditional convergence criteria, but tend to 0 rather than to 1.) In Section 4 below, we provide evidence that simple and efficiently-computed measures of correlation or rank correlation between successive iterates indeed yield useful *correlation convergence* criteria.

3.4 The Solver-Partitioner Interface

A fourth degree of freedom in implementing the quadratic placement methodology is the "solver-partitioner interface", namely, the manner in which an initial solution for min-cut partitioning is constructed from a given solver iterate. The key decision concerns how much information to retain from the iterate when "seeding" the partitioner. Above, we noted that the usual practice is to sort coordinates of the iterate, then pre-seed some percentage of modules (corresponding to the most extreme coordinates) into the left and right initial partitions. Many implementation choices must be faced, e.g., whether the pre-seeded modules

[7]This precept also applies when the iterate is used to "seed" the partitioner. For example, one can seed the initial partitioning solution with only a percentage (say, 20%) of vertices having the most extreme coordinates (with all other vertices randomly assigned), because these vertices are more likely to be "correctly" assigned. The GORDIAN and GORDIAN-L placers use such a strategy [24] [31].

should be locked or unlocked within the partitioner; how to construct initial assignments for the remaining (not pre-seeded) modules; whether the pre-seeding should be based on module areas or module cardinalities; etc. In our experiments below, we apply the following procedure to create the initial bipartitioning solution.

- The *midpoint* of the iterate is determined, such that the sums of module areas on either side of the midpoint are as close to equal as possible.
- Fixed pads are assigned to left or right partitions according to whether they are to the left or right of the midpoint in the iterate.
- On the left (right) side of the midpoint, a prescribed *seeding percentage* of the modules with smallest (largest) coordinates are pre-seeded (but not locked) into the left (right) partition. The percentage is computed on each side according to module cardinality, rather than module area. In the experiments below, we study seeding percentages of 0%, 25%, 50% and 100%.
- Remaining (not pre-seeded, not pad) modules are randomly assigned to the left and right partitions, such that the resulting partitioning is balanced. More precisely, we randomly order these remaining modules, then assign each in turn to the partition that currently has smaller total module area.

3.5 Use of Modern (Multilevel) FM Partitioners

Recall that a motivation for our present investigation is that the use of numerical linear systems solvers that minimize quadratic wirelength may be a historical accident, resulting from the pre-1990's weakness of min-cut partitioners. The standard FM bipartitioning approach consists of iterative improvement based on the Kernighan-Lin algorithm, using the improvement of Fiduccia-Mattheyses [13]. The FM algorithm begins with some initial solution $\{X, Y\}$ and proceeds in a series of *passes*. During a pass, modules are successively moved between X and Y until each module has been moved exactly once. Given a current solution $\{X', Y'\}$, the previously unmoved module $v \in X'$ (or Y') with highest *gain* ($= cut(\{X' - v, Y' + v\}) - cut(\{X, Y\})$) is moved from X' to Y'. After each pass, the best solution $\{X', Y'\}$ observed during the pass becomes the initial solution for a new pass, and the passes terminate when a pass does not improve the initial solution.

Recent work [1] [2] [17] [18] [21] [20] has illustrated the promise of *multilevel* approaches for partitioning large circuits. Multilevel partitioning recursively clusters ("coarsens") the instance until its size is smaller than a given threshold, then unclusters ("uncoarsens") the instance while applying a partitioning refinement algorithm. Work in multilevel partitioning was originally prominent in the scientific computing literature for partitioning finite-element graphs [18] [21] [28]. Hendrickson and Leland [18] developed a very efficient multilevel partitioning algorithm, included in their Chaco package. Metis, another multilevel partitioning package targeted to finite-element graphs, was developed by Karypis and Kumar [21]. In the VLSI CAD community, previous multilevel works include [1] [10] [17], [20] and [2]. As shown in [20] and [2], multilevel implementations of the FM approach give the strongest and most stable results yet reported in the VLSI partitioning literature. Thus, our fifth degree of freedom assesses the use of multilevel FM versus traditional FM implementations.

4 Experimental Results

4.1 Experimental Setup

Our top-down placement testbed includes the following elements.

- Plain FM and multilevel FM bipartitioning engines. The plain FM implementation uses a LIFO gain bucket organization for improved performance [16]. The multilevel FM implementation uses a

CLIP-FM core [11] and follows the description of [2] with respect to use of heavy-edge matching for coarsening [20, 2], and the value of the matching ratio ($r = 0.33$) for coarsening/uncoarsening.

- Numerical iterations to minimize the squared-wirelength and linear-wirelength objectives. To minimize squared wirelength, we use a BiConjugate Gradient Stabilized (BiCGS) solver without preconditioner, following the conclusions of [5]. To minimize linear wirelength, we apply the Weiszfeld iteration described in [4], using the same BiCGS solver. For this objective, we also use an ILU preconditioner since linear-wirelength minimization is inherently harder than squared-wirelength minimization.

- A top-down quadratic placer framework. Within this framework, relevant implementation choices are:

  - Final (non-overlapping) module placements are evaluated by the sum of net bounding box half-perimeters.

