Novel Convex Optimization Approaches for VLSI Floorplanning

by

Chaomin Luo

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2008

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The floorplanning problem aims to arrange a set of rectangular modules on a rectangular chip area so as to optimize an appropriate measure of performance. This problem is known to be NP-hard, and is particularly challenging if the chip dimensions are fixed. Fixed-outline floorplanning is becoming increasingly important as a tool to design flows in the hierarchical design of Application Specific Integrated Circuits and System-On-Chip. Therefore, it has recently received much attention.

A two-stage convex optimization methodology is proposed to solve the fixed-outline floorplanning problem. It is a global optimization problem for wirelength minimization. In the first stage, an attractor-repeller convex optimization model provides the relative positions of the modules on the floorplan. The second stage places and sizes the modules using convex optimization. Given the relative positions of the modules from the first stage, a Voronoi diagram and Delaunay triangulation method is used to obtain a planar graph and hence a relative position matrix connecting the two stages. An efficient method for generating sparse relative position matrices and an interchange-free algorithm for local improvement of the floorplan are also presented.

Experimental results on the standard benchmarks MCNC and GSRC demonstrate that we obtain significant improvements on the best results in the literature. *Overlap-free* and *deadspace-free* floorplans are achieved in a fixed outline and floorplans with any specified percentage of whitespace can be produced. Most important, our method provides a greater improvement as the number of modules increases. A very important feature of our methodology is that not only do the dimensions of the floorplans in our experiments comply with the original ones provided in the GSRC benchmark, but also zero-deadspace floorplans can be obtained. Thus, our approach is able to guarantee complete area utilization in a fixed-outline situation. Our method is also applicable to area minimization in classical floorplanning.

Acknowledgements

Firstly, I would like to sincerely and deeply acknowledge my academic advisors, Prof. Anthony Vannelli and Prof. Miguel F. Anjos, for their guidance, encouragement, assistance, constant patience, and continued support. I greatly appreciate them for all that they taught me during the Ph.D. program. Their suggestions were most helpful. I would like to thank my examining committee, Prof. Shawki Areibi, Prof. Samir Elhedhli, and Prof. Catherine H. Gebotys. Special thanks to my external examiner, Prof. Richard Shi.

I would like to thank our system administrators, Phil Regier and Fernando R. Hernandez, for maintaining our computers healthy.

I would like to acknowledge financial support by Postgraduate Award from the Natural Sciences and Engineering Research Council (NSERC) of Canada, Ontario Graduate Scholarship, University of Waterloo President's Graduate Scholarships, and University of Waterloo Graduate Incentive Awards; this support was a great encouragement.

I also wish to thank the following friends and officemates for their assistance and friendship: Rafael Avalos, Hemantkumar Barot, Doris Chen, Mohamed Elsalih, Alexander Engau, Xinxin Fan, Bissan Ghaddar, Mohammad Towhidul Islam, Ibi Jankovits, Hai Jiang, Christie Kong, Nathan Krislock, Xinhua Ling, Stanley Liu, Matthias Takouda, Juan Vera, Kris Vorwerk, Ping Wang, Hua Wei, Zhizhong Yan, Chenxi Zhang, and Jichen Zhang.

Last but not least, I am extremely grateful to my wife, Rong Yang, and my son, Martin Luo, for their love and patience. I also thank my dear parents Ronghua Luo and Xiuzhen Tian, my brother Youmin Luo, and my sister Ping Luo for their love, support, and encouragement despite the distance. None of this would have been possible without their understanding and patience.

To my dear parents

Ronghua Luo,

Xiuzhen Tian

and

my dear wife and son

Rong Yang,

Martin Luo.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The explosive growth in technology for very large scale integration (VLSI) circuit design and manufacturing has led to entire systems with millions of components being placed on a single chip. Due to the increasingly high complexity of modern chip design, VLSI CAD tools are vital for delivering high VLSI system performance and there is a requirement for design automation tools. For most problems in layout design, the computational complexity is NP-hard (Sherwani, 1999). The future tremendous growth of VLSI circuits will rely on the development of physical design automation tools.

1.1 Overview of the VLSI Design Process

In modern VLSI technology, a single chip typically contains millions of transistors. The overall design task is divided into a series of steps: *system specification*, *architectural design*, *behavioural* or *function design*, *logic design*, *circuit design*, *physical design* or *circuit layout*, *fabrication*, *packaging, testing* and *debugging*. This procedure produces a packed chip. These design steps are illustrated by the flow chart

shown in Fig. 1.1 and are briefly outlined below (see e.g., Hu and Kuh, 1985; Sherwani, 1999; Behjat, 2002).

The design cycle starts with the system specification. This process determines the specifications of the system, primarily *functionality*, *performance*, and *physical dimensions*. The design techniques and fabrication technology are also involved in this process. The specifications for the size, speed, power, and functionality of the VLSI system are determined by compromising among technology, market demand, and economical perspective.

In the architectural design step, the design purpose and system constraints are defined. The system is split into components that interact with each other. This process defines the tasks above. Additionally, the criteria specified during the architectural design include specifications such as power requirements, area requirements, and timing requirements (Fig. 1.1).

The third step, functional design, involves identifying the main functional components of the system as well as the interconnection requirements between the components. The behavioural facets of the system are taken into account. Normally, a timing pattern or other relationships between components are the end result of the functional design. Improvement of the overall design process and reduction of the complexity of subsequent stages are gained by means of the information resulting from the functional design.

In the logic design step, the focus is on the derivation and testing of the logic structure that conforms to the functional design. The logic design is typically represented by Boolean expressions which are simplified to generate the smallest logic design corresponding to the functional design. The effectiveness and correctness of the logic design of the system may be verified by simulation and testing.

In the fifth step, the circuit designer converts the logic design represented by

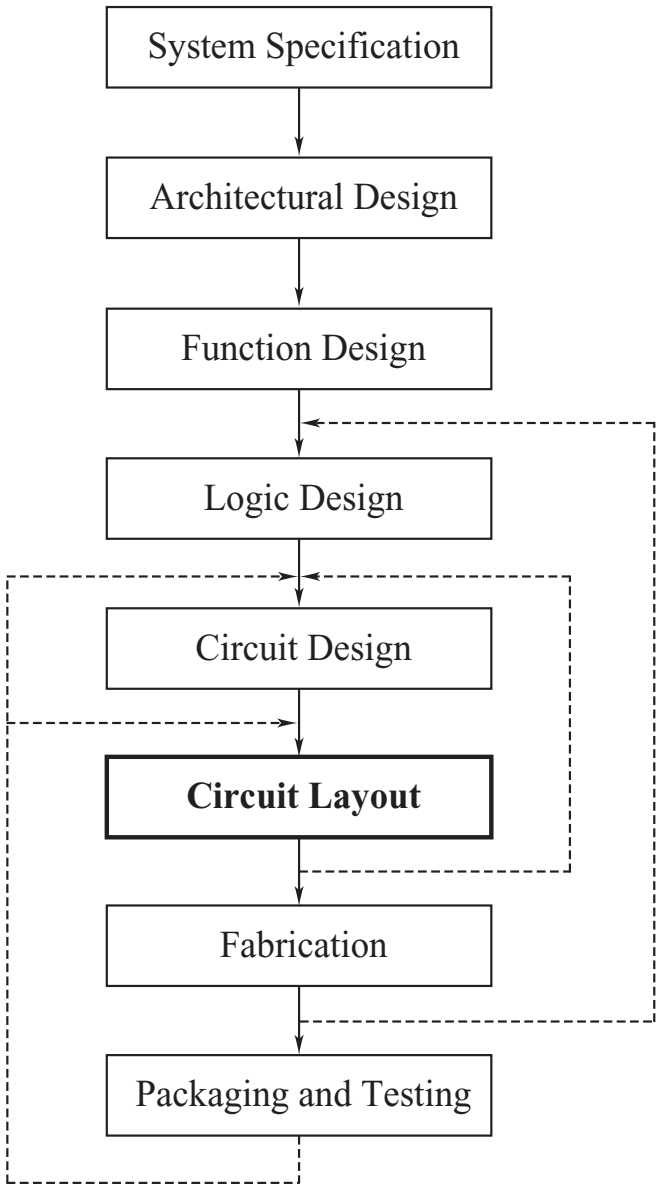


Figure 1.1: *VLSI circuit design procedure.*

Boolean expressions into a circuit representation while satisfying the power and timing requirements of the original design. A detailed circuit diagram that denotes the circuit elements is typically used to express the circuit design.

The physical design is also called the circuit layout. A *layout* is the geometric representation of a circuit. The main task of the circuit layout step is to convert the circuit representation of each component into a geometric representation, which is a set of geometric diagrams carrying out the intended logic function of the corresponding component. A *net* is a collection of pins required to be electrically connected. A circuit is usually described by a netlist, including modules and their interconnecting wires and nets. The geometrical positions of modules and the course of nets are determined by optimizing a given objective while satisfying certain design requirements. The circuit layout process is so complex that it is usually broken down into various subproblems. Each of these subproblems can be expressed as a nonlinear or discrete optimization problem. This step is described in more detail in the following section.

In the seventh step, the fabrication contains several phases: tape out of the layout data, preparation of the wafer, deposition, and diffusion of various materials on the wafer. Once the implemented prototype of a chip has been tested successfully, it may be mass produced.

In the packaging, testing and debugging step, which follows the steps above, the wafer is fabricated and diced into individual chips in a fabrication facility. Each chip may then be packaged and tested to ensure that all the design specifications are satisfied.

