DPlace: Anchor Cell based Quadratic Placement with Linear Objective

Tao Luo and David Z. Pan

Department of Electrical and Computer Engineering
The University of Texas at Austin
{tluo, dpan}@ece.utexas.edu

1 Introduction

Although circuit placement has been studied for decades, it continuously attracts research attentions. The placement problems grow rapidly in both problem size and complexity. Some industry placement problems contain multi-million gates and excessive number of blockages [1] [2]. In this chapter, we introduce DPlace, an anchor cell and diffusion spreading based quadratic placement engine that can handle large scale placement problem.

Historically, existing circuit placement algorithms can be roughly classified into three major categories, i.e. simulated annealing [3], iterative partitioning based approach [4, 5, 6], and analytical placement approach [7, 8, 9, 10, 11, 12, 13, 14].

Among existing placement works, analytical placement has been successful in recent years and achieved impressive results on wire length, scalability and the speed of convergence. A typical analytical placement formulates the wire length optimization into a mathematical problem, and minimizes a smooth, continuous, and derivable wire length formulation. According to the reported results of ISPD 2005 and 2006 placement contest [15] [2], most of the top ranked placers are analytical placers.

In placement, the Half Parameter Wire Length (HPWL) is a common estimation of the routed wire length. Since HPWL model is not smooth and derivable, quadratic placement optimizes the quadratic form of HPWL [7, 8, 9, 10, 11, 12, 14], and nonlinear model placement [16] [13] [17] adopts a nonlinear estimation of HPWL model, such as the log-sum-exponential wire length approximation patented by Naylor et al. [18].

Three placers in the ISPD 2006 placement contest using the log-sum-exponential wire model have achieved impressive wire length results. It is agreed that placement uses log-sum-exponential wire model approximates the HPWL much closer than the quadratic estimation. However, although still controversial, some researchers believe that the quadratic placement potentially has advantages for timing driven placement, as the quadratic approximation of the HPWL gives larger penalty on longer wires.

Most of analytical placements are force directed placement. The initial placement solution generated in force directed placement has excessive overlap among cells. To push cells away from congestion, in subsequent iterations, force directed placer adds "spreading force" or density constraints into the original wire length formulation. In force directed quadratic placement, the density constraints are combined into the optimization objective either by adding the spreading forces as constant force terms or by adding fixed points to implement the spreading forces.

We present a new quadratic placement, DPlace, that does not explicitly add "force" or apply density constraint into the original wire length optimization framework [19]. Different from traditional force directed quadratic placement, we divide the wire length minimization and density control tasks in two steps. A concept of anchor cell is presented to split the overlap reduction and wire length optimization objectives into two problems. DPlace is based on, but not limited to, quadratic placement, and the new framework is applicable for other analytical placements. In brief, during every iteration in DPlace, we have two steps:

1. A pre-placement step to spread cells for better density distribution. The wirelength minimization are not explicitly considered in the pre-placement step.
2. An unconstrained wire length minimization step to repair the wirelength. In this step, anchor cells are inserted as the reference of the pre-placement result and as the basis of the new wire-length optimization formulation.

In traditional force directed placement, spreading forces are used to estimate where to push cells. There is no explicit control of the cell movements and a cell may be pushed to any placement region. However, explicit cell movement control is important to cope with some challenging placement tasks, such as the ECO placement and timing-driven placement. It is possible to control the cell movement in DPlace by specifying the cell movement explicitly in the pre-placement stage. The following are a few characteristics of our approach, which differentiates DPlace from other analytical placement works.

- We propose a global placement framework that uses *anchor cells* to split a traditional placement/spreading iteration into two steps, the pre-placement step to reduce the cell overlaping, and the unconstrained wire length minimization step to reduce the wire length.
- In order to reduce the gap between the quadratic wire length vs. linear wire length objective, we introduce a net weight linearization strategy that transforms the star model based quadratic objective into HPWL objective exactly.
- The framework we propose in DPlace can be used for both the global placement and ECO placement. The pre-placement step can be extended to control the cell movements explicitly, e.g , we can specify a certain group of cells be moved to a certain position of the chip. This capability has the advantage for ECO placement where the placement stability is crucial.
- Our quadratic formulation is efficient for large scale placement. The Hessian matrix in our quadratic formulation has much lower dimension as well as extremely low density. The runtime to solve one iteration of the system of linear equations is improved by 24 times in our formulation.

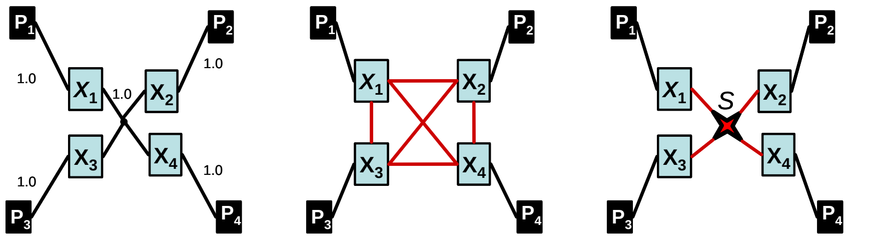
In the following, we introduce the preliminaries and current status of the force directed quadratic placement in section 2. The details of our global placement are described in section 3. The legalization and detailed placement are presented in section 4. We give the overall algorithm of DPlace in section 5 and show the experimental results in section 6, which followed by section 7.

2 Preliminaries and the motivation

To motivate our proposed approach, the following section gives an overview of the force-directed quadratic placement and the analysis of the essential concept in some of the existing force directed quadratic placement approaches.

2.1 Quadratic placement

In circuit placement, a netlist is normally modeled as a hyper-graph with each node representing an object/cell and each edge representing a net. Let x_i and y_i denote the coordinates of each cell, HPWL is used as an estimation of the routed wire length. Because the equation of HPWL is difficult to optimize mathematically, quadratic placement minimizes the square of the length and width of the bounding box of a net, commonly referred as the quadratic wire length.



(a) A circuit with 5 nets, 4 cells, and 4 pins, one of the nets is a 4-pin net

(b) Clique model transformation

(c) Star model transformation

Fig. 1. Transformations of the multi-pin net into multiple two-in nets. Only the x coordinates are showed in these figures

As multi-pin nets can not be processed in quadratic placement, each multi-pin net is transformed into multiple two pin connections with proper weights. Traditionally, clique model is used for multi-pin net transformation and one k-pin net will be transformed into C_k^2 connections in clique model. For the 4-pin nets in Figure 1(a), the clique model transformation is shown in Figure 1(b). The disadvantage of clique model is that it may increase the number of non-zero entries in the connectivity matrix significantly, as the example in Figure 6, which slows down the quadratic solver.