  - Nets are modeled as weighted graph edges for the numerical solvers using the standard clique model for nets of degree 10 or less, and the directed star model for nets of degree greater than 10.

  - Pads (or block terminals) are kept fixed in the positions originally specified by the designer.

  - For multi-pin nets with pins outside the current partitioning instance, straightforward terminal propagation is used.

We use four standard-cell test cases from industry, which we read in Cadence LEF/DEF 4.5 format (see Table 1).

Our basic experiment explores the various degrees of freedom from the previous section, as follows.

- For each bipartitioning instance with ≥ 50 modules we use a solver to obtain a one-dimensional

Test Cases			
Test Case	Pad Cells	Core Cells	Nets
Case1	1083	5840	7637
Case2	182	8829	11962
Case3	711	12146	10880
Case4	185	20392	25634

Table 1: Parameters of four standard-cell test cases from industry.

placement minimizing either squared or linear wirelength. For smaller instances, we do not produce placements and instead use a random initial solution in the bipartitioning.

- We use the linear placement to pre-seed an initial bipartitioning solution, either fully (100%), partially (50% or 25%) or not at all (0%).
- We use either LIFO FM (FM) or ML CLIP-FM (MLFM) to obtain a minimum-cut exact bisection (using exact module areas, with tolerance equal to the largest individual module area in the instance). Note that when MLFM is used, its coarsening phase is constrained by the pre-seeding. Pre-seeded modules are not allowed to be matched to modules pre-seeded in the opposite partition.
- Exhaustive enumeration of all possible placements is used for end-cases having 5 modules or fewer.
- Each minimum-cut bisection is the best result from 5 multi-starts, with randomization in the initial assignment of non-pad/non-seeded modules, and in the heavy-edge matching based coarsening stage of MLFM.

Runtimes for our placer on a 300MHz Sun Ultra-10 are given in Table 2. We emphasize that a tuned implementation would be much more efficient, e.g., in practice solvers would not be run with such rigorous convergence criteria. For the top-level bipartition instances alone, the quadratic solver required 11 seconds (178 iterations) and 45 seconds (154 iterations), for Case 1 and Case 4 respectively. The 5 starts of LIFO

Partitioner	Analytical Placer	Case 1 (sec)	Case 4 (sec)
LIFO FM	Quadratic	110	660
	Linear (Weiszfeld)	180	767
ML Clip-FM	Quadratic	156	820
	Linear (Weiszfeld)	205	802

Table 2: **Total CPU times for our placer** (300MHz Sun Ultra-10) on smallest and largest test cases, under various configurations, with 100% pre-seeding.

FM required 2 and 11 seconds, and the 5 starts of ML CLIP-FM (including all clustering operations) required 4 and 20 seconds, for Case 1 and Case 4 respectively.

4.2 Experimental Data

For each experiment configuration and test case, we examine both the final placement result as well as the results of the top-level bisection step. In the context of the top-level bisection, we save 20 different iterates of the squared-wirelength minimization (BiCGS engine) to pre-seed partitioners in separate experiments. These 20 iterates are chosen uniformly spaced in the interval between the first and final solver iterates (the stopping criterion is for successive iterates to differ by less than $1e^{-8}$ times the norm of the residual). For each of these iterates, each partitioning engine, and each level of pre-seeding, Figures 1 through 4 show the best cuts achieved in 5 random starts, averaged over 5 separate trials. Oscillations in the Figures, particularly for MLFM results, are due to the randomizations inherent in the experimental setup. For linear-wirelength minimization only a one iteration is available, as Weiszfeld typically converges in a single iteration. The best cutsizes obtained when this iterate is used for pre-seeding (100%, 50%, 25% and 0% pre-seeding) are given in the captions of each Figure.

Tables 3 through 6 document the similarity of each iterate of the squared-wirelength minimization to the next iterate and to the final iterate, using correlation and rank correlation measures [29]. We additionally report the similarity of the resulting partitioning solutions to the solution achieved using the final iterate.

Here, the similarity measure is Hamming distance, i.e., the minimum number of modules that must be moved to transform one solution into the other.[8]

In our study of complete placement results, within each experimental configuration we pre-seed each partitioner call with the corresponding final solver iterate, and report the sum of net bounding box half-perimeters in the final placement. These results are given in Table 7.

4.3 Discussion

We first note that fully pre-seeded (100%) runs are still somewhat randomized as we do not pre-seed partitioning instances of size 5 through 50 (small instances with 5 or fewer cells are solved optimally with an enumerative approach). Figures 1 through 4 justify the traditional quadratic placement methodology, in the sense that a a (LIFO) FM partitioner clearly benefits from pre-seeding by a quadratic (squared-wirelength) solver. We see that full (100%) pre-seeding reduces the FM cutsize by as much as 35no (0%) pre-seeding. On the other hand, MLFM cutsizes are clearly not improved, and in some cases *worse* when pre-seeded with results of the quadratic solver. For all test cases MLFM with 0% pre-seeding is clearly superior to FM with 100% pre-seeding. This confirms that with modern partitioners, pre-seeding from analytical placements only hurts solution quality.

