ARP: A CONVEX OPTIMIZATION BASED METHOD FOR GLOBAL PLACEMENT

Hussein Etawil, Shawki Areibi, and Anthony Vannelli

Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario
Canada N2L 3G1

ABSTRACT

In this paper, we present a new method for cell placement. The method is based on a new metric for wirelength that ensures no overlap among cells sharing common nets (repeller model). Moreover, new forces working on the cells are added to the new metric to attract the cells to the less dense regions and help spread out the cells within the placement area. Minimizing traditional metrics (linear or quadratic) results in a placement with substantial amount of overlap. The methodology iterates between global optimization and slicing the placement area to diminish cell overlap and attain uniform distribution of the cells within the placement floor. To help cells spread out, hard constraints are added to the problem in each iteration resulting in a further constrained version of the original problem. Unlike these approaches, no hard constraints are required in the new approach. Besides, the new metric is convex and versatile in the sense that it can be applied to placement problems with no fixed cells (i.e, FPGAs).

A preliminary version of the new placement method (referred to as ARP: Attractor-Repeller Placer)is tested using a set of MCNC *benchmarks* [17] and the results obtained are very competitive with up-to-date results reported in the literature.

1. INTRODUCTION

Cell placement is the subtask of the VLSI circuit layout that involves arranging the cells on the chip area such that the layout is routable and the overall area of the chip is minimum. Other objectives include minimum delay, minimum power consumption and minimum heat [2, 24, 7]. A common objective function in performing the placement is minimizing total wire-length such that overlap among cells is prevented. In [19], a force-directed formulation of the placement problem is proposed and applied to a set of industrial boards. The immense increase in problem size quickly lead to more powerful

SOME PRELIMINARY RESULTS OF THIS WORK APPEARED IN ICCAD'99

Currently at School of Engineering, University of Guelph

The research of the third author is partially supported by a Natural Sciences and Engineering Research Council of Canda (NSERC) operating grant (OGP 0044456)

techniques. Currently, placement methods that can handle large design instances can be divided into two classes based on how they deal with cell overlap while computing the placement. In the first class, the placement methods start with an initial placement and iteratively modifies the existing placement in an endeavor for better one. The placement is kept overlap-free during the search process. In this class of methods, Simulated Annealing based placers [22, 21] have been successful in handling large circuits. The second class involves exhaustive slicing of the placement floor associated with partitioning the set of the cells. Within this class, Min-cut ([16],[1],[13]) based placers have been successfully applied to industrial circuits. Analytic (quadratic or linear) objective function combined with partitioning the placement area [14, 10, 8] are also within this class of placers that can handle large instances.

Analytic placement techniques have acquired the attention of many researchers since they produce good solution quality in reasonable times. The quadratic objective has been used in most of the analytic approaches to placement [14, 2, 24, 6]. Linear objectives have been used by Gordian-L [8] method and recently a new approximation to the linear objective function for wire-length has been proposed in [4]. Numerical stability problems associated with linear objective functions made the quadratic objective more attractive. Using either metric, the placement is generated by optimizing the wire-length objective function. The result of the optimization process is typically a placement with great deal of overlap among the cells. To attain uniform distribution of cells on the placement area (fully utilize the placement area), most placement methods use min-cut partitioning. For instance, in [23], min-cut partitioning is used to create hierarchical subproblems. In [14, 2, 24] min-cut partitioning combined with first moment (center-of-gravity) constraints is used to induce further constrained version of the initial problem.

Typically, upon the first iteration of the optimization process, most of the cells collapse in the center of the layout region. Such a placement is not adequate for creating legal placement as the amount of overlap among the cells is substantial and cells are not near their final positions. In Gordian [14], Ritual [2] and Pcube [24], upon the first iteration, partitioning and first moment constraints are introduced. Specifically, the placement floor is partitioned into four quadrants with four different centers of gravity. Subsequently, the set of cells is partitioned into four by assigning each cell to one of the different quadrants of the placement floor. New constraints are added to the formulation resulting in a more constrained problem. The global optimization is performed again with the added constraints resulting in a better spread of the cells on the placement area. Following, each quadrant is again sliced into four sub-quadrants and the global optimization is performed again. The execution is terminated when each cell has been assigned to a distinct sub-quadrant.

The partitioning approach eventually leads to a final solution with minimum overlap. However, the fact that hard (center-of gravity) constraints are required to spread out the cells makes the problem harder to solve. Besides, performing the min-cut partitioning consumes a significant amount of computational efforts. Furthermore, solution quality may be deteriorated as a consequence of possible erroneous assignment of cells to the different partitions.

These deficiencies motivated researchers to seek an alternative to the partitioning approach as a means of spreading out the cells during the computation of the global placement. In [10], a free partitioning directed-force method is proposed in which additional forces have been added to the traditional quadratic wire-length. The added force vector contains forces working on the cells in the x and y directions. The existence of the force vector ensures that connected cells are placed such that their locations are spatially shifted. The force vector is computed as a function of the cell coordinates, and the supply and demand concept which requires moving cells from dense to sparse regions on the on the placement area. The force vector is updated in every iteration. In this approach, finding a placement is transformed

into determining the cell force vector. The method has been successfully applied to large industrial circuits. A disadvantage of the approach is the fact that determining the force vector requires computing the solution to Poisson's equation which can be computationally expensive, especially for large size instances.

Another approach to partitioning free placement used nonlinear programming [9]. Specifically, a target estimate is subtracted from the quadratic wire-length of each pair of connected cells and the resulting function is squared. The idea is that, upon minimization, a target estimate is maintained between the geometric locations of each pair of connected cells. The algorithm proceeds by performing global optimization followed by legalizing the placement (removing cell overlap). The process is repeated until a maximum number of iterations is exceeded, or the improvement in wire-length over three successive iterations is either worse or not significant compared to the best wire-length obtained so far. However, the proposed model is not convex and consequently, a global minima is not guaranteed. Also, it is not clear if the combination of global optimization and legalizing the placement lead to a uniform distribution of the cells with the placement area.

In this work we present a new formulation of wire-length that (like the work in [9, 10]) ensures no overlap between connected cells (that is, cell repeller model). We propose several convex functions that attain the desired repelling model. The repeller model accounts for cells sharing common nets, but cells with no common nets may still overlap, and accordingly uniform spread out of the cells within the placement floor may not be accomplished using only the repeller model. To force cells to spread out, we propose adding new forces to the repeller model that acts on the cells in the dense regions and force them to move towards the sparse regions on the placement floor. Like the approach in [10], in our approach, the placement problem problem (which has always been formulated as constrained problem), is modeled as unconstrained optimization problem. This has the advantage of optimizing a simpler problem. A major impact of our formulation would be on cell placement with no fixed cells (i.e, I/O pads). This is owing to the fact that, in the absence of fixed cells, the traditional formulation of wire-length lacks the capability of forcing cells apart even by a small margin. In such scenario, the partitioning approach would fail too. This is because, fixed cells attract movable cells to the boundaries of the chip, and accordingly play a crucial role in spreading the cells in the absence of any cell-repelling forces. Thus, for the traditional formulations to work, the existence of the I/O pads is extremely important. In our formulation, however, the existence or absence of the I/O pads is not as important as in case of the traditional formulation. This is because, as we will see later, cell overlap is diminished by the cell repellers and the uniform distribution of the cells within the placement area is a result of the joint efforts of the cell repellers and cell attractors. Furthermore, our repeller model is versatile in the sense that it is applicable to a variety of problems where a target distance is desired between connected components.

