University of California

Los Angeles

Multilevel Optimization for VLSI Circuit Placement

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Mathematics

by

Nang Keung Sze

2006

The dissertation of Nang Keung Sze is approved.

University of California, Los Angeles

2006

To my family.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank the Doctoral Committee members for their helpful comments and valuable time for my exam schedules. Especially my thesis advisors Professor Tony Chan and Professor Jason Cong's constant support and guidance made the successful completion of my Ph.D. study.

Moreover, I would like to thank all my colleagues in UCLA VLSICAD Lab. They have been the sources of many insightful discussions. In particular, Joseph Shinnerl, Xin Yuan, and Min Xie had worked closely with me and taught me a lot of technical skills in the field of Computer Science. Also, Janice Martin-Wheeler and Jeanette Miller had helped me with the proof-reading.

At last, but not the least, I would like to thank my family for their continuing encouragement and support, without which I could not continue to finish my Ph.D. study.

The work in this thesis was partially supported by SRC, NSF, and ONR.

Vita

1976	Born, Fujian Province, People's Republic of China.
1996 – 1999	B.S. (Mathematics), The University of Hong Kong.
1999 – 2001	Teaching Assistant, Mathematics Department, The University of Hong Kong.
1999 – 2001	M. Phil. (Mathematics), The University of Hong Kong.
2002	M.S. (Mathematics), UCLA.
2005	Summer Internship in Magma Design Automation, Inc..
2002 – 2006	Research Assistant, Computer Science/Mathematics Department, UCLA.
2001 – present	Ph.D. (Mathematics), UCLA.

Publications

T.F. Chan, J. Cong, T. Kong, J. Shinnerl and K. Sze. "An Enhanced Multilevel Algorithm for Circuit Placement." Proceedings of International Conference on Computer Aided Design, pp. 299–306, Nov 2003.

T.F. Chan, J. Cong, J. Shinnerl, K. Sze, M. Xie, Y. Zhang. "Multiscale Optimization in VLSI Physical Design Automation." Kluwer Academic Publishers, 2003. (Book Chapter)

T. Chan, J. Cong, and K. Sze. "Multilevel Generalized Force-directed Method for Circuit Placement." Proceedings of the International Symposium on Physical Design, San Francisco, CA, Apr 2005, pp. 185–192. (Best Paper Award)

T.F. Chan, J. Cong, J. Shinnerl, K. Sze, M. Xie and Y. Zhang, "Multilevel Optimization in VLSI Physical Design Automation", Multiscale Optimization Methods and Applications, Springer Publishers, 2005. (Book Chapter)

Tony Chan, Jason Cong, Michalis Romesis, Joe Shinnerl, Kenton Sze, and Min Xie. "Enhanced Robustness in Multilevel Mixed-Size Placement." Proceedings of SRC TECHCON, Oct 2005.

Tony Chan, Jason Cong, Joe Shinnerl, Kenton Sze, and Min Xie. "mPL6: Enhanced Multilevel Mixed-Size Placement." Proceedings of the International Symposium on Physical Design, Apr 2006, pp. 212–214.

Abstract of the Dissertation

Multilevel Optimization for VLSI Circuit Placement

by

Nang Keung Sze

Doctor of Philosophy in Mathematics

University of California, Los Angeles, 2006

Professor Tony Chan, Co-chair

Professor Jason Cong, Co-chair

Circuit placement – spatially arranging the components of an electronic circuit in a non-overlapping configuration on a chip – is a crucial step in today's VLSI physical design flow. Placement determines the basic structure of the interconnect, and interconnect delay is the bottleneck of nanoscale VLSI-system performance. Placement is challenging not only because of the enormous number of objects to be placed, but also because of many complex constraints related to non-overlap, signal timing, wireability, temperature, manufacturability, and noise.

A high quality placement tool – mPL6 – has been developed and analyzed in this thesis. It is a highly scalable non-convex nonlinear programming algorithm. It makes use of accurate and smooth approximations of a bounding-box wirelength objective function and generalized bin-based density constraints. These incorporate complicated pairwise cells non-overlapping constraints and are evaluated globally and scalably by fast numerical methods for a Poisson-based partial differential equation (PDE). The nonlinear optimization engine is embedded in a multilevel framework which enables scalability and better global optimization.

Experiments show that mPL6 is a fast placement algorithm producing the shortest wirelength among the state-of-the-art academic placers. It is a stable and robust algorithm that has consistent good performances on wide variety of publicly available benchmarks. The runtime of mPL6 on designs with problem size up to 2 millions is less than 9 hours on a Linux, 1.8GHz AMD Opteron machine.

CHAPTER 1

Introduction

1.1 Motivation

Integrated Circuit (IC) technology plays an important role in the revolution of computing and communication. ICs are used in computers for microprocessor, memory, interface chips etc. Nowadays, ICs are mainly composed of very large-scale integration (VLSI) chips. Figure 1.1 shows the procedures for manufacturing VLSI chips. The circuit placement problem, that we consider in this thesis, is a key step in physical design cycle in manufacturing VLSI chip (cf. Figure 1.2).

Gordon Moore's famous observation that the number of transistors per integrated circuit doubles every two years has held true for the last four decades. The impact of this explosive increase has already transformed practically to all areas of society, making possible all the recent revolutions in information technology. Continuation of the trend can reasonably be expected to lead to similar breakthroughs. By the year of 2010, the minimum feature size of a transistor be expected to shrink to as low as 25nm [itr].

It is possible, however, that the exponential rate of progress may cease very soon, for two reasons. The first is the enormous cost of building fabrication plants. According to the addendum to Moore's law known as Rock's law, the cost of capital equipment to build semiconductors doubles every four years. If current trends continue, the cost of a fab will exceed $10 billion by 2007 and may

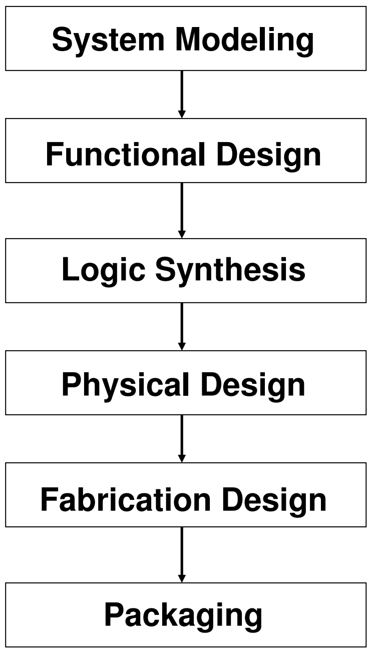

Figure 1.1: Design cycle for VLSI chip

reach $18 billion by 2010 [ICk01]. Such cost increases may not be sustainable [ICk02]. Second, thermal noise analysis suggests that transistors densities will reach fundamental physical limits before 2015, and possibly as early as 2008 [Kis02]. As the final economic and physical barriers to continued increases in CMOS circuit densities begin to take shape, automated design plays an ever more important role in determining system performance. The placement problem is the step that defines the interconnects, which dominate the system performance. Hence, it is a critical step in determining the circuit performance.

However, circuit placement is a NP hard problem which involves placing huge amount of objects (up to several millions) on a given region subject to many constraints (eg. non-overlapping constraint) to optimize the performance of the circuit. It has hampered researchers for over 30 years. Recently, [CCX03, CKS03]

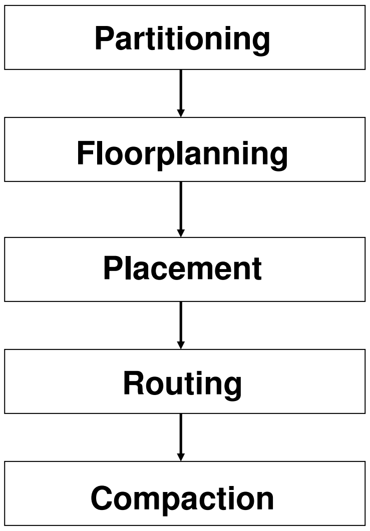

Figure 1.2: Physical design procedure

shows that the quality of the current state-of-the-art placement algorithms (placers) is still 50% to 150% from optimal. It leaves a large gap for further improvement.

Due to the complexity of the placement problem and the necessity of fast production time to market, it is undoubted that one has to use approximation/heuristic algorithm to solve the large-scale problem within a reasonable time. Several heuristic techniques for placement have been studied. In this thesis, a stable and robust placement algorithm is presented. The proposed algorithm reduces the optimality gap to around 20%. It has produced the best average quality over the state-of-the-art academic placers.

1.2 Organization of the Thesis

The organization of the thesis is as follows.

Chapter 2 is the preliminaries. It gives the notations and definitions used in the thesis, a brief description of the circuit that is considered in the thesis, and the introduction to the circuit placement problem addressed in this thesis. Chapter 3 reviews commonly used placement algorithm techniques. Chapter 4 presents a multilevel optimization framework for circuit placement problem. Chapter 5 discusses a robust and stable placement algorithm we have developed. It is the main contribution of the thesis. Experimental results are given in Chapter 6. Finally, conclusions and future work are presented in Chapter 7.

CHAPTER 2

Preliminaries

2.1 Notations and Definitions

Definition 2.1.1 *Given a real-valued $m \times n$ matrix A with a_{ij} as its ij-th entry, the transport of A, denoted as A^T, is an $n \times m$ matrix defined as*

$$A^T \equiv \begin{pmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ \vdots & & & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{pmatrix}.$$

Definition 2.1.2 *Given a real-valued $n \times 1$ vector $\vec{x} \equiv (x_1, x_2, \ldots, x_n)^T$, the L_p-norm of $\vec{x}$, denoted as $\|\vec{x}\|_p$, is defined as*

$$\|\vec{x}\|_p = (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}}.$$

Definition 2.1.3 *$A[n]$ is denoted for a real $n \times n$ square matrix A with each $ij-th$ entry denoted by $A[n]_{ij}$.*

Definition 2.1.4 *The ij-th entry, $C[n]_{ij}$, of a discrete cosine matrix $C[n]$ is defined by*

$$C[n]_{ij} = \sqrt{\tfrac{2-\delta_{i1}}{n}}\cos(\tfrac{(i-1)(2j-1)\pi}{2n}),$$

where $1 \le i, j \le n, \delta_{i1} = 1$ if $i = 1$ and $\delta_{i1} = 0$ otherwise.

Definition 2.1.5 *The tensor product of two matrices, $A[m]$ and $B[n]$ with a_{ij} and b_{ij} as the ij-th entry respectively, is defined as*

$$A[m] \otimes B[n] \equiv \begin{pmatrix} a_{11}B[n] & a_{12}B[n] & \cdots & a_{1m}B[n] \\ \vdots & & & \vdots \\ a_{m1}B[n] & a_{m2}B[n] & \cdots & a_{mm}B[n] \end{pmatrix}$$

which is an $mn \times mn$ matrix.

Definition 2.1.6 *Any local optimal solutions of a real-valued constrained or unconstrained optimization problem, that give the best objective value (smallest value for minimization problem and largest value for maximization problem), are considered global optimal solutions.*

Next, we recall some definitions and theorems from [AB84] for our analysis.

Definition 2.1.7 *An $n \times n$ matrix A is diagonally dominant if*

$$|a_{ii}| \geq \sum_{j=1, j \neq i}^{n} |a_{ij}|, \quad i = 1, 2, \ldots, n,$$

where a_{ij} is the ij-th entry of A. For any matrix A such that $a_{ii} \neq 0, i = 1, 2, \ldots, n-1$, let

$$n(i) = \max\{j \ : \ (i \leq j \leq n) \cap (a_{ij} \neq 0)\}, i = 1, 2, \ldots, n-1.$$

That is, $a_{i,n(i)}$ is the last nonzero entry in the ith row.

Definition 2.1.8 *An $n \times n$ matrix A is an $\hat{M}$ matrix if*

(i) $a_{ii} > 0, i = 1, 2, \ldots, n-1; a_{nn} \geq 0.$

(ii) $a_{ij} \leq 0, i, j = 1, 2, \ldots, n, i \neq j.$

(iii) $n(i) > i, i = 1, 2, \ldots, n-1.$

Lemma 2.1.1 *Let A be a symmetric diagonally dominant $\hat{M}$ matrix and at least one row sum is positive, then A is non-singular and hence positive definite.*

Proof. *Followed from the Theorem 1.17 in [AB84].* $\square$

Definition 2.1.9 *A graph $G = (V, E)$ contains a set of vertices $V = \{v_1, v_2, \ldots, v_n\}$ and a set of edges $E = \{e_1, e_2, \ldots, e_m\}$. Each edge is a subset of V containing 2 vertices. G is a weighted graph if each edge e has a weight $w(e) > 0$.*

Definition 2.1.10 *A hypergraph $H = (V, E)$ is a generalization of graph. It is a graph except each edge $e \in E$ is a subset of V that can contain more than 2 vertices and is called hyperedge. H is a weighted hypergraph if each edge e has a weight $w(e) > 0$.*

Definition 2.1.11 *Given a hypergraph $H = (V, E)$, vertices $v_i \in V$ and $v_j \in V$ are connected if there is an hyperedge $e \in E$ such that $v_i \in e$ and $v_j \in e$. A path $P = \{e_{i_1}, e_{i_2}, \ldots, e_{i_k}\}$ in H is an ordered subset of E such that $e_{i_j} \neq e_{i_l}$ if $j \neq l$ and $e_{i_j} \cap e_{i_{j+1}} \neq \emptyset$, for any $1 \leq j \leq k-1$. A Hypergraph is connected if for any two vertices, there is a path of hyperedges containing the two vertices.*

Using the above definitions and lemma, it is easy to prove the following theorem:

Theorem 2.1.1 *Given a connected hypergraph $H = (V, E)$, suppose each vertex v_i is associated a variable weight x_i and at least one of the vertices has a fixed given weight, then the Jacobian of the quadratic function*

$$q(\vec{x}) = \sum_{e \in E} \sum_{v_i, v_j \in e} w_{ij}(x_i - x_j)^2, \tag{2.1}$$

where $q(\vec{x})$ is a function of the variable weights and each $w_{ij} > 0$ is a given constant, is a positive definite matrix.

Proof. *By direct computation of J_q, the Jacobian of $q(\vec{x})$, one can show that J_q satisfies the conditions of Lemma 2.1.1 and hence is positive definite.* $\square$

2.2 Circuit Description

A brief description of circuits is presented in this section. For simplicity, many engineering details and functionality of a circuit are skipped here.

One of the circuit types addressed in this thesis is the standard cell circuit. Figure 2.1 shows a standard cell circuit. All cells are of rectangular shape. They have the same height but can have different widths. A placement region is given and standard cell row locations are defined. For a legal placement solution, all cells have to be placed on the rows without overlapping with each other. Cells on the boundary of the placement region are called terminal pads or pads. Their locations are given and fixed. Cells are connected by wires, characterized as nets or interconnects. Each net connects a set of cells through the pins on the cells. Figure 2.2 shows a 2-pin net e_1 connecting cells v_1 and v_2, and a 3-pin net connecting cells v_1, v_2, v_3. For a net e, $|e|$ denotes the net degree or the number of pins connected by e. For a cell v, $|v|$ denotes the cell degree or the number of pins on the cell. For example, in Figure 2.2, $|e_1| = 3$, $|v_1| = 2$.

Another circuit type considered in the thesis is the mixed-size circuit. It is similar to standard cell circuit except a small portion of the cells, named macros, can have different height and their area can be more than 1000X larger than the standard cell area. Some of the macros location may be given and fixed.

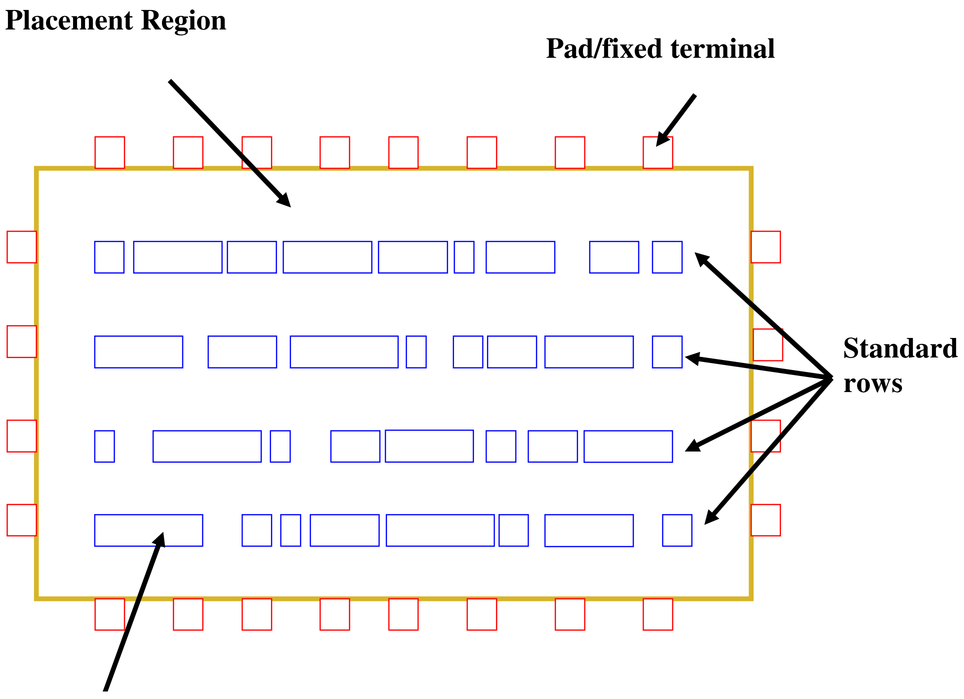

Figure 2.1: Simplified standard cell circuit

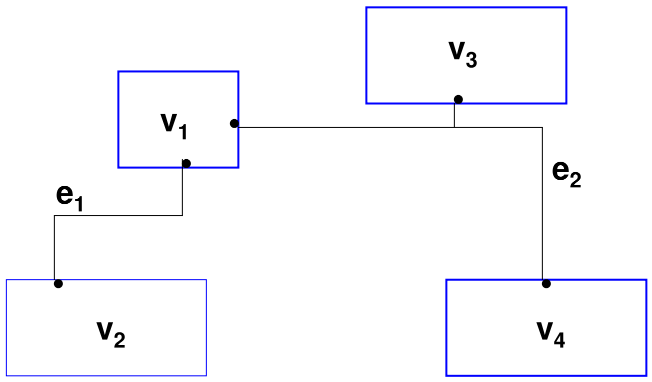

Figure 2.2: Illustration of the connections between cells v_1, v_2, v_3, v_4 by nets e_1, e_2.

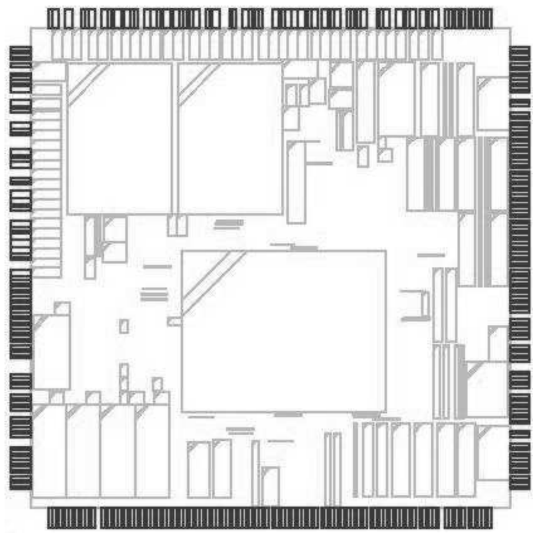

Figure 2.3: Layout of a mixed-size testcase – adaptec2 from ISPD'05 Placement Contest benchmark [NAV05]. Only pads (located on the boundary) and fixed macros are drawn.

Again, all the cell types are of rectangular shape. Figure 2.3 shows a layout of a mixed-size testcase (standard cells not drawn), adaptec2 from ISPD'05 Placement Contest benchmark [NAV05]. All the macros locations are given and fixed inside the placement region. Again, for a legal solution, overlapping between any cells is not allowed.

Given a circuit, the total placeable area is the area of the region where cells can be placed. The whitespace of a given circuit is the total placeable area subtracted by the total movable cells area. The utilization of a circuit is the total movable cells area divided by the total placeable area. For a real design circuit, there are enough whitespace so that cells can be placed within the placement region without overlapping. Note that it is more challenging to find a legal placement of the cells if there is small amount of whitespace.

2.3 Circuit Placement Problem Formulation

Circuit placement problem can be characterized as a hypergraph optimization. Let $H = (V, E)$ be the hypergraph. Let $V = \{v_1, v_2, \ldots, v_N, v_{N+1}, \ldots, v_{N+P}\}$ represents the set of cells and $E = \{e_1, e_2, \ldots, e_m\}$ represents the set of nets. Net with degree k is called $k-$pin net. The set $\{v_{N+1}, \ldots, v_{N+P}\}$ represents pads (fixed terminals) and each e_i is a subset of V that gives the connection among the cells. Figure 2.1 shows the simple characteristics of a standard cell circuit.

Let (x_k, y_k) be the center coordinate of the cell v_k. The pin-to-pin half perimeter wirelength (HPWL) of a net e, given by

$$l(e) = \max_{v_i, v_j \in e, i<j} |(x_i + p^e_{x_i}) - (x_j + p^e_{x_j})| + \max_{v_i, v_j \in e, i<j} |(y_i + p^e_{y_i}) - (y_j + p^e_{y_j})|, \quad (2.2)$$

where $(p^e_{x_i}, p^e_{y_i})$ is the relative pin position on cell v_i connecting by net e, The total HPWL $\sum\limits_{e \in E} l(e)$ is the wirelength used to measure the performance of the placement algorithms. Through out the thesis, we use HPWL or WL to denote the half-perimeter wirelength of a net or a circuit. For simplicity, many placement algorithms consider the center-to-center HPWL

$$\max_{v_i, v_j \in e, i<j} |x_i - x_j| + \max_{v_i, v_j \in e, i<j} |y_i - y_j|, \quad (2.3)$$

for each net e which is easier to minimize as there are significantly fewer variables.