With regard to the use of a linear-wirelength minimizer, we observe that the FM partitions are still generally better with more pre-seeding, but that the improvement versus pre-seeding with the quadratic solver is somewhat unpredictable (in two cases, FM results are distinctly worse when pre-seeded with the Weiszfeld solution).[9] The MLFM partitioner is still hurt by pre-seeding, but the linear-wirelength pre-

[8]More precisely, we treat each bipartitioning solution as a 0-1 vector, so the Hamming distance between two partitioning solutions is a measure of how dissimilar two solutions are. If $X = x_i$ and $Y = y_i$ are two bipartitioning solutions, their Hamming distance is $\sum_{i=1}^{n} |x_i - y_i|$. If this quantity is larger than $n/2$ then the coordinates in Y are flipped and the quantity is recomputed.

[9]From the results of [30], we would expect superior performance from a Weiszfeld-seeded FM partitioner.

seeding is less damaging than the squared-wirelength pre-seeding. Again, our main conclusion is that a modern multilevel partitioner no longer requires pre-seeding by an analytical solver, particularly for large instances.

The Hamming distance studies in Tables 3 through 6 show that pre-seeding with early iterates leads to partitioning solutions that are structurally similar to those achieved with later iterates.[10] We also see that correlation and rank correlation to the next iterate increase steadily. Since using later iterates does not usually improve cutsize, this suggests that correlation convergence measures can be used for early termination of numerical solvers. Finally, we observe that linear-wirelength minima do not seem strongly correlated with squared-wirelength minima.

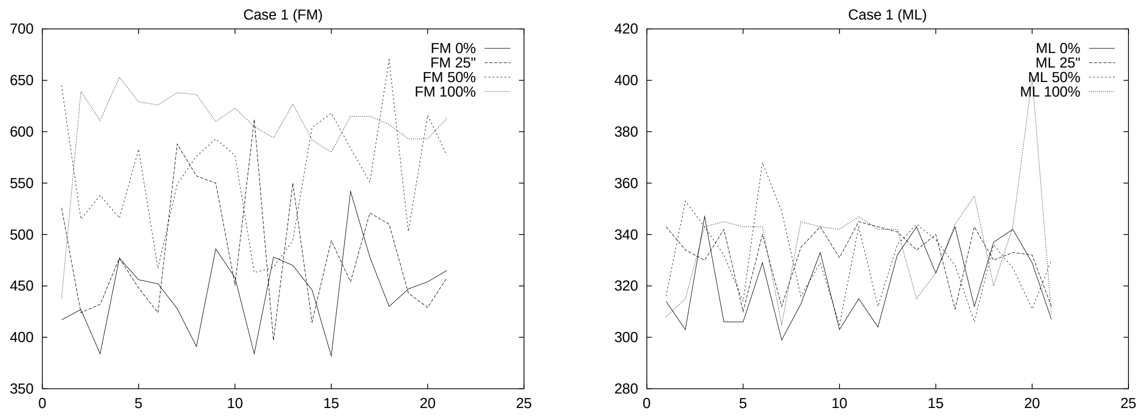


Figure 1: **Best cut after pre-seeding with solver iterates** for Case 1. The x-axis is the index of the iterate, and the y-axis is the cutsize. When pre-seeding with the final (converged) Weiszfeld iterate, the best cuts were 485, 492, 550 and 571 for FM (0%, 25%, 50% and 0% pre-seeding); and 315, 339, 328 and 307 for MLFM (0%, 25%, 50% and 0% pre-seeding).

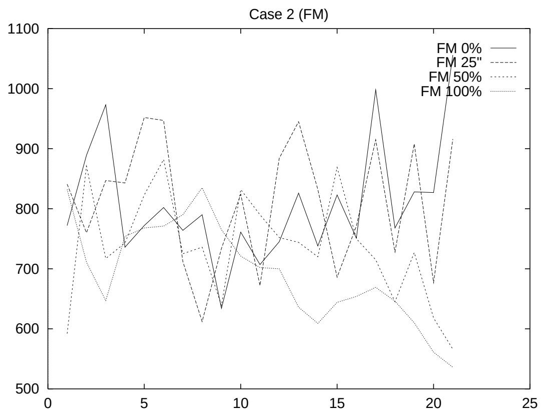
[10]The Hamming distances are occasionally surprisingly large, even though cutsizes are similar. We recognize that our experiments do not address other issues, notably (i) the number of multi-starts required for stable solution quality, and (ii) reproducibility of solution structure, that may yet reveal advantages of solver-based pre-seeding strategies. Since these issues move us into details of metaheuristics within the top-down placement, we defer them to future research.