2. PRELIMINARIES

A circuit netlist is represented by undirected hypergraph $G(V, E)$ where $V = \{v_1, v_2, \cdots, v_n\}$ is the set of vertices, and $E = \{e_1, e_2, \cdots, e_m\}$ is the set of edges of the hypergraph. The set of vertices represents the cells to be placed and the set of edges represents the signal nets connecting the cells. The hypergraph is converted into a graph (that is, a hypergraph with hyper-edge sizes of 2) via either a star or a clique model [4]. Each edge e_j is assigned a positive weight w_j. The placement task seeks to place the cells within the $XY-$plane of the placement floor such they do not overlap. Depending on whether a linear or quadratic measure is used, the cost of an edge connecting a pair of cells i and j with geometric

locations (x_i, y_i) and (x_j, y_j) is the weighted l_1 or the squared l_2 norm of the difference position vector $(x_i - x_j, y_i - y_j)$. Assuming a quadratic measure, the total cost is given as the sum of the cost over all edges; that is:

$$\phi(x, y) = \sum_{1 \le i < j \le N} w_{ij}[(x_i - x_j)^2 + (y_i - y_j)^2] \tag{1}$$

Formulation (1) can be written in matrix form:

$$\phi(x, y) = \frac{1}{2}\mathbf{x}^T \mathbf{C}\mathbf{x} + \mathbf{d}_x^T \mathbf{x} + \frac{1}{2}\mathbf{y}^T \mathbf{C}\mathbf{y} + \mathbf{d}_y^T \mathbf{y} + \mathbf{t} \tag{2}$$

Vectors $\mathbf{x}$ and $\mathbf{y}$ denote the coordinates of the N movable cells; matrix $\mathbf{C}$ is the Hessian matrix; vectors $\mathbf{d}_x^T$ and $\mathbf{d}_y^T$ and the constant term $\mathbf{t}$ result from the contributions of the fixed cells. Normally the first moment constraints are added to force the distribution of the cells to be uniform around the center of the placement area. It follows that the quadratic placement model is given as:

$$\begin{aligned} &\text{Min } \phi(x, y) \\ &\text{s.t. } A_x x = b_x \\ &\quad A_y y = b_y \\ &\quad l_x \le x_i \le u_x \\ &\quad l_y \le y_i \le u_y \end{aligned}$$

where A_x and A_y are $q \times n$ matrices; q is the number of regions into which the placement area has been partitioned. The $q \times 1$ vectors b_x and b_y represent the centers of the q regions. The parameters l_x, u_x, l_y and u_y are lower and upper bounds on the x and y coordinates of the cells. Clearly, the above optimization problem can be split into two 1-dimensional subproblems and each subproblem can then be solved independently.

3. NEW FORMULATION

As we indicated previously, we propose several functions that achieve the desired cell-repelling model. The following theorem presents the details of the new models.

Theorem 1 *Let* $\zeta : \Re^m \longrightarrow \Re$, *and given by* $\zeta = \| \mathbf{v} \|_2^2$, *and* $\mathbf{v} = (v_1, v_2, \cdots, v_m)$ *is m-dimensional vector, then* $\eta(\zeta) = \zeta + \rho(\zeta)$ *is convex for* $\zeta \in [1, \infty)$, *provided* $\rho(\zeta) \in \{-\mathbf{ln}(\zeta), \mathbf{e}^{1-\zeta}\}$.

The detailed proof of this theorem is presented in [11]. It includes two scenarios depending on whether vector $\mathbf{v}$ is independent or dependent (That is, whether the elements of $\mathbf{v}$ are independent or dependent variables). In the first scenario, we proved that the function $\eta(\zeta)$ is convex for $\zeta > 1$, quasi-convex when $\zeta = 1$ and non-convex when $\zeta < 1$. In the second scenario, we studied the case when vector $\mathbf{v} = \mathbf{u} - \mathbf{r}$ where $\mathbf{u}$ and $\mathbf{r}$ are $m-$dimensional vectors. Specifically, we proved that function $\eta(\zeta)$ is quasi-convex when $\| \mathbf{u} \|_2^2 \geq 1$ and $\| \mathbf{r} \|_2^2 \geq 1$, and strictly convex if $\| \mathbf{u} \|_2^2 > 1$ and $\| \mathbf{r} \|_2^2 > 1$, and either the elements of $\mathbf{u}$ or $\mathbf{r}$ are constant.

To adapt the model to the placement problem, we let $\mathbf{u} = p_i = (x_i, y_i)$ and $\mathbf{r} = p_j = (x_j, y_j)$ be the position vectors of the geometric locations of cells i and j. We also let $\mathbf{v} = p_i - p_j$ and we define

$$z_{ij} = \frac{(w_{ij} \| p_i - p_j \|_2^2)}{d} \tag{3}$$

where $\| p_i - p_j \|_2^2 = (x_i - x_j)^2 + (y_i - y_j)^2$ is the squared l_2 norm of $p_i - p_j$ (that is, the squared distance between the geometric locations of the pair of connected cells i and j), w_{ij} is the connectivity weight between the connected cells, and $d > 0$ is constant. Geometrically, z_{ij} is a circle of radius d.

According to our proof in [11], it follows that the function

$$f(\mathbf{z}) = \begin{cases} \sum_{1 \le i < j \le N} z_{ij} + \rho(z_{ij}) & z_{ij} > 1 \\ 0 & \text{Otherwise} \end{cases} \quad (4)$$

where $\mathbf{z} = \{z_{ij} : i, j = 1, 2 \ldots, N\}$ and $\rho(z_{ij}) \in \{-\mathbf{ln}(z_{ij}) - 1, \mathbf{e}^{z_{ij}-1} - 2\}$ is convex if, at least, one cell is fixed. The non-convex portion of $f(\mathbf{z})$ (corresponding to $z_{ij} \in [0, 1)$) is flattened out giving $f(\mathbf{z})$ a tea-cup shape. The fact that $f(\mathbf{z})$ is flat in the interval $z_{ij} \in [0, 1]$ implies that $f(\mathbf{z})$ has multiple solutions in this region. However, line search numerical optimization methods concludes the search once the first optimal answer is encountered. This observation suggests that if the initial solution is outside the flat region (that is each $z_{ij} > 1$), the search will be terminated when $z_{ij} = 1$ (geometrically, it means that z_{ij} is reduced to a *unit* circle). That is, the optimization process will conclude that $z_{ij} = 1$ is the optimal answer. From (3), $z_{ij} = 1$ implies that the weighted squared distance between the geometric locations of the pair of connected cells equals an estimate of d units. Thus, $f(\mathbf{z})$ is a *repeller* model. Clearly, if $\rho(z_{ij}) = 0$ and $d = 1$, the repeller model $f(\mathbf{z})$ reduces to the traditional wire-length formulation given by (1).