Our objective is to place the cells subject to some constraints such as cells non-overlapping such that the total wirelength $\sum\limits_{e \in E} l(e)$ is minimized. Currently, we consider standard cell and mixed-size placement problem. The cells non-overlapping constraint for the placement problem is to place all the movable cells

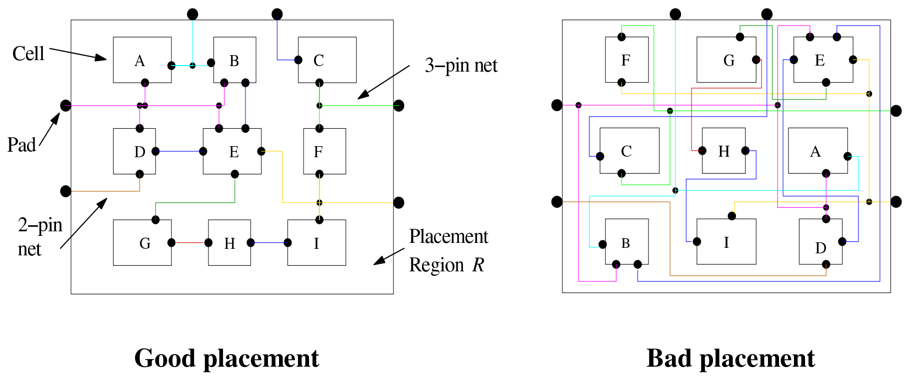

Figure 2.4: Good and bad placement layouts

on the given rows without overlapping with each other (see Figure 2.1). Figure 2.4 shows the importance of decent placement. A bad placement can degrade the performance of the circuit a lot. The congestion of the wires may also make the circuit design impossible to be manufactured.

Typically, placement problem is divided into two stages: global placement and detailed placement. For global placement, one can place the cells on the placement region without too much overlapping between cells. The global placement solution is then legalized by discrete scheme [LK03, CX06]. The wirelength is further reduced by local cell swapping where each move does not create overlapping. In this thesis, we mainly focus on global placement algorithm.

CHAPTER 3

Review of Placement Algorithm Techniques

Automatic circuit placement has received renewed interest recently given the rapid increase of circuit complexity, increase of interconnect delay, and potential sub-optimality of existing placement algorithms [CCX03, CKS03]. As integrated circuit technology further scales, the design sizes are getting larger. Recently, [Con01] shows the importance of building a good physical hierarchy from a flattened or nearly flattened logical netlist for performance optimization. Therefore, large-scale placement on a nearly flattened netlist is needed for physical hierarchy generation to achieve the best performance. This is even more critical for deep sub-micron or nanometer designs as the interconnect has become the performance bottleneck. Many placement algorithms minimize the approximated wirelength objective and reduce overlapping between cells alternatively. In the following, we discuss several techniques for handling wirelength and overlapping constraint.

Typical techniques used in the current state-of-the-art placement tools consist of min-cut partitioning [CKM00a, YM01], simulated annealing [WYS00, CCY03], analytical methods with exact HPWL minimization [HL99], quadratic wirelength minimization [KSJ91, CCK00], linear wirelength minimization [SDJ91, CCK03b] and log-sum-exponential wirelength minimization [Na01, KW04, CCS05b]. The first two techniques always produce a global placement with not much cell overlapping. On the other hand, analytical techniques for minimizing some unconstrained smoothed wirelength objectives usually introduces a lot of cell overlapping or area congestion during the global placement. Hence, area congestion removal techniques such as slot assignment [CCK00, CCK03b], recursive bisection/quadrisection partition [Vyg97], ripple-move [HL00, CCK03b], cell shifting [VC04] and grid warping [XMF04] have been introduced. Algorithms [CCS05b, KW04, EJ98] are also proposed to minimize the wirelength objective and area congestion simultaneously. Usually, those placement techniques are embedded in hierarchical/multilevel framework to speed up the placement process [CCS05b, KW04, CKM00b, WYS00, YM01, CCK00, CCK03b, CCX03]. Multilevel framework is discussed in more detail in Chapter 4.

In the following section, different wirelength minimization and overlapping removal techniques are discussed.

3.1 Wirelength Handling

Since (2.3) is not differentiable and the constraints are highly non-convex, the minimizer is hard to locate. Therefore, many direct or indirect approximations of (2.3), that can be minimized efficiently, have been studied.

Typical indirect minimization of (2.3) is used in min-cut partitioning based placers [CKM00a, YM01]. Since the cut size is related to net length, the smaller the cut size implies the smaller the HPWL. Using the cut size driven based recursive partitioning gives a fast way to minimize the HPWL and remove overlapping. However, minimizing the cut size during partitioning is not always minimizing the HPWL effectively.

An unconstrained minimization of the exact HPWL (2.3) can be formulated as a linear programming [HL99]:

$$\begin{array}{c} \min \sum_{e \in E} (r_e^x - l_e^x) + (r_e^y - l_e^y) \\ s.t. \quad l_e^x \leq x_i \leq r_e^x, l_e^y \leq y_i \leq r_e^y, \forall v_i \in e, \forall e \in E, \end{array} \tag{3.1}$$

where l_e^x and r_e^x are the leftmost and rightmost ends of net e respectively, l_e^y and r_e^y are the lowest and uppermost ends of net e respectively. It can minimize the HPWL directly. However, it is too costly to solve, not scalable, and difficult to incorporate with overlapping constraint.

Using continuous differentiable functions to approximate (2.3) is necessary for analytical placers [KSJ91, SDJ91, EJ98, CCK00, KM00, KW04, VKV04, CCS05b] since typical mathematical programming requires the smoothness of the objective function.

Many studies [KSJ91, SDJ91, EJ98, CCK00, VKV04] use quadratic function approximation:

$$\sum_{e \in E} (\sum_{v_i, v_j \in e, i<j} |x_i - x_j|^2 + \sum_{v_i, v_j \in e, i<j} |y_i - y_j|^2). \tag{3.2}$$

The advantage of using the quadratic wirelength objective is that its unconstrained minimizer can be obtained by solving a positive definite linear system of equations (cf. Theorem 2.1.1). However, it over-penalizes the long nets which gives a bad half-perimeter wirelength placement solution which has illustrated in [KSJ91, MAN94, KM00]. To avoid over-penalizing the long nets, better approximation of (3.2) is proposed in [KSJ91] by using linearized quadratic wirelength:

$$\sum_{e \in E} (\sum_{v_i, v_j \in e, i<j} |x_i - x_j| + \sum_{v_i, v_j \in e, i<j} |y_i - y_j|). \tag{3.3}$$

It certainly reduces the effects of over-penalizing the long nets. Typical Weiszfeld iteration [Eck75] is used to solve it but requires more computations than minimizing (3.2).

A more accurate approximation of (3.2) by L_p-norm [KM00] is given by

$$\sum_{e\in E}((\sum_{v_i,v_j\in e, i<j}(x_i-x_j)^p)^{\frac{1}{p}}+(\sum_{v_i,v_j\in e, i<j}(y_i-y_j)^p)^{\frac{1}{p}}). \tag{3.4}$$

Since L_p-norm tends to maximum norm when p tends to infinity, the larger p implies more accurate approximation of (2.3). However, the non-smoothness of the function at the zero point may cause difficulties during minimization.

Another accurate approximation of the half-perimeter wirelength [Ber82, Na01, KW04, CCS05b] is given by

$$\begin{aligned}&\eta\sum_{e\in E}(\log\sum_{v_k\in e}\exp(x_k/\eta)+\log\sum_{v_k\in e}\exp(-x_k/\eta)\\&+\log\sum_{v_k\in e}\exp(y_k/\eta)+\log\sum_{v_k\in e}\exp(-y_k/\eta)),\end{aligned} \tag{3.5}$$

where the smaller η, the more accurate the approximation. It is a smooth convex function. Also, the number of terms in (3.5) is much fewer than those in (3.4) which makes it easier to minimize. However, it costs more to minimize (3.5) than (3.2).

3.2 Overlapping Handling

A direct consideration of cells overlapping is by posing pairwise non-overlapping constraint [CCK00]. For simplicity, each cell v_i is considered a disk with radius $r_i=\sqrt{h_iw_i/\pi}$. Then for each pair of cells v_i and v_j, the non-overlapping constraint is given by

$$(x_i-x_j)^2+(y_i-y_j)^2\geq(r_i+r_j)^2. \tag{3.6}$$

However, the quadratic number of constraints is not scalable and this approach is used to solve small enough problems in [CCK00, CCK03b].

Another direct way to remove overlapping between cells is to pose a uniform bin grids on the placement region where the number of bins is equal or larger than the number of cells. A weighted bipartite graph between cells and bins is constructed, using the distance between the cell and the bin as the weight. A minimum weighted matching is computed so that cells are assigned to their target bins to minimize the total displacement of the cells. The process is called slot assignment which is used in [CCK00, CCK03b]. To make the slot assignment more scalable, a two-way recursive partitioning of the region and cells is done until the subregion slot assignment problem is small enough. However, it is effective to remove overlapping only when all cells are near uniform size. Also cells are moved more than necessary if the bin size is much larger than the cell area.

A recursive four-way partitioning of the cells by minimizing the total displacement of the cells, proposed in [Vyg97, BS05], can be considered a generalization of the slot assignment. It can handle cell with different sizes and is very scalable.

To avoid direct handling the quadratic number of pairwise non-overlapping constraints, many studies use indirect measurement of the cells overlapping. A bin grid structure is posed on the placement region (cf. Figure 5.1). The capacity of the bin is the bin area. The density of a bin is the sum of all the cells area overlapping with the bin. Once the density of a bin exceeding the bin capacity, it is considered overflow. If the bin area is small enough and no overflow bins exists, the placement is expected with little overlapping between cells. It is more efficient to use the bin overflow to approximate the cells overlapping since one can use linear number of bins in terms of number of cells, which is much fewer than the number of pairwise non-overlapping constraints.

There are several effective algorithms proposed to reduce the density of an overflow bin or to even the density in each bin. For example, in ripple-move algorithm [HL99], to reduce overflow in a bin, a nearby underflow bin is chosen. A directed acyclic graph is built using the overflow bin as source and the underflow

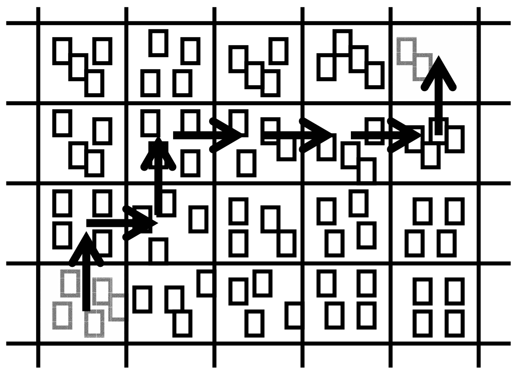

Figure 3.1: Ripple-move area-congestion control.

bin as sink. There are only edges between adjacent bins directed towards the sink. Each edge uses the most wirelength reduction to move a cell to its adjacent bin as the weight. A longest monotone path is computed (cf. Figure 3.1). Cells are moved to their adjacent bin along the longest path. Since each moved cell is perturbed by a bin size, this gives an effect of ripple move and avoids large increase in wirelength. Also, using the reduction of wirelength as the weight helps reduce the wirelength or avoid too much wirelength increase. However, this approach is effective only when all cells are nearly uniform size.

Other effective and more general algorithms to reduce overflow are cell shifting [VC04], grid warping [XMF04], diffusion based placement migration [RPA05], and force-directed method [EJ98, VKV04, CCS05b]. The main goals of these techniques consist of keeping the relative ordering and minimum displacement of the cells.

The placement algorithm proposed in this thesis is based on a force-directed method to reduce overflow. More details are given in Chapter 5.

CHAPTER 4

Multilevel Algorithm for Placement

In this chapter, multilevel framework for placement is discussed. Placement algorithms [CCK00, CCK03b] utilizes the multilevel framework are presented at the end of the chapter.

4.1 Multilevel Framework

Multigrid and algebraic multigrid have been successfully applied to solve partial differential equations and linear system of equations respectively [Bra01, BMH00, TOS00, Bra86, RS87]. It is highly effective and scalable. Similar ideas have brought to solve large-scale optimization problems [BR02]. Many studies [AHK97, KAK97, CCK00, CCK03b, CCY03, CCK03a] show that multilevel algorithm is a promising technique to handle large-scale optimization problems in VLSI domain. It is not only used for speed-up, but also for better global optimization.

Figure 4.1 shows an example of a V-cycle optimization in multilevel framework for placement which is effectively used in [CCK00, CCK03b, VK05, CCS05b]. The main idea in multilevel algorithm is to build a sequence of coarser level or simplified problems to approximate the original problem (the finest level problem). Each coarse problem is an approximated and simplified problem of the finer level problem such that a good solution obtained from optimizing the coarse prob-

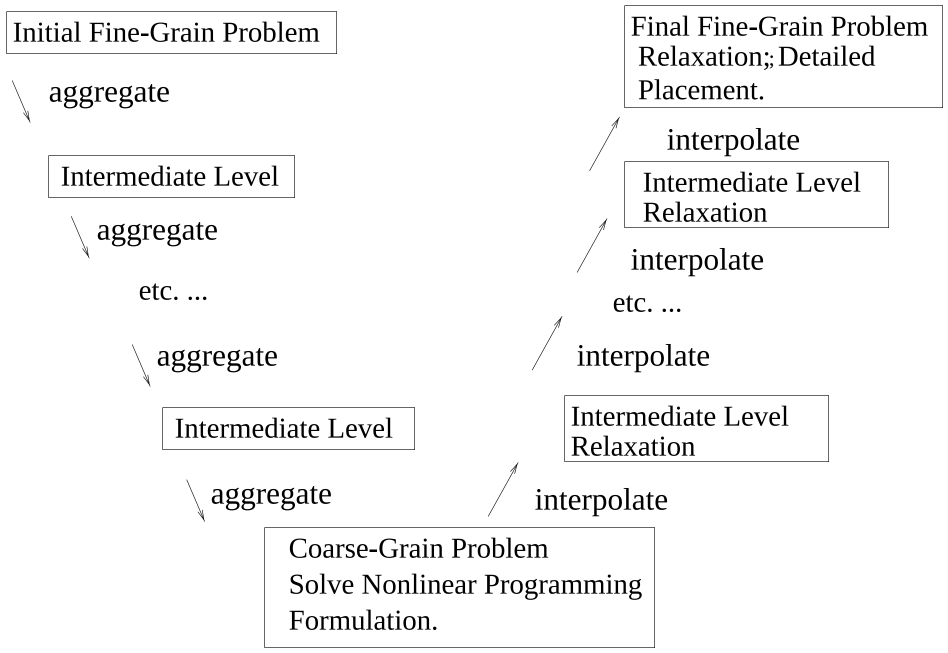

Figure 4.1: Multilevel optimization V-cycle

lem serves a good starting point for optimizing the next finer level problem. The main components are coarsening (clustering) – to build the coarse level problems, interpolation – to transfer solutions between levels, relaxation – to optimize the problem or improve a given solution, and multilevel flow – an order of sequence to optimize from the coarsest level problem to the finest level problem. Each component is discussed in the following sections.

4.1.1 Coarsening

The purpose of coarsening or clustering is to build a hierarchy for the multilevel paradigm. Due to the physical meaning of the placement problem, a natural way to reduce the problem size is by grouping a subset of cells (cluster) until the number of clusters is small enough. After clustering the cells, it becomes the placement of the clusters only and the nets connecting cells within the cluster are discarded. This indeed gives a coarse or simplified problem of the original problem.

To assure that the clustered netlist is a good approximation of the original netlist, the way to cluster the cells plays an important step in building the coarse level problems. There are many studies on clustering scheme [KAK97, CL04, HM04, AKN05]. In general, clustering scheme can be divided into two steps. The first step is to define the affinity between two connected cells, the larger the affinity the higher the chance to cluster the cells. The next step is to decide the sequence to pick a cell and cluster it with one of its neighbors that sharing the largest affinity. The coarsen hypergraph is then constructed as follows. An affinity graph is constructed by joining each vertex to exactly one of its neighbors for which it has maximal affinity. Each group of joined vertices is called a cluster and become the coarser level vertex. The cluster size equals the sum of the size of the grouped cells. Hyperedges are defined on the clusters in the obvious way: each hyperedge on the finer level becomes a hyperedge (the set of clusters containing those vertices) at the coarser level, with the singleton hyperedge simply ignored. We hence get a smaller hypergraph at the coarser level.

A clustering scheme for hypergraph coarsening, called First Choice, is proposed in [KAK97]. It first transforms the hypergraph into a clique model weighted graph (see Figure 4.2). Given a hypergraph $H = (V, E)$, the weight or affinity between any two vertices v and w in the clique model weighted graph is defined as

$$r_{vw} = \sum_{e \in E | v, w \in e} \frac{1}{(|e| - 1)}. \tag{4.1}$$

It traverses the list of vertices in a random order. Each traversed vertex is then clustered with its neighbor with the largest affinity (4.1). The clustering algorithm stops when the number of clusters has reached the target.

From the equation (4.1), the larger the affinity implies the higher the connec-

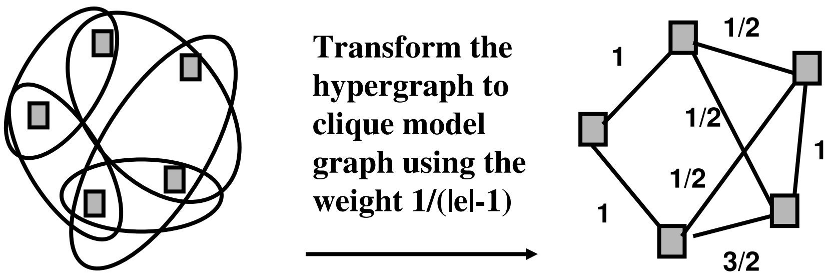
Transform the hypergraph to clique model graph using the weight 1/(|e|-1)
1/2
1
1/2
1
1/2
1
3/2

Figure 4.2: Transformation from hypergraph to clique model weighted graph.

tivity between the cells. Also, the smaller the net degree $|e|$ contributes the more affinity between the cells. The intuition for it to be a good clustering scheme for placement problem is that cells with high affinity should stay close together in a good placement solution. In the following sections, two clustering schemes, modified First Choice and Best Choice, for placement problem are presented.

4.1.1.1 Modified First Choice Clustering

For placement problem, the affinity between vertex or cell v and w is defined as

$$r_{vw} = \sum_{e \in E | v,w \in e} \frac{1}{(|e|-1)\text{area}(e)}, \tag{4.2}$$

where area(e) denotes the sum of the areas of the cells in e. The additional term area(e) is an indirect way to control the cluster size. Unlike the affinity defined in (4.1) which is mainly proposed for hypergraph partitioning problem, the modified affinity (4.2) is to target the placement problem. By controlling the cluster size, the coarsen hypergraph has less variations in the clusters size. Instead of traversing the cells in a random sequence, the cells are traversed in ascending order of the cell area (with preference to smaller cell degree to break tie). This ordering gives a more balance in clusters area. If a good initial placement is provided, the geometric information of the cells can be incorporated into the affinity between cells:

$$r_{vw} = \sum_{e \in E | v,w \in e} \frac{1}{(|e|-1)\text{area}(e)\text{dist}(v,w)}, \tag{4.3}$$

where dist(v, w) is the Euclidean distance between v and w. That is, closer neighbor contributes more to the affinity.

4.1.1.2 Best Choice Clustering

Another more sophisticated clustering scheme is Best Choice proposed in [AKN05]. The affinity between any two cells v and w used in the Best Choice is defined as

$$r_{vw} = \sum_{e \in E | v,w \in e} \frac{1}{(|e|)(\text{area}(v) + \text{area}(w))}, \tag{4.4}$$

where area(v) and area(w) denote the area of the cell v and cell w respectively. In addition to the indirect control of the cluster size, it poses a hard bound for the cluster size. when the formed cluster area larger than the predefined upper bound for the cluster size, the merged clusters are restored.

It first computes the affinity or score (4.4) for each cell. The one with largest score is chosen to cluster. Hence it gives a more globally optimal sequence for the clustering process if (4.4) is a good affinity. Once a cluster is formed, it updates the netlist immediately. That is, the coarsen hypergraph is updated immediately after clustering. And the updated hypergraph is used to compute the score for the new clusters. However, the immediate updating of the hypergraph after clustering is time consuming, a lazy update is proposed to reduce the runtime. Instead of updating the hypergraph immediately, it updates the hypergraph only when the one with maximal score is required an update. Since the changes to the netlist after clustering only affect the neighboring cells, the score for those cells can be updated when they are chosen.

Similar to the modified affinity (4.3), if a good initial placement is provided,the geometric information of the cells can be incorporated into the affinity between cells:

$$r_{vw} = \sum_{e \in E | v,w \in e} \frac{1}{(|e|)(\text{area}(v) + \text{area}(w))\text{dist}(v,w)}, \tag{4.5}$$

where $\text{dist}(v,w)$ is the Euclidean distance between v and w. That is, closer neighbors contribute more to the affinity.

4.1.2 Relaxation

Relaxation is referred to the optimization process for a given problem. In multilevel framework, the ideal complexity for relaxation at each level should be linear time in terms of the number of variables at that level so that the overall complexity for a V-cycle (cf. Figure 4.1) optimization is of linear complexity.

At the coarsest level where the problem size is small enough, the relaxation should be good enough to find a good solution or nearly optimal solution. At the other levels, an initial solution is given or transfered from the coarser level and serves as a starting point for the relaxation scheme. The initial solution is iteratively improved by the relaxation scheme.

4.1.3 Interpolation

Interpolation is used to transfer solutions from level to level. For example, given a placement solution at the coarse level, we use it to compute the placement solution at the finer level via interpolation.

A simple interpolation scheme, called constant interpolation, is to simply assign the cluster's location to its sub-clusters. This interpolation gives an immediate solution for the finer level problem. However, for placement problem, this means putting all the sub-clusters on top of each other. It creates significant overlapping between cells which may not serve as a good initial solution for some particular placement techniques.