Case 1 - Quadratic Wirelength												
Iterate	Corr.	Rank Corr.	Corr.	Rank Corr.	Hamming Distance to Final							
					LIFO FM				ML Clip-FM			
	to next		to final		0%	25%	50%	100%	0%	25%	50%	100%
1	0.98974	0.938893	0.843959	0.811998	1220	1100	1514	2594	832	723	1447	1576
2	0.998518	0.989313	0.887373	0.877041	1414	1170	1512	456	446	1069	1367	1543
3	0.996944	0.98943	0.902439	0.893202	1193	1117	1426	555	750	802	1459	1590
4	0.998996	0.992139	0.921435	0.909269	870	1264	1457	501	237	768	1179	980
5	0.999269	0.997498	0.932654	0.924666	1233	1427	1139	217	992	149	1025	971
6	0.998947	0.996191	0.941326	0.930699	1186	1276	1308	210	479	284	1443	984
7	0.999457	0.997867	0.951809	0.939749	940	1331	1481	354	99	961	1464	1028
8	0.999324	0.99767	0.958524	0.945523	1724	1189	1155	304	397	116	1447	937
9	0.999328	0.998888	0.965899	0.951144	770	1089	1267	126	209	687	1529	923
10	0.999833	0.999403	0.972098	0.954903	806	1480	1499	391	757	276	1065	1085
11	0.999643	0.998387	0.975049	0.957211	1440	1253	1149	136	156	69	1452	191
12	0.999698	0.998566	0.978848	0.96131	2065	1670	1157	155	546	606	1485	1627
13	0.999885	0.999276	0.982403	0.965475	1442	1395	1326	422	604	714	1475	273
14	0.99995	0.999491	0.98427	0.967885	1446	898	1133	213	997	91	1519	1692
15	0.999777	0.997455	0.985507	0.96939	1391	1278	957	197	301	1045	1532	901
16	0.9999	0.998801	0.988169	0.97446	1503	1368	1187	115	117	802	1437	216
17	0.99981	0.996813	0.989733	0.97735	1404	1021	1090	115	334	499	1472	184
18	0.999923	0.999515	0.991954	0.983116	1455	1372	1457	128	310	595	1101	1530
19	0.999944	0.999332	0.99318	0.984812	1049	1074	1296	205	110	977	1593	1638
20	0.999952	0.999312	0.994174	0.98658	1598	1109	1537	205	598	101	1129	1625
Final	-	-	1	1	0	0	0	0	0	0	0	0

Table 3: **Correlation convergence studies for the top-level bisection of Case 1** and pre-seeding with early iterates of the squared-wirelength minimization.

Case 2 - Quadratic Wirelength												
Iterate	Corr.	Rank Corr.	Corr.	Rank Corr.	Hamming Distance to Final							
					LIFO FM				ML Clip-FM			
	to next		to final		0%	25%	50%	100%	0%	25%	50%	100%
1	0.99445	0.705295	0.614096	0.392017	2378	2972	3731	4192	2367	299	2246	3861
2	0.995601	0.960569	0.638045	0.682129	2548	2228	2570	2145	2832	3315	2384	2668
3	0.998306	0.987488	0.659515	0.745051	2833	2165	2606	1449	278	791	2283	3895
4	0.998757	0.992223	0.675138	0.773531	3471	3478	2690	1251	1750	4019	2204	3757
5	0.998027	0.995468	0.689994	0.793538	3814	2470	2841	1311	1926	4402	917	3159
6	0.998181	0.997799	0.710364	0.81365	3756	2398	3064	1261	1689	3407	1832	3825
7	0.998545	0.998154	0.729696	0.826896	3752	2899	2291	1302	2710	339	2635	3864
8	0.999051	0.999162	0.750016	0.842943	3110	3243	2521	1382	2519	4388	2355	3791
9	0.999169	0.998872	0.7666	0.854092	2904	2628	2823	1219	2297	3307	2607	3847
10	0.999185	0.9993	0.785464	0.869296	2399	2743	2388	1028	2752	3866	2416	3829
11	0.998488	0.998786	0.803803	0.881228	2507	2923	2714	968	2529	3803	1829	3916
12	0.998938	0.999046	0.829433	0.897179	1906	3037	1883	1013	2434	402	2374	3616
13	0.998435	0.998875	0.851125	0.910766	2717	2379	2637	894	1985	4338	994	3833
14	0.998147	0.998721	0.87616	0.925172	2755	2721	2670	655	1656	3732	4062	1429
15	0.997945	0.998286	0.901718	0.939449	3107	2961	2549	902	1650	4372	2516	3162
16	0.997114	0.998252	0.925989	0.954149	2546	2510	1937	981	617	3846	2313	1387
17	0.998156	0.998953	0.95073	0.96779	2459	2642	1862	688	523	4389	2429	1495
18	0.998083	0.998727	0.967119	0.976922	2755	2307	1321	685	583	3264	3312	101
19	0.998301	0.99869	0.980637	0.98533	3324	2662	1682	478	2702	4061	2318	1224
20	0.999164	0.999046	0.999164	0.999046	2454	2606	2036	59	1744	3993	2737	1266
Final	-	-	1	1	0	0	0	0	0	0	0	0

Table 4: **Correlation convergence studies for the top-level bisection of Case 2** and pre-seeding with early iterates of the squared-wirelength minimization.



