Technical Report

Large Scale Placement with Explicit Cell Movement Control

Tao Luo and David Z. Pan

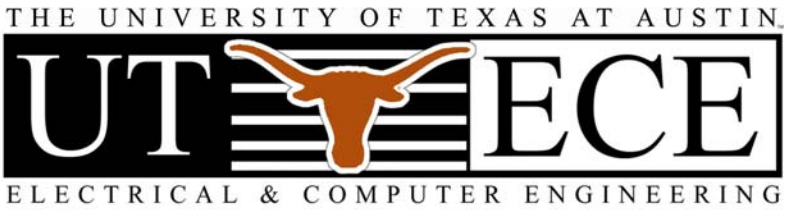
UT-CERC-06-01

April 19, 2006

Computer Engineering Research Center
The University of Texas at Austin
1 University Station, C8800
Austin, Texas 78712-0323

Telephone: 512-471-8000
Fax: 512-471-8967

http://www.cerc.utexas.edu



Large Scale Placement with Explicit Cell Movement Control

Tao Luo and David Z. Pan
Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712
tluo@ece.utexas.edu, dpan@ece.utexas.edu

ABSTRACT

Nowadays a circuit placement algorithm often faces a problem size of multi-million objects, with excessive fixed blockages. We present a new efficient quadratic placement algorithm that scales well to the large-scale circuit placement problems. Instead of adding spreading forces into the quadratic system to push cells away from congestions, we simulate the cell diffusion process and use dummy *anchor cells* to guide the real cell movements. By using the concept of *anchor cells*, our placer has explicit control on how and where to move cells. Meanwhile, the usage of anchor cells significantly reduces the complexity of the Hessian of the quadratic system, which encapsulates the netlist conductivity. The Hessian matrix in our quadratic placer is extremely sparse, with 2-3 nonzero entries in most of rows/colums. Such a matrix is trivial to solve. From our experiments on the large-scale ISPD 2005 placement benchmarks, our placer achieved a **23x** speedup on the CPU usage of the quadratic solver, and comparable wirelengths to existing state-of-art placers.

1. INTRODUCTION

Circuit placement has been studied for decades and continuously attracts research attentions because the placement algorithms are facing exponential growth of problem size and complexity. Nowadays, some industry placement problems are multi-million gate designs with a large amount of fixed blockages[1], which further increases the complexity of the placement.

Historically, existing circuit placement algorithms can be roughly classified into three major categories, such as the simulated annealing based approach [2], iterative partitioning based approach [3, 4, 5], and the analytical placement approach[6, 7, 8, 9, 10, 11, 12, 13].

Among all existing placement approaches, the force-directed analytical placements have been successful in recent years and achieved impressive results on speed of convergence and the scalability. According to the results of ISPD 2005 placement contest [14], 4 of the 5 top ranked placers are force-directed placers. Existing force-directed placers either formulate the placement objectives, such as the quadratic wirelength as the quadratic optimization problem, or use other continuous approximation of the linear wirelength.

Most of existing force-directed quadratic placements are derived from the framework in presented in [7]. There have been well developed theories for quadratic optimizations. Quadratic optimizations gives large penalty on long wires, and its flexibility makes the force-directed quadratic placement a promising technique for timing driven placement. A force-directed quadratic placer involves solving a quadratic system. Overlap constraints are not embedded in the formulation from the beginning. To reduce excessive overlap, typically, the spreading forces or the perturbing fixed points are added to the system to push cells away highly congested region [7, 9, 11] [12]. Most of existing cell spreading forces are similar as the electron charge repulsive force. Uplace [13] adopts a frequency domain based method, which approximates the density constraints into convex functions and adds in to original quadratic form.

Currently, the spreading force is used as an estimation of where to push cells. The explicit and precise control of the cell movements are not handle well by existing force-direct approaches. However, such explicit control is important to cope with challenging placement objectives, such as the timing-driven placement with fixed blockages. Besides, for all quadratic placement approaches, the growing of the problem sizes is a challenge for the scalability of the quadratic solver.