Another type of transformation is the star model [20] [11]. One *k*-pin net will be transformed into *k* connections in star model, as shown in Figure 1(c). The combination of the clique and star transformation is also referred as the hybid model [11].

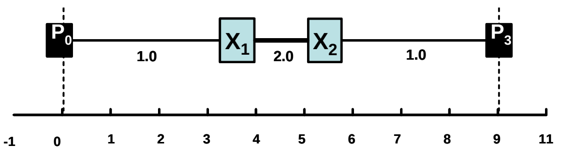
For a two pin net $e_{i,j}$ that connects cell i and j, the quadratic wire length is defined as $w_{i,j}((x_i - x_j)^2 + (y_i - y_j)^2)$, where $w_{i,j}$ denotes the weight of net $e_{i,j}$. The quadratic placement minimizes the sum of all quadratic wire length in the circuit. The optimization problems in x and y direction are separable and can be treated independently. Therefore, the cost function in x direction is given by

$$\Phi(x) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x} + const \tag{1}$$

Assuming there are n movable objects in the netlist. Let $\mathbf{A}$ denote the Hessian matrix of the quadratic system, which is essentially the n x n connectivity matrix of the netlist. $\mathbf{A}$ is symmetric and positive definite. $\mathbf{x}$ denotes the vector of x coordinates of all cells. $\mathbf{b}$ is the vector encoding all connectivity information between movable and fixed objects, and the pin offsets are captured in $\mathbf{b}$ as well. The minimizer of the cost function (1) can be obtained by taking the gradient of the cost function to zero, $\partial(\Phi(x))/\partial x = \mathbf{0}$, which is determined by the following system of linear equations

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{2}$$

Figure 2 shows a simple circuit with 2 movable cells and two fixed pins. The number associated with each net is the net weight. Cell 1 and 2 are in the force equilibrium status in Figure 2, i.e. the sum of the weighted wire length is the minimum.



$$\begin{bmatrix} 3 & -2 \\ -2 & 3 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} p_0 \\ p_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 9 \end{bmatrix}$$

$$x_1 = 3.6, \qquad x_2 = 5.4$$

Fig. 2. The quadratic placement formulation of a simple circuit in the x direction. p_0 and p_3 are the x coordinate of the fixed pins

2.2 Force-directed quadratic placement

Solving the unconstrained minimization problem in equation (1) results a placement with significant overlap among cells. A placer needs to push cells around to remove overlap. Some placers recursively partition the placement region to spread cells, such as Gordian [7]. The force-directed placers add spreading forces into the system in each solving process and reduce the overlap iteratively. Figure 3 shows that cell 1 and 2 are too close to each other, a force directed placer adds forces to push cells away from the center.

To apply spreading forces into the optimization framework, there are mainly two types of strategy to implement the force, the *constant force addition* and the *fixed point addition* approach. In each placement iteration, Kraftwork [8] and FDP [12] add a constant force vector **f** to the right hand side of equation (2). The fixed point based approach adds artificial pins and nets to move cells. mFar [9] uses multiple fixed virtual pins for each cell in every iteration, one is used to maintain a cell's force equilibrium state, and others are applied to perturb the cell. FastPlace [11] uses one fixed virtual pin for both purposes.

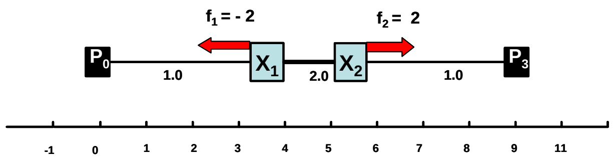


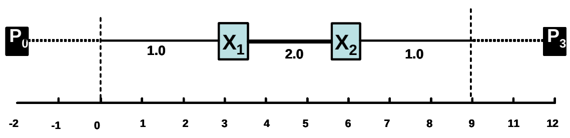
Fig. 3. Force directed placement: adding forces to push cells out of the region with congestion

Constant forces

In every iteration, the force for each cell is computed to reduce the overlap. In constant force based approach, the force vector **f** is added to vector **b** in equation 2. The solution of the modified quadratic system generates a placement with less overlap among cells. In the *i*th iteration, the force vectors used in 1 to $i-1$th iterations are accumulated to prevent cells collapsing back. The modified equation with constant forces is given by

$$\mathbf{Ax} = \mathbf{b} + \sum_{k=1}^{i-1} \mathbf{f}^k + \mathbf{f}^i \tag{3}$$

In constant force based approach, the Hessian (connectivity matrix) is not changed in each iteration unless the net re-weighting is involved. In such case, the Hessian **A** only needs to be pre-conditioned once in the beginning, which will save runtime as the matrix pre-conditioning is runtime expensive.



$$\begin{bmatrix} 3 & -2 \\ -2 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} p_0 + f_1 \\ p_3 + f_2 \end{bmatrix} = \begin{bmatrix} -2 \\ 11 \end{bmatrix}$$

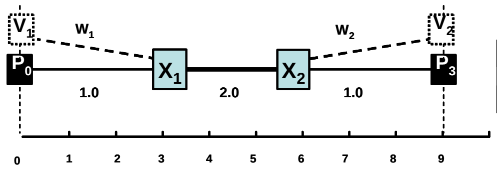
$$x_1 = 3.2, \qquad x_2 = 5.8$$

Fig. 4. Adding the constant force on a cell is equivalent to shifting its connected objects

The physical meaning of adding a spreading force to one cell is equivalent to shifting its connected pins and cells. To add the spreading force in Figure 3, a force vector is added into constant vector **b** in equation 2. To add a force vector is equivalent to shifting the connected objects of each cell, as shown in Figure 4. Pins are shifted outside of the chip, and cells may “jump” out the chip region if the magnitude and direction of the spreading forces are not properly adjusted. This tends to happen in the earlier placement iterations, where spreading forces are large and the force directions are not evenly distributed. Although such a scenario is not obvious in ISPD 2005 and 2006 benchmarks, where the initial density distributions are more even due to a large amount of fixed macros, the force scaling is tricky for placement with no fixed macros, such as the ISPD02 bechmarks [21].

Because the connectivity matrix is not strictly diagonal dominant, and often ill-conditioned, the solver of the linear system may have stability problem [12], i.e. cells may jump around when large forces are added. FDP adds a small weight to a portion of the diagonal terms of the Hessian and the new Kraftwerk [22] adds weight to all diagonal terms. Such a strategy is equivalent to adding a virtual fixed pin and net to a cell, as shown in Figure 5, which affects the quadratic objective and improves the stability of the quadratic solver.