Note that the VLSI design cycle is a complex and iterative procedure with transformations of representations in various steps. Each step has an effect on the subsequent steps and any step may send feedback to revise previous steps. For instance, a certain step may be repeated and revised if errors and/or violations are detected.

Such a cycle may be repeated as the representation is iteratively improved to ensure that it meets the circuit specifications. Several *iterations* of the design procedure are performed until all the design specifications of a chip are met. In this thesis, the focus is on the *circuit layout* step of the VLSI design cycle.

1.2 Circuit Layout Cycle

The goal of circuit layout design is to physically realize the circuit obtained from its logic and functional description represented by the circuit diagram. Naturally, a circuit diagram is the input to the circuit layout design and the layout of the circuit is its output.

Circuit layout design is such a complex process that it is typically divided into several subproblems. It is usually solved as a sequence of intractable subproblems consisting of partitioning, floorplanning and placement, and routing. The *floorplanning* problem is the main focus of this research. A flow chart of the phases of circuit layout is given in Fig. 1.2. For a global perspective, partitioning, placement, and routing are briefly described in the following sections. Floorplanning will be described in Chapter 2. Fig. 1.3 illustrates a VLSI circuit (Pan, 2004).

1.2.1 Partitioning

Modern integrated circuits contain millions of elements. Because of resource, time, and computational power limitations, it is usually difficult to handle the entire circuit simultaneously. Hence circuit partitioning is used to split a large circuit into several relatively independent subcircuits such that their sizes are small enough to be handled by the existing physical design process (Kennings, 1994). The circuit performance depends on the partition; A good partition is likely to significantly reduce circuit

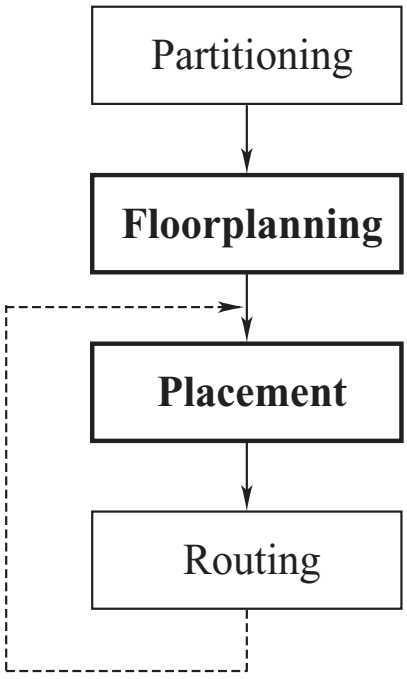


Figure 1.2: *VLSI circuit layout steps.*

layout expenses and improve circuit performance. Circuit partitioning is known to be NP-hard (Alpert and Kahng, 1995). There are a number of heuristic methods that provide approximate solutions to the partitioning problem. They can be classified into *constructive* and *iterative* algorithms (Johannes, 1996). Also, partitioning heuristics can be divided into two categories: *deterministic* and *stochastic* (Sait and Youssef, 1995).

1.2.2 Placement

The objective of placement is to minimize the total wirelength for all the nets and to find a minimum-area placement of modules that allows completion of the routing among modules (Sherwani, 1999). Traditionally, various placement techniques have

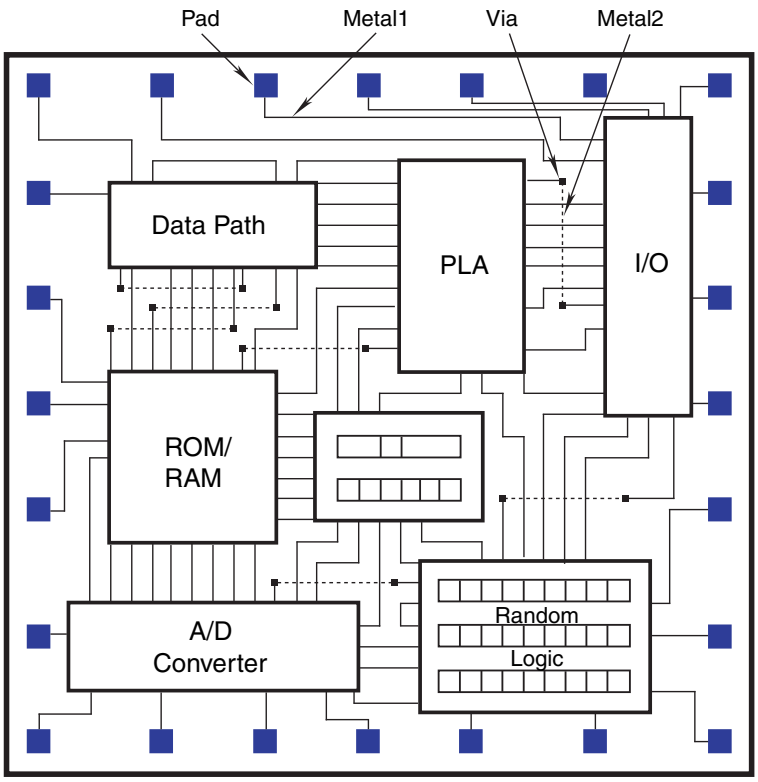


Figure 1.3: *Layout example of VLSI circuit (redrawn from Pan, 2004).*

been proposed to minimize the total area and wirelength (Shahookar and Mazumder, 1991). Research on placement can be classified into several major approaches: simulated annealing (e.g., Kleinhans *et al.*, 1991; Eisenmann and Johannes, 1998), min-cut based placement (e.g., Dunlop and Kernighan, 1985; Takahashi *et al.*, 1995), force-directed based placement (e.g., Alupoaei and Katkoori, 2002; Rajagopal *et al.*, 2003; Vorwerk *et al.*, 2004), evolution based placement (Kling and Banerjee, 1991), numerical optimization placement (Cheng and Kuh, 1984), and convex optimization placement (e.g., Etawil, 1999; Etawil *et al.*, 1999).

1.2.3 Routing

The main aim of routing is to complete the interconnection among modules of the circuit according to the specified netlists. The routing or wiring subproblem is solved to determine the geometric layout of the wires that connect the modules. As the routing problem is very difficult to solve even for small circuits, the routing stage is generally composed of two phases, global and detailed routing (Sait and Youssef, 1995; Sherwani, 1999). In global routing, the exact geometric details of each wire and pin are disregarded while the wires connecting the modules are determined. Following the global routing, detailed routing accomplishes point-to-point connections between the pins of the modules. Detailed routing involves channel routing and switchbox routing and is completed for each channel and switchbox.

1.3 Objectives of this Thesis

Floorplanning is a very important step in circuit layout design. Previous floorplanning research has focused on heuristics such as simulated annealing (e.g., Wong and Liu, 1986; Wong and Liu, 1989; Brasen and Bushnell, 1990; Young *et al.*, 2000*b*).

Optimization problems in floorplanning and placement are known to be NP-hard (Sait and Youssef, 1995; Sarrafzadeh and Wong, 1996; Sherwani, 1999). Previous research on optimization models for floorplanning lack convexity and so convergence to a global optimal solution cannot be guaranteed (e.g., Mogaki *et al.*, 1987; Onodera *et al.*, 1991; Sutanthavibul *et al.*, 1991; Chen and Kuh, 2000).

The aim of this research is to investigate and develop floorplanning strategies that use convex optimization. It will take advantage of the important property of convex optimization that any local minimum is a global solution of the problem. The proposed model should be able to generate a competitive optimal solution for area minimization and wirelength minimization. The designed and developed mathematical programming method should be effective, efficient, optimal, stable, robust, flexible, scalable, and applicable.

1.4 Contribution of this Thesis

The proposed optimization methodology demonstrates that optimization theory has powerful applications in VLSI floorplanning. Also, the model may be helpful in industrial applications. To the best of our knowledge, this is the first time that a convex-optimization-based method has been used for fixed-outline floorplanning. The contributions of this thesis can be summarized as follows:

1. A two-stage convex-optimization-based methodology is proposed for VLSI fixed-outline floorplanning (optimality).

2. The experimental results demonstrate that, compared with some state-of-the-art floorplanners, the performance of the new method is competitive on the MCNC and GSRC benchmarks. The running time is competitive on the MCNC

benchmark and the total wirelength is competitive on the MCNC and GSRC benchmarks. The quality of the total wirelength has significant impact on the floorplan in this thesis research. Minimization of total wirelength aims to minimize chip size, and thus cost, but also minimizes power and delay, which are proportional to the wirelength and wirelength squared, respectively. Minimizing the total wirelength is the main objective of most existing floorplanners. Therefore, the significantly improved quality in the total wirelength of our floorplanner is one of our most important contributions (effectiveness and efficiency).

3. The first-stage convex optimization model provides the relative positions of the modules on the floorplan.

4. The second-stage convex optimization model places and sizes the modules. *Overlap-free* and *deadspace-free* floorplans are achieved in a fixed outline with any specified percentage of **whitespace**. Our model provides flexibility to allow zero-whitespace or any specific percentage of whitespace (for buffer insertion, for example).

5. A Voronoi diagram (VD) is employed to obtain a planar graph and thus connect the two stages. The VD spreads modules and constructs a planar graph. VD-based module spreading is an efficient geometric method to evenly reduce module density in the congested areas. A relative position matrix technique is proposed to input the non-overlap constraints for the second stage model. This technique efficiently linearizes the non-overlap constraints.

6. An efficient approach is used to generate sparse relative position matrices that can improve computational efficiency. An interchange-free algorithm for local improvement of the floorplan is proposed.