4. CELL SPREADING

The repeller model ensures no overlap among connected cells. However, cells sharing no common nets still overlap as they are not accounted for in the repeller model. In fact, the repeller engine can achieve cell spreading if the estimate d in (3) is set to a relatively larger value with respect to the average cell dimensions; that is width and height. However, we found that increasing the intensity of the repelling engine (which corresponds to setting d to a higher value) causes excessive stretching of the short nets, and accordingly disperse highly connected cells on the placement floor. The result is a deterioration in the total wire-length. To spread out the cells without deteriorating the wire-length, we propose adding new forces to the repeller model to regulate the distribution of the cells within the placement floor. The basic idea of our approach is to force cells to spread over the placement area without restricting their movement. Specifically, the new forces attract cells to sparse regions within the placement area. That is, they encourage a cell to move to a sparse region in a direction conforming with the direction of the cell movement. The approach involves adding *dummy fixed* cells to the less dense regions of the placement floor and establishing connections between these new fixed cells and the movable cells in the dense regions. The new connections are, in fact, new nets added to the netlist. It follows that during the computation of the global placement, the dummy fixed cells exert additional forces on the movable cells and force them to move towards the less dense regions where the dummy fixed cells reside. Thus, movable cells fill the sparse regions, and accordingly better cell spreading is achieved in each iteration.

The first step involves identifying dense and sparse regions on the placement area. In the following section, we present how these regions are identified.

4.1. Identifying Dense and Sparse Regions

The procedure of identifying dense and sparse regions depends on the cell distribution and the variability of cell area (generally, cell width and height is not the same for all cells). Specifically, for each cell i

with geometric location (x_i, y_i), an $\ell_w \times \ell_h$ rectangular window w_i centered at (x_i, y_i) is imposed on the placement floor and the total area A_a of all cells with centers enclosed by w_i is computed; i.e:

$$A_a = \sum_{k=1}^{r} c_w^k c_h^k$$

where c_w^k and c_h^k are the width and height of cell k and r is the number of cells enclosed by window w_i. Next, region $\mathcal{R}_i$ surrounded by w_i is regarded sparse only if

$$A_a < A_w$$

where

$$A_w = \ell_w \ell_h$$

In our implementation, ℓ_w and ℓ_h have been expressed as linear functions of the chip width W and chip height H and adaptively varied from iteration to another; i.e:

$$\ell_w = \alpha_t W$$

and

$$\ell_h = \alpha_t H$$

where α is a constant that is updated from iteration to iteration according to [1]

$$\alpha_{t+1} = 0.95\alpha_t, \quad t = 1, 2, \cdots, \mathcal{I}$$

where t is the iteration number, $\mathcal{I}$ is the total number of iterations and $\alpha_1 = 0.1$. Cells that have been collapsed in a sparse region are not considered when new sparse regions are being identified. Each sparse region $\mathcal{R}_i$ is then split into 4 quadrants and the center of the sparsest quadrant (in terms of number of cells that fall inside a quadrant) (x_a^i, y_a^i) become a center of a new cell attractor. A cell attractor is, merely, a *dummy fixed* cell located at (x_a^i, y_a^i). Figure (1) illustrates an example of sparse and dense regions. Let

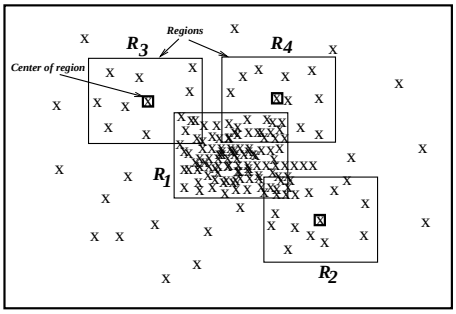

Figure 1: Region R_1 is dense, while regions R_2, R_3 and R_4 are sparse.

$$\mathcal{A} = \{(x_a^1, y_a^1), (x_a^2, y_a^2), \cdots, (x_a^q, y_a^q)\}$$

be the set of locations of the q cell attractors. The attractors in $\mathcal{A}$ are used to divide the set of cells among the different identified sparse regions on the placement floor. Precisely, for each cell i an attractor is selected based on a certain *cell-attractor* assignment criteria. Subsequently, a connection is established between cell i and the selected cell attractor. In other words, a new two-cell net is added to the netlist of the circuit. The details of the cell-attractor assignment are presented in the following section.

[1] We need to stress that this formula is empirical and it is entirely based on experimentation with the benchmarks.

4.2. Cell-Attractor Assignment

When assigning cells to sparse regions, the aim is to attain cell spreading while preventing any possibility of excessive stretching of short nets in subsequent iterations. In general, the deterioration in net wire-length is correlated with the distance between its cells and the attracting dummy cells in the sparse regions. Greater distances normally correspond to excessive net stretching. Thus, the criteria to assign a cell to a sparse region (or equivalently cell attractor) should be based on how many distance units that separates the geometric location of that cell from the geometric location of the dummy attracting cell in that particular sparse region. Based on this, we propose general *cell-attractor* assignment criteria that can be expressed in terms of the distance between the geometric locations of the cells and the attractors. Specifically, the criteria are based on the well known inequality between the *minimum*, *harmonic mean*, *geometric mean*, *arithmetic mean* and *maximum* of a set or a sequence of positive numbers [18]. To formalize the discussion, let $\mathcal{D}^i = \{d_1, d_2, \cdots, d_q\}$ be the sequence or set of distance between the geometric location of cell i and that of each attractor (dummy cell) in $\mathcal{A}$. To facilitate the analysis, we assume $\mathcal{D}^i$ is the set of distance between the geometric location of cell i and the attractors in the x-direction. For instance $d_1 = |x_i - x_a^1|$ is the distance (in the x-direction) between cell i and the attracting dummy cell with coordinates (x_a^1, y_a^1).