In this paper, we present a new **A**nchor **C**ell based quadratic **P**placement algorithm (**ACP**). ACP is guided by a diffusion process [15] and uses the concept of *anchor cells* to explicitly control the movement of cells. The method to create anchor cells is the same as the one to create stars in [11]. However, the concept and usage of anchor cells are different. In ACP, anchor cells are not active entries appear in the Hessian matrix. Instead, anchor cells are just used to mark the desired target positions for real cells, and also, anchor cells are used as the attractors to pull cells around, as well as the reference positions to restrict the movements of cells. In this paper, we also describe how to smoothly handle the fixed blockages in ACP. We believe that the following are a few advantages of our algorithm.

- Explicit and precise control on cell movement. Once we get the ideal target positions where we should move cells, the anchor cells will guarantee cells to moved around the ideal position under the quadratic placement framework.
- Our formulation is extremely efficient. By using anchor cells, the Hessian matrix in our quadratic formulation has much lower dimension as well as extremely low density, which scales well for the large-scale problems. We achieved an impressive 23x speedup on every iteration of the quadratic solver. For example, it takes only 4 seconds to solve one iteration of the quadratic system in ACP for a 2-million gates design, while it takes 76 seconds if using conventional formulation. Yet, our placer get comparable results to existing placers. Since ACP is a very new idea, we believe it has a lot of room to improve.

The rest of the paper is organized as follows: The overview and motivations are in section follows 2. We describe the details of our global placement in section 3, and the legalization and detail placement in section 4. In section 5, we show the experimental results and we conclude in section 6.

2. OVERVIEW AND MOTIVATIONS

The following section gives a brief overview of the force-directed quadratic placement. Through analyzing the advantages and disadvantages of different force-directed placement strategies, we motivate our approach.

2.1 Quadratic placement

One of the most important objectives of the circuit placement is to minimize the wirelength. Typically, the netlist is modeled as a hypergraph, with nodes representing cells and edges representing the nets. Let x_i and y_i denote the coordinates of each cell, the bounding box wirelength/HPWL is commonly used as the estimation of the routing wirelength. However, the HPWL wirelength function is not convex, the quadratic placement minimize the l_x norm of the length and width of the bounding box of a net, commonly referred as the quadratic wirelength. For a two pin net $e_{i,j}$ that connects cell i and j, the quadratic wirelength is $W_{i,j}((x_i - x_j)^2 + (y_i - y_j)^2)$. $W_{i,j}$ denotes the weight of net $e_{i,j}$. The quadratic placement minimizes the sum of all quadratic wirelength. The optimization in x and y direction are separable,and we can consider one direction for discussion purpose. The cost function in x direction is given by

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

Assuming there are n movable objects in the netlist, $\mathbf{A}$ is the Hessian of the quadratic problem, a n x n netlist connectivity matrix. $\mathbf{A}$ is symmetric, positive definite. $\mathbf{x}$ is the vector of all cells' x coordinates. $\mathbf{b}$ is the vector containing the connectivity information between movable objects and fixed objects, and the pin offsets are also captured in $\mathbf{d}$. The minimizer of the cost function (1) is obtained by the zero gradient of the cost function, $\partial(\Phi(x)) = \mathbf{0}$ which is determined by the following systems of linear equations

$$\mathbf{A}\mathbf{x} + \mathbf{b} = \mathbf{0} \tag{2}$$

2.2 Force-directed method

Solving the unconstrained minimization problem in (1) results a placement with significant overlap among cells. A placer should spread cells to reduce the overlap. To reduce the overlap and spread cells, some quadratic placers recursively partition the placement region, and a typical force-directed placer modifies the objective or constraints of the linear system of equation in (2) and iteratively adding overlap constraints.

Existing force-directed quadratic placers mainly use two types of spreading strategies, the constant force based approach and the fixed point addition based approach. In each placement iteration, Kraftwork [7] and FDP [12] add a constant spreading force vector (f) to equation (2). The force vector $\mathbf{f}$ is used to perturb the placement such that overlap is reduced. The force vector in each iteration is accumulated to prevent cells collapsing back to previous positions. The modified equation with constance forces is given by

$$\mathbf{A}\mathbf{x} + \mathbf{b} + \sum_{k=1}^{i-1}\mathbf{f}^k + \mathbf{f}^i = \mathbf{0} \tag{3}$$

The other types of force-directed quadratic placement use fixed point based approaches to move cells away from congested area. [9] uses two or three fixed point for each cell in every iteration, one fixed point is used to maintain a cell into force equilibrium state, and other fixed points are used to adjust the movability of a cell or adding perturbing force to the cell. [11] uses one fixed point per cell for both preventing the cell from collapsing back and moving away from congested region.