A more sophisticated interpolation scheme, called weighted or AMG-based interpolation, is proposed in [CCK03b]. It creates less cells overlapping comparing to the constant interpolation. A graph model of connectivity (cf. Figure 4.2) is employed to define the interpolation: the weight of edge e_{ij} is

$$w(e_{ij}) = \sum_{\{e \in E \,|\, i,j \in e\}} \frac{1}{(|e| - 1)}. \tag{4.6}$$

For efficiency, only weights above a certain threshold (currently 1/4) are used. Finer level vertices v_i within each cluster with the highest vertex degree (using cell area to break tie) are designated as "C-points" and are given the positions of their parent clusters. "C-point" locations are fixed during interpolation. The remaining points are designated as "F-points" and are placed at the weighted average of the positions of the C-points to which they are connected. Once an F-point has been placed, it can be treated like a C-point and used to influence the positioning of other F-points to which it has connections (cf. Figure 4.3. Moreover, since the process depends on the vertex order, iterations are used to allow all interconnected nodes to influence each others' positions. For this purpose, the nodes are ordered by decreasing connectivity $w(v_i) = \sum_j w(e_{ij})$, following (4.6).

If an initial solution is given for the original problem, it needs to be transfered to the coarse levels so that it can be utilized and further improved through the coarse levels optimization. The interpolation of a solution to coarser level is relatively simple – the coarser level vertex or cluster position is determined by

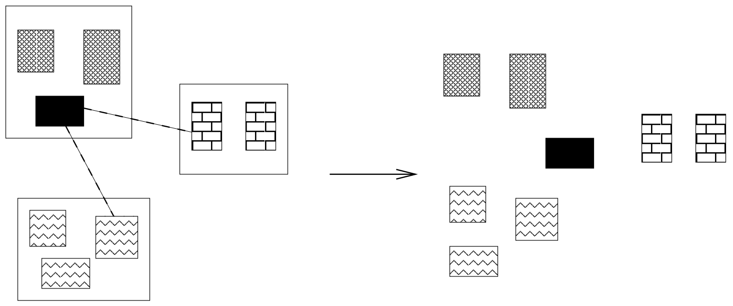

Figure 4.3: AMG-based weighted interpolation

the average positions of its sub-clusters.

4.1.4 Multilevel Flow

Multilevel flow is the sequence of coarse levels relaxation. Figure 4.4 shows three different sequences of coarse levels relaxation. The standard multilevel flow is the V-cycle optimization. In a V-cycle, coarse level problems are constructed recursively until the problem size is small enough for the relaxation to find a good solution. Then the solution at the coarsest level is interpolated and relaxed subsequently up to the finest level (the original problem). A more extensive coarse levels relaxation sequence is called F-cycle. For each coarse level problem, a V-cycle optimization flow is used to optimize the problem. It gives more extensive relaxations at coarse levels at the expenses of runtime. A back-tracking V-cycle (Figure 4.4) is a trade-off of speed and quality between V-cycle and F-cycle.

The multilevel flow can be repeated to further improve the quality of the solution. The solution from the previous multilevel flow can be incorporated into the clustering scheme affinity (4.3) and (4.5) to speed up the convergence.

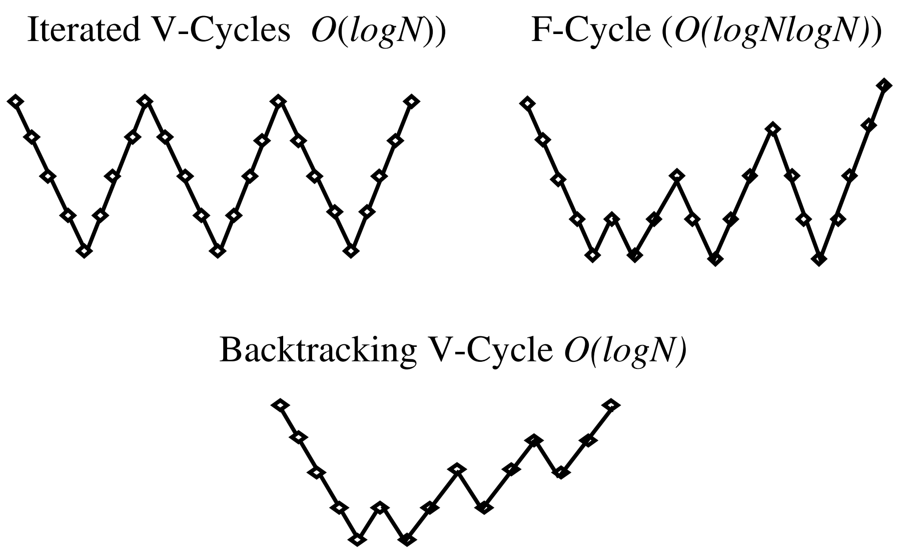

Figure 4.4: Iterated multilevel flow alternatives

4.2 Multilevel Placement Algorithm – mPL

In this section, an effective multilevel algorithm utilizes the multilevel framework is discussed. A multilevel placement algorithm, called mPL, is developed in [CCK00]. It is based on the multilevel paradigm discussed in the previous section. It makes use of edge-separability clustering scheme [CL00]. At the coarsest level, the relaxation is a customized interior-point method which is used to solve a nonlinear programming formulation – minimization of the quadratic wirelength (3.2) subject to pairwise non-overlapping constraints (3.6). Slot assignment followed by GOTO based discrete single cell optimization [CCK00] is used for relaxation at each level. It uses one V-cycle with constant interpolation.

An enhanced version of mPL, called mPL2, is proposed in [CCK03b]. The first version of mPL, is called mPL1 for clarity in subsequent discussions. mPL2 makes use of modified First Choice as the clustering scheme and AMG-based weighted interpolation discussed in the previous sections. In addition to the optimization algorithms proposed in [CCK00], it incorporates a more effective relaxation scheme, called quadratic relaxation on noncontiguous subsets (QRS) discussed in the following.

At each level of the interpolation phase, sweeps of unconstrained quadratic relaxations on small, noncontiguous movable subsets of cells are applied. Let M denote a designated set of movable cells. We denote the set of nets containing at least one movable cell by E_M and the set of fixed cells in E_M by F. That is, $F = \bigcup_{e \in E_M} e \setminus M$. For each movable set M, we minimize the total weighted quadratic-star wirelength of E_M.

For each net $e \in E$, let $|e|$ denote the number of cells in net e, (x_i, y_i) the center of cell v_i, and let

$$(x_e,\, y_e) = \left(\frac{1}{|e|} \sum_{v_i \in e} x_i, \quad \frac{1}{|e|} \sum_{v_i \in e} y_i \right).$$

By weighted quadratic-star wirelength, we mean

$$\ell_2^* = \frac{1}{2} \sum_{e \in \mathcal{E}_M} \sum_{v_i \in e} \frac{(x_i - x_e)^2}{|x_i^{(k)} - x_e^{(k)}|} + \frac{(y_i - y_e)^2}{|y_i^{(k)} - y_e^{(k)}|}$$

where the fixed-constant displacement weights
$1/|x_i^{(k)} - x_e^{(k)}|$ and $1/|y_i^{(k)} - y_e^{(k)}|$ are used at each iteration after the first, as in Gordian-L [SDJ91], in order to help the objective gradually approximate a linear-star wirelength as closely as possible in a smooth way. In the very first iteration, the weights are not used, and the objective is simply quadratic star wirelength. Currently we employ just five such iterations for each movable subset.

The movable subsets are obtained as segments of length 3 along a DFS vertex traversal of the netlist. The traversal begins with a vertex v such that the sum of the wirelength of the nets containing v is maximal. Although larger movable subsets may produce lower wirelengths at the given level, they do not appear to be cost-effective within the multilevel framework.

Because there are typically many fixed cells in E_M, the movable cells tend to remain separated and not move unreasonably far. However, some increase in overlap may be incurred, as there is currently no area-congestion modeling included in the subproblem objective. Following Mongrel [HL00], the movable cells are, therefore, not all moved to their relaxed locations right away. Instead, they are moved one at a time, and after each such move, the bin-based area density constraint(s) for the destination bin(s) are examined (a cell may be too large to fit in a single destination bin). If any such bin-density constraint is violated, its bin's area is reduced by cell-by-cell "ripple-move" propagation along monotone paths from overfull bins to underfull bins until feasibility is restored (Figure 3.1).

Currently, quadratic relaxation in mPL2 amounts simply to three cell displacements, each possibly followed by ripple-move corrections to area-congestion. Following each such subset relaxation, the net wirelength change is checked. If the relaxation increases the wirelength, its steps are reversed, and relaxation on the next movable subset in the DFS sequence of subsets proceeds. This monotone-nonincreasing-wirelength strategy was observed to produce better results in our multilevel implementation than the alternative FM-like hill-climbing strategy used in Mongrel [HL00]. In that approach, all relaxations are initially accepted, but at the end of the entire sweep, the configuration following the one that produced the least wirelength is restored.

For multilevel flow, instead of using one V-cycle optimization in mPL1, mPL2 uses two V-cycles. The placement from first V-cycle is used to guide the clustering scheme (4.3) in the second V-cycle.

Note that mPL1 and mPL2 can only handle circuit with uniform cell size because the relaxation scheme used only can handle uniform cell size placement. To handle the non-uniform standard cell size circuit (cells with same height but different width), large cells are broken into smaller cells so that the cell size variations is reduced. Artificial net with large net weight is used to connect the broken pieces of cells so that they stay close together during the optimization steps. The newer version is called mPL3.

An enhanced version of mPL3, called mPL4, uses more extensive multilevel flow. Two back-tracking V-cycles is used. Also, it incorporates a more effective detailed placement [LK03].

In the following chapter, a new powerful relaxation is presented. It is able to handle the non-uniform size standard cell placement and mixed-size placement. Enhanced versions of mPL are presented by incorporating the powerful optimization algorithm.

CHAPTER 5

Generalized Force-directed Algorithm

In this chapter, an effective constrained minimization formulation, an approximation of the placement problem, is proposed. It makes use of smooth accurate approximation to wirelength objective and smooth approximation of pairwise cells non-overlapping constraints. A nonlinear programming algorithm, named GFD algorithm, is presented to solve the problem. It is based on a well-known mathematical programming, Uzawa algorithm [AHU58], which is developed to solve constrained optimization problems.

Since one needs to solve the placement problem with up to millions of cells in fast runtime (close to linear complexity), several heuristic techniques are incorporated to sped up the convergence. The approximation algorithm for solving the constrained minimization problem can be considered a generalization of the force-directed method proposed in [EJ98]. These important techniques, contributed to the success of the GFD algorithm, are discussed in this chapter. Also, the GFD algorithm is embedded in multilevel framework for better scalability and better global optimization.

5.1 Constrained Minimization Problem Formulation

In this section we present a constrained minimization formulation to approximate the placement problem. We discuss smooth approximations to the wirelength objective (2.3) and smooth approximation to the pairwise cells non-overlapping constraint of the placement problem.

5.1.1 Smooth Wirelength Approximation

Since (2.3) is not differentiable and the constraints are highly non-convex, the minimizer is hard to locate. Therefore, using continuous differentiable functions to approximate (2.3) is necessary. Many studies, for example [KSJ91, EJ98, CCK00], use a quadratic function approximation given in (3.2). The advantage of using the quadratic wirelength objective is that its unconstrained minimizer can be obtained by solving a positive definite linear system of equations (cf. Theorem 2.1.1). However, it over-penalizes the long nets which gives a bad half-perimeter wirelength placement solution.

In this paper we use the following better half-perimeter wirelength approximation objective (3.5) proposed in [Na01] and recently used in [KW04, CCS05b], where the smaller η, the more accurate the approximation. However, η can not be chosen too small due to machine precision and numerical stability. In experiments, we scale the placement problem so that all the cell locations are between 0 and 1. η is then set to 0.01.

We have also proposed and studied another accurate approximation to (2.3) using L_p-norm:

$$\sum_{e\in E}((\sum_{v_k\in e} x_k^p)^{\frac{1}{p}} - (\sum_{v_k\in e} x_k^{-p})^{-\frac{1}{p}} + (\sum_{v_k\in e} y_k^p)^{\frac{1}{p}} - (\sum_{v_k\in e} y_k^{-p})^{-\frac{1}{p}}) \tag{5.1}$$

since the first term and the second term tend to $\max\{x_k\}$ and $\min\{x_k\}$ respectively as p tends to infinity. We set $p = 32$ in experiments so that x^p and $x^{\frac{1}{p}}$ can be computed efficiently. Numerical results verifying the effectiveness of different objectives are given in Table 6.8. We remark that a slightly different approxima-

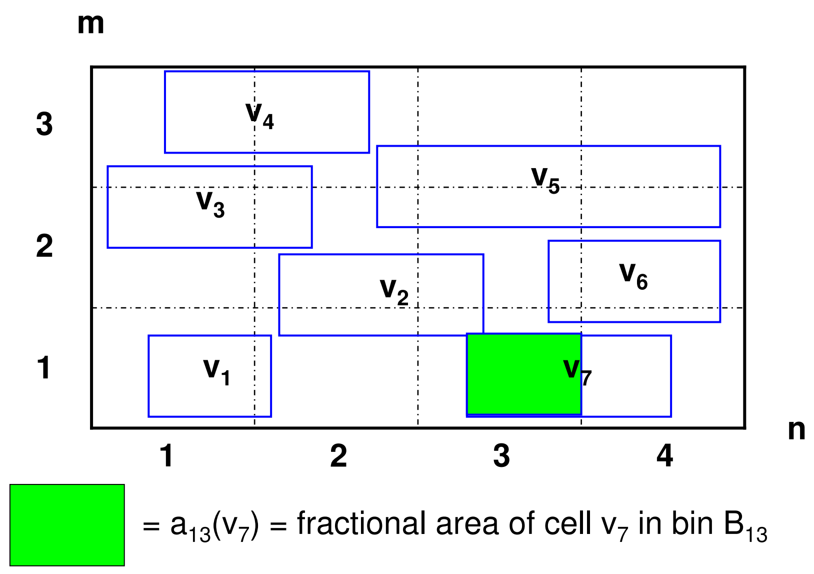

Figure 5.1: Illustration of fractional cell area in a 3x4 bins region.

tion using L_p-norm is proposed in [KM00] (cf. (3.4)).

5.1.2 Smooth Constraints Approximation

Since the pairwise cells non-overlapping constraints are highly non-convex and difficult to satisfy during the global placement, we relax the constraints by using the bin density constraints discussed in the following.

Based on the placement region R, we divide the region into $m \times n$ uniform non-overlapping sub-regions (bins) $B_{ij}, 1 \leq i \leq m, 1 \leq j \leq n$ such that $\cup_{i,j} B_{ij} = R$. Let h_x and h_y be the bin width and bin height respectively. Define D_{ij} to be the average density in the bin B_{ij} which is given by

$$D_{ij}(\vec{x}, \vec{y}) = \sum_{k=1} a_{ij}(v_k)/(h_x h_y), \tag{5.2}$$

where $a_{ij}(v_k)$ is the fractional area of cell v_k lying inside bin B_{ij} (see Figure 5.1).

We consider the constrained minimization problem:

$$\begin{aligned} \min \quad & W(\vec{x}, \vec{y}) \\ s.t. \quad & D_{ij} = K, \quad 1 \le i \le m, 1 \le j \le n \end{aligned} \tag{5.3}$$

where D_{ij} is the average density in B_{ij} defined through (5.2) and $K(\le 1)$ is the total cells area divided by the area of the placement region R. In the following discussions, we assume that $K = 1$, that is, total cells area equals the area of the placement region.

Note that in general we can have different density target $K_{ij}(\le K)$ for each bin B_{ij} to reflect uneven density requirement due to pre-placed blocks etc.. The uneven density requirements problem can be solved through the even density case if dummy fixed density F_{ij} is pre-occupied in each bin B_{ij} so that $K_{ij} + F_{ij} = K$. This can be proved in the following theorem.

Theorem 5.1.1 *A global optimal solution of the following constrained minimization problem*

$$\begin{aligned} \min \quad & W(\vec{x}, \vec{y}) \\ s.t. \quad & D_{ij} = K_{ij}, \quad 1 \le i \le m, 1 \le j \le n \end{aligned} \tag{5.4}$$

can be obtained from a global optimal of (5.3).

Proof. Let F_{ij} be the amount of fixed density added to each bin B_{ij} such that $F_{ij} + K_{ij} = K$. Then a global optimal solution of (5.3) is satisfying the constraints in (5.4). It is also the global optimal solution for (5.4) because any solution satisfying the constraints in (5.4) with smaller objective value contradicts the optimality of the solution we obtained in (5.3). $\square$

The current problem is to find a placement that minimizes the wirelength $W(\vec{x}, \vec{y})$ such that cells are evenly distributed over the region. However, it is difficult to solve the above problem since the density function is not differentiable. To make the problem easier to solve, we use the inverse Laplace transformation [Eva02] to smooth the density function. The smoothing operator $\Delta_\epsilon^{-1} d(x,y)$ is defined by solving the following Helmholtz equation:

$$\begin{cases} \Delta\psi(x,y) - \epsilon\psi(x,y) = d(x,y), \quad (x,y) \in R \\ \frac{\partial\psi}{\partial\nu} = 0, \quad (x,y) \in \partial R \end{cases} \tag{5.5}$$

where $\epsilon > 0$, ν is the outer unit normal, ∂R is the boundary of R, $d(x,y)$ is the continuous density function and Δ is a differential operator given by

$$\Delta \equiv \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}. \tag{5.6}$$

The inverse operator $\Delta_\epsilon^{-1} d(x,y)$ is well defined, as (5.5) has a unique solution for any $\epsilon > 0$ [Eva02]. Since the solution of (5.5) gains two more derivatives [Eva02] than $d(x,y)$, ψ is a smoothed version of the density function. On the other words, the solution must be smooth enough to satisfy (5.5) as it needs to be differentiated twice.

We use the finite difference method [MM94] to discretize the problem (5.5) using the bin grids we defined above. The Neumann boundary condition is used in the discretization scheme. Let $\psi_{i,j}$ be the value of ψ at the center of the bin B_{ij}. The approximation equations of (5.5) are given by

$$\begin{aligned} &\tfrac{\psi_{i+1,j} - 2\psi_{i,j} + \psi_{i-1,j}}{h_y^2} + \tfrac{\psi_{i,j+1} - 2\psi_{i,j} + \psi_{i,j-1}}{h_x^2} \\ &-\epsilon\psi_{i,j} = D_{ij}, \quad \forall\, 1 \le i \le m, 1 \le j \le n \end{aligned} \tag{5.7}$$

where

$$\begin{aligned} &\psi_{0,j} = \psi_{1,j} \quad \forall\, 1 \le j \le n \\ &\psi_{m+1,j} = \psi_{m,j} \quad \forall\, 1 \le j \le n \\ &\psi_{i,0} = \psi_{i,1} \quad \forall\, 1 \le i \le m \\ &\psi_{i,n+1} = \psi_{i,n} \quad \forall\, 1 \le i \le m, \end{aligned} \tag{5.8}$$

and D_{ij} is the average density in B_{ij}. Let $L_\epsilon[mn]$ be the matrix corresponding to the above linear system. Then $\vec{\Psi} = (\psi_{11}, \psi_{12}, \ldots, \psi_{mn})^T$ can be computed by solving the following linear system

$$L_\epsilon[mn]\vec{\Psi} = \vec{D}, \tag{5.9}$$

where $\vec{D} = (D_{11}, D_{12}, \ldots, D_{mn})^T$. Note that the problem (5.9) can be solved in $O(mn \log mn)$ by fast discrete cosine transform [CCN00]. The matrix $L_\epsilon[mn]$ can be diagonalized by discrete cosine matrix (2.1.4), that is,

$$L_\epsilon[mn] = (C[m] \otimes C[n])^T \Lambda_\epsilon[mn](C[m] \otimes C[n]), \tag{5.10}$$

where $C[n])$ is the $n \times n$ discrete cosine matrix (2.1.4) and $\Lambda_\epsilon[mn]$ is a diagonal matrix with diagonal entries being the eigenvalues of $L_\epsilon[mn]$. The eigenvalues of $L_\epsilon[mn]$ can be computed analytically (see Appendix A).

Since $(C(m) \otimes C[n])^{-1} = (C[m] \otimes C[n])^T$ and the multiplication $(C[m] \otimes C[n])\vec{D}$ or $(C[m] \otimes C[n])^T \vec{D}$ can be computed in $O(mn \log mn)$ time [SB93, FFT], the solution of (5.9) is given by

$$\vec{\Psi} = (C[m] \otimes C[n])^T \Lambda_\epsilon^{-1}[mn](C[m] \otimes C[n])\vec{D}, \tag{5.11}$$

which can be computed in $O(mn \log mn)$ time.

Now we can reformulate the problem (5.3) as

$$\begin{aligned} \min \quad & W(\vec{x}, \vec{y}) \\ s.t. \quad & \psi_{ij} = \bar{K}_\epsilon, \quad 1 \le i \le m, 1 \le j \le n \end{aligned} \tag{5.12}$$

where $\vec{\Psi} = L_\epsilon[mn]^{-1}\vec{D}$ and $\bar{K}_\epsilon \vec{1} = L_\epsilon[mn]^{-1} K \vec{1} = -K/\epsilon \vec{1}$ is a constant vector where $\vec{1} = (1, \ldots, 1)^T$. We have smooth objective function and constraints which enable (5.12) ready to be solved by standard mathematical programming. In the following section, we discuss how to solve the above problem (5.12).

5.2 Problem Solver

In this section, a generalization of the force-directed method [EJ98], called GFD algorithm, is presented to solve (5.12).

5.2.1 GFD Algorithm

There are many nonlinear programming techniques to solve (5.12). We use the Uzawa algorithm [AHU58] to solve (5.12). The advantage is that it does not require a Hessian inversion to find a minimizer satisfying the KKT condition [Ber82]. Another reason is that the iterative scheme can be viewed as a generalization of the force-directed method [EJ98]. An analysis of applying Uzawa algorithm to solve linear constrained quadratic programming problem is given in Appendix B.