7. Our methodology is capable of achieving scalability under even the tightest fixed-die constraints. Kahng (2000) criticizes classical floorplaning that seems to lack scalability and the ability to handle tight fixed-die constraints. Our methodology provides a greater improvement over other floorplanners as the number of modules increases (scalability).

8. A very important feature of our methodology is that the dimensions of the floorplans in our experiments comply with the original ones provided in the MCNC and GSRC benchmarks, and moreover zero-deadspace floorplans can be obtained. Thus, our approach is able to guarantee complete area utilization in a fixed-outline situation. Furthermore, our methodology also produces floorplans with any specified percentage of whitespace (stability, robustness).

9. The proposed methodology has been successfully applied to facility layout problems and excellent results have been obtained (Jankovits *et al.*, 2007) (applicability).

10. Although our methodology is currently focused on the soft module case and wirelength minimization, it is in principle applicable to the hard module case and area minimization. Also, the model can be applied in classical floorplanning (flexibility, applicability).

1.5 Flowchart of the Proposed Model

A flowchart is shown in Fig. 1.4 that represents the computation process of our model. It consists of initial configuration, first stage model, Voronoi diagram, second stage model, local improvement and final floorplan. Every step will be described in the following chapters, respectively. The second stage model takes most computation

time. Fig. 1.4 shows that these steps execute tasks in sequence and each component includes several sub-tasks. The following data are given and input into the first stage model: n modules with a list of areas a_i, $1 \leq i \leq n$; an interconnection matrix $C_{n\times n} = [c_{ij}], 1 \leq i, j \leq n$, where c_{ij} captures the connectivity between modules i and j that is symmetric (given by netlist). Finally, the coordinates (x_i, y_i), height h_i and width w_i for each module should be obtained and also the total wirelength should be minimized. The first stage model uses an attractor-repeller (AR) convex model, which globally minimizes the total wirelength, to provides relative position of modules. A Voronoi diagram (VD) is used to spread out modules on the floorplan and obtain a planar graph. The non-overlap constraints are enforced in the second stage model by a relative position matrix (RPM) achieved by VD stage. In the second stage, we use Semidefinite Programming (SDP) and Second Order Cone Programming (SOCP) model, which are both convex optimization problems, to size and place the modules thus obtain the final floorplan.

1.6 Organization of this Thesis

The remaining chapters of this thesis are organized as follows.

In Chapter 2 placement and floorplanning (both classical and fixed-outline) are introduced. We describe different types of placement and floorplanning and previous models and algorithms in these areas. Their advantages and limitations are explained. We also introduce zero-deadspace (ZDS) floorplanning, explaining the background and importance of ZDS floorplans. Wirelength estimation and the clique model for nets are also described in this chapter.

Chapter 3 addresses two-stage nonlinear optimization techniques for fixed-outline floorplanning in which the first stage uses a convex optimization technique and the

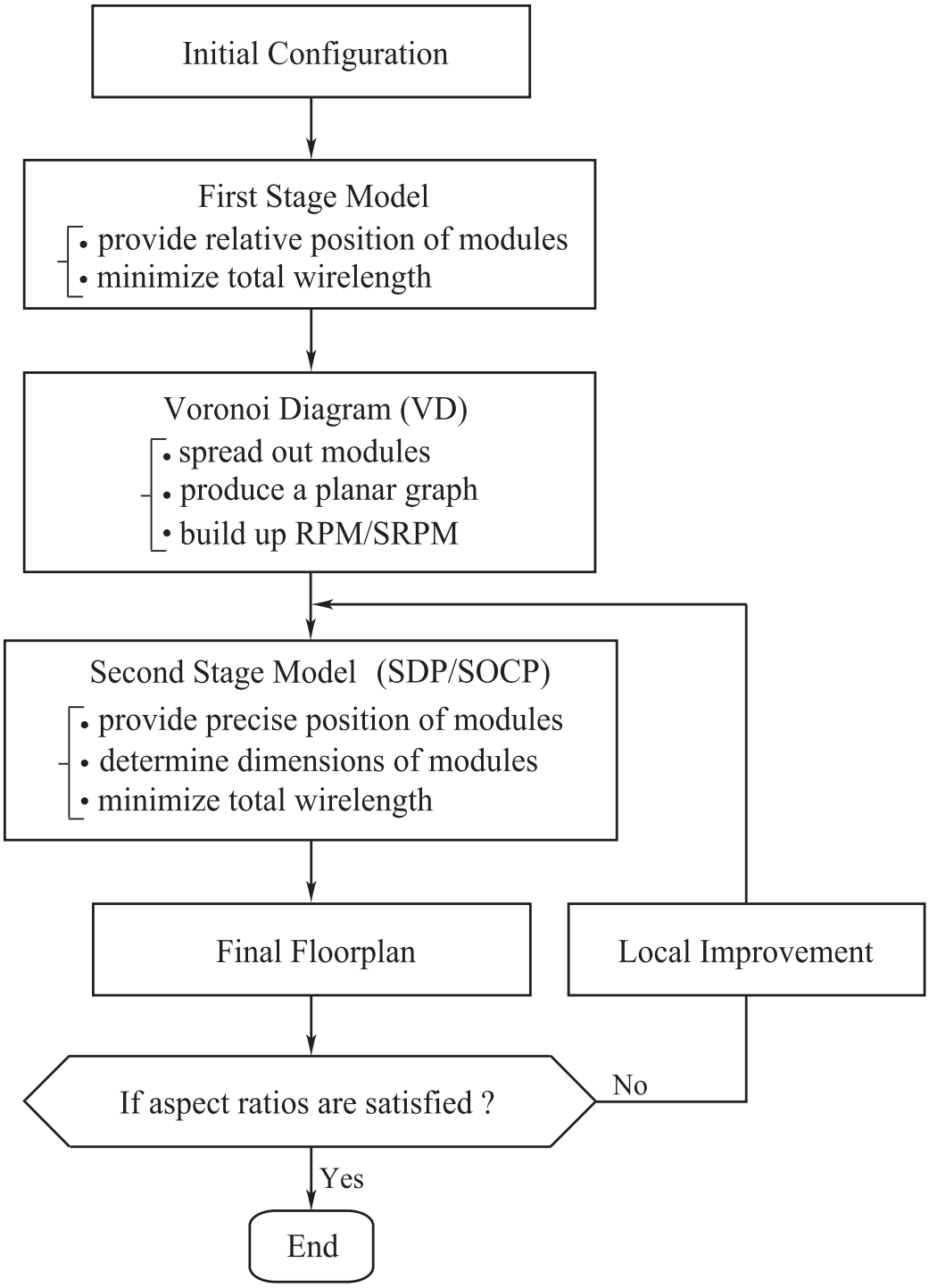


Figure 1.4: *Flowchart of the Proposed Model.*

second stage a nonlinear optimization model. An attractor-repeller model that is a convex optimization problem is introduced, which also provides the relative positions of the modules for the second stage in Chapter 5. The nonlinear optimization methodology is applied to obtain zero-deadspace floorplans by minimizing the total wirelength using rectilinear and quadratic objective functions.

Chapter 4 discusses the creation of a relative position matrix (RPM) provided by the first stage model and a sparse relative position matrix (SRPM). The SRPM technique is intended to reduce the computational effort. Additionally, we employ Voronoi diagrams (VD) and Delaunay triangulation (DT) to convert the relative position graph into a planar graph.

Chapter 5 describes in detail the second stage model, a completely convex optimization problem, based on Semidefinite Programming (SDP) and Second Order Cone Programming (SOCP) models. A variety of experiments on the MCNC and GSRC benchmarks using state-of-the-art floorplanners are performed in this chapter. The results demonstrate that the two-stage method is competitive with existing floorplanners on these benchmarks. The zero-deadspace fixed-outline floorplans are also obtained in this chapter.

In Chapter 6, an algorithm for interchange-free local improvement is described. The motivation is to avoid violating the required upper bound on the aspect ratios of the modules. The local relaxation of module position does not negatively affect the total wirelength.

Finally, in Chapter 7, the contributions of this thesis are highlighted and some important features are summarized. Recommendations for future work are presented. Feasibility for some future work is also discussed.

Chapter 2

VLSI Placement and Floorplanning

The rapid growth in the complexity, size, and density of VLSI systems has made placement and floorplanning challenging and these are critical phases that affect the performance of the resulting system on a chip. Some placement algorithms and floorplanning models are described in this chapter. Additionally, some background on wirelength estimation and the clique model is presented. We introduce placement because our first stage model, described in Chapter 3, may be regarded as a relative placement problem. Both placement and floorplanning arrange a set of overlap-free modules on a chip so that an appropriate objective is optimized. The main difference between placement and floorplanning is shown as follows: shapes of modules and pin positions on the periphery of circuit components are fixed for placement, while shapes of modules are flexible for floorplanning.

2.1 VLSI Placement

The VLSI cell placement problem is known to be NP-hard (Sait and Youssef, 1995; Sarrafzadeh and Wong, 1996; Sherwani, 1999). The input to the placement problem

is a set of modules with fixed shapes and fixed terminal positions, and a netlist representing connection information among modules. The placement typically aims to find the best locations for each module throughout the placement region while optimizing the appropriate objective functions. There are two approaches: constructive placement and iterative improvement placement (Kennings, 1997; Sherwani, 1999). A subset of modules has typically pre-assigned positions (I/O pads) (Fig. 1.3).