2.3 Existing forces directed approaches

There are advantages and disadvantages for different types of force-directed approach.

2.3.1 Constant forces

For the constant force based approach, in equation 2, the connectivity matrix is never changed. Only the vector is updated in each solving iteration. Therefore, there is no need to pre-conditioning the matrix in each iteration. The matrix pre-conditioning is often more costly than solving the linear system of equations. However, the connectivity matrix is not strictly diagonal dominant, and often ill- conditioned. It is not trivial to scale a proper magnitude of the spreading forces, especially in the initial spreading stages. The physical meaning of adding a spreading force to one cell in the system can be understand as shifting its connected pins and cells. If a large amount of forces are pointing to the same direction, which often happens in the earlier spreading iterations, the large forces will "throw" cells out of the chip boundary.

Without a proper force scaling strategy, the spreading process will be either too slow, or causing stability problems. FDP [12] proposed to solve the stability problem by adding "friction" to a portion of the cells, which will increase the connectivity weight of those cells. One potential problem is, adding virtual connections to a cell will change the original relative movability of the cell, and also affect the meaning of the original cost function.

2.3.2 Fixed point forces

In fixed-point methods, the forces or the target positions of cells are computed first, then virtual fixed points and nets are added to the original linear system of equations to perturb the placement. For fixed-point based methods, adding a virtual fixed-point connection to a cell is to add a diagonal term in the corresponding entry of the cell in matrix $\mathbf{A}$. Adding a virtual connection to a cell will make the corresponding row and column strictly diagonal dominant, thus improve the condition number of the matrix. Therefore, in general, the fixed-point addition based method is more stable.

The fixed point addition method guarantees cells moving inside the convex hull defined by the fixed points. If uses a large weight for the virtual nets, cells tend to move toward pure force direction and cell movements are under tight control. However, the large virtual net weights added to the $\mathbf{A}$ may dominate the actual net connections, and affecting the meaning of the optimization objective. On the contrary, if uses a small weight for virtual nets, fixed-points will be off chip and the movement of cells is less stable. In mFar[9], the success of the fixed addition based method relies on a proper virtual net weighting and perturbing strategy.

FastPlace [11] uses one virtual pin on the chip boundary for both perturbing cells and preventing cells collapse back. Cell shifting is used to generate a target position for each cell, and compute a large enough force to pull the cell away connected cells to the target position. The force computed in such way is very large in the early stage of placement. In the first a few spreading iterations, cells will "explode" out and may move hundreds of times further than their target positions. Such a method converges fast, but also requires non-trivial efforts to control the wirelength damage caused by cell explosion in the earlier stages.

Although fixed-point based methods is more stable, virtual nets added to the original connectivity will change the diagonal term of the original connectivity matrix, and affect the cost function. At meantime, matrix recondition is needed in each solving iteration.

2.4 Motivation of the proposed approach

It is difficult to explicitly control the distance and direction of the cell movements by using existing force-directed methods. A typical force-directed approach computes the directions of spreading forces to move cells, and uses a scaling method to control the magnitudes of the forces. Where a cell eventually go is usually out of control. Such precise control may not be very important for many global placement problems. However, the technique that explicitly controls cell movements will have advantages for constrained placement, such as placement with excessive blockages and the timing driven placement.

To explicitly control the movement of cells, ACP simulates the diffusion process to guide the movement of cells. Anchor cells are used to "memorize' the target positions, and to pull real cells close to where the placer want them be. Furthermore, the fast growing of the problem sizes is a challenge to existing quadratic solvers. We need a method that scales well to the scaling of the problem size. The concept of anchor cells significantly simplifies the solving of the quadratic problem .