By applying the Uzawa algorithm to solve (5.12), we get the following iterative scheme:

$$\begin{cases} \nabla W(\vec{x}^{k+1}, \vec{y}^{k+1}) + \sum_{i,j} \lambda_{ij}^{k} \nabla \psi_{ij} = 0 \\ \lambda_{ij}^{k+1} = \lambda_{ij}^{k} + \alpha(\psi_{ij} - \bar{K}_{\epsilon}) \end{cases} \tag{5.13}$$

where $\vec{\lambda}^k$ is the Lagrange multiplier at $k-$th iteration, α is a parameter to control the rate of convergence, and $\vec{x}^k$ and $\vec{y}^k$ are the cells center locations at the $k-$th iteration.

The gradient of ψ_{ij} with respect to the center location of cell v_k is approximated by the forward difference scheme [MM94]

$$\nabla_{x_k}\psi_{ij} = \frac{\psi_{i,j+1} - \psi_{i,j}}{h_x} \quad \text{and} \quad \nabla_{y_k}\psi_{ij} = \frac{\psi_{i+1,j} - \psi_{i,j}}{h_y} \tag{5.14}$$

if the center location of cell v_k is inside B_{ij} and zero otherwise. Note that using backward difference scheme or central difference scheme [MM94] instead of forward difference scheme (5.14) gives similar results.

In each step of the iterative scheme (5.13), we have to solve a nonlinear equation which can be solved by the time marching scheme [ROF92, AE04]. The solution of the nonlinear equation is a steady solution of the following ordinary differential equation (ODE):

$$\begin{cases} \begin{pmatrix} \frac{\partial \vec{x}(t)}{\partial_t} \\ \frac{\partial \vec{y}(t)}{\partial_t} \end{pmatrix} = -(\nabla W(\vec{x}(t), \vec{y}(t)) + \sum_{i,j} \lambda_{ij} \nabla \psi_{ij}) \\ (\vec{x}(0), \vec{y}(0)) \text{ is a given initial placement,} \end{cases} \tag{5.15}$$

where $(\vec{x}(t), \vec{y}(t))$ denotes the placement at time t. It can be considered a gradient descent scheme for the Lagrangian function

$$L(\vec{x}(t), \vec{y}(t), \vec{\lambda}) = W(\vec{x}(t), \vec{y}(t)) + \sum_{i,j} \lambda_{ij}(\psi_{ij} - \bar{K}_\epsilon) \tag{5.16}$$

since

$$\frac{\mathrm{d}L(\vec{x}(t), \vec{y}(t), \vec{\lambda})}{\mathrm{dt}} = -\|\frac{\partial L(\vec{x}(t), \vec{y}(t), \vec{\lambda})}{\partial \mathrm{t}}\|_2^2 < 0. \tag{5.17}$$

One can think of it as minimizing the wirelength objective and constraints penalty simultaneously at each iteration. We solve the above ODE by the explicit Euler method [MM94] which gives the following iterative scheme:

$$\begin{cases} \begin{pmatrix} \vec{x}^{k+1} \\ \vec{y}^{k+1} \end{pmatrix} = \begin{pmatrix} \vec{x}^{k} \\ \vec{y}^{k} \end{pmatrix} - \tau(\nabla W(\vec{x}^k, \vec{y}^k) + \sum_{i,j} \lambda_{ij} \nabla \psi_{ij}(\vec{x}^k, \vec{y}^k)) \\ (\vec{x}^0, \vec{y}^0) \text{ is a given initial placement,} \end{cases} \tag{5.18}$$

where $(\vec{x}^k, \vec{y}^k)$ are the locations of cells at $k-th$ step and τ is the time step. The time step τ has to be smaller enough to guarantee convergence. Analytical upper bound for τ is depended on the Hessian of the Lagrangian function (5.16) that is hard to be determined. In practical, an initial value of τ is determined. It is reduced by a constant ratio and the previous solution is restored if the iterative scheme (5.18) does not converge.

The algorithm we used to solve (5.12) is given in Table 5.1. It is called GFD (Generalized Force-directed) algorithm. The algorithm takes in the number of outer iterations and the stopping criterion for inner iteration. α is a parameter to speed up the convergence. μ is the increasing rate for α. κ is the percentage of the number of non-zero density bins for the stopping criterion. N is the set of movable cells, and P is the set of pads and fixed cells. Since one can only get a local minimizer by solving (5.13), the initial solution is important. The outer iterations can be considered a continuation method where the solution at each outer iteration is used as an initial solution for the next iteration.

We use uniform bin grids, and the number of bins is roughly equal to the number of cells. Since the global placement produced by the GFD algorithm contains cell overlapping, a discrete algorithm is used to legalize the solution. We use a simple effective greedy algorithm [LK03, CX06] to place the cells in standard rows without overlapping. Local greedy cell swapping, where each move does not create overlapping, is then applied to reduce wirelength.

5.2.2 Comparisons with APlace and Kraftwerk

In this section, we compare GFD algorithm with other famous analytical placers APlace [KW04] and Kraftwerk [EJ98].

APlace [KW04] is an analytical placer based on [Na01]. The problem formulation considered in [KW04] is the same as (5.3). It uses a bell shape function [Na01] to smooth the density constraint locally. In our case, however, the inverse Laplace transformation (5.5) smoothes the density function globally. APlace uses a penalty method utilizing nonlinear conjugate gradient algorithm to solve the smoothed version of (5.3).

Kraftwerk is an analytical placement algorithm utilizing the force-directed

```
GFD($outer\_iters$, $stop\_percent$)
if initial placement not given
  use the unconstrained minimizer of the quadratic
  wirelength objective as an initial solution.
endif
compute $nnb$ = number of non-zero density bins.
set $P$ = the set of pads and fixed cells.
set $M$ = the set of movable cells.
set $inner\_iters = |M|$.
set $\mu = 1.5$. (Experiments show that it is a good
trade-off between runtime and wirelength)
for $i = 1$ to $outer\_iters$
  set $\alpha = \frac{\sqrt{\max\{|P|,1\}}}{h_x h_y \log |M|}$.
  $\kappa = \min\{\frac{100i}{outer\_iters}, stop\_percent\}$.
  $\vec{\lambda} = 0$.
  for $j = 1$ to $inner\_iters$
    if $nnb$ not increased
      $\alpha = \mu\alpha$.
    endif
    $\vec{\lambda} = \vec{\lambda} - \alpha(\vec{\Psi} - \vec{K}_\epsilon)$.
    solve the ODE (5.15) by explicit Euler method (5.18).
    compute $nnb$.
    if more than $\kappa$% non-zero density bins
      break.
    endif
  endfor
call detailed placement.
endfor
```

Table 5.1: GFD algorithm.

method proposed in [EJ98]. In [EJ98], it derives that the divergence of the forces $\vec{f}(x,y) = (f_1, f_2)^T$ is proportional to the density; that is,

$$\frac{\partial f_1}{\partial x} + \frac{\partial f_2}{\partial y} = c \cdot d(x,y), \tag{5.19}$$

where c is a constant. Also, there exists a scalar function $\phi(x,y)$ satisfying

$$\nabla \phi(x,y) = \vec{f}(x,y). \tag{5.20}$$

Combining (5.19) and (5.20) gives the following equation

$$\Delta \phi(x,y) = c \cdot d(x,y) \tag{5.21}$$

with boundary conditions that the magnitude of the forces $\nabla \phi(x,y)$ is zero at infinity.

Comparing (5.5) with (5.21), the main difference is the boundary condition if we choose small ϵ. The boundary condition in our formulation (5.5) tells that the forces pointing outside the boundary are zero, which makes more sense than assuming the forces being zero at infinity as in [EJ98] since we want to place the cells inside a finite region.

Moreover, the force-directed method in [EJ98] can be considered a special case of (5.13). It uses the quadratic wirelength objective (3.2) for $W(\vec{x},\vec{y})$ and iteratively solves

$$\begin{pmatrix} C & 0 \\ 0 & C \end{pmatrix} \begin{pmatrix} \vec{x}^{k+1} \\ \vec{y}^{k+1} \end{pmatrix} + \begin{pmatrix} \vec{p}_x \\ \vec{p}_y \end{pmatrix} + \tau_k \begin{pmatrix} \vec{f}_x^k \\ \vec{f}_y^k \end{pmatrix} = 0 \tag{5.22}$$

until all cells are well distributed over the chip region. C, $\vec{p}_x$ and $\vec{p}_y$ are derived from $\nabla W(\vec{x},\vec{y})$. The τ_k is a scalar to control the movement of cells in each iteration. The horizontal force $\vec{f}_x^k$ and the vertical force $\vec{f}_y^k$ acting on the cells are given by $\sum(\nabla_{x_1}\phi_{ij}, \ldots, \nabla_{x_N}\phi_{ij})^T$ and $\sum(\nabla_{y_1}\phi_{ij}, \ldots, \nabla_{y_N}\phi_{ij})^T$ respectively computed based on the placement solution at the $k-$th iteration. Clearly, this is a particular case of (5.13) by setting $\lambda_{ij}^k = \tau^k$. One can expect the above fixed point iteration requiring a small enough τ_k for convergence. But we know $\vec{\lambda}^k$ is the Lagrange multiplier for (5.12) which has to be large enough to get a well-distributed placement. Also, the $\vec{\lambda}^k$ is a vector in (5.13) and has each of its component acting as a scaling factor for the forces induced in the corresponding bins. These show that our new algorithm is more general and robust, and, overcomes the shortcoming of ad hoc force scaling selection used in [EJ98].

5.3 Analysis and Enhancements of the GFD Algorithm

In this section, analysis and enhancements of the GFD algorithm are presented. Due to the high complexity of the placement problem: up to millions of cells to be placed subject to pairwise cells non-overlapping constraints, heuristic approximations or approaches are necessary in building a highly scalable placement algorithm. Several important heuristic enhancement techniques to the GFD algorithm are discussed in the following, which makes the GFD algorithm much more robust and stable.

5.3.1 Multilevel Implementation

It is undoubted that the constrained minimization problem (5.12) has a lot of local minimas. For a good local optimal solution to be found by the nonlinear programming GFD algorithm, a good initial solution is necessary. The multilevel optimization framework, presented in Chapter 4, not only provides a scalable framework but also is able to provide such a good initial placement for the GFD algorithm.

By recursive coarsening of the problem such that the coarsest level problem is smaller enough for the GFD algorithm to find a good solution, the coarsest level good placement solution is then recursively interpolated and iteratively improved to finer levels. For a good multilevel framework, one should be able to obtain a good approximation solution for the original problem through coarse levels refinement. It then provides a good starting point for the GFD algorithm at solving the original problem.

In our multilevel framework, the cluster ratio ranges from 0.1 to 0.4, where cluster ratio is the number of clusters created at the current level divided by the number cells at the finer level. The coarsest level problem size is less than around 1% of the original problem size or 500, depending which one is smaller.

Figure 5.2 shows an example of multilevel optimization framework making use of the GFD algorithm. It is a two V-cycle placement and each V-cycle consists of three levels. Level 3 in the figure represents the original problem and Level 1 represents the coarsest level problem. In practice, the relaxation at the finest level in the first V-cycle is skipped to reduce the computations, which is found to be a good trade-off between quality and speed. A more detailed algorithm is shown in Table 5.2. It is named mPL5.

A fast mode of mPL5, named mPL5-fast, is obtained by:

- set the $stop\_percent = 95$;
- increase α in the GFD algorithm whether nnb is increased or not;
- reduce the number of bins to half of the default;
- reduce the amount of cell swapping in the detailed placement.

```
use modified FC (cf. (4.2)) to coarsen the hypergraph,
with cluster ratio 0.4, until the number of cells reaches target.
set $nl$ = number of levels.
set $stop\_percent = 97$
% suppose level $nl$ is the finest level corresponding
% to the original hypergraph.
for $i = 1$ to $nl - 1$
    set $distri\_percent = min(50 + 50 * i/nl, 90)$.
    call GFD$(1, distri\_percent)$ for level $i$.
    interpolate placement from level $i$ to level $i + 1$.
endfor
% start the second V-cycle.
use modified geometric based FC (cf. (4.3))
to coarsen the hypergraph until the number of
cells reaches target.
placement from first V-cycle is interpolated to coarse
levels during the coarsening.
set $nl$ = number of levels.
for $i = 1$ to $nl - 1$
    set $distri\_percent = min(50 + 50 * i/nl, 90)$.
    call GFD$(1, distri\_percent)$ for level $i$.
    interpolate placement from level $i$ to level $i + 1$.
endfor
call GFD$(1, stop\_percent)$ for level $nl$.
call detailed placement.
```

Table 5.2: mPL5 algorithm.

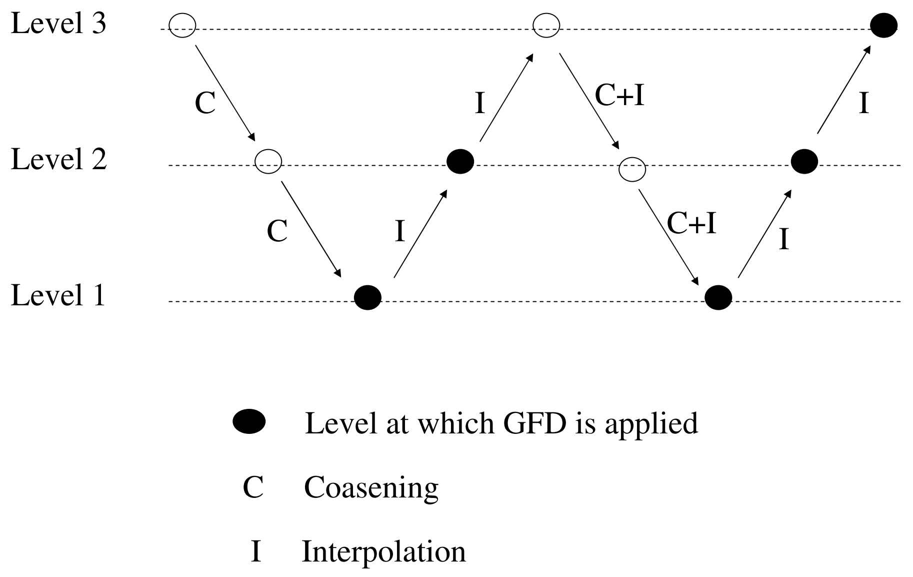

Figure 5.2: Two V-cycle multilevel framework makes use of GFD algorithm.

5.3.2 Effects of Density Smoothing

Density based constraint is used to approximate the pairwise cells non-overlapping constraint which significantly reduces the $O(n^2)$ complexity due to the pairwise cells non-overlapping constraints. It is because one can use $O(n)$ number of bin based density constraints which makes the algorithm more scalable.

However, density function is not differentiable which makes the problem (5.3) difficult to be solved by standard mathematical programming. Smooth approximation (5.5) of the density function is necessary. The ϵ in (5.5) plays an important role in the smoothing process. It not only makes the smoothing process well defined but also controls the smoothness of the smoothed density function. In figure 5.3, it shows the density function of an one dimension placement and the smoothed density functions under $\epsilon = 10, 1, 0.1$. We see that the smaller the ϵ the more global smoothing or smoother of the density function. In experiments, we observe that the larger the ϵ the slower the convergence of the

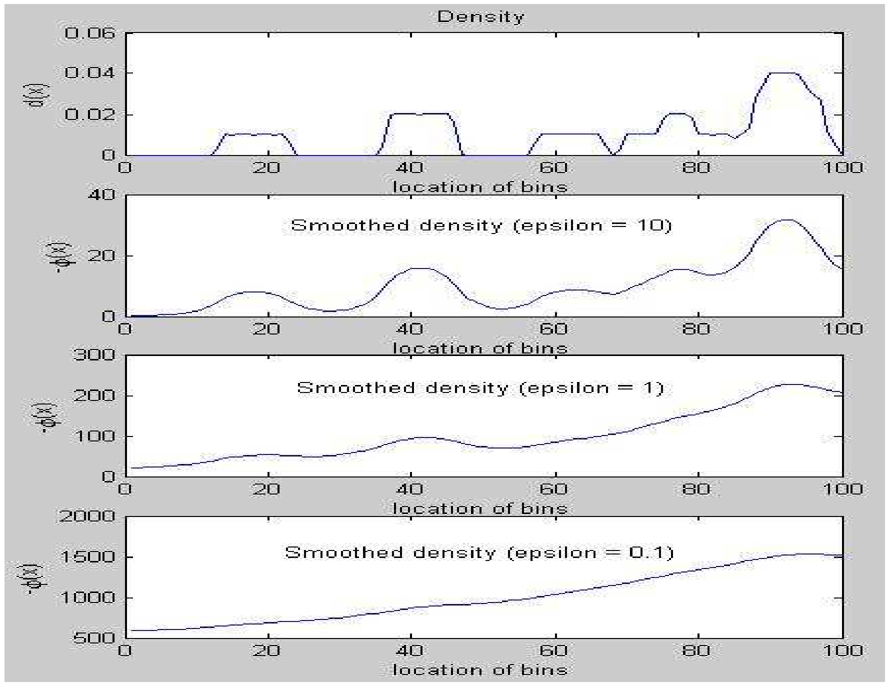

Figure 5.3: Smoothness of the density function under different epsilon (ϵ) = 10, 1, 0.1.

GFD algorithm. The reason is that the larger the ϵ the less smoothness of the density function causes forces acting on cells smaller and more local, leading to slowly cells spreading. Hence, the choice of smaller ϵ gives faster convergence, however, at the expense of worse wirelength quality. It certainly provides the freedom for the trade-off between speed and quality of the placement algorithm. In experiments, $\epsilon = 1$ is found to be a good trade-off.

5.3.3 Forces Weighting

The gradients of the constraints with respect to cell center locations are approximated by (5.14) which not only speeds up the computations but also leads to faster convergence of the algorithm. The approximations (5.14) are viewed as cells spreading forces in [EJ98]. Recently, a more stable implementation [VKV04] of

[EJ98] shows that weighting of the forces acting on cell by the corresponding cell area gives more stable spreading of the cells. The intuition is that each unit area of the cell should be getting the forces. Similar ideas are tried in our approximations (5.14). A more general form to approximate the gradient of the constraint with respect to cell v_k is given by

$$\nabla_{x_k}\psi_{ij} = \frac{\psi_{i,j+1} - \psi_{i,j}}{h_x} * w(\text{area}(v_k)) \quad \text{and} \quad \nabla_{y_k}\psi_{ij} = \frac{\psi_{i+1,j} - \psi_{i,j}}{h_y} * w(\text{area}(v_k)) \tag{5.23}$$

if the center of cell v_k is inside B_{ij} and zero otherwise, where $w(x)$ is a monotone increasing function and area(v_k) denotes the area of cell v_k. In experiments, $w(x) = cx^{0.8}$ is used. The weighted approximation (5.23) significantly improves the quality of placement, especially for the circuits with huge variations in cell area. Also, the scaler $c(> 0)$ in $w(x)$ provides the freedom for trade-off between speed and quality since the larger the c, the faster the convergence or spreading of cells.

5.3.4 Gradual Legalization and Fixing of Large Cells

The weighting of forces (cf. (5.23)) does improve the placement quality, however, it may cause instability for the spreading of large cells when most of the cells are well spread over the placement region. It is because (5.23) shows that larger cells have larger scaler for the forces acting on them which may cause large perturbations of the large cells. It is a good idea to fix the large cells when they are well placed. By fixing the large cells, it means that those large cells are first legalized [CX06] (no overlapping between those large cells) then their positions are fixed during the consequent placement.

In the multilevel framework, the placement solutions at coarse levels provide good intermediate steps for fixing the relatively large cells. Note that similar ideas

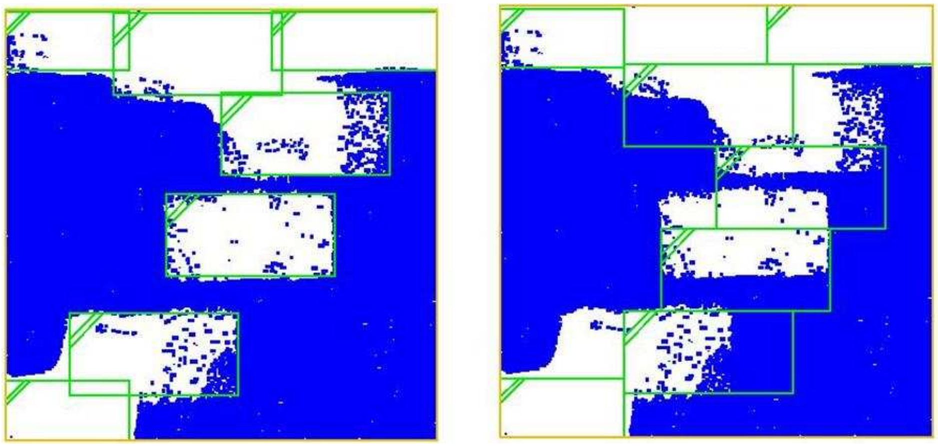

Figure 5.4: Global placement is shown in the left figure and the placement after legalization of macros is shown in the right figure.

of fixing the large cells during coarse level placement has been used in [CCY03]. Fixing large cells during coarse level not only makes the GFD algorithm more stable but also accelerates the convergence as it creates more connections to fixed cells.

Moreover, earlier legalization of the large cells also helps for detailed placement [CCS05a]. In Figure 5.4, the left placement plot shows the global placement without fixing the large cells during the placement. The small amount of overlapping between large cells, that is allowed for a global placement, could create troubles for detailed placement. The right-hand side placement plot in Figure 5.4 shows that huge perturbations of the larges cells are produced after legalization which then causes huge perturbations during legalization of standard cells. Those huge perturbations created after legalization have no doubt significantly increased wirelength of the placement.