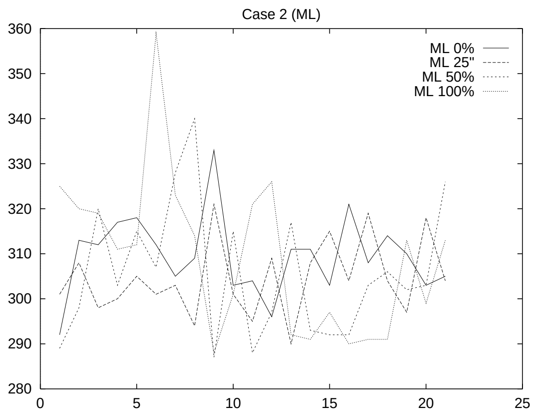
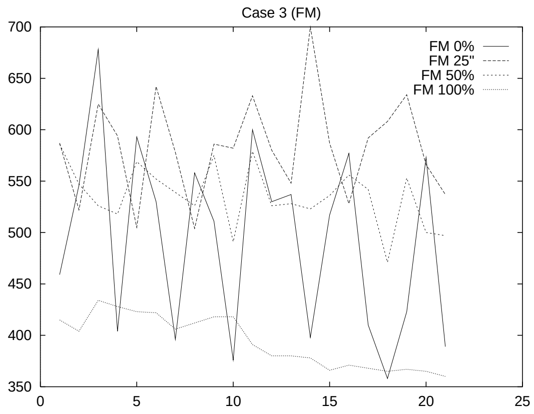


Figure 2: **Best cut after pre-seeding with solver iterates** for Case 2. The x-axis is the index of the iterate, and the y-axis is the cutsize. When pre-seeding with the final (converged) Weiszfeld iterate, the best cuts were 737, 745, 687 and 605 for FM (0%, 25%, 50% and 100% pre-seeding); and 303, 310, 302 and 315 for MLFM (0%, 25%, 50% and 100% pre-seeding).

Case 3 - Quadratic Wirelength												
Iterate	Corr.	Rank Corr.	Corr.	Rank Corr.	Hamming Distance to Final							
					LIFO FM				ML Clip-FM			
	to next		to final		0%	25%	50%	100%	0%	25%	50%	100%
1	0.977774	0.805804	0.607565	0.638514	3169	3114	3475	3871	3813	4266	1450	2905
2	0.990913	0.979015	0.690443	0.830686	1851	3700	2971	3886	277	3879	2984	2469
3	0.993253	0.995148	0.744298	0.875779	2723	3525	3683	3680	2838	2646	2838	1111
4	0.995701	0.996973	0.789769	0.898059	2669	3334	2955	2882	258	3948	1429	4361
5	0.997248	0.998161	0.825311	0.915763	2788	3745	3404	2546	453	4350	376	4414
6	0.997744	0.998767	0.852773	0.929199	3120	2974	3619	3044	187	4004	1405	4376
7	0.998409	0.999132	0.87671	0.940352	2868	3725	2866	3121	887	293	3486	791
8	0.998692	0.999151	0.895874	0.94939	2716	3143	3631	3069	2735	3264	1453	4331
9	0.998422	0.999177	0.912526	0.957789	2774	4148	3215	2682	3140	3955	1690	1224
10	0.998945	0.999413	0.92911	0.96547	2405	3593	2976	2827	3954	329	2857	2616
11	0.999375	0.999638	0.941674	0.971347	2508	3380	2999	2927	279	3923	1309	3004
12	0.998969	0.999268	0.950603	0.975456	3610	3974	3658	2954	3151	4377	1052	4036
13	0.99946	0.999728	0.961839	0.981017	2960	3512	3512	3155	4379	3844	3311	4170
14	0.999278	0.999556	0.96884	0.984048	2417	4267	3679	3056	4039	2638	1727	253
15	0.999619	0.999797	0.976438	0.98757	2775	3662	3285	2899	345	3765	1535	3960
16	0.999688	0.999816	0.981134	0.98958	2939	2911	3852	2992	1570	2494	1254	4397
17	0.999829	0.999903	0.985025	0.991328	3259	3432	4040	3157	3147	2953	1785	4400
18	0.999607	0.99973	0.98757	0.992461	2073	3582	3606	3214	2702	4010	1400	260
19	0.999868	0.999921	0.991186	0.994243	3657	3548	4144	3195	3139	2930	3185	4387
20	0.992901	0.995008	0.992901	0.995008	3184	2973	3892	3230	2713	318	3067	3748
Final	-	-	1	1	0	0	0	0	0	0	0	0

Table 5: **Correlation convergence studies for the top-level bisection of Case 3** and pre-seeding with early iterates of the squared-wirelength minimization.