3. THE GLOBAL PLACEMENT

APC has a standard two stage placement steps, the global placement stage and the legalization and improvement step. The global placement in ACP is guided by a diffusion process and the anchor cells formulation.

3.1 The diffusion

Diffusion is a natural process that material from highly concentrated areas flow into less concentrated area. Diffusion has been used for placement migration successfully [15]. The cell spreading in placement shares a similar as the nature diffusion process, where cells are driven from high density area to low density area. Diffusion in placement is driven by the density gradient, i.e. the slope and steepness of the density difference. Mathematically, the diffusion process is given 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. We used the discrete approximation method in [15] to solve the diffusion equation .

To use diffusion to move cells, first, the placement region is cut into equal size bins, with each bin holds a certain number of cells, e.g. 50 cells. The bin density is computed as the areas of cells covered by the bin dividing by the bin area. The discrete solver for the diffusion equation in [15] is straightforward, basically is to even out the densities between neighboring bins as time proceeds. In every global placement iteration of ACP, cells are *imaginarily* diffused from high density area to low density area. Every virtual diffusion process takes k iterations. k is a small number in the earlier stages of the global placement and becomes larger in the later stages. The actual cell movements will not happen until we place and lock anchor cells. Anchor cell positions are updated into vector **b** as fixed pins. Real cells are move by solving the system of linear equations in 2. Anchor cells ensure that their connected cells will be pull around the target positions. The anchor cell concept is described in the following subsection.

3.2 Anchor cells

To build the connectivity matrix **A** from a netlist, a hyperedge, which is a net connecting more than two cells, will be transformed into *clique* or *star* model. The *clique* model for a n pins net will be transformed into 2^n clique edges, which are non-zero entries in the Hessian **A**. Because a large number of non-zero entries in the matrix **A** will significantly increase the runtime of the quadratic solver, a common approach is to transform nets with high conductivity into *star* model [11].

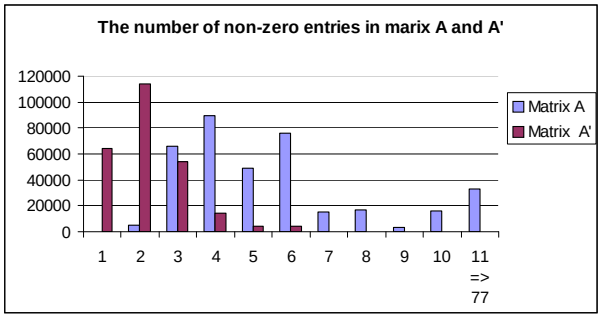


Figure 1: The comparison of non-zero entries in all rows in the sparse matrix A and A′. The x-axis is the number of non-zero entries, the y-axis is the row counts. Note: most of rows in matrix A′ has only 2-3 non-zero entries

Stars are virtual cells with no width and height, and are used to reduce the number of non-zero entries in matrix **A**. However, adding stars will increase the dimension of the matrix. Table 1 is statistics of the ISPD 2005 benchmark sponsored by IBM [1]. The ISPD 2005 benchmarks are derived from real industry designs with circuit size up to 2-million gates. Column #Star in table 1 are the number of stars added to Hessian **A** if setting a connectivity threshold of 5. Note that by transform cliques into stars, the dimension of the matrix will increase up to 40% percent. For the largest benchmark, the dimension of Hessian **A** grows from 2.2 million to 2.8 million. Even uses a very efficient quadratic solver, to solve one iteration of the system of linear equations with the dimension of 2.8 million using conventional formulation will take minutes.

The anchor cell use in ACP is similar as the the stars used in star model, but are used for different purpose. By setting a connectivity threshold t, we assign a anchor cell to the net with the number of pin connections larger or equal to t. We set the t as 3 in our experiments. The smaller t is, the more steadily cells will move to the target position. Note that a small t may over constraint the movement of cells.