2.1.1 Constructive Placement

Constructive approaches generate a placement directly from the circuit netlists. These approaches can be classified into partition algorithms (e.g., Dunlop and Kernighan, 1985; Suaris and Kedem, 1988; Suaris and Kedem, 1989; Takahashi *et al.*, 1995; Huang and Kahng, 1997; Yildiz and Madden, 2001), placement by block Gauss-Seidel optimization (Tsay *et al.*, 1988), resistive network algorithms (Cheng and Kuh, 1984), and analytical algorithms (e.g., Sigl *et al.*, 1991; Kleinhans *et al.*, 1991; Vygen, 1997; Eisenmann and Johannes, 1998; Etawil *et al.*, 1999; Hu and Marek-Sadowska, 2002; Hur *et al.*, 2003; Kahng and Wang, 2004*b*; Viswanathan and Chu, 2004; Vorwerk *et al.*, 2004). The effective placement may be produced by combining partitioning and analytical algorithms (e.g., Wipfler *et al.*, 1983; Tsay *et al.*, 1988).

In constructive methods, there exists no initial placement. The coordinates of each module are viewed as variables. The advantage of these methods is that they can *rapidly* build up reasonably acceptable layouts for large circuits, as they take all the circuit interconnections into account simultaneously. As constructive approaches can not generate the highest quality placements, they are typically used to produce an initial placement for iterative improvement algorithms. The constructive approaches are more practical for placement problems with a huge number of cells such as a sea-of-gates placement (Tsay *et al.*, 1988; Sarrafzadeh and Wong, 1996; Sherwani, 1999).

2.1.2 Iterative Improvement Placement

Iterative improvement approaches start with an initial placement and modify it in search of an improved placement by making local changes to the existing placement. If a cost reduction is achieved, the new placement is accepted; otherwise, it is abandoned. The process iterates until no further improvement can be obtained. Iterative improvement approaches can be divided into two groups: deterministic and randomized algorithms. Deterministic algorithms accept *only* changes which produce an improved solution. In order to escape from local minima, randomized algorithms also accept, with a pre-defined small probability, changes that generate worse solutions. Simulated annealing is currently the most popular technique in terms of placement quality, but it takes an excessive amount of computation time (e.g., Sechen and Sangiovanni-Vincentelli, 1986; Wang *et al.*, 2000).

2.2 VLSI Floorplanning

Floorplanning has received much attention recently due to the increasingly high complexity of modern chip design and the importance of hierarchical design and intellectual property (IP) blocks. The future growth of VLSI circuits will rely on the development of physical design automation tools. The floorplanning problem consists of arranging a set of rectangular modules on a rectangular chip area so that an appropriate measure of performance is optimized. The resulting layout is called a floorplan. The floorplanning is also to decide the relative position of each module. Modules with relatively high connections are arranged close to one another for routability. In this section, we first describe floorplan representations. The problem statement and definition of the floorplaning will be presented, and classical and fixed-outline floorplanning will then be discussed.

2.2.1 Floorplan Representations

As the floorplanning is the early stage of physical design, it significantly determines the overall chip performance. Floorplanning is becoming an increasingly important tool for designing flows in the hierarchical design of Application Specific Integrated Circuit (ASIC) and System-on-Chip (SoC) (Adya and Markov, 2003; Bourbakis, 2008). Additionally, IP(module reuse)-based design methodology has been widely adopted and this makes floorplanning even more important. Floorplanning is a type of placement in which the embedded modules are flexible. The area of each rectangular module is assumed to be fixed while its height and width are allowed to vary subject to aspect ratio constraints (Wong and Liu, 1986; Sait and Youssef, 1995). A VLSI circuit consists of a collection of variable-dimension rectangular modules interconnected by nets, each module with its own prescribed fixed area (Fig. 1.3). The floorplanning problem is to determine the positions and dimensions of the circuit modules or IP modules on a chip to optimize the circuit performance such that all the modules are enveloped in the rectangular floorplan. A complete and formal representation of the floorplanning problem is described below (Wong and Liu, 1986; Sutanthavibul *et al.*, 1991; Sait and Youssef, 1995):

Generally, the inputs for a floorplanning problem are given as follows:

- a set of n rectangular modules $S = \{1, 2, \ldots, n\}$ with a list of areas a_i, $1 \leq i \leq n$;
- a partition of S into sets S_1 and S_2 representing the modules with fixed and free orientations respectively;
- an interconnection matrix $C_{n \times n} = [c_{ij}], 1 \leq i, j \leq n$, where c_{ij} captures the connectivity between modules i and j (we assume C is symmetric, i.e., $c_{ij} = c_{ji}$, given by netlist);

- values a_i for the area of each module i;
- bounds R_i^{low} and R_i^{up} on the aspect ratio R_i of each module i;
- bounds w_F^{low}, w_F^{up}, h_F^{low}, and h_F^{up} on the width and height respectively of the floorplan, for an instance of outline-free floorplanning; and
- values w_F and h_F for the width and height of the floorplan, for an instance of fixed-outline floorplanning.

The required outputs are as follows. The floorplanning problem is to determine the location, width, and height of each module on the floorplan so that:

- there is no overlap between the modules;
- coordinates (x_i, y_i), height h_i and width w_i for each module such that $w_i \times h_i = a_i, 1 \leq i \leq n$;
- $R_i^{low} \leq \frac{h_i}{w_i} \leq R_i^{up}$ for every module i with fixed orientation ($i \in S_1$);
- $R_i^{low} \leq \frac{h_i}{w_i} \leq R_i^{up}$ or $\frac{1}{R_i^{up}} \leq \frac{h_i}{w_i} \leq \frac{1}{R_i^{low}}$ for every module i with free orientation ($i \in S_2$);
- all the modules are enveloped in the floorplan;

and the total wirelength is minimized.

An optimum floorplan is achieved by optimizing the desired objective function. Possible objectives are (Lu *et al.*, 2008):

- minimize area (bus area);
- minimize wirelength;

- maximize routability;
- minimize power dissipation;
- minimize timing/delays; or
- any combination of the above.

Practically, it is difficult to achieve all these objectives simultaneously as they mutually conflict. Moreover, it is algorithmically difficult to take all the objectives into consideration simultaneously.

Floorplan representations are typically grouped into two categories: slicing and nonslicing. A slicing floorplan is a floorplan that is obtained by recursively cutting a rectangle horizontally or vertically. A binary tree with n leaves and n-1 nodes used to represent a slicing structure is called a slicing tree. Each node corresponds to a vertical cut line or horizontal cut line relevant to the model and each leaf to a basic rectangle (Sarrafzadeh and Wong, 1996). For example, a slicing floorplan is shown in Fig. 2.1A and its corresponding slicing tree is obtained as Fig. 2.1B.

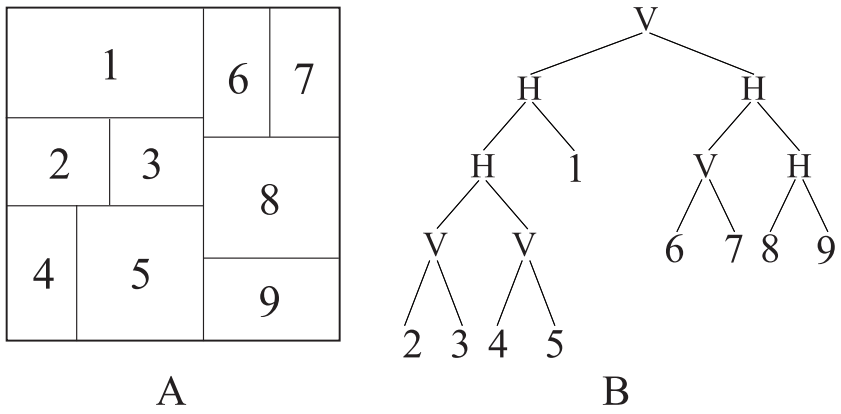


Figure 2.1: *Illustration of slicing floorplan. A: floorplan; B: slicing tree.*

A nonslicing floorplan is one that is not restricted to be slicing. The floorplan of Fig. 2.2A is nonslicing, where modules 4, 5, 6, 7, and 8 constitute a floorplan structure known as a *wheel*, which is also the smallest nonslicing floorplan. Its corresponding nonslicing tree is shown in Fig. 2.2B.

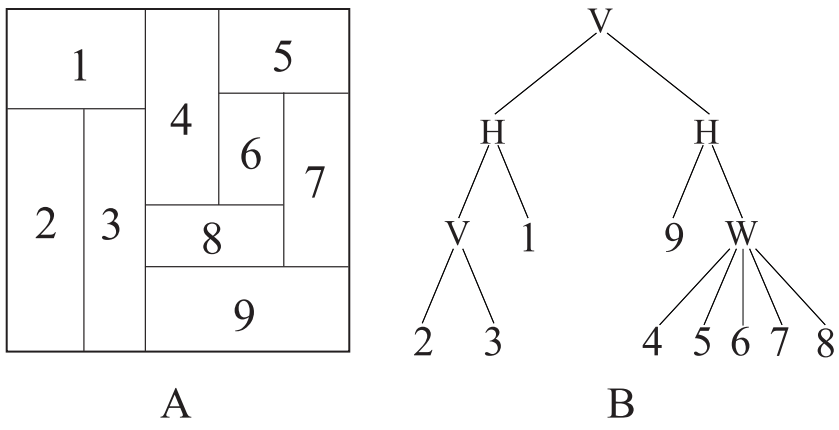


Figure 2.2: *Illustration of nonslicing floorplan. A: floorplan; B: nonslicing tree.*

The computation time of slicing floorplans is faster than that of nonslicing floorplans because the slicing structure has a smaller solution space. However, the nonslicing category is a more general representation for all types of packings. Otten (1982) first proposed a binary-tree representation for a slicing floorplan; Wong and Liu (1986) then improved on it by suggesting a normalized Polish expression.