In experiments, macros with area $100X$ larger than the average cell area are fixed before GFD algorithm is applied during intermediate coarse levels (except

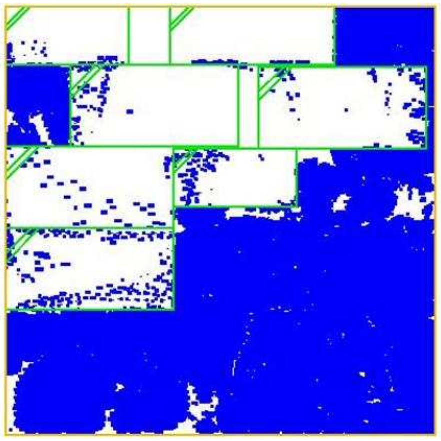

Figure 5.5: Global placement with macros legalized and fixed after coarse levels placement.

the coarsest level). Hence, all relatively large macros are fixed right before the placement of the finest level that gives the room for standard cells to be placed adaptly to the fixed cells.

5.3.5 Wirelength Weighting

Proper weighting for spreading forces is seen to be crucial. However, it may create significant degradation of placement quality in some cases. From the GFD algorithm (cf. Table 5.1) and the iterative scheme (5.18), the scaling factor α for the forces $\sum\limits_{i,j} \lambda_{ij} \nabla \psi_{ij}$ is kept increasing when cells are not spreading. The τ in the iterative scheme (5.18) is kept decreasing when the scheme does not converge. In the case when τ is too small and force scaler α is too large, the term $\nabla W(\vec{x}^k, \vec{y}^k)$ in (5.18), which corresponding to the wirelength objective, is diminishing. This causes the degradation of the placement quality. Hence weighting for the term

$\nabla W(\vec{x}^k, \vec{y}^k)$ in (5.18) is necessary which gives the following iterative scheme:

$$\begin{cases} \begin{pmatrix} \vec{x}^{k+1} \\ \vec{y}^{k+1} \end{pmatrix} = \begin{pmatrix} \vec{x}^{k} \\ \vec{y}^{k} \end{pmatrix} - \tau(\beta \nabla W(\vec{x}^k, \vec{y}^k) + \sum_{i,j} \lambda_{ij} \nabla \psi_{ij}(\vec{x}^k, \vec{y}^k)) \\ (\vec{x}^0, \vec{y}^0) \text{ is a given initial placement,} \end{cases} \tag{5.24}$$

where $\beta(> 0)$ is kept increasing at the outer iterations of the GFD algorithm (cf. Table 5.3). Experiments show that the wirelength weighting has significant improved the placement wirelength in many testcases, though at the expense of longer runtime. It is because the weighting of the wirelength causes more attractions between cells, hence causes slow down of cell spreading.

5.3.6 Pin-to-Pin Half-perimeter Wirelength Minimization

The half-perimeter wirelength in (2.3) is measured in terms of center locations of the cells. That is, assuming all the pins belonging to a cell are located on the center of the cell. In real situation, pins of a cell are usually located on the boundary of the cell (cf. Figure 2.2). Therefore, the center-to-center half-perimeter wirelength (2.3) may not a good approximation of the pin-to-pin wirelength (2.2). Since each net connects a set of unique pins, it is more costly to minimize (2.2). However, if (2.2) is approximated by log-sum-exp function (3.5), the complexity of minimizing center-to-center half-perimeter wirelength is the same as minimizing pin-to-pin half-perimeter wirelength. Thank to the equality $\exp(x_i + p^e_{x_i}) = \exp(p^e_{x_i}) \exp(x_i)$, $\exp(p^e_{x_i})$ is a constant that can be computed before the optimization steps. Similarly thing is done for $y-$direction.

5.3.7 Whitespace Handling by Filler Cells

In the formulation (5.3), we have assumed the placement region has zero whitespace, that is, $K = 1$. In practical, there is usually 10-40% whitespace which is

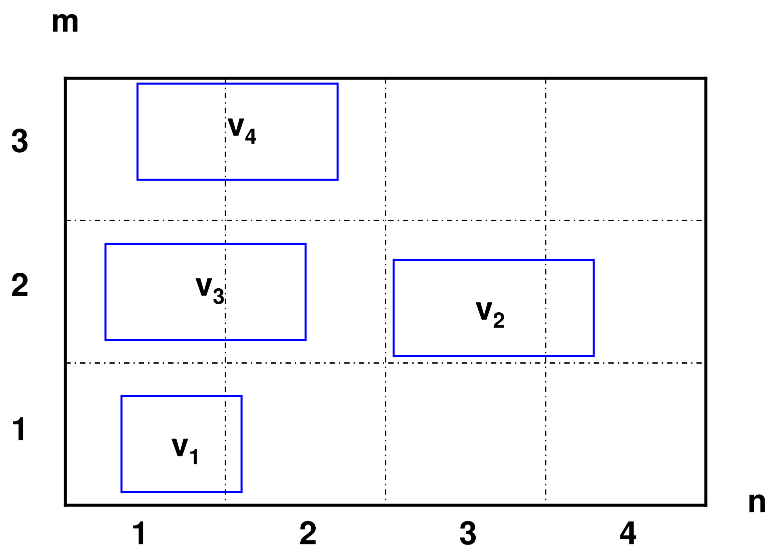

Figure 5.6: A small example of circuit with low utilization.

reserved for post placement purpose. Also, cells can not be packed too closely. Sufficient space around cells for putting the wires is necessary for a circuit to function properly. However, when K is significantly less than one, it would cause trouble in the GFD algorithm.

Figure 5.6 shows a placement problem with total cells area much less than the placement region. One can easily see that the density equality cannot be satisfied because the cells cannot be broken into pieces. Due to the forces driven by density constraint, cells are spread to occupy the whole placement region which causing over separations between cells and hence the degradation of the wirelength. Figure 5.7 shows the global placement on a testcase with 60% whitespace. We can see that the cells are placed evenly through out the region which maybe over spread.

One simple fix is to shrink the placement region, from either directions of the

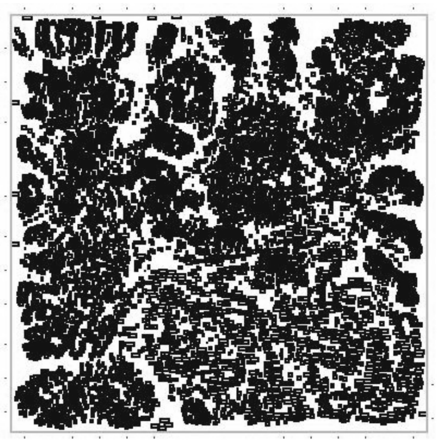

Figure 5.7: Global placement on a testcase (#cells = 12506) with utilization = 0.4. The half-perimeter wirelength is 2.2024×10^6.

chip boundary, so that the utilization ratio is close to one. Figure 5.8 shows the global placement of the same testcase as used in Figure 5.7. During the placement, placement region is shrunken by same amount from both left and right boundary such that the utilization is one during the placement. It significantly improves the wirelength, near 13%. However, it is easy to see that shrinking the region or trim away the excess whitespace has certainly limited the solution space. Also, there are several other directions to shrink the region. Picking the best from the placement solutions produced by different ways of region shrinking is not cost effective.

A reasonable way to handle the whitespace is to reformulate (5.3) as an inequality constrained minimization problem:

$$\begin{aligned} \min \quad & W(\vec{x}, \vec{y}) \\ s.t. \quad & D_{ij} <= 1, \quad 1 \le i \le m, 1 \le j \le n. \end{aligned} \tag{5.25}$$

However, the Uzawa algorithm [AHU58] is developed for solving equality constrained optimization problem. The GFD algorithm (cf. Table 5.1), based on the Uzawa algorithm, cannot be applied to solve (5.25). Whitespace handling by

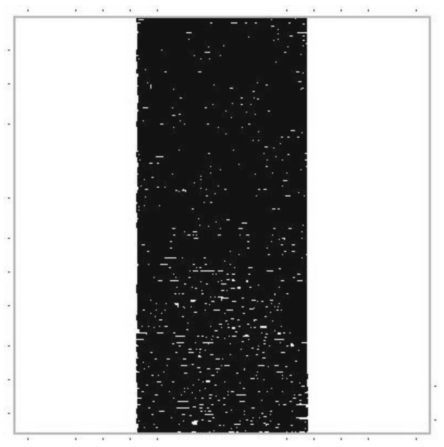

Figure 5.8: Global placement on a testcase (#cells = 12506) with utilization = 0.4, with placement region shrunken from left and right boundary such that the utilization = 1 during the placement. The half-perimeter wirelength is 1.9250×10^6.

adding extra dummy cells or filler cells to the netlist is proposed in [AMV03] for min-cut based placement algorithms. The idea gives a way to transform (5.25) into an equality constrained minimization problem. A set of artificial cells, named dummy/filler cell, with total area summed up to the total amount of whitespace is added to the netlist. Hence, the utilization is one and the problem becomes an equality constrained minimization problem which is ready to be solved by the GFD algorithm.

Since there are no nets connecting dummy cells, their movements are driven by density only (cf. (5.18)). Initial placement of dummy cells are based on the density distribution of the initial placement of the cells from the original netlist. Recursively four way partition hierarchical distribution of the dummy cells is used. Number of dummy cells assigned to each partition are proportional to the whitespace available for the region. The idea is to place dummy cells to low density regions initially. In addition to using forces to drive the dummy cells, re-

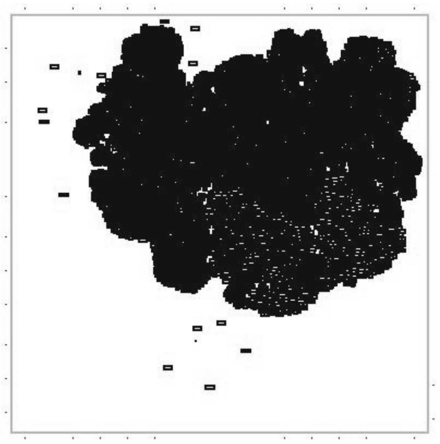

Figure 5.9: Global placement on a testcase (#cells = 12506) with utilization = 0.4, with dummy cells added such that the utilization = 1 during the placement. The half-perimeter wirelength is 1.7203×10^6.

distribution of dummy cells is carried out during intermediate stage of placement. This is to avoid dummy cells occupying the regions where the cells can move in to get a better placement wirelength. Figure 5.9 shows the global placement by using dummy cells to handle whitespace. It significantly improves the wirelength by 12% over the method of region shrinking.

5.3.8 Stopping Criterion of GFD

The stopping criterion in the GFD algorithm (cf. Table 5.1), measured in terms of the percentage of the number of non-zero density bins, is not robust enough to guarantee a decent global placement, especially in the case where there are many large cells. Since each large cell automatically covers a significant number of bins, the overlapping between small cells can be a lot even when each bin has non-zero density. Also, it is not able to achieve the stopping criterion if the circuit has low utilization (see Figure 5.6). A better stopping criterion is proposed in this section.

The new stopping criterion for a global placement is measured in terms of the average bin overflow. The amount of overflow for a bin is the amount of bin density exceeding the bin capacity. The average overflow for a set of cells V_c is defined as

$$\mathrm{OVL}(V_c) = \sum_{i=1}^{m}\sum_{j=1}^{n} \max(D(V_c)_{ij} - 1, 0) h_x h_y / \mathrm{area}(V_c), \tag{5.26}$$

where $D(V_c)_{ij}$ is the density of ij-th bin and the density function is measured in terms of the area of the cells in V_c, area(V_c) is the total area of the cells in V_c. Under sufficiently small size of bins and small overflow, the global placement is guaranteed little overlapping between cells and can be legalized without much increase in wirelength.

5.3.9 Minimum Perturbation Formulation

It is expected that the GFD algorithm is in slow convergence when the cells are well distributed. Slow reduction of overflow (cf. (5.26)) happens during the last few steps of the GFD algorithm. It is because the forces acting on cells diminish when the density at each bin becomes more even. To sped up the convergence and to reduce computations, (5.12) is reformulated as

$$\begin{aligned} \min \quad & \sum_{v_i \in M} \{(x_i - x_i^o)^2 + (y_i - y_i^o)^2\} \\ s.t. \quad & \psi_{ij} = \bar{K}_\epsilon, \quad 1 \le i \le m, 1 \le j \le n, \end{aligned} \tag{5.27}$$

where M is the set of movable cells and (x_i^o, y_i^o) is the initial center location of cell v_i. It is to minimize the displacement from a given placement subject to the density constraint. The objective in (5.27) is easier and faster to minimize comparing to (3.5). It is cost effective to switch to solve (5.27) when slow convergence happens in the GFD algorithm. The GFD algorithm can be terminated earlier and the placement solution is a good starting point for (5.27). It also gives a more smooth transaction from global placement to detailed placement. Experiments show that it is a good trade-off between quality and speed.

5.3.10 Enhanced GFD Algorithm

In this section, an enhanced GFD algorithm, combined the enhancement techniques in the previous sections, is presented. Table 5.3 shows the new GFD algorithm combined the enhancement techniques, named EGFD.

In Table 5.3, P is the set of pads, F is the set of fixed cells, M is the set of movable cells not including dummy cells, and H is the set of dummy cells added to the netlist (cf. Section 5.3.7). In EGFD, it takes in a percentage of overflow (cf. (5.26)) as stopping criterion. The algorithm terminates when both overflow of movable cells and overflow of total cells (excluding pads and dummy cells) less than a target overflow.

Similar to the GFD algorithm (Table 5.1), it uses the unconstrained minimizer of the quadratic wirelength (3.2) as the starting point if an initial placement is not given. In EGFD, the scaling factor α for forces is guided by the overflow of the movable cells instead of the number of non-zero density bins used in GFD. Also, there is a scaling factor β for the wirelength term in the iterative scheme (5.24). In addition, dummy cells are added to the netlist and the distribution of dummy cells is done at every 3% reduction of the overflow of the movable cells. It proceeds to solve (5.27) when the overflow is 5% away from the $stop\_percent\_ovl$.

A multilevel framework utilizes the EGFD algorithm is done in a similar way as shown in Table 5.2. The multilevel algorithm using EGFD as relaxation is named mPL6. The mPL6 algorithm is shown in Table 5.4. It uses Best Choice clustering scheme [AKN05] with cluster ratio 0.25. Since Best Choice is a better clustering scheme, usually one V-cycle with cluster ratio 0.25 is enough to get a decent placement. mPL6 uses one V-cycle instead of two V-cycles used in mPL5. Also, relatively large macros are fixed after the coarse levels placement. Experimental results of different versions of mPL are given in the following chapter.

5.4 Our Contributions

The novel analytical placement algorithm, EGFD, is based on a mathematically sound foundation for supporting the density constraint, and can be viewed as a generalization of the force-directed method in [EJ98]. In [EJ98], it uses a quadratic wirelength objective and adds forces on cells based on area density of the placement. Adding forces on cells is equivalent to modifying the right-hand size of the linear system arising from the quadratic wirelength minimization. Hence each iteration can be solved easily. However, it suffers from the inaccurate approximation by quadratic wirelength objective as illustrated in [KM00, KSJ91, MAN94] and the ad hoc scaling of the spreading forces for supporting the density constraint.

The new contributions and enhancements presented in this thesis are as follows:

- We develop a new analytical placement algorithm using a density constrained minimization formulation which can be viewed as a generalization of the force-directed method in [EJ98].

- We analyze and identify the advantages of our new algorithm over the force-directed method in [EJ98].

```
EGFD($stop\_percent\_ovl$)
set $P$ = the set of pads.
set $F$ = the set of fixed cells inside the placement region.
set $M$ = the set of movable cells.
set $H$ = the set of dummy cells added to the netlist.
set $\eta = 100$.
set $\mu = 1.5$.
set $\alpha = \frac{\sqrt{\max\{|P|+|F|,1\}}}{h_x h_y \log|M|}$.
set $\beta = 1$.
set $\kappa = stop\_percent\_ovl/100$.
set $\lambda = 0$.
if initial placement not given
  use the unconstrained minimizer of the quadratic
  wirelength objective as an initial solution.
endif
Hierarchical distribution of dummy cells.
set $old\_std\_ovl = curr\_std\_ovl = \mathrm{OVL}(M)$.
for $j = 1$ to $|M|$
  $\lambda = \lambda - \alpha(\psi - K_\epsilon)$.
  solve the iterative scheme (5.24).
  set $old\_std\_ovl = curr\_std\_ovl$.
  set $curr\_std\_ovl = \mathrm{OVL}(M)$.
  if $curr\_ovl \leq \kappa$ and $\mathrm{OVL}(M \cup F) \leq \kappa$
    break.
  endif
  if $curr\_std\_ovl \geq old\_std\_ovl$
    $\alpha = \mu\alpha$.
  endif
  $\beta = \beta + \eta$.
  redistribution of dummy cells for every 3% reduction of OVL($M$).
endfor
```

Table 5.3: Enhanced GFD algorithm.

```
use Best Choice (cf. (4.4)) to coarsen the hypergraph,
with cluster ratio 0.25, until the number of cells reaches target
set $nl$ = number of levels created.
set $stop\_percent\_ovl = 15$
% suppose level $nl$ is the finest level corresponding
% to the original hypergraph.
for $i = 1$ to $nl - 1$
    call EGFD($stop\_percent\_ovl + 5$) for level $i$.
    interpolate placement from level $i$ to level $i + 1$.
    legalize and fix macros with area > 4X the average area at level $i$.
endfor
call EGFD($stop\_percent\_ovl + 5$) for level $nl$.
call EGFD($stop\_percent\_ovl$) by minimizing (5.27) for level $nl$.
call detailed placement.
```

Table 5.4: mPL6 algorithm.

• We successfully incorporate the generalized force-directed algorithm into a multilevel framework which significantly improves the wirelength and speed. We use the multilevel framework proposed in mPL [CCK03b]. The new algorithm is named mPL5 (cf. Table 5.2).

• We carefully analyze and enhance our placement algorithm and propose several important enhancement techniques which make our placement algorithm mPL6 (cf. Table 5.4) much more stable and robust (cf. Chapter 6).

CHAPTER 6

Experimental Results

In this chapter, experimental results of different versions (2–6) of mPL (can be downloaded at [UCL]) are presented and discussed. Comparisons with other state-of-the-art placers are given. Comparisons are based on half-perimeter wirelength (HPWL or WL) of legal placement solution (unless specified in the context) and runtime (time) of the algorithm. GP HPWL is stand for the half-perimeter wirelength of the global placement solution.

Stable and good performances on wide varieties of circuits are necessary for good heuristic algorithms. Six set of benchmarks, from synthetic circuit design to real industrial circuit design, are used to evaluate the performances of our proposed placement algorithms. The statistics of the circuit testcases are given in the following section.

6.1 Circuit Benchmark Statistics

In this section, six sets of benchmark: ICCAD00, ISPD04, ICCAD04, FARADY, ISPD05, and PEKO are presented. A table summarizing the circuit statistics of each benchmark is given. In each table, we show the circuit name, the number of pads (#Pads), the number of standard cells (#Std. cell), the number of macros (#Macro), the number of fixed macros (#Fixed macro), the number of nets (#Net), and the percentage of whitespace (%WS).

Circuit	#Pad	#Std. cell	#Macro	#Fixed macro	#Net	%WS
ibm04	287	27335	0	0	31970	5
ibm07	287	45789	0	0	48117	5
ibm09	285	53271	0	0	60902	5
ibm10	744	68869	0	0	75196	5
ibm14	517	147358	0	0	152772	5
ibm16	504	183281	0	0	190048	5
ibm17	743	185054	0	0	189581	5
ibm18	272	210664	0	0	201920	5

Table 6.1: ICCAD00 Benchmark Statistics.

ICCAD00 is a set of standard cell circuit with uniform cell size. The netlist is the same as the ISPD98 benchmark released in [Alp98]. The cells size are modified to uniform size. The circuit statistics are shown in Table 6.1. Total number of cells ranges from 20k to 210k. Each testcase has around 5% whitespace. The benchmark can be downloaded at [UCL].

ISPD04 is a set of standard cell circuit. The circuit statistics are shown in Table 6.2. The set of benchmark is the same as that used in [VC04], and is provided by the authors of FastPlace1.0 [VC04]. It is originally derived from the ISPD02 suite [AM02] downloaded from [ISPa]. The macros are modified to be standard cells in a way that the height of macro blocks is brought down to the standard cell height and the width of macro blocks, if exceeding 4X average width, is changed to a value of 4X average width. Each testcase has around 10% whitespace. Total number of cells ranges from 10K to 210K. The ISPD04 benchmark can be downloaded at [ISPb].

ICCAD04 is a set of mixed-size benchmark where each testcase may have

Circuit	#Pad	#Std. cell	#Macro	#Fixed macro	#Net	%WS
ibm01	246	12506	0	0	14111	10
ibm02	259	19342	0	0	19584	10
ibm03	283	22853	0	0	27401	10
ibm04	287	27220	0	0	31970	10
ibm05	1201	28146	0	0	28446	10
ibm06	166	32332	0	0	34826	10
ibm07	287	45639	0	0	48117	10
ibm08	286	51023	0	0	50513	10
ibm09	285	53110	0	0	60902	10
ibm10	744	68685	0	0	75196	10
ibm11	406	70152	0	0	81454	10
ibm12	637	70439	0	0	77240	10
ibm13	490	83709	0	0	99666	10
ibm14	517	147088	0	0	152772	10
ibm15	383	161187	0	0	186608	10
ibm16	504	182980	0	0	190048	10
ibm17	743	184752	0	0	189581	10
ibm18	272	210341	0	0	201920	10

Table 6.2: ISPD04 Benchmark Statistics.

both standard cells and macros. The benchmark is used in [ACJ04] and can be downloaded at [ICC]. The netlist connectivity is similar to ISPD04. The circuit statistics are given in Table 6.3. Each testcase has around 20% whitespace. Total number of cells ranges from 10K to 210K.

FARADY is also a set of mixed-size benchmark where each testcase may have both standard cells and macros. The benchmark is used in [ACJ04] and can be downloaded at [ICC]. The netlist connectivity is derived from industrial designs. Testcase statistics are shown in Table 6.4. Whitespace percentage for each testcase ranges from 5% to 10%. Total number of cells ranges from 10K to 210K.