To transform the netlist, we use a weighting strategy similar as that in[11]. However, anchor cells are not movable objects and will not appear in the new Hessian matrix **A'**. The new Hessian **A'** in ACP has a dimension the same as the number of movable objects in netlist. In each placement iteration in ACP, Anchor cells are ignored in the cell diffusion, then placed to the gravity centers of their connected cells and locked after diffusion. Anchor cells are used to mark the positions diffusion wants cells to move, and act as anchors to pull real cells around target region in the subsequent quadratic optimization step. The new Hessian **A'** has a dimension much smaller than that in the conventional quadratic placement methods. Meanwhile, the number of non-zero entries in each row of **A'** is the number of pins on the cell. Such an extremely sparse matrix is trivial to solve.

Figure 1 shows the statistics of the number of non-zero entries in old Hessian **A** and new Hessian **A′** for circuit *adaptec*2 in figure 5. The dimension of the Hessian **A** is 354K, while only 254K for the new Hessian **A′**. In most of rows, the number of non-zero entries in **A** are around 3-6, and 1-2 in new Hessian **A'**. Through our

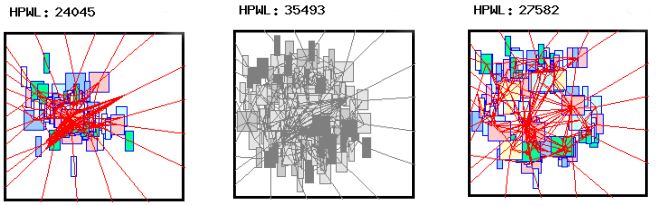
experiments, for the 2M cells circuit *bigblue*4 in table 1, it takes 200 seconds for pre-conditioning and 75 seconds for solving if using the conventional quadratic formulation. However, it takes 11 seconds for preconditioning and 4 seconds for solving if using our anchor cells based formulation.

3.3 Global placement

The initial placement seed is generated by the conventional quadratic formulation. Then all following iterations in ACP start with the previous placement solutions, and cells are *imaginarily* diffused to reach a desired density distribution extent. Then anchor cells are used to mark the suggested positions and fixed. Because quadratic wirelength is an indirect estimation of the linear HPWL wirelength, to further reduce the wirelength, wirelength improving heuristics are performed between each iteration. Figure 4 illustrates how to formulate an efficient diffusion guided placement.

In figure 2(a), an initial placement is generated after one iteration of the standard quadratic placement. Cell are congested in the middle of the placement region and the initial HPWL is 24045. After a few iterations of diffusion process performed, cells are imaginarily spread in figure 2(b). Note that cells have not actually moved yet. Although diffusion improves the density distribution, it does not have explicit control on wirelength, and the total wirelength increases to 35493 in figure 2(b).

As we described earlier, every net with a connectivity larger than k has a anchor cell assigned. Anchor cells are moved to the gravity center of its connected cells basing on the imaginary cell positions. By locking all anchor cells, ACP remembers where diffusion suggested cells to go. The fixed anchor cells are treated as fixed pins and their positions are updated to vector **b** in equation 2. After solving equation 2, cells are spreading out as shown in figure 2(c), and the new HPWL became 27582, which only increases slightly.



(a) Initial placement. HPWL: 24045 (b) Diffusion simulation. HPWL: 35493 (c) Actual movement of cells: HPWL: 27582

Figure 2: One iteration of the diffusion guided placement

3.4 Fixed blockages

Fixed blockages are barriers to block cells' spreading path. However, modern design may contain a large number of fixed-blockages, for example, the circuit *adaptec*2 in figure 5. Figure 5 shows an initial solution of the quadratic placement iteration, and cells are mostly placed over fixed-blockages. Fixed blockages are density obstacles to prevent cells to pass over, and the cost for cells to flow around fixed-blockage is very high.

Aplace [16] applies a landscape smoothing technique to smooth the density distribution. mFar [17] proposes a set of techniques, such as the constrained global placement as well as the ignoring and repairing strategies to cope with blockages. In ACP, we use a contour-based density smoothing technique to alleviate the density obstacles.