There have been many studies using nonslicing floorplan representations in VLSI floorplanning. The representations of the geometric relationships between modules have been extensively studied so as to implement certain algorithms such as simulated annealing more effectively. Typical representations in VLSI floorplanning are as follows: SP (sequence pair) (Murata *et al.*, 1995; Murata and Kuh, 1998), BSG (bounded-sliceline grid) (Nakatake *et al.*, 1996; Kang and Dai, 1997), O-tree (Guo

et al., 1999; Pang *et al.*, 2000), CBL (corner block list) (Hong *et al.*, 2000; Zeng *et al.*, 2003), B*-tree (Chang *et al.*, 2000; Maruvada *et al.*, 2002), MB*-tree (multilevel B*-tree) (Lee *et al.*, 2003), TCG (transitive closure graph) (Lin and Chang, 2001), and CS (corner sequence) (Lin *et al.*, 2003).

2.2.2 Classical and Fixed-Outline Floorplanning

In terms of practical physical designs, floorplans can be classified as variable-die or fixed-die. The variable-die methodology uses 2-layer metal processes while the fixed-die methodology is a modern standard that is suitable for metal processes with three or more layers. Floorplanning handling fixed-die is called *fixed-outline* floorplanning, while *classical* floorplanning handles variable-die (Adya and Markov, 2003). Fixed-outline floorplanning works with a fixed floorplan outline, and aims to simultaneously minimize an estimate to the total interconnecting wirelength and overlap, and possibly also timing (Kahng, 2000), while classical floorplanning seeks to place the modules inside a rectangular floorplan whose outline is variable so as to minimize the area of the floorplan.

Classical floorplanning aims to find a feasible floorplan to envelop every module without any overlap while minimizing the overall area (see Fig. 2.3). It is similar to a generalized two-dimensional bin-packing problem. Even this simplified version of floorplanning is known to be NP-hard (Baker *et al.*, 1980; Sait and Youssef, 1995).

Fixed-outline floorplanning is significantly more difficult than outline-free floorplanning (Adya and Markov, 2003; Kahng, 2000). Fixed-outline floorplanning typically aims to optimize the following circuit performance:

- total wirelength of the circuit;
- delay of critical path;

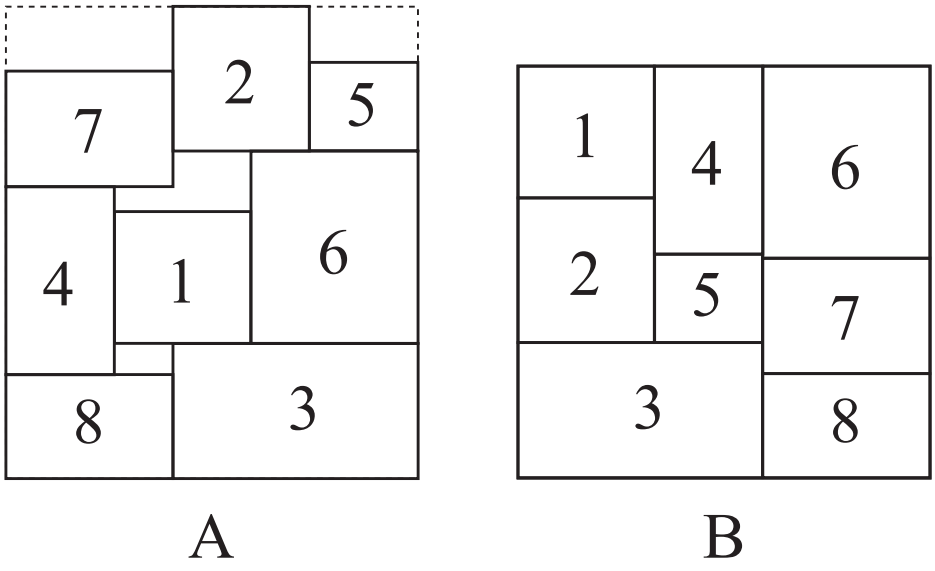


Figure 2.3: *Comparison of classical floorplanning and fixed-outline floorplanning. A: classical floorplan; B: fixed-outline floorplan.*

- estimated timing performance;
- routability;
- heat dissipation;
- a combination of two or more of the above criteria.

Kahng (2000) pointed out five main problems with classical floorplanning: 1) an excessive attention to packing-driven instead of connectivity-driven approaches and corresponding benchmarks; 2) an unnecessary limitation of modules to only rectangular, L, or T shapes; 3) a lack of emphasis on the register-transfer level (RTL)-down methodology context; 4) a lack of attention to the fixed-outline constraints; and 5) an inability to handle scalability. Adya and Markov (2003) have attempted to deal

with some of the issues above. However, the last two concerns are difficult. Our methodology is focused on solving these two problems.

We use a 4-module instance to demonstrate fixed-outline floorplanning. Four modules represented by $M1$, $M2$, $M3$, and $M4$ are interconnected via nets and also connected to I/O pads $P1$, $P2$, $P3$, ..., and $P8$ that are fixed on the boundary of the floorplan shown in Fig. 2.4. The netlists describe a list of all the nets in the circuit. For instance, Net 1 connects $P1$, $M3$, $M1$, and $P6$. The sum of the area of the four modules is equal to the entire area of the floorplan. A deadspace-free fixed-outline floorplan can be achieved by enveloping all four modules within the floorplan. We first randomly pack four modules inside the outline (Fig. 2.4A). The measured wirelength is 127 and the length for Net 1 is 36. It is easy to observe that module 3 connects to $P1$ and $P7$ that are located around the top-left corner, and module 4 connects with $P2$ that is fixed in the left boundary. Therefore, modules 3 and 4 should be moved to the left side of the floorplan (Fig. 2.4B). The improved floorplan has a total wirelength of 91 and Net 1 becomes 36. The optimal floorplan finally attains a total wirelength of 32 with Net 1 as 18 (Fig. 2.4C). The wirelength and length for Net 1 with these cases are shown in Table 2.1.

Table 2.1: Comparison of four-module case for fixed-outline floorplnning

Illustration	**Wirelength**	**Length for Net 1**
Fig. 2.4A	127	36
Fig. 2.4B	91	36
Fig. 2.4C	32	18

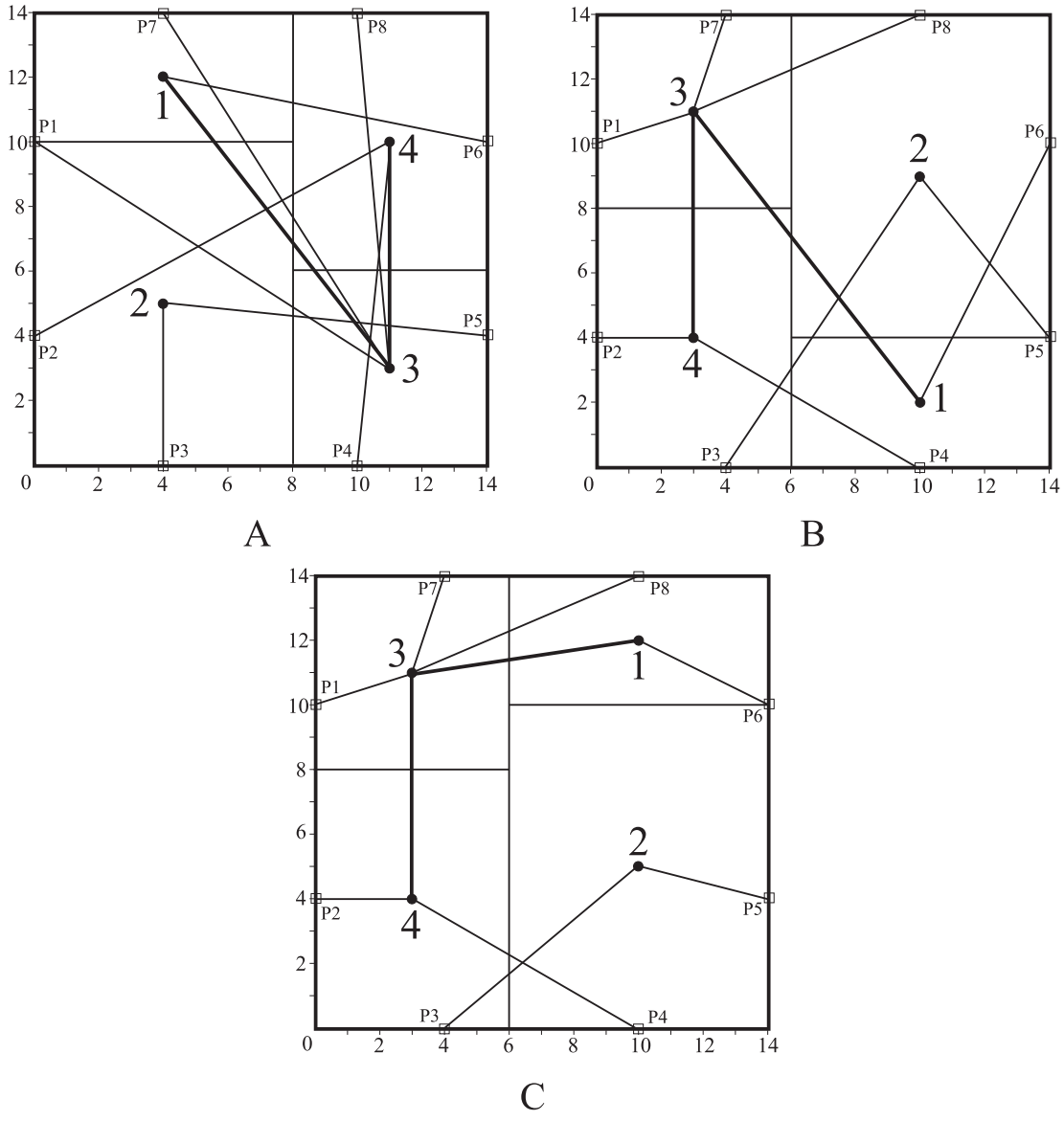


Figure 2.4: *Instance of four-module case for fixed-outline floorplanning. A: random floorplan; B: improved floorplan; C: final floorplan.*

2.3 Previous Research on Classical Floorplanning

The floorplanning problem has been tackled using various approaches: *constructive* approaches, *iterative* approaches, *knowledge* approaches, and *mathematical programming* approaches (Sait and Youssef, 1995).