The harmonic mean of the nonnegative sequence of numbers $\mathcal{D}^i$ is defined as

$$H(\mathcal{D}^i) = \frac{q}{1/d_1 + 1/d_2 + \ldots + 1/d_q}$$

The geometric mean of the same sequence is defined as

$$G(\mathcal{D}^i) = (d_1 d_2 \ldots d_q)^{1/q}$$

and the arithmetic mean is given by

$$A(\mathcal{D}^i) = \frac{d_1 + d_2 + \ldots + d_q}{q}$$

We also define

$$D_{min} = min\{d_1, d_2, \ldots, d_q\}$$

and

$$D_{max} = max\{d_1, d_2, \ldots, d_q\}$$

For the finite sequence of positive numbers $\mathcal{D}^i$, we have [18]

$$D_{min} \leq H(\mathcal{D}^i) \leq G(\mathcal{D}^i) \leq A(\mathcal{D}^i) \leq D_{max} \tag{5}$$

with equality if and only if

$$d_1 = d_2 = \ldots = d_q$$

Inequality (5) provides a set of an intact criteria to control the degree of cell spreading and accordingly the extent of net stretching. For instance, if D_{min} is used as a cell-attractor assignment criterion, cell i will be connected to an attractor in the closest zone of sparse regions. In such case, cell i would be displaced by a relatively small distance with respect to its most recent geometric location in the subsequent iteration. Thus, excessive stretching in its nets is not highly likely to take place. If $H(\mathcal{D}^i)$ is

employed as a cell-attractor assignment criterion, each cell i will be assigned to a cell attractor located in a sparse region that is at least (or it can be at most) $H(\mathcal{D}^i)$ units from its most recent location, depending on whether $H(\mathcal{D}^i)$ is used as lower (or upper) bound on the distance separating cell i from the different sparse regions. Hence, more spreading but more stretching in short nets is expected compared to the previous case.

To make the discussion formal, let $\mathbf{s}_1$, $\mathbf{s}_2$, $\mathbf{s}_3$ and $\mathbf{s}_4$ be disjoint subsets such that

$$\mathcal{D}^i = \cup_{j=1}^{4} \mathbf{s}_j$$

and are given by

$$\mathbf{s}_1 = \{d_k : D_{min} \leq d_k < H(\mathcal{D}^i), k = 1, \ldots, q\}$$
$$\mathbf{s}_2 = \{d_k : H(\mathcal{D}^i) \leq d_k < G(\mathcal{D}^i), k = 1, \ldots, q\}$$
$$\mathbf{s}_3 = \{d_k : G(\mathcal{D}^i) \leq d_k < A(\mathcal{D}^i), k = 1, \ldots, q\}$$
$$\mathbf{s}_4 = \{d_k : A(\mathcal{D}^i) \leq d_k \leq D_{max}, k = 1, \ldots, q\}$$

Each subset or subsequence $\mathbf{s}_j$ corresponds in essence to a group of cell attractors or equivalently a group of sparse regions. In fact, each group of sparse regions represents a sparse zone on the placement floor. The group of sparse regions located at distances given in $\mathbf{s}_1$ represents the closest sparse zone to cell i. Subset or subsequence $\mathbf{s}_2$ corresponds to the next closest sparse zone to cell i and so on.

Clearly, inequality (5) provides an ordering of the sparse zones according to how far they are from cell i. It follows that, a cell attractor can be selected based on inequality (5) and the corresponding ordering of the sparse regions given by $\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3$ and $\mathbf{s}_4$. Dispersion of the distance sequence $\mathcal{D}^i$ gives clues on how to select a cell attractor for cell i. Specifically, the *variability* $\varrho(\mathcal{D}^i)$ of sequence $\mathcal{D}^i$

$$\varrho(\mathcal{D}^i) = \frac{\sigma(\mathcal{D}^i)}{A(\mathcal{D}^i)}$$

(where $\sigma(\mathcal{D}^i)$ is the standard deviation of sequence $\mathcal{D}^i$ given by $\sigma(\mathcal{D}^i) = \sqrt{\frac{1}{q}\sum_{k=1}^{q}(d_k - A(\mathcal{D}^i))^2}$) can

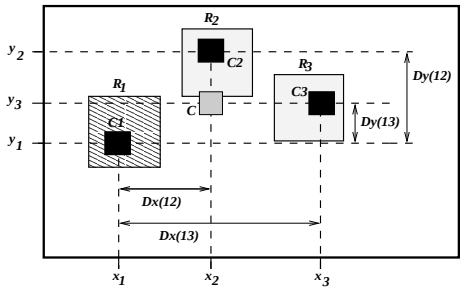

Figure 2: Region R_1 is dense while regions R_2 and R_3 are sparse. Cell C_1 located in R_1 with geometric location (x_1, y_1) is hooked to the dummy fixed cell C with coordinates (x_2, y_3) where x_2 and y_3 are the x and y coordinates of C_2 and C_3 (which are the closest cells to C_1 in the x and y directions respectively). Note that $Dx(ij)$ and $Dy(ij)$ are the distances in the x and y directions between cell i and cell j.

be utilized to characterize the dispersion of the distance sequence. Accordingly $\varrho(\mathcal{D}^i)$ provides clues on a suitable zone of sparse regions where cell i can be displaced to. If $\varrho(\mathcal{D}^i)$ is small, then any group

of cell attractors can be chosen without expecting any major change in the solution ($\varrho(\mathcal{D}^i)$ is zero, or equivalently $\sigma(\mathcal{D}^i)$ is zero, implies that cell i is at equidistant from all cell attractors). On the other hand, if $\varrho(\mathcal{D}^i)$ is relatively large, then a group can be selected depending on the circuit design style, size and netlist sparsity. For instance, thresholds of $\varrho(\mathcal{D}^i)$ can be designated for each group of cell attractors based on experimentation with different problem size and design style.

In our implementation, we applied the new placement method to *standard-cell* design style which is widely used in ASIC (Application Specific Integrated Circuits). In this design style, as we pointed out in previous chapters, cells are rectangular in shape with same height but not necessarily same width. Also, short nets represent the majority of the nets in this design style [12]. Based on our experience, we found that connecting cells to cell attractors corresponding to the distance subsequence $\mathbf{s}_1$ (closest zone of sparse regions) yields the least net stretching and quite good cell spreading. Accordingly, the results reported in this paper are all based on cell spreading obtained through establishing connections between each cell and the closest dummy fixed cell in the closest zone of cell attractors, see Figure (2).