First, we identify large blockages, which are those fixed macros

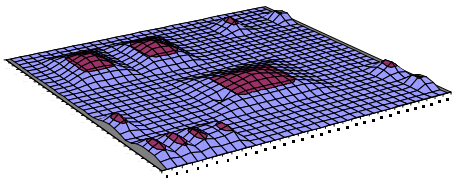


Figure 3: Initial adjusted densities for blockages

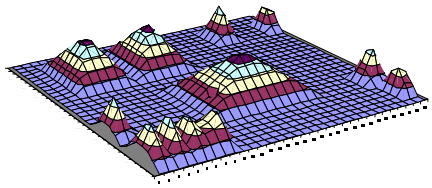


Figure 4: Blockage densities increase gradually during the placement

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 middle bin. Figure 4 shows the adjusted blockage density distribution of circuit *adaptec*2 in figure 5.

In the earlier stages of the global placement, the adjusted fixed blockage density is set to a very small value to allow cells passing over. As the cells spreading stabilized, the adjusted density increases gradually, as shown in figure 4. The density in the middle of the fixed blockage rises to push overlap cells out of blockages smoothly.

4. LEGALIZATION AND THE DETAILED PLACEMENT

4.1 Overall algorithm

The overall algorithm of ACP is summarized in **algorithm** 1

4.2 HPWL improvement heuristics

Equation 2 optimizes the quadratic wirelength, which is an indirect estimation of the linear wirelength. In certain extent, the wirelength improvement heuristics determine the quality of final linear wirelength. The inaccuracy of using the quadratic wirelength as the objective is magnified in large-scale ISPD2005 benchmarks, which contains a large amount of fixed macros.

Figure 5 is the initial quadratic solution for circuit *adaptec*2. Without any spreading, the unconstrained quadratic optimizer generates a solution of 8.306e+7 in HPWL, which is already close to the our final solution in figure 6 (HPWL 9.895e+7). Some of the circuits, for example, the bigblue2, has an initial HPWL of 2.25e+7, which is much larger than most of the final solutions in table 2. Therefore, in quadratic placement, the wirelength improving heuristics are crucial for the final HPWL results. In our placer, the quadratic optimization step is extremely fast, most of CPU time is on wirelength improving heuristics.

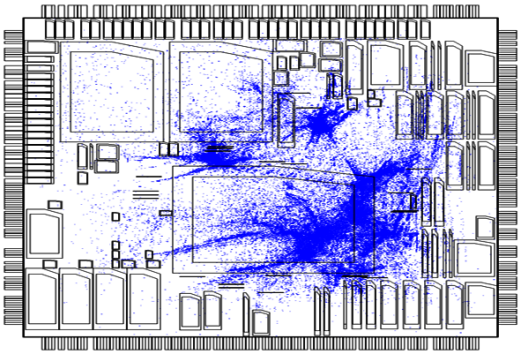


Figure 5: Initial placement of adaptec2: with 254K movable objects and 566 blockages. The initial HPWL: 8.306e+7

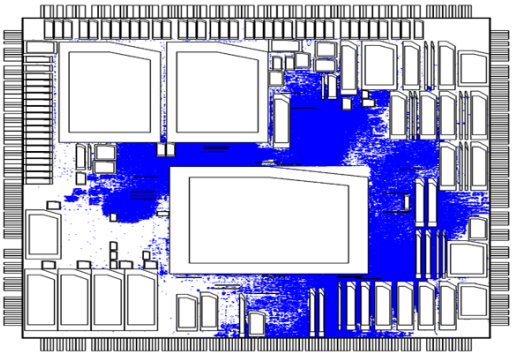


Figure 6: ACP solution of adaptec2 with HPWL: 9.895e+7

From our experiments, the medium improvement heuristics used in FDP[12] is very effective in the earlier stages of the global placement. And the iterative local refinement [11] is more helpful in the later stages of placement. We used different heuristics in different stages of ACP.

4.3 Legalization and detailed placement

The legalization algorithm in ACP is similar as the Tetris[18]. The first step in our legalization stage is to legalize all movable macros and adjust movable macros slightly to ensure no overlaps exist between all macros. All movable macros were fixed before the standard cell legalization. Blockages will split the placement region into row segments. We identify all row segments, sort all cells and pack cells into the closet row segment with the minimum cost.

The detailed placement is performed on standard cells only. We use a window-based greedy swapping technique to further improve the wirelength. That is, we use a fixed window to slide through the placement region and test each cell pairs inside the window. We greedily swap two cells if such swapping helps reducing the total wirelength.