Fixed point forces



$$\begin{bmatrix} 3+w_1 & -2 \\ -2 & 3+w_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} p_0 + w_1 v_1 \\ p_3 + w_2 v_2 \end{bmatrix}$$

Fig. 5. An example of the fixed point addition formulation, p_0, p_3, v_1, and v_2 are the x coordinate of the fixed real and virtual pins, respectively

In fixed-point methods, the fixed points and nets are added to the original system of linear equations to perturb the placement. In fixed-point based methods, adding a virtual fixed-point connection to a cell will add a diagonal term in the corresponding entry of the cell in the Hessian matrix **A** and the term in the constant vector **b**. In Figure 5, to add force to each cell, a virtual pin and connection are added to each cell with proper weight, and we can see the change in the Hessian and the constant vector in the figure. Therefore, adding a cell will make the corresponding row and column strictly diagonal dominant in Hessian **A**, and improve the condition number of the matrix. As a result, the fixed-point addition based method tends to be more stable.

The fixed point addition method guarantees cells moving inside the convex hull defined by the fixed points. If using a large weight for the virtual nets, cells have less mobility and tend to move steadily toward force directions. However, the added large virtual net weights may dominate the actual net connections and affect the optimization objective. On the contrary, if using very small virtual net weights, fixed-points will be off chip and cells may start to jump out of the boundary. In other words, the fixed point placement starts to behave similar as the constant force addition based method. Furthermore, in fixed point based approach, the connectivity weights will be updated in every iteration and the matrix needs to pre-conditioned in every solving iteration.

2.3 The proposed approach

DPlace does not fall into the above categories. In each iteration in DPlace, the *cell anchoring* divides the constrained wire-length minimization problem into two steps, the overlap reduction pre-placement step and the unconstrained wire length minimization step. There is no “force” added to the quadratic system in DPlace, and there is no need to control the magnitude of forces, which is a non-trivial part in conventional force directed placements. As a result, no solver stability issue exists in DPlace.

Any smooth cell spreading techniques can be used for the density optimization pre-placement step. We use the diffusion cell spreading [23] for pre-placement step. Anchor cells are used to mark the pre-placement result. We use the nets connecting anchor cells and real cells to formulate an unconstrained wire length minimization problem.

If a netlist is changed, without explicit cell movement control, the placement solutions before and after the changes could be completely different. In our approach, the explicit cell movement control can be naturally applied in pre-placement step, which potentially provides flexibility for ECO placement. Furthermore, the fast growing of the problem sizes is a challenge to existing quadratic solvers. Our approach scales well to the problem size. The anchor cells used in quadratic framework significantly reduces the complexity of the problem.

3 Global Placement in DPlace

The global placement in DPlace is guided by a density driven pre-placement method. We use the diffusion based cell spreading technique [23] for the spreading smoothness.

3.1 Diffusion pre-placement

The global placement is guided by a diffusion based cell spreading technique. Diffusion is the flow of particles from a region of highly concentration to a region with lower concentration, until the concentration on both regions is equal. The cell spreading in placement shares similar philosophy as the natural diffusion process, where cells are driven from high density areas to low density areas. Diffusion in placement is driven by the density gradient, i.e. the steepness of the density difference. Mathematically, the diffusion process is characterized by the following differential equation.

$$\frac{\partial d_{x,y}(t)}{\partial t} = D\nabla^2 d_{x,y}(t) \tag{4}$$

In the context of placement, $d_{x,y}(t)$ is the cell density at position (x,y) at time t. D is the diffusivity constant, which determines the speed of the diffusion process. The discrete approximation method in [23] can be used to solve the diffusion equation.

In diffusion based pre-placement, the placement region is cut into equal size bins. The bin density is computed as the total cell area enclosed in the bin divided the bin area. The discrete solver we use to solve the diffusion equation evens out the densities between neighboring bins as time proceeds.

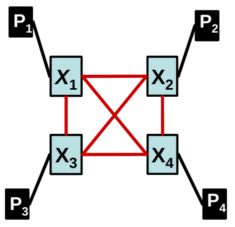
In every global placement iteration of DPlace, cells are pre-diffused from high density area to low density area. The diffusion based pre-placement takes k substeps, where k is relatively small in earlier placement iterations and becomes larger in the later iterations. The cells will not be moved until we placed and locked all anchor cells.

3.2 Anchor cells

Once a pre-placement result is generated, we need to “memorize” the pre-placement solution, in which cells have been spread out. Since the pre-placement solution is often poor on wire-length, we need to use the quadratic placement formulation to repair the wire length. To prevent cells collapsing back to the initial placement, we can fix a small percentage of cells in pre-placement, and let the quadratic solver rearrange other cells. Another way is that we use virtual cells to mark the pre-placement solution and replace some nets with virtual nets connecting the virtual cells and real cells. By updating the virtual connections into the wire length optimization objective and solving the unconstrained wire length minimization problem, cells will be “pulled” toward their anchors due to the wire tensions. In above scenarios, the fixed real or virtual cells are used as anchors to control the movement of real cells, and we name them “anchor cells”.

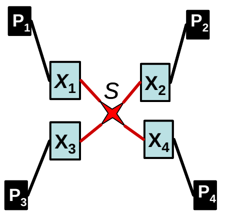
We do not need to use one anchor per cell, which may over-restrict the movements of real cells. Instead, we can use one anchor for several cells, which gives more freedom for cells to move during the wirelength optimization. We use star model to transform a portion of multi-pin nets into two-pin connections and use the star as the anchor of real cells. Compared with the method to use one anchor per cell, using stars as anchors will have much less impact to the original wire objective and imposes less constraint on cell movements.

In the hybrid model based wire length transformation, the multi-pin nets are converted into star and clique model. All stars will be added back into the Hessian matrix **A** as moveable objects, which may increase the dimension of the matrix significantly. In ISPD 2005 benchmark, by using star model with a pin threshold as 5 will increase the dimension of the matrix up to 40 percent. For example, the dimension of the Hessian **A** for circuit *bigblue*4 in ISPD 2005 benchmark grows from 2.2 million to 2.8 million. Under conventional formulation, solving one iteration of the system of linear equations with a dimension over 2 million will take several minutes.