5. THE ATTRACTOR-REPELLER MODEL

We are now in a position to give the full Attractor-Repeller model for the global placement; i.e:

$$\begin{aligned} Min \quad & f(\mathbf{z}) + g(x) + h(y) \\ s.t \quad & \\ & l_x \leq x_i \leq u_x \\ & l_y \leq y_i \leq u_y \end{aligned} \tag{6}$$

Parameters l_x, l_y, u_x and u_y are lower and upper bounds on x and y. The first term, $f(\mathbf{z})$, represents the repelling terms or shortly the *repellers* and is given by equation (4). The second and the third terms, $g(x)$ and $h(y)$, represent the attracting terms or simply the *attractors*. We repeat again, for each movable cell, a connection is established with the closest cell attractors in the x and y directions. It follows that, the geometric location of the cell attractor to which a movable cell is hooked with is given by the x and y coordinates of the closest cell attractor in the x and y directions respectively, see Figure (2). In other words, Each movable cell i is assigned to an attractor with a coordinate (x_a^μ, y_a^ν) where μ and ν are the closest attractors to cell i in the x and y directions respectively. Accordingly, $g(x)$ and $h(y)$ are given as

$$g(x) = \sum_{1 \leq i \leq N} min\{(x_i - x_a^1)^2, \cdots, (x_i - x_a^q)^2\}$$

$$h(y) = \sum_{1 \leq i \leq N} min\{(y_i - y_a^1)^2, \cdots, (y_i - y_a^q)^2\}$$

Before we conclude this section, we would like to point out that throughout the remaining parts of the paper, we will refer to formulation (6) as the Attractor-Repeller (AR) formulation (or model), and to the new placement method (based on the AR model) as the **A**ttractor-**R**epeller **P**lacer (**ARP**).

6. THE ATTRACTOR-REPELLER PLACER: BASIC ALGORITHM

Figure (3) illustrates the flow of the new placement algorithm ARP. Following the parsing of the circuit information, the quadratic formulation of wire-length is solved to obtain an initial placement. Obviously, the majority of cells are on top of each other with only a small portion shifted towards the boundaries of the placement area as a result of the attracting forces due to the I/O pads. In the subsequent iterations, the algorithm proceeds iteratively via minimizing the AR model. In each iteration, following the termination of the global optimization, the AR model is updated as a result of the new connections between the movable cells and the cell attractors. Specifically, based on the resulting cell positioning, dense and sparse regions are identified according to the window-based technique presented previously. Next, the global placement is legalized through snapping the cells to the rows to ensure the starting solution in the next iteration is outside the flat region of $f(\mathbf{z})$. The next step involves improving the global placement slightly through a sequence of cell swapping and cell displacement within and among the different rows. The idea behind perturbing the global placement for better positioning of the cells is to increase the likelihood of connecting cells to cell attractors in sparse regions that enclose, or at least, located near their final ideal positions. In other words, by improving the current cell positioning, we hope to displace highly connected cells to the same zone on the placement floor so that in the subsequent iteration they end up hooked to the same cell attractors. In each iteration, new attractors are created and attractors

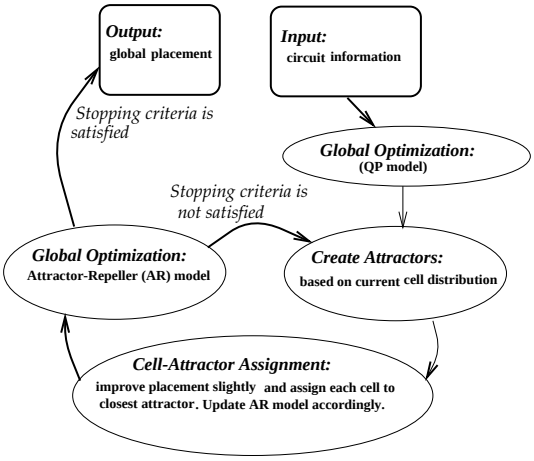

Figure 3: An outline of the new global placement method **ARP**.

from the previous iteration are deleted. As depicted in Figure (3), the sequence of creating attractors, establishing new connections with cell attractors and, solving the updated AR model continues until the termination criteria is met. Specifically, the algorithm stops when the number of iterations exceeds an upper bound $\mathcal{K}$, or if the *ratio* of the total area of the identified sparse regions to that of the placement area is $< \kappa\%$. Experimentally, we found that $5 \leq \mathcal{K} \leq 10$ and $\kappa = 10\%$ are quite sufficient to yield uniform cell spreading and accordingly adequate utilization of the placement area.

The solution methodology used is a *quasi-Newton* nonlinear algorithm that calculates a step direction based on the gradient and an approximation to the Hessian [5]. It then performs a line search along that direction to minimize the objective function, generating a new iterate. The optimization stops if the difference in the objective function values over two successive iterations is sufficiently small. The Hessian approximation is generated using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm.

This method has the advantage of having super-linear convergence rate.

7. QUALITATIVE ANALYSIS

In this section, we present a qualitative analysis. In particular, we consider the effect of the repellers and attractors on the spreading of the cells, and the effect of the attractors on the convergence of the global optimization.

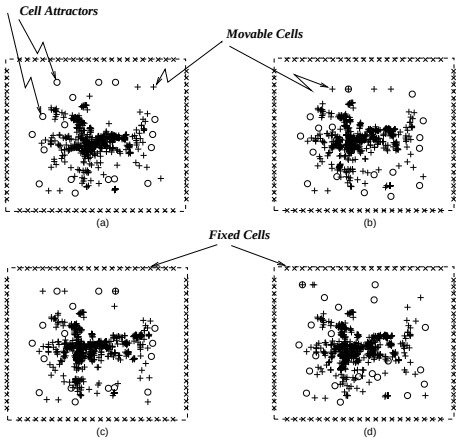

Figure 4: Benchmark *Primary1*: cell spreading after each pass using no repellers, $\rho(z_{ij}) = 0$ and $d = 1$ in $f(\mathbf{z})$ ("+" represent locations of movable cells, "x" represent locations of fixed cells (I/O pads), and "o" represent locations of attractors).

7.1. Attractors-Repellers and Cell Spreading

Cell attractors and repellers complement each other in the sense that cell repellers prevent connected cells from folding on top of each other and cell attractors prevent excessive displacement of connected cells by the cell repellers.

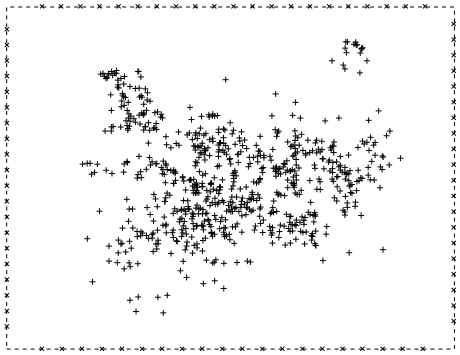

Figure 5: Cell spreading using strict repeller model. "+" represents movable cells and "x" represents fixed cells.

To demonstrate this, and to draw a general conclusion on whether a combination of attractors and repellers is better or not, we conducted different scenarios in which

- repellers are in-activated; i.e, $\rho(z_{ij}) = 0$ and $d = 1$ in equation (4). That is, the AR model is reduced to a combination of quadratic estimate of wire-length and cell attractors.
- no attractors (i.e, $g(x) = 0$ and $h(y) = 0$ in equation (6) are used and the model reduces to a strict repelling engine.
- the AR model (the combination of attractors and repellers) is used.