5. EXPERIMENTS

We implemented our placer in C++ and run the experiments on a Linux box with 3.4 GHZ 64-bit Xeon processor and 16G of memory, we tested our placer on ISPD 2005 placement contest benchmarks [1] and use the ISPD 2005 placement contest results [14] for comparison. The quadratic solver is an important part of a force-directed quadratic placer. We tested the LASPack CG solver [19] and the Hybrid solver [20]. In our experiments, the Hybrid solver is at least twice faster than the LASpack CG solver. We use the Hybrid solver as the quadratic engine in our placer.

Algorithm 1 Major steps in our algorithm

```
1:  The global placement
2:      Build matrix A, and matrix A'
3:      Generate an initial quadratic placement with matrix A
4:      Repeat
5:          Diffuse cells for k iterations
6:          Move anchor cells to the gravity centers
7:          Lock the anchor cells, update vector b
8:          Solve the quadratic system x = A'^-1 b
9:          if(earlier iterations)
10:             Use medium heuristic to repair wirelength
11:         else if(Later iterations)
12:             Use iterative local refinement to repair wirelength
13:     Until reaches a desired density distribution)
14:     Further diffuse cells to remove remaining overlap
15: The legalization
16:     Legalize the macros
17:     legalize the standard cells
18: The detailed placement
19:     Further check the cell density and congestion
20:     Greedy window based standard cell swapping
```

Table 1: Benchmarks from ISPD 2005 placement contest. adaptec2 is shown in Figure 6

Circuits	#Obj	#Fix	#Net	Util.%	**#Stars**
adaptec2	255K	566	266K	44.32	**100K**
adaptec4	496K	1329	516K	27.23	**179K**
bigblue1	278K	560	284K	44.67	**114K**
bigblue2	558K	23084	577K	37.94	**193K**
bigblue3	1097K	1293	112K	56.68	**293K**
bigblue4	2177K	8170	2230K	44.35	**662K**

The statistics of the ISPD 2005 contest benchmarks are shown in table 1. Column #Star shows the number of star cells added to the linear system if setting the connectivity threshold as 5. All HPWL results are shown in table 2. There is no runtime information available for other placers. We show our runtime for the global placement and the detailed placement in table 4. Although there is no subject comparisons for runtime, we can get a feeling about how fast is our placer from the data in table 4. e.g. our placer finished the 255K cells circuit *adaptec*2 within 15 minutes. From the HPWL results in table 2, in current stage, our placer produces a result close to existing stage-of-art placers, such as the FastPlace and Capo.

Table 3 shows the statistics of the new Hessian **A'** used in our placer, versus the Hessian **A** in conventional formulation. Column *Size* shows the dimension of the Hessian, and column #Non-zeros

Table 4: Statistics of ACP CPU time on ISPD2005 benchmarks

Circuits	HPWL (e+6)	Global (s)	Detail(s)
adaptec2	98.95	839	48
adaptec4	215.72	1375	158
bigblue1	104.95	1058	45
bigblue2	191.62	1751	180
bigblue3	426.81	3754	339
bigblue4	980.07	14741	1405

Table 2: Wirelength results (All wirelength results except those for ACP are from ISPD2005 placement contest slides)

Placers	Adaptec2	Adaptec4	bigblue1	bigblue2	bigblue3	bigblue4	ratio
Aplace	87.31	187.65	94.64	143.82	357.89	833.21	1.00
mFAR	91.53	190.84	97.70	168.70	379.95	876.28	1.06
dragon	94.72	200.88	102.39	159.71	380.45	903.96	1.08
mPL	97.11	200.94	98.31	173.22	369.66	904.19	1.09
FastPlace	107.86	204.48	101.56	169.89	458.49	889.87	1.16
Capo	99.71	211.25	108.21	172.30	382.83	1098.76	1.17
ACP	**98.95**	**215.72**	**104.95**	**191.62**	**426.81**	**980.07**	**1.18**
NTUP	100.31	206.45	106.54	190.66	411.81	1154.15	1.21
fs50	122.89	337.22	114.57	285.43	471.15	1040.05	1.50
K&D	157.65	352.01	149.44	322.22	656.19	1403.79	1.84