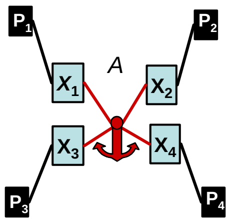
$$\Rightarrow \begin{bmatrix} 1.25 & -0.25 & -0.25 & -0.25 \\ -0.25 & 1.25 & -0.25 & -0.25 \\ -0.25 & -0.25 & 1.25 & -0.25 \\ -0.25 & -0.25 & -0.25 & 1.25 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}$$

Fig. 6. The quadratic placement formulation by using clique model. For simplicity, we assume the weight of each transformed two-pin net is weight 0.25.



$$\Rightarrow \begin{bmatrix} 1.25 & & & & -0.25 \\ & 1.25 & & & -0.25 \\ & & 1.25 & & -0.25 \\ & & & 1.25 & -0.25 \\ -0.25 & -0.25 & -0.25 & -0.25 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ s \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ 0 \end{bmatrix}$$

Fig. 7. The quadratic placement formulation by using star model. S is the x coordinate of the star, which is a moveable object in the placement. For simplicity, we assume the weight of each transformed two-pin net is weight 0.25. The dimension of the Hessian matrix **A** is equal to the number of cells plus the number of stars



$$\Rightarrow \begin{bmatrix} 1.25 & & & \\ & 1.25 & & \\ & & 1.25 & \\ & & & 1.25 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} p_1 + 0.25C \\ p_2 + 0.25C \\ p_3 + 0.25C \\ p_4 + 0.25C \end{bmatrix}$$

Fig. 8. The quadratic placement formulation after the anchor cell insertion. C is the x coordinate of the anchor cell, which is a constant. The new Hessian matrix **A** is extremely sparse compared with that by using the star or clique formulation.

Figure 6 shows the quadratic placement formulation of the circuit in Figure 1(a) by using the clique model. The dimension of the Hessian matrix is the same as the number of movable cells. Figure 7 is the formulation by using the star model. The dimension of the Hessian increases, but the matrix is more sparse compared with that by using the clique model. Unlike stars, anchor cells are fixed objects in our formulation. Therefore, anchor cells will not increase the dimension of the Hessian **A**. Furthermore, in the anchor cell based quadratic formulation in Figure 8, we see that the Hessian matrix is extremely sparse compared with that by using both the star and clique models.

We assign anchor cells to the nets with a pin degree above th (e.g. 3 as in our implementation) only. With a small th, more anchor cells will be inserted, which may over-restrict the movement of cells. However, if the pin threshold *th* is too large, it will be more difficult to spread cells.

Let $\mathbf{A}'$ denotes the Hessian matrix in our new formulation. Anchor cells are not movable objects, thus do not appear in $\mathbf{A}'$. Matrix $\mathbf{A}'$ has the dimension as the number of movable objects in the netlist. We insert anchors after the completion of pre-placement stage in each iteration. Once cells are pre-placed, anchor cells are inserted at the gravity centers of their connected cells and locked. In such a way, anchor cells mark the pre-placement result, and act as anchors to pull other cells around in the subsequent wire length minimization step. The new Hessian $\mathbf{A}'$ has a dimension much smaller than that in the conventional quadratic placement methods. Most importantly, the number of non-zero entries in each row of the new formulation $\mathbf{A}'$ is close to the number of pins on the cell, which is mostly around 2 to 4. A matrix with 2-4 non-zero entries is extremely sparse, and the linear system is trivial to solve using the anchor cell formulation.

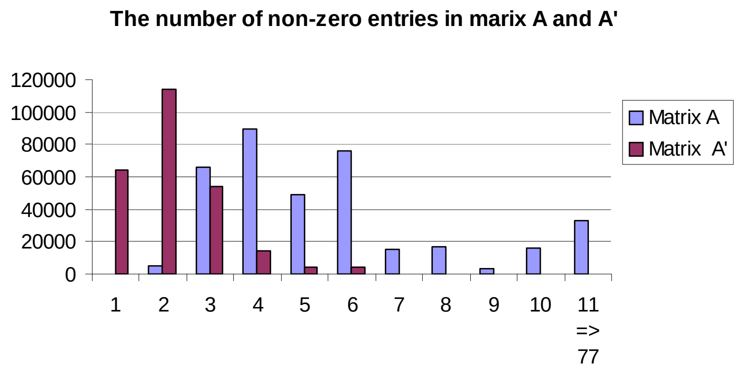
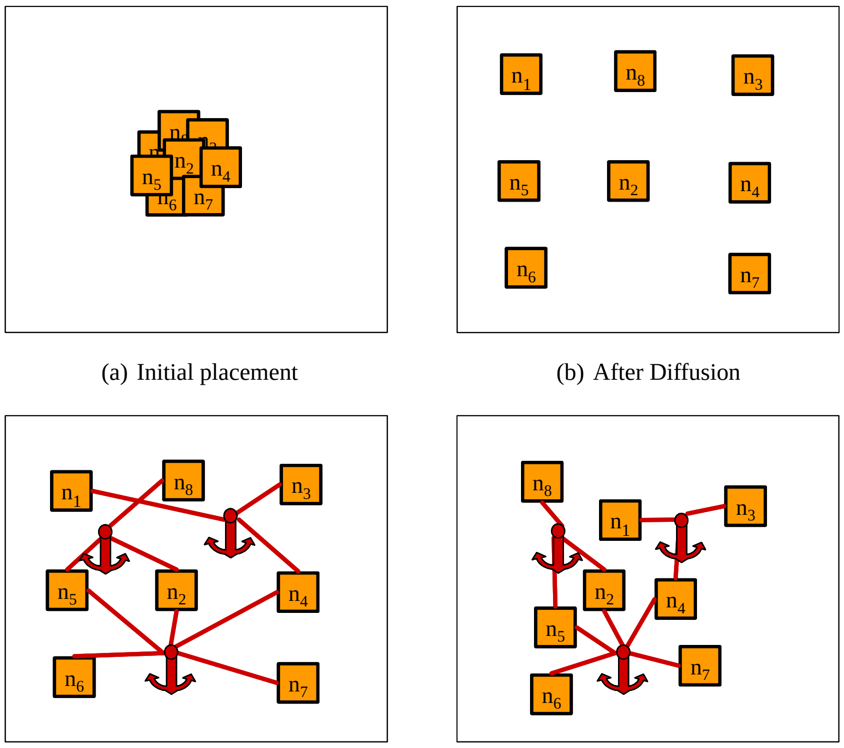


Fig. 9. The comparison of non-zero entries in all rows in the sparse matrix $\mathbf{A}$ and $\mathbf{A}'$. The x-axis is the number of non-zero entries, the y-axis is the row counts. Note: most of rows in matrix $\mathbf{A}'$ has only 2-3 non-zero entries

Figure 9 shows the statistics of the number of non-zero entries in old Hessian $\mathbf{A}$ and new Hessian $\mathbf{A}'$ for circuit *adaptec*2 in ISPD 2005 benchmark. The dimension of the Hessian $\mathbf{A}$ is 354K, while only 254K for the new Hessian $\mathbf{A}'$. In most of rows, the number of non-zero entries in $\mathbf{A}$ are around 3-6, and 1-2 in new Hessian $\mathbf{A}'$. Circuit *bigblue*4 in ISPD 2005 benchmark contains 2 million objects. In our experiments for bigblue4, it takes 200 seconds for pre-conditioning and 75 seconds for solving using the conventional quadratic formulation, while only 11 seconds for preconditioning and 4 seconds for solving using our anchor cells based formulation.