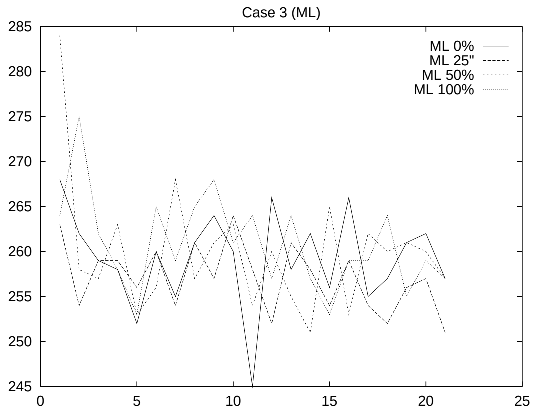
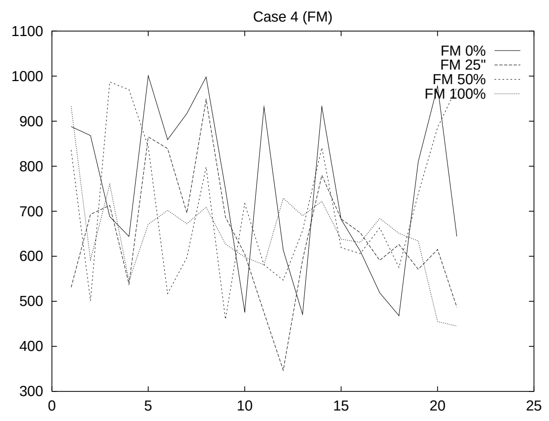


Figure 3: **Best cut after pre-seeding with solver iterates** for Case 3. The x-axis is the index of the iterate, and the y-axis is the cutsize. When pre-seeding with the final (converged) Weiszfeld iterate, the best cuts were 515, 492, 465 and 340 for FM (0%, 25%, 50257, 250, 262 and 251 for MLFM (0%, 25%, 50% and 100% pre-seeding).

Case 4 - Quadratic Wirelength												
Iterate	Corr.	Rank Corr.	Corr.	Rank Corr.	Hamming Distance to Final							
					LIFO FM				ML Clip-FM			
	to next		to final		0%	25%	50%	100%	0%	25%	50%	100%
1	0.982226	0.627657	0.449465	0.500442	10060	1694	2664	7230	3736	3491	274	3843
2	0.997357	0.955058	0.501609	0.822398	4321	3456	4176	3972	1165	141	91	549
3	0.996133	0.997686	0.528977	0.897615	3985	2479	4134	1951	3697	2972	3744	495
4	0.998251	0.998819	0.565941	0.911946	6956	5965	3863	2073	672	147	3235	2387
5	0.997302	0.99901	0.594907	0.922433	8704	3442	3410	3312	3799	128	197	492
6	0.997553	0.99931	0.633184	0.932492	6606	4215	3142	3434	3721	186	3244	522
7	0.997205	0.999182	0.670902	0.940816	4795	2242	3975	1379	1185	3472	321	525
8	0.996913	0.999497	0.713008	0.949826	7036	3866	4573	1038	1050	118	307	673
9	0.997377	0.999591	0.755451	0.95665	9381	4113	4587	818	3655	106	63	618
10	0.997351	0.999652	0.793298	0.962594	7964	4774	2866	908	506	119	105	2360
11	0.996165	0.999566	0.829006	0.967859	6906	3039	4214	891	3643	116	257	517
12	0.997065	0.999662	0.868215	0.97342	7586	2797	4029	1819	3700	171	3237	499
13	0.997577	0.999622	0.898789	0.977999	4857	5310	3685	1749	3669	3060	88	294
14	0.998607	0.999793	0.92385	0.982489	4596	5907	3705	1696	3698	3643	2554	313
15	0.998706	0.999715	0.94039	0.985462	7030	2051	4050	1044	259	98	3775	343
16	0.998054	0.999573	0.954994	0.988675	6853	3956	3746	951	713	141	277	340
17	0.998796	0.999746	0.970299	0.992146	5066	2175	3856	1446	3630	206	282	3348
18	0.999387	0.999803	0.980137	0.994402	5102	2505	2538	1108	3696	137	190	2338
19	0.999211	0.999756	0.986279	0.996104	4929	1803	4577	880	631	153	78	2388
20	0.999047	0.999685	0.999047	0.999685	7567	3330	3763	657	1111	210	116	413
Final	-	-	1	1	0	0	0	0	0	0	0	0

Table 6: **Correlation convergence studies for the top-level bisection of Case 4** and pre-seeding with early iterates of the squared-wirelength minimization.



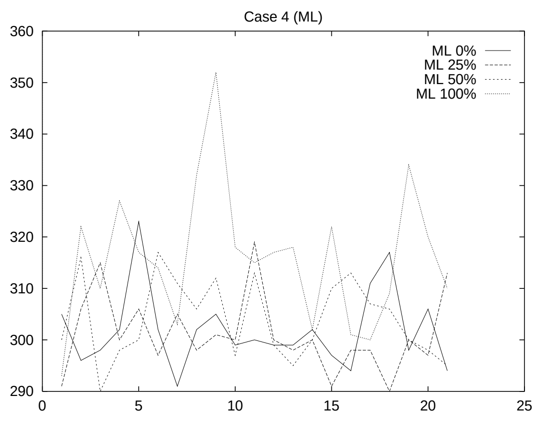


Figure 4: **Best cut after pre-seeding with solver iterates** for Case 4. The x-axis is the index of the iterate, and the y-axis is the cutsize. When pre-seeding with the final (converged) Weiszfeld iterate, the best cuts were 737, 754, 643 and 590 for FM (0%, 25%, 50% and 100% pre-seeding); and 318, 328, 337 and 331 for MLFM (0%, 25%, 50% and 100% pre-seeding).