ISPD05 is a set of mixed-size benchmark where each testcase may have both standard cells and macros. The majority of the macros in this benchmark are fixed. The benchmark is used in ISPD'05 Placement Contest [NAV05] and can be downloaded at [ISPc]. Testcase statistics are shown in Table 6.5. The netlist connectivity is derived from industrial designs. Whitespace percentage for each testcase ranges from 14% to 46%. Total number of cells ranges from 210K to 2200K.

PEKO, stand for placement examples of known optimal, is developed in UCLA VLSICAD [UCL]. It is a set of synthetic benchmark designed to evaluate the optimality gap of a placement algorithm. The set of PEKO benchmark used in this thesis is a set of standard cell circuit with connections to pads, classified as suiteIII in [UCL]. Testcase statistics are similar to Table 6.6 since the netlist is derived from ISPD02. The netlist connectivity is derived such that each net is constructed locally optimal which also implies global optimal [CCX03]. Each testcase has around 15% whitespace. Total number of cells ranges from 10K to 210K.

Circuit	#Pad	#Std. cell	#Macro	#Fixed macro	#Net	%WS
ibm01	246	12260	246	0	14111	20
ibm02	259	19071	271	0	19584	20
ibm03	283	22563	290	0	27401	20
ibm04	287	26925	295	0	31970	20
ibm05	1201	28146	0	0	28446	20
ibm06	166	32154	178	0	34826	20
ibm07	287	45348	291	0	48117	20
ibm08	286	50722	301	0	50513	20
ibm09	285	52857	253	0	60902	20
ibm10	744	67899	786	0	75196	20
ibm11	406	69779	373	0	81454	20
ibm12	637	69788	651	0	77240	20
ibm13	490	83285	424	0	99666	20
ibm14	517	146474	614	0	152772	20
ibm15	383	160794	393	0	186608	20
ibm16	504	182522	458	0	190048	20
ibm17	743	183992	760	0	189581	20
ibm18	272	210056	285	0	201920	20

Table 6.3: ICCAD04 Benchmark Statistics.

Circuit	#Pad	#Std. cell	#Macro	#Fixed macro	#Net	%WS
DMA	945	11734	0	0	12613	5
DSP1	842	26299	2	0	28400	9
DSP2	842	26279	2	0	28384	10
RISC1	625	32615	7	0	33762	5
RISC2	625	32615	7	0	33762	5

Table 6.4: FARADY Benchmark Statistics.

Circuit	#Pad	#Std. cell	#Macro	#Fixed macro	#Net	%WS
adaptec1	480	210904	63	63	221142	24
adaptec2	407	254457	159	159	266009	21
adaptec3	0	451023	723	723	466758	25
adaptec4	0	494812	1329	1329	515951	37
bigblue1	528	277604	32	32	284479	46
bigblue2	0	534878	23084	23084	577235	38
bigblue3	0	1093130	3778	1293	1123170	14
bigblue4	0	2169279	8170	8170	2229886	35

Table 6.5: ISPD05 Benchmark Statistics.

Circuit	#Pad	#Std. cell	#Macro	#Fixed macro	#Net	%WS
Peko01	246	12506	0	0	14111	16
Peko02	259	19342	0	0	19584	16
Peko03	283	22853	0	0	27401	16
Peko04	287	27220	0	0	31970	15
Peko05	426	28146	0	0	28446	15
Peko06	166	32332	0	0	34826	15
Peko07	287	45639	0	0	48117	15
Peko08	286	51023	0	0	50513	15
Peko09	285	53110	0	0	60902	15
Peko10	566	68685	0	0	75196	15
Peko11	406	70152	0	0	81454	15
Peko12	637	70439	0	0	77240	15
Peko13	490	83709	0	0	99666	15
Peko14	517	147088	0	0	152772	15
Peko15	383	161187	0	0	186608	15
Peko16	504	182980	0	0	190048	15
Peko17	743	184752	0	0	189581	15
Peko18	272	210341	0	0	201920	15

Table 6.6: PEKO Benchmark Statistics.

All experiments on benchmarks ICCAD00, ISPD04, ICCAD04, FARADY, and PEKO are run on a Linux, 2.4GHz, Intel 32-bit machine. Experiments on ISPD05 benchmark are run on a Linux, 1.8GHz, AMD Opteron 64-bit machine.

6.2 Introduction to State-of-the-art Placement Algorithms

In this section, a brief description of several state-of-the-art placement algorithms, with the executable of the algorithm available for conducting experiments, are presented.

APlace [KW04, KRW05] is a high quality analytical placer that solves the nonlinear programming formulation (5.3) by penalty method utilizing nonlinear conjugate gradient algorithm. It uses log-sum-exp (3.5) function to approximate the HPWL and a bell-shape function to smooth the density function locally.

Capo [ACJ04, RPN06] and FengShui [YM01, AOM05] are well-established min-cut partition based placers.

Dragon [WYS00] is top-down hierarchical approach based on min-cut partitioning and simulated annealing placer.

FastPlace [VC04] is an ultra fast analytical placer that minimizes weighted quadratic wirelength with cell shifting techniques to remove area congestion.

6.3 Performances of Our Multilevel Placement Algorithms

In this section, we evaluate the performances of different versions of our multilevel placement algorithm – mPL. More detailed performances on GFD algorithm – a powerful optimization framework for placement, are evaluated.

In Table 6.7, we run the GFD algorithm (cf. Table 5.1) with different number

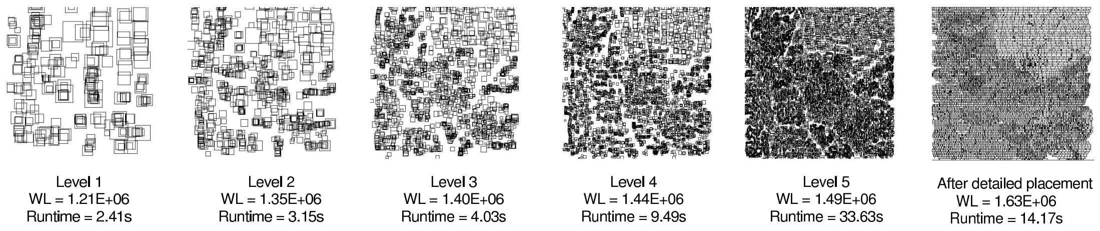

Figure 6.1: Placement solutions at each level in the second V-cycle of mPL5 for ibm01 of ISPD04 benchmark.

of *outer_iters* shown in the parenthesis. It also shows comparisons with mPL5, the multilevel GFD (cf. Table 5.2). The wirelength and runtime are relative to GFD(10), the GFD algorithm with 10 outer iterations. The *stop_percent* in the GFD algorithm is set to 97 in the comparisons. We can see the wirelength is getting shorter as we increase the number of outer iterations. We remark that keeping increase of the number of outer iterations, though increasing the runtime, does not further significantly reduce the wirelength. However, the multilevel GFD (mPL5 given in Table 6.7) significantly outperforms the GFD both in quality and runtime. This shows that our multilevel algorithm is a very effective technique that gives better scalability and better global optimization. Figure 6.1 shows the placement solutions of mPL5 at each level in the second V-cycle. We can see that cells are distributed more evenly from level to level.

In Table 6.8, we compare the performances of different objectives: log-sum-exp (3.5), Lp-norm (5.1) and quadratic (3.2) under the mPL5 platform. It shows that the global placement wirelength by Lp-norm is 3% longer than log-sum-exp and with a 67% longer runtime. The runtime for quadratic is around 20% shorter than log-sum-exp, but its wirelength is 61% longer. This demonstrates that the LogsumExp gives the best approximation to (2.3).

In Table 6.9 and Table 6.10, we compare mPL5 with Capo9.0, Dragon3.01, FastPlace1.0, and FengShui5.0. All the placers are run in default mode. Ta-

	GFD(20)	GFD(30)	mPL5
Circuit	Rel. WL, Rel. time	Rel. WL, Rel. time	Rel. WL, Rel. time
ibm01	0.96 , 1.34	0.93 , 1.75	0.87 , 0.58
ibm02	0.84 , 1.31	0.82 , 1.66	0.78 , 0.67
ibm03	0.97 , 1.39	0.95 , 1.84	0.94 , 0.56
ibm04	0.93 , 1.38	0.90 , 1.74	0.76 , 0.63
ibm05	0.93 , 1.13	0.83 , 1.14	0.63 , 0.57
ibm06	0.96 , 1.47	0.90 , 1.70	0.76 , 0.58
ibm07	0.95 , 1.52	0.92 , 2.20	0.77 , 0.43
ibm08	0.96 , 1.37	0.93 , 1.78	0.86 , 0.45
ibm09	0.94 , 1.48	0.93 , 1.91	0.83 , 0.44
ibm10	0.89 , 1.35	0.86 , 1.73	0.77 , 0.41
ibm11	0.92 , 1.23	0.89 , 1.73	0.77 , 0.39
ibm12	0.95 , 1.41	0.90 , 1.75	0.87 , 0.47
ibm13	0.95 , 1.45	0.92 , 1.92	0.80 , 0.41
ibm14	0.94 , 1.47	0.93 , 1.94	0.77 , 0.34
ibm15	0.96 , 1.47	0.94 , 1.94	0.80 , 0.33
ibm16	0.96 , 1.39	0.93 , 1.73	0.78 , 0.34
ibm17	0.95 , 1.37	0.91 , 1.81	0.71 , 0.37
ibm18	0.93 , 1.33	0.89 , 1.72	0.69 , 0.34
average	0.94 , 1.38	0.90 , 1.78	0.79 , 0.46

Table 6.7: Relative wirelength (Rel. WL) and relative runtime (Rel. time) of GFD(20), GFD(30) and mPL5 with respect to GFD(10) on ISPD04 benchmark.

	log-sum-exp	Lp-norm	quadratic
Circuit	WL, runtime(s)	Rel. WL, Rel. runtime	Rel. WL, Rel. runtime
ibm01	1.57E+06 , 45	1.05 , 1.71	1.73 , 0.81
ibm02	3.51E+06 , 66	1.02 , 1.80	1.84 , 1.65
ibm03	4.83E+06 , 66	0.99 , 1.82	1.63 , 0.64
ibm04	5.90E+06 , 117	0.98 , 1.47	1.48 , 0.47
ibm05	9.85E+06 , 94	1.06 , 1.74	1.49 , 1.17
ibm06	4.73E+06 , 161	1.03 , 1.36	1.82 , 0.50
ibm07	8.14E+06 , 209	1.04 , 1.62	1.50 , 0.67
ibm08	9.49E+06 , 321	1.01 , 1.45	1.79 , 0.72
ibm09	9.25E+06 , 313	1.05 , 1.53	1.65 , 0.53
ibm10	1.74E+07 , 302	1.03 , 2.07	1.47 , 0.72
ibm11	1.39E+07 , 379	1.04 , 1.62	1.54 , 0.52
ibm12	2.31E+07 , 367	1.02 , 1.68	1.34 , 0.67
ibm13	1.67E+07 , 404	1.03 , 1.72	1.69 , 0.63
ibm14	3.25E+07 , 1164	1.03 , 1.46	1.60 , 0.71
ibm15	3.92E+07 , 1250	1.04 , 1.84	1.61 , 0.80
ibm16	4.32E+07 , 1387	1.04 , 1.63	1.66 , 0.79
ibm17	6.27E+07 , 1347	1.03 , 1.73	1.42 , 0.85
ibm18	4.18E+07 , 1502	1.07 , 1.78	1.77 , 0.97
average	1.00 , 1.00	1.03 , 1.67	1.61 , 0.77

Table 6.8: Comparisons of half perimeter wirelength (WL) and runtime of global placement under different objectives on ISPD04 benchmark. Wirelength and runtime using Lp-norm ($p = 32$) and quadratic are divided by those of using log-sum-exp respectively.

Circuit	mPL5 HPWL	Capo9.0 Rel. WL	Dragon3.01 Rel. WL	FastPlace1.0 Rel. WL	FengShui5.0 Rel. WL	mPL5-fast Rel. WL
ibm01	1.67E+06	1.08	1.02	1.09	1.08	1.09
ibm02	3.62E+06	1.09	1.02	1.06	1.02	1.02
ibm03	4.57E+06	1.10	1.05	1.12	1.03	1.05
ibm04	5.75E+06	1.06	1.00	1.04	1.05	1.04
ibm05	9.92E+06	1.02	0.98	1.05	1.00	1.05
ibm06	5.10E+06	1.11	0.98	1.04	1.02	1.03
ibm07	8.23E+06	1.11	1.04	1.08	1.09	1.09
ibm08	9.38E+06	1.05	0.96	1.02	n/a	1.05
ibm09	9.33E+06	1.08	1.07	1.12	1.06	1.07
ibm10	1.73E+07	1.10	1.04	1.07	1.07	1.11
ibm11	1.40E+07	1.09	1.03	1.09	1.04	1.06
ibm12	2.23E+07	1.11	1.03	1.08	1.07	1.06
ibm13	1.66E+07	1.10	1.05	1.11	1.09	1.07
ibm14	3.16E+07	1.10	1.05	1.11	1.04	1.08
ibm15	3.85E+07	1.09	1.04	1.13	1.07	1.07
ibm16	4.30E+07	1.10	1.05	1.07	1.09	1.08
ibm17	6.13E+07	1.09	1.08	1.08	1.08	1.09
ibm18	4.10E+07	1.09	1.02	1.10	1.04	1.07
average	1.00	1.09	1.03	1.08	1.06	1.07

Table 6.9: Comparisons of HPWL between mPL5, mPL5-fast, Capo 9.0, Dragon 3.01, FastPlace 1.0 and FengShui 5.0 on ISPD04 benchmark. Rel. WL is stand for the relative wirelength with respect to mPL5 HPWL. Program failure is denoted by n/a.

	mPL5	Capo9.0	Dragon3.01	FastPlace1.0	FengShui5.0	mPL5-fast
Circuit	Time (s)	Rel. time	Rel. time	Rel. time	Rel. time	Rel. time
ibm01	64	1.90	16.81	0.10	2.06	0.23
ibm02	126	1.82	7.34	0.13	1.93	0.26
ibm03	113	2.56	8.04	0.13	2.39	0.28
ibm04	151	2.47	10.95	0.13	2.16	0.35
ibm05	158	2.52	17.36	0.14	2.29	0.30
ibm06	200	2.01	10.44	0.12	2.12	0.28
ibm07	259	2.45	9.61	0.19	2.32	0.30
ibm08	389	1.73	15.47	0.14	n/a	0.38
ibm09	342	2.30	16.26	0.17	1.72	0.28
ibm10	450	2.42	10.96	0.20	1.56	0.30
ibm11	437	2.70	7.81	0.19	2.31	0.30
ibm12	482	2.48	11.15	0.20	2.03	0.36
ibm13	596	2.32	7.73	0.20	1.80	0.28
ibm14	1064	2.49	10.65	0.21	1.20	0.33
ibm15	1379	2.41	11.14	0.23	2.15	0.30
ibm16	1577	2.29	11.09	0.23	2.18	0.30
ibm17	1705	2.32	22.22	0.23	2.17	0.30
ibm18	1904	2.00	17.84	0.25	2.07	0.29
average	1.00	2.29	12.38	0.18	2.03	0.30

Table 6.10: Comparisons of runtime between mPL5, mPL5-fast, Capo 9.0, Dragon 3.01, FastPlace 1.0 and FengShui 5.0. Rel. time is stand for relative runtime with respect to mPL5 runtime. Program failure is denoted by n/a.

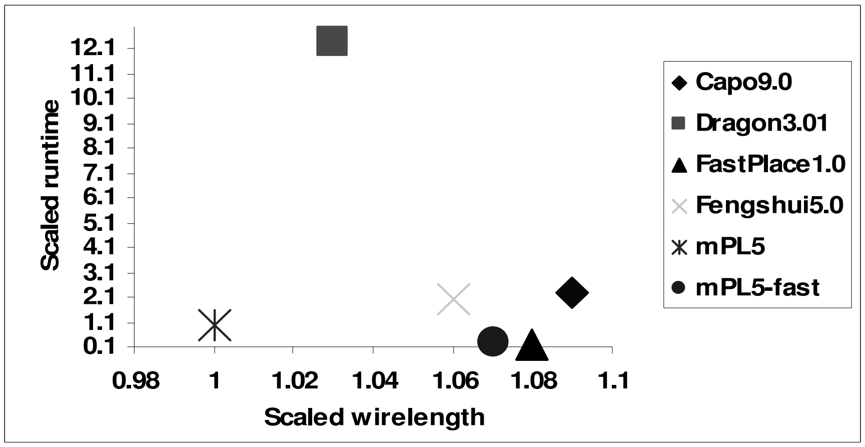

Figure 6.2: Wirelength and runtime comparisons on FastPlace1.0 IBM circuits.

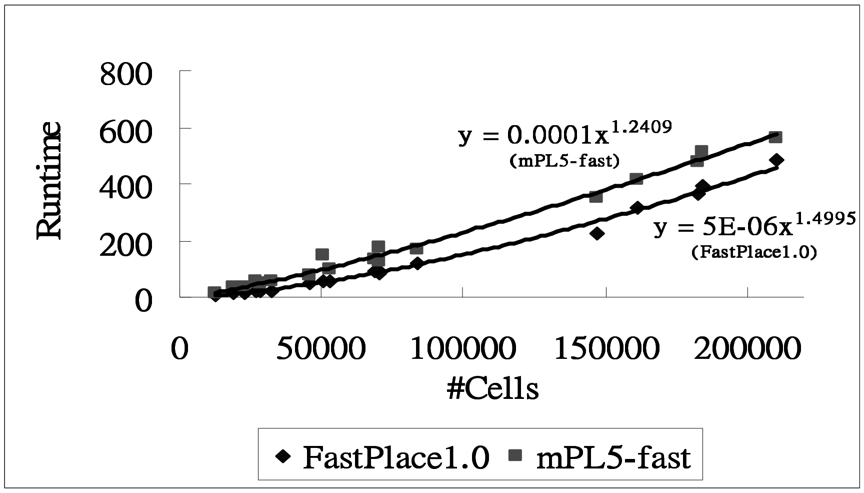

Figure 6.3: Scalability plot of FastPlace1.0 and mPL5-fast on ISPD04 benchmark.

ble 6.9 shows that overall mPL5 produces the shortest wirelength. Compared to Capo9.0, mPL5 has 8% shorter wirelength and is 2X faster. Compared to Dragon3.01, mPL5 produces 3% shorter wirelength and is 12X faster. Compared to FastPlace1.0, it outperforms the wirelength by 8% but is 6X slower. Compared to FengShui5.0, mPL5 has 5% shorter wirelength and is 2X faster. The fast mode of mPL5, mPL5-fast (cf. Section 5.3.1), can produce 1% shorter wirelength than FastPlace1.0 and is only 2X slower in average. A scalability plot of FastPlace1.0 and mPL5-fast is shown in Figure 6.3. It shows that mPL5-fast is more scalable and it is expected that mPL5-fast will be faster than FastPlace1.0 on design with millions of cells. Figure 6.2 shows the average performance of each placer. The wirelength and runtime shown are divided by mPL5's wirelength and runtime respectively. We remark that Dragon's fixed-die mode (routability congestion driven) results, longer runtime and wirelength than the results of the default mode (wirelength driven), are used for comparisons in [VC04]. We compare mPL5 with Dragon's wirelength driven mode in this thesis. Also, the new binary of FastPlace1.0 (used for comparisons in this thesis) produces 5% shorter wirelength than the results published in [VC04] with similar runtime.

We also run mPL5 on PEKO [CCX03] which is a set of synthetic benchmarks used to evaluate how far placement tools are from optimal. In [CCX03], it shows that the quality of the current state-of-the-art placers is 50% to 150% from optimal. The results of mPL5 on PEKO suiteIII, circuits with pad connections, as well as other placers' results are shown in Figure 6.4. mPL5 can produce placement solution that is very close to the optimal – only around 25% away, which again is the best among the compared placers.

Next, we compare the performances between different versions (2 – 6) of mPL. A brief description of mPL2, mPL3, and mPL4 can be found in Section

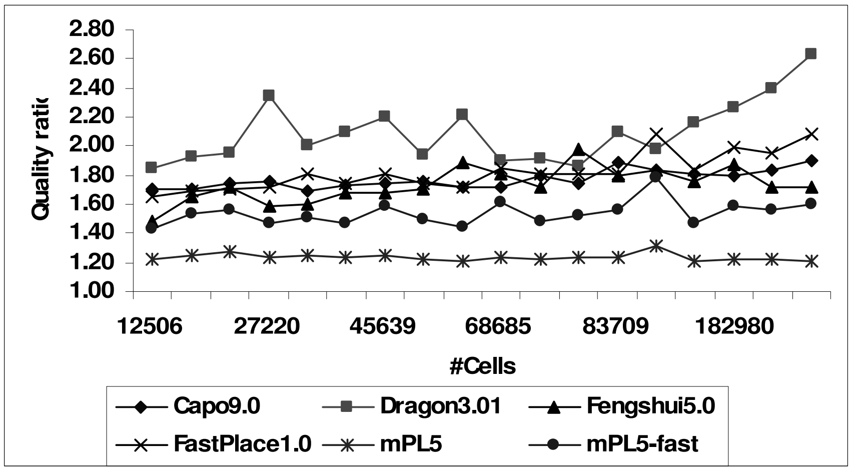

Figure 6.4: Quality ratio on PEKO suiteIII.

4.2. mPL6 (cf. Table 5.4) is built on top of mPL5 by incorporating the enhanced GFD algorithm (cf. Table 5.3).