3.3 Unconstrained wire length minimization

The initial placement seed is generated by solving a conventional quadratic formulation. In each following iteration, cells are diffused to obtain the desired density distribution, and anchor cells are inserted and locked. In the successive quadratic formulation, the locked anchor cells will be treated as fixed objects, which will be added to the constant vector **b** in equation 2. Therefore, in this step, the quadratic engine minimize an unconstrained wire length objective. It is to be noted that anchor cells are used in hyper-nets decomposition, no forces or artificial fixed points are used in our formulation.



(a) Initial placement

(b) After Diffusion

(c) Inserting anchor cells and replacing a few multi-pin nets with virtual two-pin nets

(d) Solution of the new quadratic system, cells will not collapse back to the initial status due to anchor cells

Fig. 10. Anchor cell based placement illustration (regular nets are not showed for simplicity

Figure 10 illustrates the idea of one placement iteration. In Figure 10(a), an initial placement is generated and cells are congested in the middle of the placement region. After the pre-placement, cells are spread, as shown in Figure 10(b). But the wire length after spreading could be very bad. It should be noted that cells have not actually moved yet in this step.

In Figure 10(c), we insert a few anchor cells to convert a few nets into the star model, one for each high pin net. All anchor cells are locked once inserted. The locked anchor cells are used as fixed pins and their positions are updated into the quadratic system. After solving the new quadratic formulation, cells are rearranged, but will not collapse back to the initial placement due to the tension from their anchors, as shown in Figure 10(d). We can proceed to next iteration, or we can go over the anchoring cells insertion and wire length optimization sub-step multiple times before proceeding to the next iteration, to further reduce the wire length.

Figure 11 plots the first placement iteration of a circuit. In Figure 11(a), the wire length of the initial placement is 0.48×10^6, and cells are congested in the middle of the placement. After a few iterations of diffusion, cells are spread as shown in Figure 11(b). Although the diffusion explicitly controls the cell movement and improves the density distribution, it is not explicitly wire length aware. The total wire length increases to 2.56×10^6 in Figure 11(b). Once anchor cells are used. The new quadratic formulation leads to the placement in Figure 11(c), with the new HPWL 1.19×10^6, which improved significantly compared with that after the pre-placement.

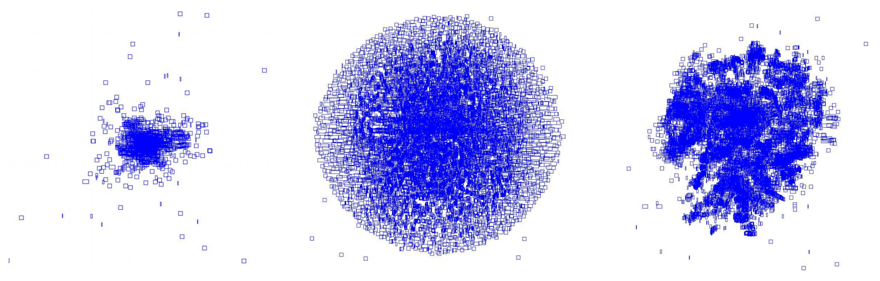

(a) Initial placement. HPWL: 0.48×10^6 (b) After diffusion. HPWL: 2.56×10^6 (c) Actual HPWL: 1.19 $\times10^6$

Fig. 11. One iteration of the diffusion guided placement

3.4 HPWL transformation in a quadratic system

A major weakness of a quadratic wire formulation is that the quadratic objective is an approximation of HPWL for a two pin nets. Transforming a multi-pin net into multiple two-pin nets may enlarge the gap between HPWL and the actual objective to optimize. To alleviate such a problem, existing techniques iteratively linearize the quadratic wire length objective [24]. Recently, the $Kraftwerk$ proposes a method to linearize the quadratic objective into HPWL in the clique model based transformation [22]. Here we propose a method to transform the quadratic objective into HPWL by using the star model based transformation, which helps to reduce the gap between quadratic wire length and HPWL in the DPlace framework.

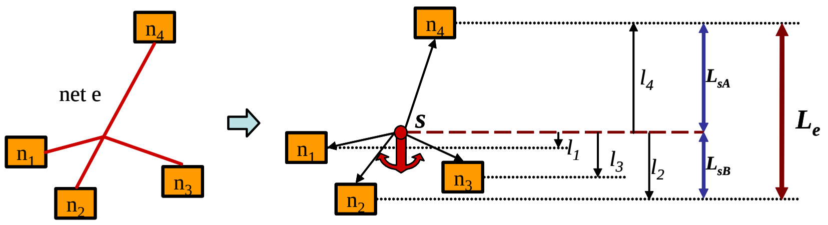


Fig. 12. Net weights computation. $A = \{n_4\}, B = \{n_1, n_2, n_3\}$ in this example

As the wire length minimization problem is independent in x and y directions, here we show the formulation in y direction only. Assuming net e is connected with n cells, and HPWL in direction y is L_e. We add a star cell s to decompose the net e into n two-pin connections. Let l_i denotes the distance between star s and cell i and let w_i denote the weight of each two pin connection. We assign all cells into two sets based on if the cell n_i has a y coordinate large than that of star s. As a result, we have two sets, set $A = \{n_i : y_i > y_s\}$ and set $B = \{n_i : y_i < y_s\}$ for each star model transformation. We define the weight of each two pin net as follows.

$$
\begin{aligned}
w_i &= \frac{L_{sA}}{S_{AB} \times |y_i - y_s|}, \forall n_i \in A \\
w_i &= \frac{L_{sB}}{S_{AB} \times |y_i - y_s|}, \forall n_i \in B \\
&where \\
S_{AB} &= 0.5 \sum_{n_i} |y_i - y_s| \\
L_{sA} &= max\{y_i\} - y_s \\
L_{sB} &= y_s - max\{y_i\} \\
L_e &= L_{sA} + L_{sB}
\end{aligned}
\tag{5}
$$

The anchor cell s is placed at the gravity center of all cells on net e, and S_{AB} is defined as the half of the sum of all distances from cell i to the star. Star s splits the length L_e into two parts, L_{sA} and L_{sB}, as shown in Figure 12.