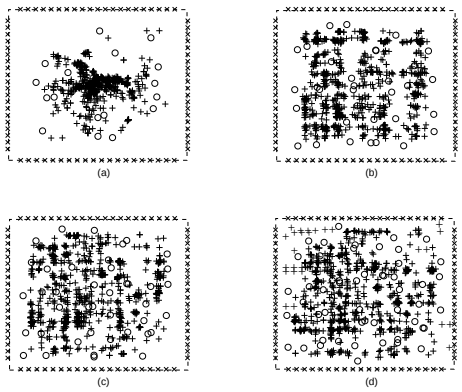

Figure 6: Benchmark *Primary1*: cell spreading after each iteration ("+" represents locations of movable cells, "x" represents locations of I/O pads on the chip periphery, and "o" represents locations of attractors).

The industrial MCNC circuits listed in Table (1) are used in each scenario. To discern between the different scenarios, Figures demonstrating spreading of cells and attractor distribution for benchmark *Primary1* are presented.

Figure (4) illustrates cell spreading in the *first scenario*. Part (a) shows the initial solution obtained via minimizing a quadratic wire-length objective function. Parts (b)-(d) illustrates the spreading after including the cell attractors. Clearly, no substantial improvement in cell spreading with respect to the first iteration is noteworthy. This is owing to the fact that in the absence of the repelling forces, the attraction forces between the connected cells outweigh the forces exerted by the cell attractors. In other words, the attraction forces between the connected cells overwhelmingly dominate the resultant forces acting on the cells.

On the contrary, Figure (5) demonstrates cell spreading in the *second scenario* in which no attractors are included and the model is strictly repelling. Clearly, the amount of cell spreading is remarkable, but the resultant wire-length is found to be higher and extra efforts by the final placement improver are necessary to offset this undesirable effect. As we suggested previously, this is owing to the fact that, a strictly repelling engine tends to stretch short nets and accordingly deteriorate total wire-length.

Figure (6) illustrates cell spreading in the *third scenario* (AR model). Part (a) shows the initial solution obtained from minimizing the quadratic wire-length. Clearly, the majority of the cells are clustered in the center of the placement region and the amount of overlap between the cells is substantial. The cell attractors (shown as circles in the Figure) are created according to the current cell positions. In parts (b)-(c), the global optimization involves minimizing the AR model. Better spreading of the cells can be noticed in each iteration compared to the preceding iterations. In the second iteration, clusters of overlapped cells tend to move to same sparse regions and in subsequent iterations, some of these clusters

tend to flatten out filling existing empty space in their immediate neighborhood on the placement floor. In this scenario, there is a trend of improvement in both cell spreading and wire-length reduction in each subsequent iteration. Figure (7) illustrates the *variability* or *relative spread* of the average wire-length as the global optimization is carried out for a finite number of iterations (specifically, the algorithm is executed for each benchmark and the average of wire-length across all benchmarks is computed). Clearly,

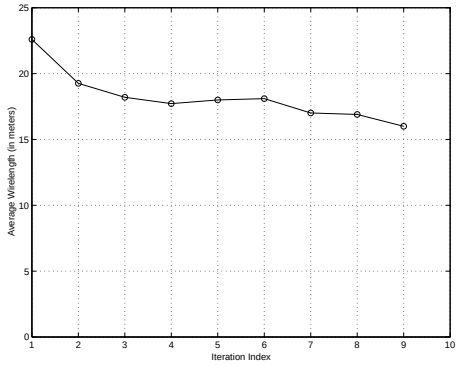

Figure 7: Variability of average wire-length over the different iterations of the algorithm.

the wire-length decreases as the algorithm proceeds from one iteration to another. In fact, for some benchmarks, we observed that if the algorithm is executed for a fairly large number of iterations, the wire-length obtained is quite close to the final answers obtained after improving (using the final placer) the initial placement generated from a relatively smaller number of iterations.

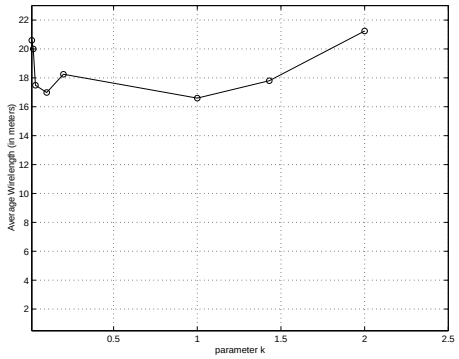

Figure 8: Variability of average wire-length versus the scaling parameter k, (equation (7)).

Figure (8), depicts the variability of the average wire-length as a function of the intensity of the repelling forces (that is, parameter d in equation (4)). Based on experimentation, we found that there is correlation between the intensity of the repelling forces required to spread the cells apart (prohibit overlap) and average cell width of a particular circuit. Accordingly, the following empirical formula for estimating d is used:

$$d = \frac{\sqrt{w_a}}{k} \tag{7}$$

where w_a is the average cell width (which is constant for a given circuit) and k is scaling factor. In fact formula (7) was found quite useful. To examine the correlation between parameter d (which reflect

the strength of the repelling terms) and the wire-length, parameter k is varied between 0.01 to 2.5 for each test case. Experimentally we found that, on average, $k = 1$ (or equivalently $d = \sqrt{w_a}$) yields best answers in terms of wire-length. This is demonstrated in Figure (8). No correlation between d and the speed of convergence of the global optimization is observed.

7.2. Cell Attractors and Convergence of the Global Optimization

As we have shown in [11], the attractor-repeller objective function is convex as long as the whole circuit forms one set; that is the netlist graph is connected and there exist, at least, one fixed cells. The convexity and smoothness of the objective function improves as the number of fixed cells increases.

As indicated previously (Figure (3)), the AR model is used after the first iteration (an initial solution is generated in the first iteration via minimizing a quadratic model; i.e, $\rho(z_{ij}) = 0$ and $d = 1$ and no attractors exist). Figure (9-a) shows the variability (*relative spreading*) of the algorithm run time as a function of the number of iterations. The relationship between the algorithm run time and the number of attractors is illustrated in Figure (9-b) (precisely, the algorithm is executed for many iterations and the *average run* time as well as the *average number of attractors* are computed and plotted in Figure (9)).

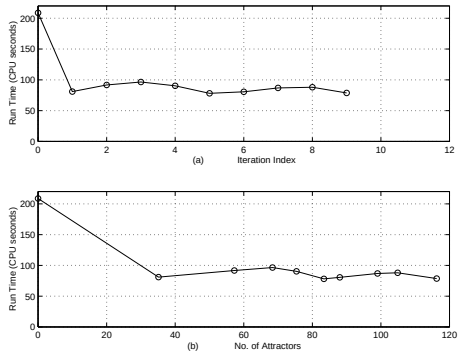

Figure 9: Variability of run time. (a) versus passes (b) versus number of attractors.