Table 6.11 and Table 6.12 show the comparisons of placement HPWL and runtime between mPL2, mPL3, mPL4, mPL5, and mPL6 on ICCAD00 benchmark. Note that mPL2 is developed to address uniform size cell circuit. We see that mPL2 and mPL3 has similar HPWL but mPL3 is slightly faster. It is because mPL3 has better handling of non-uniform cell size placement at coarse levels. Also, mPL3 incorporates speed-up technique in ripple-move area congestion remover. mPL4 is using similar techniques as mPL3 except it is using a more expensive but more effective multilevel flow – back-tracking V-cycles (cf. Figure 4.4). Comparing with mPL2, mPL4 has 5% shorter HPWL and only has around 16% longer runtime. mPL5, using the GFD algorithm as the relaxation, significantly outperforms mPL2. Comparing with mPL2, mPL5 has 10% shorter HPWL and is near 2X faster. mPL6 has similar HPWL as mPL5 but has much longer runtime. The increase in runtime is due to more even distribution of placement at coarse levels (cf. Figure 6.5) and the wirelength weighting scheme (cf.

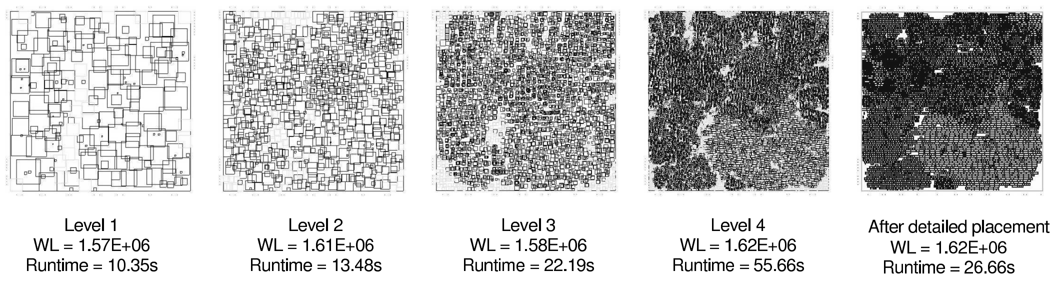

Figure 6.5: Placement solutions at each level in the first V-cycle of mPL6 for ibm01 of ISPD04 benchmark.

(5.24)).

In Table 6.13 and Table 6.14, we compare the placement HPWL and runtime on ISPD04 benchmark between mPL3, mPL4, mPL5, and mPL6. This benchmark has non-uniform cell size and mPL2 is not able to handle it. Comparing to mPL3, mPL4 has 8% shorter HPWL and is near 2X faster. This shows that mPL4 can handle the non-uniform standard cell size placement better. While mPL5 and mPL6 has similar performances as on ICCAD00 benchmark. Their performances are stable regarding the standard cell size. Comparing to mPL4, mPL5 has 5% shorter HPWL and is around 2X faster. Similar comparisons between mPL4 and mPL5 are observed on ICCAD00 benchmark.

Table 6.11 shows the quality of ratio of mPL2, mPL3, mPL4, mPL5, and mPL6 on PEKO benchmark. The average optimality gap for each version of mPL ranges from 24% to 43%. It consistently reflects the relative performance of each placer on the real design.

So far we see that mPL5 and mPL6 have similar placement HPWL and mPL5 is much faster on the ICCAD00 and ISPD04 benchmarks. They have consistently better performance than mPL2, mPL3, and mPL4. In the following, we compare mPL5 and mPL6 on more different and complex circuit designs. For more spe-

Circuit	mPL2 HPWL	mPL3 Rel. WL	mPL4 Rel. WL	mPL5 Rel. WL	mPL6 Rel. WL
ibm04	6.70E+06	1.02	0.96	0.93	0.95
ibm07	1.04E+07	1.03	0.97	0.91	0.91
ibm09	1.19E+07	0.96	0.93	0.88	0.87
ibm10	1.88E+07	0.98	0.96	0.93	0.94
ibm14	4.08E+07	0.98	0.93	0.88	0.88
ibm16	5.48E+07	0.94	0.92	0.89	0.86
ibm17	6.88E+07	1.01	0.96	0.90	0.89
ibm18	5.20E+07	1.04	0.97	0.91	0.89
average	1.00	1.00	0.95	0.90	0.90

Table 6.11: Comparisons of placement HPWL between mPL2, mPL3, mPL4, mPL5, and mPL6 on ICCAD00 benchmark. Rel. WL is stand for relative HPWL with respect to mPL2 HPWL.

	mPL2	mPL3	mPL4	mPL5	mPL6
Circuit	Time (s)	Rel. time	Rel. time	Rel. time	Rel. time
ibm04	474	0.91	1.31	0.64	1.61
ibm07	795	0.85	1.34	0.75	1.69
ibm09	954	0.83	1.38	0.85	2.44
ibm10	1808	0.70	1.11	0.56	1.59
ibm14	3652	0.74	1.06	0.58	1.30
ibm16	5557	0.72	1.09	0.57	1.12
ibm17	7165	0.73	1.04	0.48	1.06
ibm18	7953	0.70	0.94	0.44	1.34
average	1.00	0.77	1.16	0.61	1.52

Table 6.12: Comparisons of placement runtime (s) between mPL2, mPL3, mPL4, mPL5, and mPL6 on ICCAD00 benchmark. Rel. time is stand for relative runtime with respect to mPL2 runtime.

	mPL3	mPL4	mPL5	mPL6
Circuit	HPWL	Rel. WL	Rel. WL	Rel. WL
ibm01	1.88E+06	0.94	0.88	0.86
ibm02	3.77E+06	1.00	0.96	0.95
ibm03	5.45E+06	0.93	0.84	0.87
ibm04	6.34E+06	0.96	0.90	0.92
ibm05	1.06E+07	0.98	0.94	0.88
ibm06	5.88E+06	0.90	0.87	0.83
ibm07	9.62E+06	0.89	0.85	0.84
ibm08	9.77E+06	0.96	0.95	0.91
ibm09	1.10E+07	0.88	0.84	0.83
ibm10	1.98E+07	0.94	0.88	0.88
ibm11	1.67E+07	0.89	0.84	0.83
ibm12	2.51E+07	0.94	0.89	0.87
ibm13	2.04E+07	0.88	0.81	0.82
ibm14	3.74E+07	0.90	0.84	0.85
ibm15	4.57E+07	0.90	0.85	0.83
ibm16	5.10E+07	0.88	0.85	0.84
ibm17	7.07E+07	0.93	0.87	0.84
ibm18	4.86E+07	0.89	0.85	0.84
average	1.00	0.92	0.87	0.86

Table 6.13: Comparisons of placement HPWL between mPL3, mPL4, mPL5, and mPL6 on ISPD04 benchmark. Rel. WL is stand for relative HPWL with respect to mPL3 HPWL.

	mPL3	mPL4	mPL5	mPL6
Circuit	Time	Rel. time	Rel. time	Rel. time
ibm01	211	0.71	0.30	0.67
ibm02	456	0.97	0.28	0.54
ibm03	501	0.60	0.22	0.53
ibm04	565	0.61	0.26	0.64
ibm05	876	0.80	0.18	0.51
ibm06	756	0.60	0.26	0.63
ibm07	1000	0.60	0.26	0.65
ibm08	1578	0.68	0.24	0.48
ibm09	1243	0.59	0.30	0.63
ibm10	2017	0.63	0.22	0.57
ibm11	1568	0.58	0.28	0.65
ibm12	2124	0.67	0.24	0.59
ibm13	2281	0.56	0.25	0.55
ibm14	3776	0.64	0.29	0.68
ibm15	5289	0.63	0.26	0.60
ibm16	5891	0.64	0.26	0.56
ibm17	7022	0.67	0.24	0.57
ibm18	7127	0.68	0.26	0.51
average	1.00	0.66	0.26	0.59

Table 6.14: Comparisons of placement runtime between mPL3, mPL4, mPL5, and mPL6 on ISPD04 benchmark. Rel. time is stand for relative runtime with respect to mPL3 runtime.

Circuit	mPL2 QoR	mPL3 QoR	mPL4 QoR	mPL5 QoR	mPL6 QoR
Peko01	1.37	1.39	1.25	1.24	1.24
Peko02	1.35	1.38	1.25	1.27	1.24
Peko03	1.40	1.42	1.27	1.29	1.24
Peko04	1.34	1.39	1.28	1.24	1.24
Peko05	1.45	1.50	1.26	1.27	1.22
Peko06	1.39	1.39	1.29	1.25	1.23
Peko07	1.39	1.41	1.26	1.26	1.24
Peko08	1.38	1.38	1.27	1.25	1.24
Peko09	1.40	1.45	1.25	1.23	1.24
Peko10	1.47	1.55	1.27	1.26	1.25
Peko11	1.38	1.40	1.27	1.24	1.25
Peko12	1.39	1.41	1.27	1.26	1.25
Peko13	1.38	1.42	1.27	1.25	1.26
Peko14	1.48	1.50	1.30	1.31	1.24
Peko15	1.38	1.40	1.28	1.22	1.24
Peko16	1.41	1.44	1.28	1.23	1.24
Peko17	1.40	1.42	1.28	1.23	1.24
Peko18	1.57	1.43	1.28	1.23	1.23
average	1.41	1.43	1.27	1.25	1.24

Table 6.15: Comparisons of placement QoR (= placement HPWL/Optimal HPWL) between mPL2, mPL3, mPL4, mPL5, and mPL6 on PEKO benchmark.

cific comparisons, we only compare the global placement wirelength and runtime of mPL5 and mPL6. This comparison can exclude the impact of the detailed placement.

Table 6.16 shows that mPL5 has 5% shorter global placement HPWL and is around 2X faster than mPL6 on ISPD04 benchmark. The shorter global HPWL of mPL5 is due to the more even distribution for the global placement in mPL6. mPL6 has a more accurate overflow measurement of the global placement. Comparing Figure 6.1 and Figure 6.5, we see that mPL6 achieves more even distribution of the placement solution at each coarse level. This gives more smooth transaction from global placement to detailed placement. We have seen in Table 6.13 that mPL5 and mPL6 has similar HPWL after detailed placement.

Table 6.17 shows the comparisons of global placement HPWL and runtime on ICCAD04 benchmark between mPL5 and mPL6. This benchmark is more difficult to place since there are large movable macros. In average, mPL6 has around 5% shorter global HPWL and slightly faster than mPL5. In some testcases, mPL6 is more than 2X faster. The main reason is that mPL6 fixes the large macros during the placement that not only causes more stable convergence but also accelerates the convergence due to more connections to fixed cells.

In Table 6.18, we compare mPL5 and mPL6 on FARADY benchmark. It is a more challenging mixed-size benchmark. The netlist is more ill-conditioned in the way that many connections to the terminal pads clustered at a particular boundary. We see that mPL6 outperforms mPL5 significantly, around 40%, in terms of the global HPWL. And mPL6 is slightly faster and more stable. This is mainly not only due to fixing the large macros during placement but also the weighting scheme for the wirelength.

In Table 6.19, we compare mPL5 and mPL6 on ISPD05 benchmark where

	mPL6		mPL5	
Circuit	GP HPWL	GP runtime (s)	rel. GP HPWL	rel. GP time
ibm01	1.62E+06	113	0.95	0.43
ibm02	3.62E+06	191	0.93	0.51
ibm03	4.80E+06	214	0.89	0.42
ibm04	5.94E+06	305	0.92	0.40
ibm05	9.41E+06	363	1.02	0.30
ibm06	4.90E+06	397	0.99	0.41
ibm07	8.22E+06	543	0.97	0.39
ibm08	9.38E+06	632	1.00	0.48
ibm09	9.44E+06	635	0.93	0.50
ibm10	1.79E+07	867	0.93	0.40
ibm11	1.42E+07	825	0.93	0.45
ibm12	2.25E+07	933	0.96	0.42
ibm13	1.72E+07	1014	0.92	0.46
ibm14	3.31E+07	1999	0.95	0.45
ibm15	3.95E+07	2478	0.97	0.46
ibm16	4.48E+07	2647	0.96	0.45
ibm17	6.16E+07	2919	1.00	0.44
ibm18	4.29E+07	2929	0.95	0.50
average	1.00	1.00	0.95	0.44

Table 6.16: Comparisons between global placements of mPL5 and mPL6 on ISPD04 benchmark. Rel. GP HPWL is stand for relative GP HPWL with respect to mPL6 GP HPWL. Rel. GP time is stand for relative GP runtime with respect to mPL6 GP runtime.

Circuit	mPL6		mPL5	
	GP HPWL	GP runtime (s)	rel. GP HPWL	rel. GP time
ibm01	2.11E+06	164	1.01	0.54
ibm02	4.83E+06	241	0.99	1.13
ibm03	6.54E+06	261	1.12	1.24
ibm04	7.38E+06	288	1.07	0.95
ibm05	9.43E+06	330	1.04	0.32
ibm06	5.72E+06	366	1.11	1.53
ibm07	9.96E+06	500	1.03	1.49
ibm08	1.23E+07	749	1.05	1.74
ibm09	1.24E+07	799	1.14	1.95
ibm10	2.80E+07	1001	1.05	1.75
ibm11	1.80E+07	998	1.05	1.16
ibm12	3.25E+07	1116	1.15	1.47
ibm13	2.29E+07	1246	1.00	1.21
ibm14	3.71E+07	1881	1.00	1.03
ibm15	4.73E+07	2375	1.17	2.58
ibm16	5.79E+07	3205	1.02	2.13
ibm17	6.73E+07	2802	1.05	0.79
ibm18	4.42E+07	2840	1.03	0.84
average	1.00	1.00	1.06	1.33

Table 6.17: Comparisons between global placements of mPL5 and mPL6 on ICCAD04 benchmark. Rel. GP HPWL is stand for relative GP HPWL with respect to mPL6 GP HPWL. Rel. GP time is stand for relative GP runtime with respect to mPL6 GP runtime.

	mPL6		mPL5	
Circuit	GP HPWL	GP runtime (s)	rel. GP HPWL	rel. GP time
DMA	4.11E+08	71	1.36	1.00
DSP1	8.67E+08	153	1.68	1.19
DSP2	8.56E+08	160	1.55	1.00
RISC1	1.35E+09	223	1.72	1.59
RISC2	1.30E+09	219	1.70	2.10
average	1.00	1.00	1.60	1.38

Table 6.18: Comparisons between global placements of mPL5 and mPL6 on FARADY benchmark. Rel. GP HPWL is stand for relative GP HPWL with respect to mPL6 GP HPWL. Rel. GP time is stand for relative GP runtime with respect to mPL6 GP runtime.

many macros are fixed throughout the placement region (cf. Figure 2.3). The netlist is considered well-conditioned as there are lots of connections to the fixed cells that are evenly distributed over the placement region. However, there are more whitespace which causes over-spreading of the cells in mPL5. We see that mPL6 has around 19% shorter global HPWL than mPL5. Also, mPL6 is slightly faster. We remark that by using cluster ratio of 0.1 (cf. Table 5.4), mPL6 can produce similar results for ISPD05 benchmark with around 30% runtime speed-up, due to the sufficient connections to fixed cells.

Next, we compare the HPWL of mPL5 and mPL6 after detailed placement. We see that the quality gap of HPWL after detailed placement between mPL5 and mPL6 is smaller. This is because the local cells swapping in the detailed placement can further reduce the HPWL and correct the mistakes made in the global placement. We also compare to APlace version 2 and Capo version 10.

Circuit	mPL6		mPL5	
	GP HPWL	GP runtime (s)	rel. GP HPWL	rel. GP time
adaptec1	8.00E+07	2107	1.14	2.14
adaptec2	9.36E+07	2203	1.20	2.23
adaptec3	2.15E+08	7745	1.25	0.69
adaptec4	1.97E+08	7246	1.20	0.67
bigblue1	1.02E+08	2694	1.16	0.93
bigblue2	1.56E+08	7355	1.35	0.59
bigblue3	3.59E+08	9990	1.33	2.93
bigblue4	8.79E+08	22827	1.17	1.25
average	1.00	1.00	1.23	1.43

Table 6.19: Comparisons between global placements of mPL5 and mPL6 on ISPD05 benchmark. Rel. GP HPWL is stand for relative GP HPWL with respect to mPL6 GP HPWL. Rel. GP time is stand for relative GP runtime with respect to mPL6 GP runtime.

We remark that APlace2 is a high quality analytical placer which is the winner in the ISPD'05 Placement Contest. Capo10 is a well-established and open source min-cut based placer. It's runtime is known to be scalable.

From Table 6.20, we see that mPL6, APlace2 and mPL5 have similar HPWL. mPL6 has around 8% shorter HPWL than Capo10. Table 6.21 shows that mPL6 is around 2X faster than APlace10, 2X slower than mPL5, and 60% faster than Capo10.

Table 6.22 gives the comparisons of HPWL between mPL6, APlace2, Capo10, and mPL5 on ICCAD04. mPL6 and APlace has similar quality of HPWL. mPL6 has around 10% and 2% shorter HPWL than Capo10 and mPL5 respectively. The runtime comparisons in Table 6.23 show that mPL6 is the fastest placer in average. It is around 2.5X faster than APlace2, 2X faster than Capo10, and slightly faster than mPL5.

In Table 6.24 and Table 6.25, we compare the performances of mPL6, APlace2, Capo10, and mPL5 on ISPD05 benchmark that is used in ISPD'05 placement contest. Comparing with APlace2, mPL6 has around 5% shorter HPWL and is near 4X faster. Comparing with Capo10, mPL6 has around 9% shorter HPWL and is near 3X faster. Comparing with mPL5, mPL6 has around 17% shorter HPWL and is slightly faster. We can see that mPL6 is more scalable and is able to place a 2.2 millions cells design within 9 hours.

	mPL6	APlace2	Capo10	mPL5
Circuit	HPWL	Rel. WL	Rel. WL	Rel. WL
ibm01	1.62E+06	1.00	1.12	1.02
ibm02	3.59E+06	0.97	1.06	1.01
ibm03	4.73E+06	0.95	1.03	0.97
ibm04	5.80E+06	0.97	1.02	0.99
ibm05	9.32E+06	1.02	1.07	1.06
ibm06	4.85E+06	0.99	1.09	1.05
ibm07	8.07E+06	0.98	1.10	1.02
ibm08	8.88E+06	1.05	1.10	1.05
ibm09	9.21E+06	0.97	1.14	1.01
ibm10	1.75E+07	0.97	1.07	0.99
ibm11	1.38E+07	0.97	1.09	1.01
ibm12	2.17E+07	0.99	1.12	1.02
ibm13	1.67E+07	1.00	1.11	0.99
ibm14	3.17E+07	0.97	1.07	0.99
ibm15	3.79E+07	1.02	1.08	1.02
ibm16	4.26E+07	0.97	1.09	1.01
ibm17	5.92E+07	1.00	1.14	1.03
ibm18	4.07E+07	0.96	1.10	1.01
average	1.00	0.99	1.09	1.01

Table 6.20: Comparisons of HPWL between APlace2, Capo10, mPL5, and mPL6 on ISPD04 benchmark. Rel. WL is stand for relative HPWL with respect to mPL6 HPWL.

Circuit	mPL6 Time (s)	APlace2 Rel. time	Capo10 Rel. time	mPL5 Rel. time
ibm01	141	1.75	1.35	0.45
ibm02	247	1.96	1.49	0.51
ibm03	264	2.69	1.58	0.42
ibm04	364	2.57	1.49	0.41
ibm05	445	2.22	1.48	0.36
ibm06	478	1.93	1.23	0.41
ibm07	652	2.51	1.57	0.40
ibm08	765	1.82	1.64	0.49
ibm09	780	2.28	1.53	0.48
ibm10	1155	2.25	1.59	0.39
ibm11	1025	2.74	1.68	0.44
ibm12	1248	2.19	1.72	0.40
ibm13	1266	2.79	1.61	0.45
ibm14	2550	2.20	1.69	0.42
ibm15	3193	2.56	1.65	0.43
ibm16	3276	2.24	1.98	0.47
ibm17	3973	2.36	1.89	0.43
ibm18	3654	2.79	1.69	0.51
average	1.00	2.32	1.60	0.44

Table 6.21: Comparisons of runtime between APlace2, Capo10, mPL5, and mPL6 on ISPD04 benchmark. Rel. time is stand for relative runtime with respect to mPL6 runtime.

Circuit	mPL6 HPWL	APlace2 Rel. WL	Capo10 Rel. WL	mPL5 Rel. WL
ibm01	2.24E+06	0.95	1.11	0.96
ibm02	4.80E+06	0.97	1.06	0.97
ibm03	6.71E+06	1.00	1.11	1.01
ibm04	7.31E+06	1.04	1.20	1.05
ibm05	9.37E+06	1.04	1.10	1.08
ibm06	5.73E+06	1.05	1.17	1.05
ibm07	9.82E+06	1.02	1.18	1.01
ibm08	1.19E+07	1.05	1.14	1.03
ibm09	1.24E+07	0.98	1.19	1.02
ibm10	2.78E+07	1.04	1.19	1.06
ibm11	1.77E+07	1.05	1.20	1.01
ibm12	3.19E+07	1.05	1.18	1.08
ibm13	2.25E+07	1.01	1.22	1.00
ibm14	3.57E+07	1.01	1.09	0.98
ibm15	4.71E+07	0.99	1.15	1.03
ibm16	5.56E+07	0.98	1.13	0.98
ibm17	6.47E+07	1.01	1.12	1.03
ibm18	4.21E+07	1.00	1.10	1.03
average	1.00	1.01	1.15	1.02

Table 6.22: Comparisons of HPWL between APlace2, Capo10, mPL5, and mPL6 on ICCAD04 benchmark. Rel. WL is stand for relative HPWL with respect to mPL6 HPWL.

	mPL6	APlace2	Capo10	mPL5
Circuit	Time (s)	Rel. time	Rel. time	Rel. time
ibm01	200	1.96	1.51	0.57
ibm02	315	2.92	1.77	1.03
ibm03	317	3.37	2.71	1.14
ibm04	374	2.76	2.71	0.86
ibm05	408	2.05	1.70	0.43
ibm06	450	2.32	2.22	1.37
ibm07	648	2.24	2.24	1.28
ibm08	933	1.78	1.94	1.54
ibm09	997	1.95	1.76	1.69
ibm10	1287	2.35	2.33	1.53
ibm11	1263	3.55	2.01	1.02
ibm12	1424	3.00	2.56	1.34
ibm13	1597	3.14	2.24	1.04
ibm14	2550	2.26	2.18	0.89
ibm15	3001	2.10	2.40	2.18
ibm16	3901	1.87	2.12	1.88
ibm17	3659	2.10	2.32	0.81
ibm18	3777	3.28	1.80	0.79
average	1.00	2.50	2.14	1.19

Table 6.23: Comparisons of runtime between APlace2, Capo10, mPL5, and mPL6 on ICCAD04 benchmark. Rel. time is stand for relative runtime with respect to mPL6 runtime.