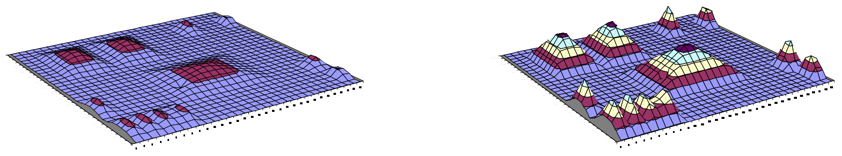
In the following, we show that the above net weighting strategy transforms the quadratic wire length objective into HPWL objective exactly.

$$\begin{aligned}\sum_{i=1}^{n} w_i (y_i - y_s)^2 &= \sum_{i \in A} \frac{(y_i - y_s)^2 \times L_{sA}}{|y_i - y_s| \times S_{AB}} + \sum_{i \in B} \frac{(y_i - y_s)^2 \times L_{sB}}{|y_i - y_s| \times S_{AB}} \\ &= \frac{L_{sA}}{S_{AB}} \sum_{i \in A} |y_i - y_s| + \frac{L_{sB}}{S_{AB}} \sum_{i \in B} |y_i - y_s| \\ &= L_{sA} + L_{sB} = L_e \end{aligned} \tag{6}$$

Figure 12 shows an example of 4-pin nets transformation.

3.5 Fixed blockages

Fixed blockages are obstacles to cell spreading. Modern design may contain a large number of fixed-blockages, which disrupt the cells from smooth spreading. Fixed blockages are density obstacles to prevent cells to pass over and cells are often placed on top of the fixed-blockages in initial placement. If not properly handled, the wire length may grow dramatically while push cells passing over or out of blockages.



(a) Initial adjusted densities for blockages (b) Blockage densities increase gradually during the placement

Fig. 13. Dynamic density on blockages

In DPlace, we use a contour-based density smoothing technique to alleviate the density obstacles. First, we identify large blockages, which are those fixed macros with width and height larger than a certain threshold, such as 1% size of the chip size. In the beginning of the global placement, we adjust the density on bins covered by blockages, the adjusted density distribution is contour based. For a bin covered by a big blockage, the bin density is set to be proportional to the distance between the bin to the blockage boundary. Therefore the highest density is in the bin lying in the middle of the blockage.

In the earlier stages of the global placement, the adjusted fixed blockage density is set to a very small value to allow cells to flow over. As the cells spreading stabilizes, the adjusted density increases gradually, as shown in Figure 13(b). The density in the middle of the fixed blockage rises to push overlapping cells out of blockages smoothly. The diffusion based pre-placement pushes cells over blockage easily according to the adjusted density distribution.

3.6 Wirelength improvement heuristics

Beside the core techniques proposed above, there are more issues that will affect the quality of the placer. Pushing cells away from a region of congestion often contradicts with wire length optimization objective. To further improve the wire length, wire length improving heuristics are performed between each iteration. In certain extent, the wire length improvement heuristics largely determine the quality of the final HPWL. The inaccuracy of using the quadratic wire length as the objective is magnified in large-scale ISPD2005 benchmarks, which contains a large amount of fixed macros.

Therefore, the wire length improvement heuristics are crucial for the HPWL results in quadratic placement. In DPlace, the quadratic optimization step is fast and thus most of the CPU time is spent on wire length improving heuristics. In our experiments, the medium improvement heuristics used in FDP [12] was found effective in the earlier stages of the global placement. Similar technique can also be used in detailed placement for cell swapping. However, the medium improvement heuristic tends to create a lot of overlap in global placement. We use the iterative local refinement [11] to improve the density distribution and further reduce the wire length during the later stages of global placement.

4 Legalization and detailed placement

Legalization and detailed placement are non-trivial for the final wire length quality of the placer. Before legalization, we divide the placement region into regular bin structures and analyze the density overflow in each bin. We swap cells out of the overflowed bins and swap cells between bins if such a swap helps to further reducing the wirelength. Once the bin density overflow is below a threshold, we run a Tetris [25] like legalization flow. We first legalize all movable macros such that no overlap exist between macros. Blockages/macros will split the placement region into row segments. We identify all row segments, sort cells and pack cells into the closet row segment with the minimum cost.

The detailed placement is based on the technique used in [26] and the standard cells are moved between different rows to improve the wire length. We use the global swapping technique to swap cells relatively larger distance to improve the wire length and we use a fixed window to slide through the placement region and test every cell pairs inside the window, and greedily swap two cells if such swapping helps reducing the total wire length.

5 Overall algorithm

The overall algorithm of DPlace is summarized in **algorithm** 1. The first iteration of the global placement stage is illustrated in Figure 10. In every global placement iteration, cells are diffused to reach a specified density distribution, and the anchor cell based wire length optimization is performed *m* times to reduce the wire length. The larger *m*, the shorter the wire length, and the worse the density distribution. Therefore, *m* is normally less than 3. The legalization and detailed placement stages are divided in a fashion similar as most of existing placement tools. To legalize the placement before improving the wire length in the detailed placement.

Algorithm 1 The DPlace

```
 1: The global placement
 2:     Build matrix $\mathbf{A}$, and matrix $\mathbf{A}'$
 3:     Generate an initial quadratic placement with matrix $\mathbf{A}$
 4:     Repeat
 5:         Do diffusion based pre-placement for $k$ iterations
 6:         Do $m$ iterations
 7:             Generate anchor cells and lock them at the gravity centers
 8:             Compute HPWL transformations net weights, update $\mathbf{A}'\mathbf{x} = \mathbf{b}$
 9:             Solve $\mathbf{x} = \mathbf{A}'^{-1}\mathbf{b}$
10:         end
11:         if (In first a few iterations)
12:             Use medium improvement heuristic to repair wire length
13:         else if (Cells are roughly spread)
14:             Use iterative local refinement to repair wire length
15:     Until (reaches a desired density distribution)
16:     Further diffuse cells to remove remaining overlap
17: The legalization
18:     Legalize the macros
19:     legalize the standard cells
20: The detailed placement
21:     Further check the cell density and congestion
22:     Global cell swapping
23:     Greedy window based cell swapping
```

6 Experiments

We test our placer on a Linux server with 3.4 GHz 64-bit Xeon processors. We give the wire length and runtime results on 4 set of benchmarks, the ISPD 2005 and ISPD 2006 placement contest benchmarks [1] and the PEKO-MS 2005/2006 benchmarks. We tested both the LASPack CG solver [27] and the Hybrid solver [28] as our quadratic system solver. Our experimental results are reported based on the Hybrid solver.