Constructive approaches start from a seed module and then select one or several modules to add to the partial floorplan. This procedure continues until all modules have been selected. This approach produces a floorplan directly from the netlist. Typical constructive approaches are cluster growth, partitioning and slicing, connectivity clustering, and rectangular dualization.

Iterative approaches search for an improved floorplan by making local changes until a feasible floorplan is gained or no more improvements can be obtained. Popular iterative techniques are simulated annealing (Sechen, 1988*b*; Wong *et al.*, 1988; Wong and Liu, 1989; Murata *et al.*, 1998; Ranjan *et al.*, 2001; Young *et al.*, 2000*b*), force-directed interchange/relaxation (Brasen and Bushnell, 1990; Choi and Kyung, 1991; Youssef *et al.*, 1995), and genetic algorithms (Cohoon *et al.*, 1991; Fernando and Katkoori, 2008).

Knowledge based approaches implement a knowledge expert system (Odawara *et al.*, 1985; Dickinson, 1986; Ackland, 1988; Ball *et al.*, 1994; Xu *et al.*, 2006). For example, Ball *et al.* (1994) and Xu *et al.* (2006) used fuzzy-based floorplanning.

Mathematical programming approaches use linear programming (Sutanthavibul *et al.*, 1991; Kim and Kim, 2003; Chu and Young, 2004), the branch-and-bound algorithm (Wimer *et al.*, 1989; Onodera *et al.*, 1991), convex programming (Moh *et al.*, 1996), or nonlinear programming (Qi *et al.*, 1994*a*).

In this section we focus on rectangular dualization, simulated annealing, force-directed methods, and mathematical programming methods. The emphasis is on the

latter three as they are most closely related to our methodology. The floorplans are classified into slicing and nonslicing floorplans, which are different representations of the floorplans (Sarrafzadeh and Wong, 1996). With the representation of floorplans, the floorplanning still needs to use some techniques such as simulated annealing and force-directed floorplanning to implement it. We review previous work using both floorplanning approaches and floorplan representations since they are closely related.

2.3.1 Rectangular Dualization

A graph theoretic rectangular dualization method may be used to construct rectangular floorplans (Lai and Leinwand, 1988; Sait and Youssef, 1995). For instance, Tani *et al.* (1988), and Lokanathan and Kinne (1989) proposed rectangular dual graph approaches. A quadratic area for floorplanning with one or more linear constraints is minimized by approximation using heuristics. The modules are compacted by a rectangular dual approach. The rectangular dualization problem is transformed into a matching problem on bipartite graphs. Floorplanning design using rectangular dualization follows five steps (Sait and Youssef, 1995):

Step 1: represent the given circuit netlist as a graph;

Step 2: transform this graph into a planar graph (He, 1997);

Step 3: convert this planar graph into a planar triangulated graph (Kozminski and Kinnen, 1984);

Step 4: check whether this planar triangulated graph is a properly triangulated graph (PTP);

Step 5: seek a rectangular dual graph for floorplans (Tani *et al.*, 1988; Lokanathan and Kinne, 1989).

With a properly triangulated graph, a rectangular dual may be constructed and thus a floorplan may be formed (Sait and Youssef, 1995).

2.3.2 Simulated Annealing

Simulated annealing (SA) has been widely applied in VLSI layout design (e.g., Ho *et al.*, 2004; Murata *et al.*, 1998; Ranjan *et al.*, 2001; Sechen, 1988*b*; Tang *et al.*, 2001; Wong *et al.*, 1988; Wong and Liu, 1989; Young *et al.*, 2000*b*; Chen and Chang, 2006; Lee *et al.*, 2007; Law and Young, 2008). SA is particularly useful when the solution space of the problem is not well understood. When using SA, the solution is often restricted to be a slicing floorplan (Wong and Liu, 1989) (recall that a slicing floorplan is a floorplan that is obtained by recursively cutting a rectangle horizontally or vertically). The quality of floorplans obtained using SA is very competitive but SA is time-consuming and takes substantial computation resources. Therefore, the applicability of SA is restricted to floorplans with a small number of modules (Wong and Liu, 1986; Wong and Liu, 1989). Murata *et al.* (1998) proposed a sequence-pair-based SA model to implement floorplans by an adaptation strategy. Ranjan *et al.* (2001) suggested a constructive technique for predicting floorplan metrics which is used to obtain a fast and accurate SA-based floorplan prediction. Tang *et al.* (2001) proposed a model based on SA with sequence pair (SP) utilization. The SP is used for floorplanning where fast SP evaluation can be achieved to improve the quality of floorplans. Adya and Markov (2003) also used SP utilization to represent the topology of a floorplan, but proposed a moving technique based on slack computation and SA to optimize the wirelength as well as the aspect ratio of the soft modules. Ho *et al.* (2004) considered an orthogonal SA with an efficient generation mechanism to solve large floorplanning problems. Among the SA methods listed above, the model of Adya and Markov (2003) is a state-of-the-art approach, and they reported little deadspace in their floorplans for the MCNC benchmarks. Therefore, we compare their reported results with our own results in Chapter 3.

2.3.3 Force-directed Methods

Force-directed methods are analogous to Hooke's law for the mechanics of a spring. The floorplanning problem is solved by a set of simultaneous linear equations that give the equilibrium positions of the modules. The force-directed method is used to find a timing- and connectivity-driven topological arrangement. Then the entire area of the floorplan is minimized from the topological arrangement. The forced-directed approach is particularly suitable for a two-stage model.

Usually, competitive efficiency can be obtained by combining the force-directed method with other approaches (Brasen and Bushnell, 1990; Choi and Kyung, 1991; Youssef *et al.*, 1995; Mohamood *et al.*, 2007). For example, Brasen and Bushnell (1990) implemented the timing-driven MHERTZ floorplanner using a two-step strategy. In the first stage, a sequence of gradient descent manipulations based on force-directed objective functions are adopted and timing constraints are taken into consideration. In the second step, an SA optimization algorithm is used to minimize the total wirelength and overall area of the floorplan and to remove module overlaps. Youssef *et al.* (1995) combined a force-directed approach with a constraint graph approach in a two-stage method. In the first stage, a force-directed approach solves the timing- and connectivity-driven floorplanning problem. The overall area is then minimized in the second stage while legalizing the floorplan. Qi *et al.* (1994*b*) suggested a force-directed relaxation approach used for timing-driven floorplanning that optimizes the interconnection delays between modules. Murofushi *et al.* (1990) also implemented timing-driven floorplanning based on a force-directed approach that minimizes the total wirelength and removes overlaps. They consider area utilization simultaneously.

2.3.4 Mathematical Programming Methods

Mogaki *et al.* (1987) use linear programming (LP) to minimize chip width and height simultaneously within given constraints on module size, relative module position, and width of the intermodule routing space. Sutanthavibul *et al.* (1991) formulated the problem as a mixed integer LP problem that minimizes the overall area of the floorplan. Routability is simultaneously considered in the LP model. Kim and Kim (2003) combined LP and SA: the floorplan is obtained from an SP by considering an LP model or an alternative construction method. (For simplicity, they assume that the pins of all the modules are located at the centres of the modules.) Chu and Young (2004) applied a Lagrangian relaxation technique to solve their proposed LP problem. They improved the area usage in the chip by changing the shapes and dimensions of the soft modules to fill up the empty space. Chen and Kuh (2000) minimized chip area using LP. Timing constraints are also modelled in the LP problem. The method is iterative, and proceeds by solving a sequence of LP problems until it converges to a local minimum.

Prasitjutrakul and Kubitz (1989) incorporated timing and geometrical constraints into the process with the formulation of the mathematical programming problem. The floorplanning problem with path-delay constraints is modelled and mathematically formulated as a constrained nonlinear programming problem. Herrigel and Fichtner (1989) suggested an analytic optimization technique based on nonlinear programming to minimize wirelength and chip area simultaneously for floorplanning with R- or L-shaped modules. The pin positions are also optimized in their second stage. Wang *et al.* (2003) proposed Lagrangian relaxation for soft module floorplanning based on an SP algorithm. They used an average-value method to compute initial values for the Lagrange multipliers so as to reduce the running time. Computational effort was significantly reduced but the total wirelength is longer. Ying and Wong (1989)

proposed a two-stage approach to minimize total wirelength and area by an unconstrained minimization model. Their model minimizes total wirelength and area using a potential energy method in the first stage, and removes overlap in the second stage. There is unavoidably much deadspace and the computational effort is high.