Partitioner	Analytical Placer	Ave Final WL for % seeded			
		0%	25%	50%	100%
CASE 1					
LIFO FM	Quadratic	2.461	2.580	2.699	2.871
	Linear (Weiszfeld)	2.468	2.539	2.752	3.105
ML CLIP-FM	Quadratic	2.329	2.365	2.395	2.443
	Linear (Weiszfeld)	2.329	2.358	2.370	2.452
CASE 2					
LIFO FM	Quadratic	7.463	7.535	7.114	6.390
	Linear (Weiszfeld)	7.463	7.574	7.111	6.414
ML CLIP-FM	Quadratic	5.250	5.344	5.353	5.365
	Linear (Weiszfeld)	5.250	5.219	5.235	5.345
CASE 3					
LIFO FM	Quadratic	3.970	3.917	3.874	3.633
	Linear (Weiszfeld)	3.970	3.976	3.853	4.129
ML CLIP-FM	Quadratic	3.139	3.172	3.168	3.233
	Linear (Weiszfeld)	3.139	3.152	3.202	3.320
CASE 4					
LIFO FM	Quadratic	9.436	9.297	9.503	8.921
	Linear (Weiszfeld)	9.436	8.575	8.724	8.523
ML CLIP-FM	Quadratic	6.860	6.948	6.953	7.047
	Linear (Weiszfeld)	6.860	6.859	6.767	6.878

Table 7: **Average final wirelength results for top-down placement of Case 4**.

We conclude this section by discussing the total wirelength results for complete placements obtained with each of the experimental configurations. From Table 7, we see that conclusions obtained for individual bisection instances still apply to complete placements: (i) FM is weaker than MLFM; (ii) FM benefits from 100% pre-seeding (but not always from 25% pre-seeding) but MLFM does better with no pre-seeding than with pre-seeding from quadratic placement; and (iii) MLFM performs somewhat better when partially pre-seeded with the placements produced by the Weiszfeld algorithm, while FM does not benefit from such pre-seeding. Overall, the worst results were achieved by FM with no pre-seeding, with partial pre-seeding by a squared-wirelength solver, or with full pre-seeding by a linear-wirelength solver. The best results were achieved by MLFM with no pre-seeding or with linear-wirelength based pre-seeding.

5 Conclusions and Futures

We have synthesized the motivations and structure for a generic "quadratic placement" methodology, and developed a testbed that allows exploration of several key implementation decisions. Our experiments compare different combinations of partitioners, analytical solvers, and pre-seeding strategies within the solver-partitioner interface. We observe that:

- Traditional pre-seeding with a quadratic (squared-wirelength) engine does not improve either cutsize or placement results if a strong partitioner (e.g. ML Clip-FM) is used.
- If pre-seeding is used, earlier iterates are often as good as later iterates, and *correlation convergence* tests based on correlation and rank correlation between iterates can save CPU time by detecting when the relative order of module locations stabilizes.
- Pre-seeding with a linear-wirelength engine *may* be useful if issues such as stability and reproducibility of solution structure are considered – but such a conclusion, if true, would require a more elaborate experimental design to demonstrate.

These observations suggest that with the transition from classic FM partitioners to modern multilevel partitioners, quadratic engines may no longer be necessary in top-down placement, and may even lead to a loss of solution quality when applied.

Our ongoing research encompasses the following areas.

- We are improving our placement testbed to enable "apples-to-apples" comparison with commercial tools. This entails building infrastructure for timing-driven layout (industry-standard timing models, timing constraints formats, delay calculation and static timing analysis algorithms, etc.) as well as interfaces to leading routing engines. Our existing placement capability is extremely competitive for wirelength minimization, but routability analyses and legality-checking are inferior to those in commercial systems, making direct comparisons difficult.

- We are studying non-hierarchical alternatives for the interface between analytical solvers and the layout substrate. Several recent approaches, based on novel formulations for both solver and "legalizer", appear promising.

- Finally, several drivers suggest looking beyond the quadratic placement methodology. (1) Well-known limits of quadratic placers include inability to naturally model path timing constraints, invariance of orderings to unequal horizontal and vertical routing resources, and the requirement of pre-placed pads to "anchor" the analytical placement. (2) Future top-down design methodologies will tend to have smaller random-logic blocks in order to gain predictability; these may not be large enough for a quadratic placer to show its "global awareness" and runtime advantages. (3) The advent of block-based designs, with synthesized glue logic spread out over disconnected regions, may lead to a design planning - block building - assembly flow that is also ill-matched to quadratic placers. Thus, we must also seek new placement approaches that can be better suited to future placement contexts.

References

[1] C. J. Alpert, L. W. Hagen, and A. B. Kahng, "A Hybrid Multilevel/ Genetic Approach for Circuit Partitioning", *Proc. ACM SIGDA Physical Design Workshop*, April 1996, pp. 100-105.

[2] C. J. Alpert, D. J.-H. Huang and A. B. Kahng, "Multilevel Circuit Partitioning", *Proc. ACM/IEEE Design Automation Conference*, June 1997, pp. 530-533.