6.1 Advantages of our new formulation

Table 1 shows the statistics of the new Hessian matrix **A'** used in our placer, versus the Hessian matrix **A** in conventional formulation. Column *Size* shows the dimension of the Hessian, and column $Non-0s$ shows the non-zero entries in the Hessian. Column $Precon.$ shows the CPU time to preconditioning each Hessian matrix. Same preconditioning quality targets are used for the comparison. Column $Solve$ shows the CPU time to solve one iteration of the quadratic system. Comparing with the conventional Hessian **A**, the new Hessian **A'** is about 30% smaller on the dimension of the matrix. Furthermore, because **A'** is extremely sparse (Figure 9 and Table 1), the runtime to precondition and solve the new quadratic system are improved significantly. The quadratic solver achieved a 24x times speed up on solving time.

Table 1. Statistics on new Hessian **A'** and the Hessian **A** for conventional formulation, and the quadratic solver runtime comparisons

	Matrix A				Matrix **A'**				Solver
	Size	Non-0s	Precon(s)	Solve (s)	Size	Non-0s	Precon(s)	Solve(s)	speedup
adaptec1	243K	196K	15.85	4.65	211K	430K	0.53	0.19	24.5x
adaptec2	355K	2099K	25.61	7.38	254K	557K	0.90	0.30	24.6x
adaptec3	674K	3713K	38.18	15.61	494K	1131K	1.74	0.58	26.9x
adaptec4	508K	3676K	38.42	15.51	451K	997K	1.97	0.49	31.7x
bigblue1	392K	2287K	29.78	6.87	278K	603K	1.16	0.36	19.1x
bigblue2	729K	3937K	47.79	22.78	535K	1178K	2.29	0.82	27.8x
bigblue3	1389K	7290K	103.93	39.32	1096K	2714K	4.54	1.70	23.1x
bigblue4	2831K	16850K	221.47	75.70	2169K	5190K	10.66	3.91	19.4x
									24.6x

6.2 ISPD placement contest benchmarks

Table 2 gives the HPWL results of DPlace on ISPD 2005 contest benchmarks. We also show the runtime for the global placement and the detailed placement in Table 2.

Table 2. Wire length and runtime results for ISPD 2005 benchmarks

	HPWL ($\times 10^6$)	GP(s)	DP(s)	Total(s)
adaptec1	82.56	778	173	951
adaptec2	91.64	952	343	1295
adaptec3	229.63	2219	712	2931
adaptec4	201.42	1591	874	2465
bigblue1	100.14	1412	312	1724
bigblue2	173.51	2451	1114	3565
bigblue3	383.33	4814	1529	6343
bigblue4	926.53	15482	4870	20352

The placement objectives in ISPD 2006 placement contest include both the wire length and the density distribution. As HPWL stands for the half parameter wire length, we use DHPWL representing the density weighted HPWL, which is $HPWL(1 + Density\_Target\_penalty\_factor)$. The $Density\_Target\_penalty\_factor$ is the scaled density overflow in the placement.

Table 3. Wire length and runtime results for ISPD 2006 benchmarks

	HPWL ($\times 10^6$)	DHPWL ($\times 10^6$)	GP (s)	DP (s)	Total (s)
adaptec5	433.06	497.56	3276	1474	4750
newblue1	89.18	89.46	1227	578	1805
newblue2	215.12	217.19	1724	768	2492
newblue3	322.39	324.55	1929	1168	3097
newblue4	266.52	324.56	2141	1361	3502
newblue5	578.52	725.12	5233	1852	7085
newblue6	579.86	599.44	4712	2863	7575
newblue7	1089.15	1215.32	13625	4475	18100

6.3 PEKO-MS Benchmarks

Table 4 and 5 show the HPWL and runtime results of the PEKO-MS 2005 and 2006 benchmarks, which are transformed from ISPD 2005 and 2006 benchmarks in a way that the optimal wire length is known. Column $OPTWL$ stands for the known optimal wire length. $HPWL/OPT$ shows the ratio of the generated wire length against the optimal wire length. GP, DP, and $Total$ show the runtime of global placement, the detailed placement and the total runtime, respectively. Circuits in the PEKO benchmark have no global nets and all local nets. As a results, cells have been roughly spread in the first iteration of the global placement. Therefore, the runtime on global placement for PEKO-MS 2005/2006 benchmarks is relatively small compared with that in ISPD 2005/2006. The PEKO-MS benchmarks show that there are still obvious gap between the DPlace results and the known optimal wire length, both in global and detailed placement stage. Due to high percentage of local nets, the deficiency of the detailed placement is magnified.

7 Conclusions

In this chapter, we present a new quadratic placement tool, DPlace. DPlace uses the diffusion based spreading technique to generate a golden placement for improved density distribution, and uses the anchor cells based formulation to repair the wirelength. Different from existing force directed approaches, we do not add forces or extra fixed points in the DPlace formulation. An anchor cell is the part of the internal net model, and the functions of anchor cells include both the net model transformation and cell movement control.