Some approaches use convex optimization (e.g., Chen and Fan, 1998; Moh *et al.*, 1996; Murata and Kuh, 1998; Rosenberg, 1989; Wimer *et al.*, 1988). An important property of convex optimization problems is that any local minimum is a global solution of the problem. The minimal area floorplanning problem was formulated as an optimization problem by Wimer *et al.* (1988). The existence and uniqueness of a floorplan is proven but aspect ratio constraints on modules are not taken into consideration. Moh *et al.* (1996) formulated the problem as a geometric programming problem and thus transformed it into a convex optimization problem. The numbers of variables and constraints in their convex formulation for area minimization were greatly decreased by Chen and Fan (1998).

Murata and Kuh (1998) combined the convex optimization technique with a non-slicing floorplan representation including variable modules and preplaced modules. While their method to find a solution is very time-consuming, they do achieve very little deadspace in their floorplans for the MCNC benchmarks, and thus their results are compared to ours in Chapter 3 and Chapter 5. The rectangular modules they used may encompass soft modules, hard modules, and semi-soft modules. Soft modules have fixed areas with continuously variable aspect ratios. Hard modules have fixed aspect ratios and pin locations. Semi-soft modules have fixed areas with discrete aspect ratios.

Onodera *et al.* (1991), and Wimer *et al.* (1989) proposed branch-and-bound techniques. Their approaches search the solution space and find optimal solutions by a branch-and-bound technique subject to constraints on critical nets and the shape of

a chip. However, the solution obtained is only near-optimal.

Finally, the recent development of semidefinite programming (SDP) has led to its application to VLSI physical design. Vandenberghe and Boyd (1996) applied SDP to VLSI transistor sizing and pattern recognition by using ellipsoids. More recently, Takouda *et al.* (2005) proposed a mixed integer SDP model to find global lower bounds for the floorplanning problem. Their technique successfully provides global lower bounds for the smaller MCNC benchmark circuits.

2.4 Previous Research on Fixed-Outline Floorplanning

As described previously, the objective of fixed-outline floorplanning problem is to pack all the modules within a given fixed floorplan outline that simultaneously minimizes wirelength and overlap. Most floorplanning studies focus on area minimization for variable-die floorplanning. Recall that handling fixed-outline floorplanning is significantly more difficult than outline-free floorplanning. Only recently have floorplanners begun tackling fixed-die floorplanning by minimizing the total wirelength (Adya and Markov, 2003; Adya *et al.*, 2004; Tang *et al.*, 2006).

Parquet (Adya and Markov, 2003) and Capo (Adya *et al.*, 2004) are two typical floorplanners for dealing with fixed-die constraints when minimizing total wirelength. Adya and Markov (2003) suggested a technique based on slack computation and SA to optimize the wirelength by using SP that represents the topology of a floorplan. The fixed outline is satisfied by using their local search technique. Capo (Adya *et al.*, 2004) effectively combines min-cut placement with SA to form a fixed-outline floorplanner. At the stage of partitioning and placement, fixed-outline floorplanning is completed while taking routability into account.

Recently, several fixed-outline floorplanners have been developed based on SA. Chen *et al.* (2005) and Chen *et al.* (2008) developed a so-called IMF multilevel floorplanner using a two-phase technique. IMF consists of a top-down partitioning phase that partitions the floorplan into subregions, and a bottom-up merging phase that merges subregions. The total wirelength is optimized via min-cut partitioning in the first phase and by SA-based fixed-die floorplanning implemented in each subregion in the second phase. The floorplanner IMFAFF incorporates accelerative fixed-outline floorplanning (AFF) into IMF, and is 11 times faster than IMF but at the expense of a 9% increase in the wirelength. Chen and Chang (2006) proposed an adaptive fast-SA scheme based on a B*-tree floorplan representation. The total wirelength can be effectively minimized by dynamically varying the weights of the objective function. Fixed-die floorplanning is achieved by taking outline constraints into account.

A fast geometric algorithm, called Traffic, has recently been proposed for wirelength minimization without SA (Sassone and Lim, 2006). Traffic also decomposes floorplanning into two steps. In the first step, the modules are grouped by local and global connectivity into several layers as rows by a partitioning algorithm. In the second step, the modules in the same layer are moved to trapezoidal form by wirelength minimization. Squeezing the constructed rows transforms them into the required floorplans. The main advantage of this approach is that it is much faster than SA-based models.

Genetic algorithm (GA)-based fixed-outline floorplanners have recently been studied (Lin *et al.*, 2004; Chen *et al.*, 2007). Lin *et al.* (2004) proposed a model that places all the modules in enlarged floorplan outlines. The outlines were enlarged to 115% of the total module area. Based on the GA concept, Lin *et al.* (2006) developed a genetic clustering algorithm for slicing floorplans considering fixed-outline and boundary constraints. Those modules that should have stronger connectivity with I/O pads are

forced along the chip boundary to improve the total wirelength. Two evolutionary operations for searching the solution space to find a fixed-outline floorplan based on the GA are described by Chen *et al.* (2007).

Most recently, some other approaches were also proposed for fixed-outline floorplanning. For example, Liu *et al.* (2005) achieved fixed-outline floorplanning by a local search method based on instance augmentation adopting a sequence-pair representation of the floorplan. Their algorithm explores both instances and sub-instances in search of feasible and optimal solutions in the solution space but has difficulty attaining zero-deadspace. Feng and Mehta (2006) suggested a geometry-based moving approach which minimizes the standard deviation of module densities. Zhan *et al.* (2006) proposed an analytical approach for fixed-outline floorplanning dealing with soft modules that minimizes total wirelength in two phases. In the first phase, the wirelength and area distribution density of modules are minimized. In the second phase, the overlap area and wirelength are optimized to achieve an overlap-free floorplan. Cong *et al.* (2006) developed a floorplanner that uses recursive bipartitioning flow to minimize total wirelength and arranges soft and hard modules within a fixed-outline floorplan. Chen and Yoshimura (2007) proposed a technique called Insertion After Remove (IAR) for solution perturbation for the SA and arrangement of modules in SP. They suggested a new objective function with width, height, and aspect ratio of the chip. This objective function improves the success rate and minimizes area and total wirelength simultaneously.

There have been few studies on fixed-outline floorplanning by *convex* optimization (Moh *et al.*, 1996; Murata and Kuh, 1998; Luo *et al.*, 2007; Luo *et al.*, 2008*b*). Moh *et al.* (1996) formulated the floorplanning problem as a geometric program and thus transformed it into a convex optimization problem. The aspect ratios of the modules are also considered in their approach. As the geometric program problem

is formulated subject to height and width constraints in a given partition, their approach is suitable for area minimization. Murata and Kuh (1998) proposed a convex optimization approach that minimizes the total wirelength by SA using a nonslicing floorplan and SP representation. Their approach is time-consuming and has difficulty achieving zero-deadspace.

2.5 Zero-Deadspace Fixed-Outline Floorplanning

The zero-deadspace (ZDS) floorplanning problem aims to pack a given set of soft modules inside a fixed-die floorplan without any deadspace and/or overlap among modules. In modern VLSI design, deadspace-free and overlap-free fixed-outline floorplanning is required because the size of the fixed-die has usually been pre-determined during the chip synthesis process. Therefore, it is necessary to pack a zero-deadspace and zero-overlap layout into the fixed die. Furthermore, top-level routing and pin assignment are iteratively performed. Fixed-die floorplanning is incorporated in a top-down hierarcgical flow that uses multi-level floorplanning. One of the reasons to seek ZDS and zero-overlap layout is that there are no unused resources and full area utilization is required at top level (Kahng, 2000; Adya and Markov, 2003). Mosaic floorplan is a term used by Hong *et al.* (2000) if the floorplan achieves ZDS.

Fixed-outline floorplanning is important in designing flows in the hierarchical design of ASICs and SoC (Kahng, 2000; Adya and Markov, 2003). It is relevant to ASIC design and its formulation is easily enforced by application-specific constraints such as alignment, abutment, order, region and symmetry (Adya and Markov, 2003).

Several authors have investigated ZDS in floorplanning (e.g., Wimer *et al.*, 1988; Wang and Chen, 1993; Peixoto *et al.*, 2000; Young and Wong, 1997; Kahng, 2000; Cong *et al.*, 2006; Feng and Mehta, 2004; Mehta and Sherwani, 2000; Feng *et al.*, 2004).

Methods for ZDS floorplanning include the network-flow method (e.g., Wimer *et al.*, 1988; Feng *et al.*, 2004), the partition-based method (Cong *et al.*, 2006), and the resistive network approach (Wang and Chen, 1993). Previous research work achieved ZDS on slicing floorplans (Peixoto *et al.*, 2000; Young and Wong, 1997; Cong *et al.*, 2006) and on nonslicing floorplans (Wang and Chen, 1993). Some approaches allow general floorplans (Wimer *et al.*, 1988; Cong *et al.*, 2006).

The first algorithms are due to Wimer *et al.* (1988) and Wang and Chen (1993). However, these algorithms did not consider the aspect ratios of the soft modules. ZDS floorplans with area minimization were achieved by Wang and Chen (1993) by transforming the nonslicing floorplanning problem into a resistive network problem. Wimer *et al.* (1988) described a network flow and planar graph approach that implements ZDS layout for area minimization and also proved the existence and uniqueness of a ZDS floorplan.