Examining these Figures, it is clear that there is a substantial decrease in the run time after the first iteration. That suggests a strong correlation between the convergence of the global optimization and the existence of the cell attractors. On average, the percentage of decrease in run time in the second iteration (that is, when the cell attractors are added) is almost 62%. That is, on average, the run time improves by an average of 62% compared to the first iteration. After the second iteration, the rate of decrease in run time is relatively lower, but its general trend is downward.

Based on these observations, it is evident that there is a strong tendency of the optimization process to converge in a much shorter time when the attractors exist. Again, this indicates that the smoothness and curvature (convexity) of the objective function improve as the number of fixed cells in the circuit increases.

8. NUMERICAL RESULTS

The new method is implemented in C language on a Sun-Ultra1/140 workstation. Half-perimeter wire-length (HPWL) is used in estimating the wire-lengths. The HPWL has been used in other results presented in the literature ([20, 21, 22, 14, 15]), since rectilinear wiring is typically used in routing a circuit.

As a final placer, we used a Tabu-search based local improvement algorithm that had been developed in collaboration with other members in our group [3].

To assess the new placement method ARP, we consider comparing solution quality and efficiency (computation times) of ARP to the state-of-art placers. Specifically, Simulated Annealing (SA), and a combination of analytic and iterative improvement based placers. Namely, TimberWolf v6.0 (TW v6.0) [20], TimberWolf v7.0 (TW v7.0) [22, 21] and Gordian/Domino [14, 15].

TW v7.0 [22, 21] is the latest release of TimberWolf placement package. It has two modes of operation, namely, a *flat* mode and *hierarchical* mode. In the flat mode, the original netlist is placed. In the hierarchical mode, however, the original netlist is clustered into various levels of netlists. At each level, SA tries to find an optimal placement for the clustered netlist, followed by flattening the netlist. Subsequently, the flattened netlist is admitted to the next level and the process is repeated until the original netlist is placed. The clustering approach has the advantage of greatly reducing the complexity of the circuit and consequently, reducing the computational efforts required to solve the problem.

The method of Gordian [14] has been described in the introduction. The method of Domino is an iterative improvement technique that has been applied to initial placements generated by Gordian [15]. In Domino, the placement problem is modeled as a *transportation problem* and *network flow* algorithms are employed to solve the resulting optimization problem. Specifically, cells positioned near each other in the existing placement are considered for improvement. The iterative process produces a sequence of intermediate placements. In each iterative step, an improved placement is generated from the current placement. The process terminates when after several generations no significant improvement is obtained.

The metrics used in the comparison of the new method ARP and the other different approaches are *total wire-length* and *longest-row width* which reflect solution quality, and *computation times* which typify the efficiency of the method. Longest row width determines chip width and accordingly total chip area. Thus, longest row width is an important parameter in the assessment of a given placement. Numerical results of the other approaches are taken from the literature [21]. For some benchmarks, no results have been reported in [21] and their entries in the results tables are left empty.

Tables (2), (3) and (4) list the final wire-length, longest row width and computation times for ARP and the other approaches. The reported computation times include times for final placements. ARP outperforms TW v6.0, TW v7.0 and Gordian/Domino on the majority of the benchmarks. For benchmark *Ind3*, ARP outperforms TW v6.0 but lacks compared to TW v7.0 and Gordian/Domino, and for benchmark *Bio* and *Prim2*, ARP lacks insignificantly compared to TW v7.0 in flat mode (FM). We suspect a different positioning of the I/O pads compared to the other methods, or a scaling problem is the reason for this difference in results.

On average, ARP achieves 7.78%, 1.02%, 3.96% and 8.68% reduction in wire-length compared to TW v6.0, TW v7.0-FM, TW v7.0-H (Hierarchical) and Gordian/Domino respectively.

Tables (3) and (6) show comparisons of longest row width obtained by ARP and the other approaches. On average, ARP outperforms TimberWolf v6.0 and TimberWolf v7.0 by 4.28% and 0.13%. No longest row width was reported for Gordian in [21] to compare to.

As for computation times, the results are illustrated in Tables (4) and (7) (computation times of the other approaches are scaled). Evidently, ARP consumes less computation times to place each test case compared to the other approaches. On average, ARP outperforms TimberWolf v6.0 by 83%, TimberWolf v7.0 (flat mode) by 87%, TimberWolf v7.0 (Hierarchical mode) by 7.6% and Gordian/Domino by 13.8%.

Circuit	**Cells**	**Pads**	**Nets**	**Pins**	**Rows**
Fract	125	24	147	462	6
Prim1	752	81	904	5526	16
Struct	1888	64	1920	5471	21
Ind1	2271	580	2478	8513	15
Prim2	2907	107	3029	18407	28
Bio	6417	97	5742	26947	46
Ind3	15059	374	21940	176584	54
Avq.small	21854	64	22124	82601	80
Avq.large	25114	64	25384	82751	86

Table 1: MCNC Benchmarks used as test cases

Ckt	TW6	TW7.FM	TW7.H	Gord/Dom	ARP
Fract	-	-	-	-	0.034
Prim1	1.0	0.93	0.99	1.08	0.79
Prim2	3.71	3.53	3.72	4.02	3.61
Struct	-	-	-	-	0.34
Ind1	-	-	-	-	1.50
Bio	1.97	1.8	1.88	1.98	1.83
Ind3	48.38	43.08	44.67	44.94	48.12
Avq.s	6.72	6.45	6.13	6.42	6.06
Avq.l	6.93	6.50	6.81	7.16	6.54

Table 2: Wire Length Comparison, TW v7.0 flat and hierarchical modes, Gordain/Domino and ARP

Ckt	TW6	TW7.0	ARP
Fract	-	-	704
Prim1	5260	5100	5170
Prim2	8380	8210	8201
Struct	-	-	2360
Ind1	-	-	4810
Bio	5114	4936	4928
Ind3	28832	26368	26176
Avq.s	9560	9128	9080
Avq.l	9744	9400	9344

Table 3: Chip width comparison: TimberWolf v7.0 and ARP. Width is measured in *microns*.

Ckt	TW6	TW7.FM	TW7.H	Gor/Dom	ARP
Fract	-	-	-		12
Prim1	467	488	130	168	95
Prim2	3127	4307	736	542	504
Struct	-	-	-		116
Ind1	-	-	-	-	376
Bio	8606	12224	1273	1553	1290
Ind3	38619	70873	5156	6087	4253
Avq.s	54681	78248	7657	10261	8534
Avq.l	56802	97612	9175	12403	11202

Table 4: Run-time comparison in CPU seconds: TimberWolf v7.0 in flat and hierarchical modes, Gordian/Domino and ARP.