Furthermore, the Hessian matrix of the anchor cells based quadratic formulation is extremely sparse. As a result, the runtime to solve such a linear system is improved

Table 4. Wire length and runtime results for PEKO-MS-2005 benchmarks

	OPTWL ($\times 10^6$)	HPWL ($\times 10^6$)	HPWL/OPT	GP(s)	DP(s)	Total (s)
adaptec1	20.06	29.53	1.47	269	240	509
adaptec2	24.97	40.56	1.62	349	257	606
adaptec3	40.95	72.59	1.77	521	485	1006
adaptec4	39.39	68.13	1.73	689	389	1078
bigblue1	20.86	31.62	1.52	325	247	572
bigblue2	42.26	67.89	1.61	835	312	1147
bigblue3	94.4	235.45	2.49	943	516	1459
bigblue4	171.48	377.94	2.20	2129	2429	4558
Average			1.80			

Table 5. Wire length and runtime results for PEKO-MS-2006 benchmarks

	OPTWL ($\times 10^6$)	HPWL ($\times 10^6$)	DHPWL ($\times 10^6$)	HPWL/OPT	GP (s)	DP (s)	Total (s)
adaptec5	81.89	207.1	972.96	2.53	1051	599	1650
newblue1	20.5	49.58	107.13	2.42	627	321	948
newblue2	328.69	715.12	1603.56	2.18	1407	313	1720
newblue3	73.51	160.69	388.48	2.19	774	287	1061
newblue4	49.14	109.03	417.07	2.22	1222	295	1517
newblue5	102.08	302.33	1392.88	2.96	1183	811	1994
newblue6	90.66	207.09	556.05	2.28	1482	1140	2622
newblue7	206.18	763.08	2754.53	3.70	4242	2752	6994
Average				2.56			

by 24 times in our experiments. In DPlace framework, since it is possible to affix explicit cell movement control in the pre-placement stage, our new formulation has the advantages for ECO and timing driven placement, in which precise cell movement control is important.

References

1. G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter, and M. Yildiz, “The ispd2005 placement contest and benchmark suite,” in *Proc. Int. Symp. on Physical Design*, (New York, NY, USA), pp. 216–220, ACM Press, 2005.
2. G.-J. Nam, “Ispd 2006 placement contest: Benchmark suite and results,” in *Proc. Int. Symp. on Physical Design*, (New York, NY, USA), pp. 167–167, ACM Press, 2006.
3. TimberWolf Systems, Inc., “Timberwolf placement & global routing software package,” in http://www2.twolf.com/benchmark.html.
4. A. E. Caldwell, A. B. Kahng, and I. L.Markov, “Can recursive bisection alone produce routable, placements?,” in *Proc. Design Automation Conf.*, pp. 477–482, 2000.
5. M. Wang, X. Yang, and M. Sarrafzadeh, “Dragon2000: Standard-cell placement tool for large industry circuits,” in *Proc. Int. Conf. on Computer Aided Design*, pp. 260–263, 2000.
6. M. C. Yildiz and P. H. Madden, “Improved cut sequences for partitioning based placement,” in *Proc. Design Automation Conf.*, (New York, NY, USA), pp. 776–779, ACM Press, 2001.
7. J. Kleinhans, G. Sigl, F. M. Johannes, and K. Antreich, “GORDIAN: VLSI placement by quadratic programming and slicing optimization,” *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. CAD-10, pp. 356–365, Mar. 1991.
8. H. Eisenmann and F. M. Johannes, “Generic global placement and floorplanning,” in *Proc. Design Automation Conf.*, pp. 269–274, 1998.
9. B. Hu and M. Marek-Sadowska, “Far: fixed-points addition & relaxation based placement,” in *Proc. Int. Symp. on Physical Design*, (New York, NY, USA), pp. 161–166, ACM Press, 2002.
10. A. B. Kahng and Q. Wang, “An analytic placer for mixed-size placement and timing-driven placement,” in *Proc. Int. Conf. on Computer Aided Design*, pp. 565–572, November 2004.
11. N. Viswanathan and C. C. N. Chu, “Fastplace: Efficient analytical placement using cell shifting, iterative local refinement and a hybrid net model,” in *Proc. Int. Symp. on Physical Design*, pp. 26–33, 2004.
12. K. Vorwerk, A. Kennings, and A. Vannelli, “Engineering details of a stable force-directed placer,” in *Proc. Int. Conf. on Computer Aided Design*, 2004.
13. T. Chan, J. Cong, and K. Sze, “Multilevel generalized force-directed method for circuit placement,” in *Proc. Int. Symp. on Physical Design*, 2005.
14. B. Yao, H. Chen, C.-K. Cheng, N.-C. Chou, L.-T. Liu, and P. Suaris, “Unified quadratic programming approach for mixed mode placement,” in *Proc. Int. Symp. on Physical Design*, 2005.
15. ISPD_2005_Placement_Contest, “http://www.sigda.org/ispd2005/ispd05/slides/10-1-placement-contest-ispd05.ppt,” 2005.
16. A. B. Kahng, S. Reda, and Q. Wang, “Aplace: A general analytic placement framework,” in *Proc. Int. Symp. on Physical Design*, pp. 233–235, April 2005.
17. T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang, “A high quality analytical placer considering preplaced blocks and density constraint,” in *Proc. Int. Conf. on Computer Aided Design*, 2006.
18. W. C. Naylor, R. Donelly, and L. Sha”, “Non-linear optimization system and method for wire length and dealy optimization for an automatic electric circuit placer,” US patent 6,301,693, 2001.
19. T. Luo and D. Z. Pan, “Large scale placement with explicit cell movement control,” in *Technical Report UT-CERC-06-01*, April 2006.

20. F. Mo, A. Tabbara, and R. K. Brayton, “A force-directed macro-cell placer,” in *Proc. Int. Conf. on Computer Aided Design*, p. 4, EECS, UC Berkeley, November 2000. A demo can be found at: http://www-cad.eecs.berkeley.edu/ fanmo/PlacementAlgorithm/index.html.
21. ISPD_2002_Benchmark, “http://vlsicad.eecs.umich.edu/bk/ispd02bench/,”
22. P. Spindler and F. M. Johannes, “Fast and robust quadratic placement combined with an exact linear net model,” in *Proc. Int. Conf. on Computer Aided Design*, 2006.
23. H. Ren, D. Z. Pan, C. J. Alpert, and P. Villarrubia, “Diffusion-based placement migration,” in *Proc. Design Automation Conf.*, June, 2005.
24. G. Sigl, K. Doll, and F. M. Johannes, “Analytical placement: A linear or a quadratic objective function?,” in *DAC ’91: Proceedings of the 28th conference on ACM/IEEE design automation*, (New York, NY, USA), pp. 427–432, ACM Press, 1991.
25. D. Hill, “Method and system for high speed detailed placement of cells within an integrated circuit design,” US patent 6,370,673, 2002.
26. M. Pan, N. Viswanathan, and C. C. N. Chu, “An efficient and effective. detailed placement algorithm,” in *Proc. Int. Conf. on Computer Aided Design*, 2005.
27. LASpack, “http://www.mgnet.org/mgnet/codes/laspack/html/laspack.html,” 1995.
28. H. Qian and S. S. Sapatnekar, “A hybrid linear equation solver and its application in quadratic placement,” in *Proc. Int. Conf. on Computer Aided Design*, 2005.