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

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

[5] C. J. Alpert, T. Chan, D. J.-H. Huang, I. Markov and K. Yan, "Quadratic Placement Revisited", *Proc. ACM/IEEE Design Automation Conference*, June 1997, pp. 752-757.

[6] S. Balay, W. Gropp, L. Curfman McInnes, B. Smith, "PETSc 2.0 User's Manual", Argonne National Laboratory, 1995 `http://www.mcs.anl.gov/petsc/petsc.html`

[7] R. Barrett, M. Berry, T. F. Chan, et al. "Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods", *SIAM Press* 1994, `http://netlib2.cs.utk.edu/linalg/html_templates/Templates.html`

[8] T. Bui, C. Heigham, C. Jones and T. Leighton, "Improving the Performance of the Kernighan-Lin and Simulated Annealing Graph Bisection Algorithms", *Proc. ACM/IEEE Design Automation Conference*, June 1989, pp. 775-778.

[9] C. K. Cheng and E. S. Kuh. "Module Placement Based on Resistive Network Optimization", *IEEE Trans. on Computer Aided Design* 3 (1984), pp. 218-225.

[10] J. Cong and M'L. Smith, "A Parallel Bottom-Up Clustering Algorithm with Applications to Circuit Partitioning in VLSI Design", *Proc. ACM/IEEE Design Automation Conference*, June 1993, pp. 755-760.

[11] S. Dutt and W. Deng, "VLSI Circuit Partitioning by Cluster-Removal Using Iterative Improvement Techniques", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, November 1996, pp. 194-200.

[12] H. C. Elman, M. P. Chernesky, "Ordering Effects on Relaxation Methods Applied to the Discrete One-Dimensional Convection-Diffusion Equation", *SIAM J. Numer. Anal.*, 30(5) (1993), pp. 1268-1290.

[13] C. M. Fiduccia and R. M. Mattheyses, "A Linear-Time Heuristic for Improving Network Partitions", *Proc. ACM/IEEE Design Automation Conference*, June 1982, pp. 175-181.

[14] K. Fukunaga, S. Yamada, H. S. Stone, and T. Kasai, "Placement of Circuit Modules Using a Graph Space Approach", *Proc. ACM/IEEE Design Automation Conference*, June 1983, pp. 465-471.

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

[16] L. W. Hagen, D. J.-H. Huang, and A. B. Kahng, "On Implementation Choices for Iterative Improvement Partitioning Algorithms", *Proc. European Design Automation Conference*, September 1995, pp. 144-149.

[17] S. Hauck and G. Borriello, "An Evaluation of Bipartitioning Techniques", *Proc. 16th Conf. on Advanced Research in VLSI*, 1995, pp. 383-402.

[18] B. Hendrickson and R. Leland, "A Multilevel Algorithm for Partitioning Graphs", *Proc. Supercomputing*, 1995. See also Tech. Rep. SAND93-1301, Sandia National Laboratories, 1993.

[19] D. J.-H. Huang and A. B. Kahng, "Partitioning-Based Standard-Cell Global Placement with an Exact Objective", *Proc. ACM/IEEE Intl. Symp. on Physical Design*, Napa, April 1997, pp. 18-25.

[20] G. Karypis, R. Aggarwal, V. Kumar and S. Shekhar, "Multilevel Hypergraph Partitioning: Application in VLSI Domain", *Proc. ACM/IEEE Design Automation Conference*, June 1997, pp. 526-529.

[21] G. Karypis and V. Kumar, "Multilevel Graph Partitioning Schemes", P. Banerjee and P. Boca, editors, *Proc. Intl. Conf. on Parallel Processing*, 1995, vol. 3, pp. 113-122.

[22] B. W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs", *Bell Syst. Tech. J.* 49(2) (1970), pp. 291-307.

[23] B. Krishnamurthy, "An Improved Min-Cut Algorithm for Partitioning VLSI Networks", *IEEE Trans. on Computers*, 33(5) (1984), pp. 438-446.

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

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

[26] 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) (1994), pp. 719-725.

[27] R. H. J. M. Otten, "Automatic Floorplan Design", *Proc. ACM/IEEE Design Automation Conference*, June 1982, pp. 261-267.

[28] R. Ponnusamy, N. Mansour, A. Chaudhary, and G. C. Fox, "Graph Contraction for Mapping Data on Parallel Computers: A Quality-Cost Tradeoff", *Scientific Programming* 3(1) (1994), pp. 73-82.

[29] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd Ed., Cambridge: Cambridge University Press, 1992.

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

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

[32] 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.

[33] R. S. Tsay and E. Kuh, "A Unified Approach to Partitioning and Placement", *IEEE Trans. on Circuits and Systems*, 38(5) (1991), pp. 521-633.

[34] J. Vygen, "Algorithms for Large-Scale Flat Placement", *Proc. ACM/IEEE Design Automation Conference*, June 1997, pp. 746-751.

[35] 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.

[36] G. J. Wipfler, M. Wiesel and D. A. Mlynski, "A Combined Force and Cut Algorithm for Hierarchical VLSI Layout, *Proc. ACM/IEEE Design Automation Conference*, June 1982, pp. 671-677.