Table 3: Statistics on new Hessian A' and the Hessian A for conventional formulation, and the quadratic solver CPU time

	Matrix A				Matrix **A'**				Solving time
	Size	#Non-0s	Precond.(s)	Solve (s)	Size	#Non-0s	Precond.(s)	Solve(s)	speedup
adaptec2	355K	2099K	25.61	7.38	254K	557K	0.9	0.3	**24.6x**
adaptec4	674K	3713K	38.18	15.61	494K	1131K	1.74	0.58	**26.9x**
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**
									23.5x

shows the non-zero entries in the Hessian. Column *Precond*. 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. All data are the average of 2 solving iterations. Comparing with the conventional Hessian **A**, new Hessian **A'** is about 30% smaller on matrix dimension. Furthermore, because **A'** is extremely sparse (figure 1 and table 3), both the preconditioning and the solving CPU time for **A'** are much less than those for **A**. The quadratic solver achieved an **23x** times speeding up on solving time.

6. CONCLUSIONS AND FUTURE WORKS

In this paper, we proposed the **ACP**, an **A**nchor **C**ell based quadratic **P**lacer, which is guided by the diffusion process, and uses anchor cells to explicitly control the movement of cells. The anchor cell structure makes the Hessian in the quadratic system extremely sparse. As a result, our new method significantly improves the runtime the quadratic solving process, which is the CPU dominant part of existing quadratic placers. We achieved a very impressive **23x** speedup on the quadratic solver on large-scale ISPD 2005 placement benchmark, and our placement results are comparable to existing state-of-art placers.

Since ACP is a brand new placement engine, we believe there is still a lot of room to improve, given that it is ultra fast. Furthermore, since it has explicit cell movement control, we believe it has significant advantage for timing and congestion driven placement where precise cell movement control is needed.

7. REFERENCES

[1] G.-J. Nam, C. J. Alpert, P. Villarrubia, B. Winter, and M. Yildiz, "The ispd2005 placement contest and benchmark suite," in *ISPD '05: Proceedings of the 2005 international symposium on Physical design*, (New York, NY, USA), pp. 216–220, ACM Press, 2005.

[2] TimberWolf Systems, Inc., "Timberwolf placement & global routing software package," in http://www2.twolf.com/benchmark.html.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] H. Eisenmann and F. M. Johannes, "Generic global placement and floorplanning," in *Proc. Design Automation Conf.*, pp. 269–274, 1998.

[8] T. F. Chan, J. Cong, T. Kong, and J. R. Shinnerl, "Multilevel optimization for large-scale circuit placement," in *ICCAD '00: Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design*, (Piscataway, NJ, USA), pp. 171–176, IEEE Press, 2000.

[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 *Proceedings of the IEEE/ACM international conference 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] A. K. K. Vorwerk and A. Vannelli, "Engineering details of a stable force-directed placer," 2004.

[13] 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 *ISPD '05: Proceedings of the 2005 international symposium on Physical design*, (New York, NY, USA), pp. 193–199, ACM Press, 2005.

[14] ISPD_2005_Placement_Contest, "http://www.sigda.org/ispd2005/ispd05/slides/10-1-placement-contest-ispd05.ppt," 2005.

[15] H. Ren, D. Z. Pan, C. J. Alpert, and P. Villarrubia, "Diffusion-based placement migration," in *Proc. Design Automation Conf.*, June, 2005.

[16] S. R. A. B. Kahng and Q. Wang, "Aplace: A general analytic placement framework," in *International Symposium on Physical Design*, pp. 233–235, April 2005.

[17] B. Hu and M. Marek-Sadowska, "Multilevel expansion-based vlsi placement with blockages," in *ICCAD '04: Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, (Washington, DC, USA), pp. 558–564, IEEE Computer Society, 2004.

[18] D. Hill, "Method and system for high speed detailed placement of cells within an integrated circuit design," US patent 6,370,673, 2002.

[19] LASpack, "http://www.mgnet.org/mgnet/codes/laspack/html/laspack.html," 1995.

[20] 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.