Circuit	mPL6 HPWL	APlace2 Rel. WL	Capo10 Rel. WL	mPL5 Rel. WL
adaptec1	7.79E+07	1.01	1.14	1.12
adaptec2	9.20E+07	1.04	1.12	1.14
adaptec3	2.14E+08	1.02	1.09	1.24
adaptec4	1.94E+08	1.08	1.06	1.18
bigblue1	9.68E+07	1.03	1.12	1.16
bigblue2	1.52E+08	1.01	1.05	1.32
bigblue3	3.44E+08	1.20	1.17	1.26
bigblue4	8.29E+08	1.05	1.16	1.15
average	1.00	1.05	1.11	1.20

Table 6.24: Comparisons of HPWL between APlace2, Capo10, mPL5, and mPL6 on ISPD05 benchmark. Rel. WL is stand for relative HPWL with respect to mPL6 HPWL.

	mPL6	APlace2	Capo10	mPL5
Circuit	Time (s)	Rel. time	Rel. time	Rel. time
adaptec1	2894	3.02	2.20	1.83
adaptec2	2995	4.22	2.60	1.90
adaptec3	9353	3.27	1.94	0.75
adaptec4	8812	3.90	2.19	0.73
bigblue1	3636	3.16	2.58	0.95
bigblue2	10207	2.67	2.19	0.70
bigblue3	13564	3.90	5.04	2.42
bigblue4	30540	5.26	4.37	1.19
average	1.00	3.67	2.89	1.31

Table 6.25: Comparisons of runtime between APlace2, Capo10, mPL5, and mPL6 on ISPD05 benchmark. Rel. time is stand for relative runtime with respect to mPL6 runtime.

CHAPTER 7

Conclusions and Future Work

To conclude, we have developed a multilevel generalized force-directed placement algorithm, mPL5. With important enhancement techniques, mPL5 is advanced to mPL6 which is a more stable and robust placement algorithm. It significantly improves the older versions (2–5) of mPL. Experiments show that mPL6 is a fast placement algorithm producing the shortest wirelength among the state-of-the-art academic placers. It is a stable and robust algorithm that has consistent good performances on wide varieties of publicly available benchmarks. We remark that the GFD algorithm presented in the paper is not limited to wirelength driven placement. It is a general algorithm that can be extended to handle different objectives and density constraints. In the future, we will extend mPL6 to handle more complex constraints such as routability, thermal, and timing.

APPENDIX A

Fast Discrete Cosine Transform for Solving Helmholtz Equation

In this appendix, a fast discrete cosine transform solver for the discretized Helmholtz equation (5.7) is presented.

The matrix $L_\epsilon[mn]$ arising from the linear system (5.7) can be written as [CCN00]

$$L_\epsilon[mn] = I[m] \otimes T[n]/h_y^2 + T[m] \otimes I[n]/h_x^2 - \epsilon I[mn], \tag{A.1}$$

where $I[n]$ is an $n \times n$ identity matrix and $T[n]$ is an $n \times n$ matrix defined as

$$T[n] = \begin{pmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{pmatrix}. \tag{A.2}$$

$T[n]$ can be diagonalized by the discrete cosine matrix $C[n]$ (cf. 2.1.4), that is, $T[n] = C[n]^T \Lambda[n] C[n]$ where $\Lambda[n]$ is a diagonal matrix with eigenvalues of $T[n]$ as the diagonal entries. Also, it is easy to verify that $C[n]^T = C[n]^{-1}$. Therefore, we have $T[n]C[n]^T = C[n]^T \Lambda[n]$. That means

$$\vec{c}_j \equiv \sqrt{\frac{2-\delta_{j1}}{n}}(\cos(\frac{(j-1)\pi}{2n}), \cos(\frac{3(j-1)\pi}{2n}), \ldots, \cos(\frac{(j-1)(2n-1)\pi}{2n}))^T, \tag{A.3}$$

the j-th column of $C[n]^T$, is an eigenvector of $T[n]$ with $\Lambda[n]_{jj}$ as the corresponding eigenvalue. Due to the sparsity of the matrix $T[n]$, one can verify that the eigenvalues of $T[n]$ satisfy

$$\Lambda[n]_{jj} = -4\sin^2(\frac{(j-1)\pi}{2n}), \quad j = 1, 2, \ldots, n. \tag{A.4}$$

Since we have

$$(C[m] \otimes C[n])(I[m] \otimes T[n])(C[m] \otimes C[n])^T = (C[m]I[m]C[m]^T) \otimes (C[n]T[n]C[n]^T), \tag{A.5}$$

$L_\epsilon[mn]$ (cf. A.1) can be diagonalized as follows

$$(C[m] \otimes C[n])L_\epsilon[mn](C[m] \otimes C[n])^T = I[m] \otimes \Lambda[n] + \Lambda[m] \otimes I[n] - I[mn]. \tag{A.6}$$

Hence the eigenvalues of $L_\epsilon[mn]$ are given by

$$-\frac{4}{h_y^2}\sin^2(\frac{(i-1)\pi}{2m}) - \frac{4}{h_x^2}\sin^2(\frac{(j-1)\pi}{2n}) - \epsilon, \quad 1 \le i \le m, 1 \le j \le n. \tag{A.7}$$

Since the matrix-vector multiplication $C[m] \otimes C[n]\vec{x}$ can be computed in $O(mn \log mn)$ [SB93], the solution of the linear system (5.9) given in (5.11) can be computed in $O(mn \log mn)$.

APPENDIX B

Uzawa Algorithm

Uzawa algorithm [AHU58] is an iterative scheme developed to solve constrained optimization problems. In this appendix, analysis of Uzawa algorithm on solving a linear constrained quadratic programming problem is presented.

Given the linear constrained quadratic programming formulation:

$$\begin{aligned} \min \quad & \vec{x}^T A\vec{x} - \vec{b}^T\vec{x} \\ s.t. \quad & B\vec{x} = \vec{c}, \end{aligned} \tag{B.1}$$

where A is a given real-valued $n \times n$ symmetric positive definite matrix, B is a given real-valued $m \times n$ full rank matrix, $\vec{b}$ and $\vec{c}$ are given real-valued $n \times 1$ vectors, the optimality condition for $\vec{x}$ satisfies [Ber82]

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \vec{x} \\ \vec{\lambda} \end{pmatrix} = \begin{pmatrix} \vec{b} \\ \vec{c} \end{pmatrix}, \tag{B.2}$$

where $\vec{\lambda}$ is the Lagrange multipliers. Instead of solving the above $(n+m)\times(n+m)$ linear system directly, Uzawa algorithm solves the following iterative scheme:

$$\begin{cases} A\vec{x}^{k+1} = -B^T\vec{\lambda}^k + \vec{b} \\ \vec{\lambda}^{k+1} = \vec{\lambda}^k + \alpha(B\vec{x}^{k+1} - \vec{c}) \end{cases}, \tag{B.3}$$

where $\alpha(> 0)$ is a parameter to control the convergence, $\vec{\lambda}^0 = 0$ and $\vec{x}^0$ can be any given vector. By substituting the first equation for $\vec{x}^{k+1}$ into the second equation of (B.3), it gives

$$\vec{\lambda}^{k+1} = (I - \alpha BA^{-1}B^T)\vec{\lambda}^k + \alpha(BA^{-1}\vec{b} - \vec{c}). \tag{B.4}$$

One can see that $\vec{\lambda}^k$ converges to a limit, say $\vec{\lambda}^*$, if $\alpha < \sigma_{\max}(BA^{-1}B^T)$ [GV96] where $\sigma_{\max}(BA^{-1}B^T)$ is the maximum eigenvalue of the symmetric positive definite matrix $BA^{-1}B^T$. Then $\vec{x}^k$ converges to $\vec{x}^*(= -A^{-1}(B^T\vec{\lambda}^* + \vec{b}))$. One can easily verify that the solution $(\vec{x}^*, \vec{\lambda}^*)^T$ satisfies the optimality condition (B.2). Hence the Uzawa algorithm (B.3) converges to the optimal solution of (B.1) if α is chosen properly.

References

[AB84] O. Axelsson and V. A. Barker. *Finite Element Solution of Boundary Value Problem.* Academic Press, Inc., 1984.

[ACJ04] S. N. Adya, S. Chaturvedi, D. A. Papa J. A. Roy, and I. L. Markov. "Unification of Partitioning, Floorplanning and Placement." In *Proceedings of the International Conference on Computer Aided Design*, pp. 550–557, Nov 2004.

[AE04] C.R. Anderson and C. Elion. "Accelerated Solutions of Nonlinear Equations Using Stabilized Runge-Kutta Methods." Report, UCLA CAM, Apr 2004.

[AHK97] C. Alpert, J.-H. Huang, and A.B. Kahng. "Multilevel Circuit Partitioning." In *Proceedings of the Design Automation Conference*, pp. 627–632, 1997.

[AHU58] K. Arrow, L. Huriwicz, and H. Uzawa. *Studies in Nonlinear Programming.* Stanford University Press, 1958.

[AKN05] C. Alpert, A.B. Kahng, G. Nam, S. Reda, and P. Villarrubia. "A Semi-Persistent Clustering Technique for VLSI Circuit Placement." In *Proceedings of the International Symposium on Physical Design*, pp. 200–207, Apr 2005.

[Alp98] C. J. Alpert. "The ISPD98 circuit benchmark suite." In *Proceedings of the International Symposium on Physical Design*, pp. 80–85, 1998.

[AM02] S. N. Adya and I. L. Markov. "Consistent Placement of Macro-Blocks using Floorplanning and Standard-Cell Placement." In *Proceedings of the International Symposium on Physical Design*, pp. 12–17, Apr 2002.

[AMV03] S. N. Adya, I. L. Markov, and P. G. Villarrubia. "On Whitespace and Stability in Mixed-Size Placement." In *Proceedings of the International Conference on Computer Aided Design*, pp. 311–318, Nov 2003.

[AOM05] A.R. Agnihotri, S. Ono, and P.H. Madden. "Recursive Bisection Placement: Feng Shui 5.0 Implementation Details." In *Proceedings of the International Symposium on Physical Design*, pp. 230–232, Apr 2005.

[Ber82] D.P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods.* Academic Press, New York, 1982.

[BMH00] W. L. Briggs, S. F. McCormick, and V. E. Henson. *A Multigrid Tutorial.* SIAM, Philadelphia, second edition, 2000.

[BR02] A. Brandt and D. Ron. *Multigrid Solvers and Multilevel Optimization Strategies*, chapter 1 of *Multilevel Optimization and VLSICAD.* Kluwer Academic Publishers, Boston, 2002.

[Bra86] A. Brandt. "Algebraic Multigrid Theory: The Symmetric Case." *Appl. Math. Comp.*, **19**:23–56, 1986.

[Bra01] A. Brandt. "Multiscale Scientific Computation: Review 2001." In T. Barth, R. Haimes, and T. Chan, editors, *Multiscale and Multiresolution Methods.* Springer Verlag, 2001.

[BS05] Ulrich Brenner and Markus Struzyna. "Faster and Better Global Placement by a New Transportation Algorithm." In *Proceedings of the Design Automation Conference*, pp. 591–596, 2005.

[CCK00] T.F. Chan, J. Cong, T. Kong, and J. Shinnerl. "Multilevel Optimization for Large-Scale Circuit Placement." In *Proceedings of the International Conference on Computer Aided Design*, pp. 171–176, San Jose, CA, Nov 2000.

[CCK03a] T.F. Chan, J. Cong, T. Kong, and J. Shinnerl. *Multilevel Circuit Placement*, chapter 4 of *Multilevel Optimization in VLSICAD.* Kluwer Academic Publishers, Boston, 2003.

[CCK03b] T.F. Chan, J. Cong, T. Kong, J. Shinnerl, and K. Sze. "An Enhanced Multilevel Algorithm for Circuit Placement." In *Proceedings of the International Conference on Computer Aided Design*, pp. 299–306, San Jose, CA, Nov 2003.

[CCN00] R. Chan, T. Chan, M. K. Ng, and A. Yip. "Cosine Transform Preconditioner for High Resolution Image Reconstruction." *Linear Algebra and its Applications*, **316**:89–104, 2000.

[CCS05a] T. Chan, J. Cong, J. Shinnerl, K. Sze, and M. Xie. "Enhanced Robustness in Multilevel Mixed-size Placement." In *SRC TECHCON*, Oct 2005.

[CCS05b] T. Chan, J. Cong, and K. Sze. “Multilevel Generalized Force-directed Method for Circuit Placement.” In *Proceedings of the International Symposium on Physical Design*, pp. 185–192, Apr 2005.

[CCX03] C.C. Chang, J. Cong, and M. Xie. “Optimality and Scalability Study of Existing Placement Algorithms.” In *Proceedings of the Asia South Pacific Design Automation Conference*, pp. 621–627, Kitakyushu, Japan, Jan 2003.

[CCY03] C.C. Chang, J. Cong, and X. Yuan. “Multi-level Placement for Large-scale Mixed-size IC Designs.” In *Proceedings of the Asia South Pacific Design Automation Conference*, pp. 325–330, 2003.

[CKM00a] A.E. Caldwell, A.B. Kahng, and I.L. Markov. “Can Recursive Bisection Alone Produce Routable Placements?” In *Proceedings of the Design Automation Conference*, 2000.

[CKM00b] A.E. Caldwell, A.B. Kahng, and I.L. Markov. “Improved Algorithms for Hypergraph Partitioning.” In *Proceedings of the Asia South Pacific Design Automation Conference*, pp. 661–666, Jan 2000.

[CKS03] J. Cong, T. Kong, J. Shinnerl, M. Xie, and X. Yuan. “Large-Scale Circuit Placement: Gap and Promise.” In *Proceedings of the International Conference on Computer Aided Design*, pp. 883–890, Nov 2003.

[CL00] J. Cong and S.K. Lim. “Performance-driven Multiway Partitioning.” In *Proceedings of the Asia South Pacific Design Automation Conference*, 2000.

[CL04] J. Cong and S.K. Lim. “Edge Separability-Based Circuit Clustering With Application to Multi-level Circuit Partitioning.” *IEEE Tran. on Computer-Aided Design of Integrated Circuits and Systems*, **23(3)**:346–357, 2004.

[Con01] J. Cong. “An Interconnect-Centric Design Flow for Nanometer Technologies.” *Proceedings of the IEEE*, **89**(4):505–528, 2001.

[CX06] J. Cong and M. Xie. “A Robust Detailed Placement for Mixed-Size IC Designs.” In *Proceedings of the Asia South Pacific Design Automation Conference*, pp. 188–194, Jan 2006.

[Eck75] U. Eckhardt. “Weber’s Problem and Weiszfeld’s Algorithm in General Spaces.” *Mathematical Programming*, **18**:186–196, 1975.

[EJ98] Hans Eisenmann and Frank M. Johannes. "Generic Global Placement and Floorplanning." In *Proceedings of the Design Automation Conference*, pp. 269–274, 1998.

[Eva02] L. C. Evans. *Partial Differential Equations.* American Mathematical Society, 2002.

[FFT] `http://momonga.t.u-tokyo.ac.jp/~ooura/fft.html`.

[GV96] G. H. Golub and C. F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, Maryland, third edition, 1996.

[HL99] S.-W. Hur and J. Lillis. "Relaxation and Clustering in a Local Search Framework: Application to Linear Placement." In *Proceedings of the Design Automation Conference*, pp. 360–366, New Orleans, LA, Jun 1999.

[HL00] S.-W. Hur and J. Lillis. "Mongrel: Hybrid Techniques for Standard-Cell Placement." In *Proceedings of the International Conference on Computer Aided Design*, pp. 165–170, San Jose, CA, Nov 2000.

[HM04] B. Hu and M. Marek-Sadowska. "Fine Granularity Clustering for Large Scale Placement Problems." *IEEE Tran. on Computer-Aided Design of Integrated Circuits and Systems*, **23(4)**:527–536, 2004.

[ICC] `http://vlsicad.eecs.umich.edu/BK/ICCAD04bench/`.

[ICk01] ICknowledge. "Can the semiconductor industry afford the cost of new fabs?", 2001.

[ICk02] ICknowledge. "Revenue trends – updated April 23rd.", 2002.

[ISPa] `http://vlsicad.eecs.umich.edu/BK/ISPD02bench`.

[ISPb] `http://www.public.iastate.edu/~nataraj/ISPD04_Bench.html`.

[ISPc] `http://www.sigda.org/ispd2005/contest.htm`.

[itr] "International Technology Roadmap for Semiconductors, 2000 Update.".

[KAK97] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. "Multilevel Hypergraph Partitioning: Application in VLSI Domain." In *Proceedings of the Design Automation Conference*, pp. 526–529, 1997.

[Kis02] L. B. Kish. "End of Moore's law: thermal (noise) death of integration in micro and nano electronics." *Physica Letters A*, **305**:144–149, 2002.

[KM00] A. Kennings and I. L. Markov. "Analytical Minimization of Half-Perimeter Wirelength." In *Proceedings of the Asia South Pacific Design Automation Conference*, pp. 179–184, Jan 2000.

[KRW05] A.B. Kahng, S. Reda, and Q. Wang. "APlace: A General Analytic Placement Framework." In *Proceedings of the International Symposium on Physical Design*, pp. 233–235, Apr 2005.

[KSJ91] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich. "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization." *IEEE Trans. on Computer-Aided Design*, **CAD-10**:356–365, 1991.

[KW04] A.B. Kahng and Q. Wang. "Implementation and Extensibility of an Analytic Placer." In *Proceedings of the International Symposium on Physical Design*, pp. 18–25, 2004.

[LK03] C. Li and C.-K. Koh. "On Improving Recursive Bipartitioning-based Placement." Report tr-ece-03-14, Purdue University ECE, 2003.

[MAN94] I. Mahmoud, K. Asakura, T. Nishibu, and T. Ohtsuki. "Experimental Appraisal of Linear and Quadratic Objective Functions Effect on Force Directed Method for Analog Placement." *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, **4(E77-A)**:710–725, 1994.

[MM94] K.W. Morton and D.F. Mayers. *Numerical Solution of Partial Differential Equations.* Cambridge University Press, 1994.

[Na01] W. Naylor and et. al. "Non-linear Optimization System and Method for Wire Length and Delay Optimization for an Automatic Electric Circuit Placer." *US Patent 6301693*, Oct 2001.

[NAV05] G.-J. Nam, C.J. Alpert, P. Villarubbia, B. Winter, and M. Yildiz. "The ISPD2005 Placement Contest and Benchmark Suite." In *Proceedings of the International Symposium on Physical Design*, pp. 216–219, 2005.

[ROF92] L. I. Rudin, S. J. Osher, and E. Fatermi. "Nonlinear Total Variation Based Noise Removal Algorithms." *Physica D*, **60**:259–268, 1992.

[RPA05] H. Ren, D. Z. Pan, C. Alpert, and P. Villarrubia. "Diffusion Based Placement Migration." In *Proceedings of the Design Automation Conference*, pp. 515–520, Jun 2005.

[RPN06] J.A. Roy, D.A. Papa, A.N. Ng, and I.L. Markov. "Satisfying Whitespace Requirements in Top-down Placement." In *Proceedings of the International Symposium on Physical Design*, pp. 206–208, Apr 2006.

[RS87] J. Ruge and K. Stüben. "Algebraic Multigrid." In S.F. McCormick, editor, *Multigrid Methods*, pp. 73–80. SIAM, Philadelphia, 1987.

[SB93] H. V. Sorensen and C. S. Burrus. "Fast DFT and convolution algorithms." In S. K. Mitra and J. F. Kaiser, editors, *Handbook for Digital Signal Processing.* New York: Jo hn Wiley & Sons, 1993.

[SDJ91] G. Sigl, K. Doll, and F. M. Johannes. "Analytical Placement: A Linear or a Quadratic Objective Function?" In *Proceedings of the Design Automation Conference*, pp. 427–432, 1991.

[TOS00] U. Trottenberg, C.W. Oosterlee, and A. Schüller. *Multigrid.* Academic Press, London, 2000.

[UCL] `http://cadlab.cs.ucla.edu`.

[VC04] Natarajan Viswanathan and Chris Chong-Nuen Chu. "FastPlace: Efficient Analytical Placement Using Cell Shifting, Iterative Local Refinement and a Hybrid Net Model." In *Proceedings of the International Symposium on Physical Design*, pp. 26–33, 2004.

[VK05] K. P. Vorwerk and A. Kennings. "An Improved Mulit-level Framework for Force-directed Placement." In *Proceedings of the Design Automation and Test in Europe*, volume 2, pp. 240–245, 2005.

[VKV04] K. P. Vorwerk, A. Kennings, and A. Vannelli. "Engineering Details of A Stable Force-directed Placer." In *Proceedings of the International Conference on Computer Aided Design*, pp. 573–580, Nov 2004.

[Vyg97] Jens Vygen. "Algorithms for Large-Scale Flat Placement." In *Proceedings of the Design Automation Conference*, pp. 746–751, 1997.

[WYS00] M. Wang, X. Yang, and M. Sarrafzadeh. "Dragon2000: Fast Standard-Cell Placement for Large Circuits." In *Proceedings of the International Conference on Computer Aided Design*, pp. 260–263, Apr 2000.

[XMF04] Z. Xiu, J. D. Ma, S. M. Fowler, and R. A. Rutenbar. "Large-Scale Placement by Grid-Warping." In *Proceedings of the Design Automation Conference*, pp. 351–356, Jun 2004.

[YM01] M. C. Yildiz and P. H. Madden. "Improved Cut Sequences for Partitioning Based Placement." In *Proceedings of the Design Automation Conference*, pp. 776–779, 2001.