Ckt	TW v6 %impr.	TW7.FM %impr.	TW7.H %impr.	Gor/Dom %impr.
Prim1	+21	+15	+20.2	+26.8
Prim2	+2.7	-2.2	+2.9	10.2
Bio	+7.1	-1.64	+2.9	+7.5
Ind3	+0.53	-10.4	-7.2	-6.6
Avq.s	+9.8	+6.0	+1.1	+5.6
Avq.l	+5.6	-0.6	+3.9	+8.6
avg.	+7.78	+1.02	+3.96	+8.68

Table 5: Relative wirelength improvement with respect to other approaches (+ means ARP is better).

Ckt	TW v6.0 %impr.	TW v7.0 %impr.
Prim1	+1.7	-1.3
Prim2	+2.1	+0.11
Bio	+3.6	+0.16
Ind3	+9.2	+0.73
Avq.s	+5.0	+0.53
Avq.l	+4.1	+0.60
avg.	+4.28	+0.13

Table 6: Relative chip-width improvement with respect to other approaches (+ means ARP is better).

Ckt	TW v6 %impr.	TW7.FM %impr.	TW7.H %impr.	Gor/Dom %impr.
Prim1	+79	+80	+27	+3.8
Prim2	+83	+88	+31	+7.0
Bio	+85	+89	-1.3	+17
Ind3	+88	+93	+17	+30
Avq.s	+84	+89	-10	+16
Avq.l	+80	+88	-18	+9
avg.	+83	+87	+7.6	+13.8

Table 7: Relative CPU time improvement with respect to other approaches (+ means ARP is better).

9. CONCLUSION

Future placement tools face new challenges as placement tasks are much more complicated due to the immense complexity of the new generations of integrated circuits. In light of this fact, we presented new formulations for estimating the wire-length in global placement. We proposed a new procedure to force cells to spread out within the chip area without causing any excessive stretching of the nets. Moreover, We proposed a new generic placement algorithm based on the new formulation. A study of the different parameters of the algorithm is presented including empirical formulae for these parameters.

The feasibility of the approach is demonstrated using a set of MCNC standard cell benchmarks. We that the results are comparable to the other popular placers (despite the fact that our implementation is not optimized and was chiefly intended to the show the feasibility of the approach).

Future work should focus on accounting for timing information and other constraints like power dissipation. Also, as we indicated in the introduction, the new approach can be applied to circuits with no fixed cells, FPGA placement for example. In fact, applying the new approach to FPGA placement is our next target.

10. REFERENCES

[1] B. W. Kernighan A. Dunlop. A procedure for placement of standard-cell VLSI placement. *IEEE Trans. on CAD*, 4, no. 4:92–98, 1985.

[2] K. Chaudhary A. Srinivasan and E. Kuh. RITUAL: A Performane-Driven Placement Algorithm. *IEEE Trans. on Circuits and Systems, vol. 39, No. 11*, pages 825–839, November 1992.

[3] S. M. Areibi. *Towards Optimal Circuit Layout Using Advanced Search Techniques*. PhD thesis, University of Waterloo, Ont. Canada, 1995.

[4] I. Markov C. J. Alpert T. F. Chan A. Kahng and P. Mulet. Faster Minimization of Linear Wirelength for Global Placement. *IEEE Trans. on CAD of Integ. Circ. and Syst., 17(1)*, January 1998.

[5] P. Lu C. Zhu, R. H. Byrd and J. Nocedal. L-BFGS-B: Fortran subroutines for large-scale bound

constrained optimization. Technical report, Dept. of Elec. Eng. and Comp. Scien., NorthWestern University, 1996.

[6] C. K. Cheng and E. S. Kuh. Module placement based on resistive network optimization. *IEEE Trans. on Comp. Aided Design, 3 (3)*, pages 218–225, 1984.

[7] C. C. N. Chu and D. F. Wong. A Matrix Synthesis Approach to Thermal Placement. *IEEE Trans. on CAD, vol. 17, No. 11*, pages 1166–1174, November 1998.

[8] G. Sigl K. Doll and F. Johannes. Analytical placement: A linear or quadratic objective function. In *In proc. 23rd DAC*, pages 57–62, 1991.

[9] Y. Du and A. Vannelli. A Nonlinear Programming and Local Improvement Method for Standard Cell Placement. In *Proc. of IEEE Custom Integrated Circuit Conf.*, 1998.

[10] H. Eisenmann and F. M. Johannes. Generic global placement and floorplanning. In *Proc. of the 35th Design Automation Conference*, 1998.

[11] H. Etawil and A. Vannelli. Optimization in VLSI Design: Target Distance Models for Cell Placement. *To appear in Handbook of Applied Optimization. Oxford University Press, 198 Madison Av. NY 10016-4314, USA*, 2000.

[12] T. Hamada, C-K Cheng, and P. M. Chau. A wire length estimation technique utilizing neighborhood density equations. *IEEE Trans. on Computers, 15(8)*, August 1996.

[13] D. J. H. Huang and A. B. Kahng. Partitioning-based standard-cell global placement with an exact objective. In *Inter. Symp. on Physical Design (ISPD)*, pages 18–25, 1997.

[14] F. Johannes J. M. Kleinhans, G. Sigl and K. Antreich. GORDIAN: VLSI placement by quadratic programming and slicing optimization. *IEEE. Trans. on CAD, vol. 10, no. 3*, pages 356–365, 1991.

[15] K. Doll F. Johannes and K. Antreich. Iterative placement improvement by network flow methods. *IEEE. Trans. on CAD, vol. 13, no. 10*, pages 1189–1200, 1994.

[16] U. Lauther. A min-cut placement algorithm for general cell assemblies based on a graph representation. In *proc. of the 16th Design Automation Conference*, pages 1–10, 1979.

[17] www.cbl.ncsu.edu/benchmarks/layoutsynth92/.

[18] D. S. Mitrinovic'. *Analytic Inequalities.* Springer-Verlag, Berlin. Heidelberg, 1970.

[19] N. R. Quinn. The placement problem as viewed from the physics of classical mechanics. In *Proc. of the 12th Design Automation Conference*, pages 173–187, 1975.

[20] Carl Sechen. *VLSI Placement and Global Routing Using Simulated Annealing*. Kluwer Academic Publishers, 1988.

[21] Wern-Jieh Sun and Carl Sechen. Efficient and effective placement for very large circuits. In *in IEEE/ACM ICCAD*, pages 170–177, 1993.

[22] Wern-Jieh Sun and Carl Sechen. Efficient and effective placement for very large circuits. *IEEE Trans. on CAD , vol. 14, No. 3*, pages 349–359, March 1995.

[23] R. S. Tsay and E. Kuh. A Unified Approach to Partitioning and Placement. *IEEE Trans. on Circuits and Syst., 45*, pages 521–533, May 1991.

[24] H. Vaishnav and M. Pedram. PCUBE: A Performance Driven Placement Algorithm for Low Power Designs. In *Proc. of the EURO-DAC*, pages 72–77, 1993.