There was then little research on ZDS until modern hierarchical design flows on ASIC and SoC became popular. ZDS was now regarded as a constraint instead of as an objective in the formulation (Adya and Markov, 2003). Kahng (2000) questioned and challenged the supremacy of classical floorplanning and his work contains the earliest suggestion that ZDS *fixed*-outline floorplanning is more consistent with the requirements of modern design. He developed a formulation called the Perfect Rectilinear Floorplanning Problem (PRFP) that produces provable ZDS perfectly packed rectilinear floorplans with the fixed-outline constraint. Actually, the ZDS floorplan is a compacted floorplan.

Some methods implement and achieve ZDS floorplans by experiment (e.g., Wimer *et al.*, 1988; Feng *et al.*, 2004; Cong *et al.*, 2006). Others propose theoretical analysis and potential results (e.g., Peixoto *et al.*, 2000; Young and Wong, 1997; Mehta and Sherwani, 2000; Hong *et al.*, 2000). For example, Peixoto *et al.* (2000) and Young

and Wong (1997) used theoretical analysis to propose upper bounds on the total area minimization. Mehta and Sherwani (2000) considered a grid data representation and developed corresponding algorithms by replacing the module rectilinear shapes such as L-shapes with arbitrary rectilinear shapes. They stated that their algorithms are theoretically able to implement ZDS.

Some methods directly achieve ZDS floorplans, while others obtain ZDS floorplans by minimizing the whitespace inside floorplans. For example, the algorithm of Cong *et al.* (2006) directly obtains a provable ZDS fixed-outline floorplan by a recursive top-down area bi-partitioning algorithm. In contrast, in other algorithms, whitespace is initially allowed. Feng and Mehta (2004) use an iterative refinement and area redistribution approach to achieve ZDS. The ZDS is potentially a result of their min-cut max-flow floorplanner. Feng *et al.* (2004) introduced three BFS (breadth first search)-based algorithms following the work of Kahng (2000). They used a min-cost max-flow network formulation that packs rectilinear-shaped modules into a ZDS floorplan. Theoretically, some floorplan representations can implement the compacted floorplan and thus a ZDS floorplan. For instance, the CBL (Corner Block List) representation invented by Hong *et al.* (2000) potentially achieves ZDS floorplans without empty rooms by assigning each room only one module in the slicing and nonslicing floorplans.

The ZDS floorplanning problem may be defined as follows:

Definition 2.5.1 (*ZDS Floorplanning*) *Given chip area $\mathcal{A}$ and n modules with areas a_i ($i = 1, 2, ..., n$), the objective is to arrange all the modules inside the floorplan without any overlap, and to minimize the total wirelength such that $\sum_{i=1}^{n} a_i = \mathcal{A} = \bar{w}_F \times \bar{h}_F$, where $\bar{w}_F$ and $\bar{h}_F$ are the fixed width and height of the floorplan.*

In our methodology, the sum of the module areas is equal to the area $\mathcal{A}$ of the

chip. We enforce this constraint into the second stage formulation by the height $\bar{h}_F$ and width $\bar{w}_F$ of the chip, whose area is $\mathcal{A}$ ($\bar{w}_F \times \bar{h}_F = \mathcal{A}$). We are able to obtain ZDS solutions and also to allow any percentage of deadspace in the fixed-outline floorplan. One of the main advantages of our methodology is that it is not restricted to slicing structures.

Whitespace in the floorplan is also an important consideration. Whitespace is needed by designers due to power-density and temperature restrictions. Buffer insertion that requires area at appropriate places on the timing paths is necessary. The locations of buffers cannot be predicted prior to floorplanning. Therefore, whitespace throughout the floorplanning is required by buffer insertion (Adya *et al.*, 2006).

2.6 The Clique Model for Nets

A hypergraph is a generalization of a graph in which the edges link not just two but any number of vertices. A circuit to be placed on a chip can be represented by a hypergraph $G(V, E)$. The vertices V represent the modules of the circuit, and the hyperedges E represent the nets.

In the clique model, a k-pin net is typically transformed into $k(k-1)/2$ two-pin nets with certain weights. Different values are used for the weight of the two-pin nets. Commonly used values are $2W/k$ (Kleinhans *et al.*, 1991; Eisenmann and Johannes, 1998) and $W/(k-1)$ (Vygen, 1997) where W is the weight of the k-pin net. In the star model, a hypergraph is converted into a graph by adding an artificial centre vertex so as to construct two-pin nets by connecting this to the existing modules. Obviously, a k-pin net generates k two-pin nets. The clique and star models for a six-pin net are illustrated in Fig. 2.5.

We use the clique model to transform hypergraphs to two-pin nets and the result-

ing weights are used in our objective functions that minimize wirelength and area. The fact that weights between *pairs of modules* are used in the objective functions supports the use of the clique model. In our experiment in Chapter 3 and Chapter 5, we use $W/(k-1)$ for the weight of the two-pin nets.

The connectivity information is typically expressed in the form of a symmetric $n \times n$ matrix C, where element c_{ij} is the connectivity between modules i and j. If w_{ij} represents the weight of a two-pin net (clique) between modules i and j, the sum of weights w_{ij} on all the cliques between modules i and j forms the connectivity c_{ij}.

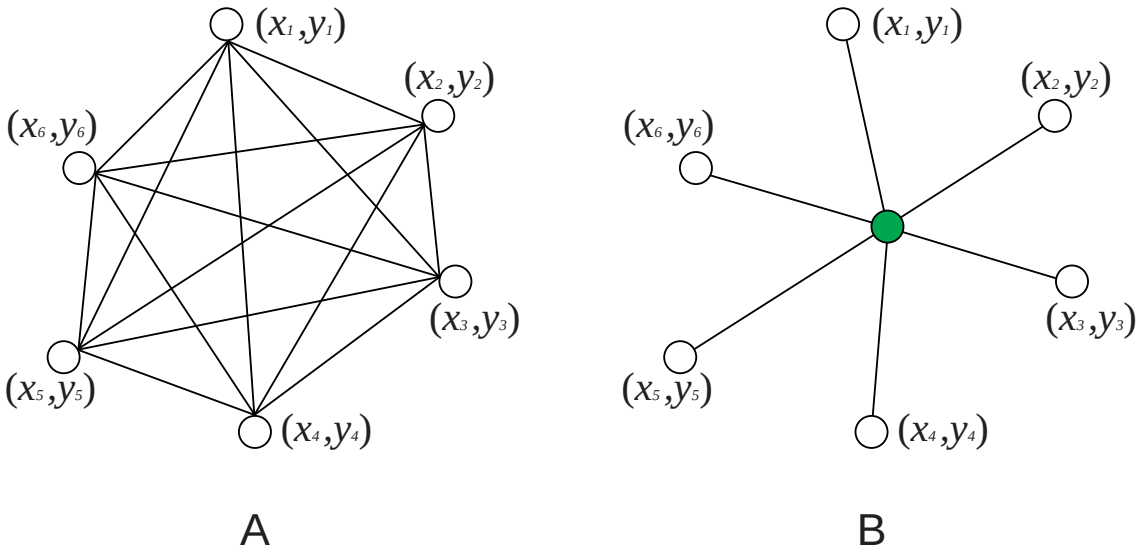


Figure 2.5: *An example of clique and star models converted from a six-pin net. A: the clique model; B: the star model.*

2.7 Wirelength Estimation

We estimate total wirelength to evaluate the quality of the floorplan. Wirelength estimation has been extensively studied and used in floorplanning and placement (e.g., Gamal, 1981; Hamada *et al.*, 1996; Hebgen and Zimmermann, 1996; Pedram and Preas, 1989).

A circuit is commonly represented by a hypergraph $G_h = (V, E)$ with its vertex set $V = \{v_1, v_2, \ldots, v_m\}$ representing modules and its set of hyperedges $E = \{e_1, e_2, \ldots, e_n\}$ representing the nets connecting the modules. The first step in our approach is to construct a simple graph capturing this information. A clique model is commonly used to form a graph from the hypergraph describing a circuit. The resulting graph is represented by a symmetric $n \times n$ adjacency matrix C, where entry $c_{ij} \geq 0$ is an aggregate measure of the connectivity between modules i and j. A positive weight W is associated with the net to indicate its criticality. In the clique model, a k-pin net is typically transformed into $k(k-1)/2$ two-pin nets with certain edge weights. If W is the weight of the k-pin net, commonly used values for the edge weights are $2W/k$ (e.g., Eisenmann and Johannes, 1998; Kleinhans *et al.*, 1991) and $W/(k-1)$ (Vygen, 1997). We use the clique model for transforming hypergraphs to two-pin nets, and a value of $1/(k-1)$ for the edge weights of a net with k pins. If w_{ij} represents the weight of a two-pin net between modules i and j, their total connectivity c_{ij} is obtained by summing up w_{ij} over all the cliques involving both i and j. The resulting connectivities are used in the objective functions of our mathematical programming models.

The techniques of wirelength estimation can be classified into:

1. *Complete graph technique*: Every module in the netlist is connected to every other module. For a k-pin net, the complete graph has $k(k-1)/2$ edges.

2. *Source-sink connection technique*: One of the modules is assumed to be a source and the rest to be sinks. All the sinks are connected to the source by separate wires. The total wirelength of this method is high and therefore it is suitable for approximation for heavily congested placement and floorplanning.

3. *Minimum chain technique*: Start from a vertex and connect the closest one, and then the next closest, in a certain sequence, until all the vertices are enclosed. The wirelength is estimated by all the lengths of chain included.

4. *Spanning tree estimation*: A spanning tree of a connected graph is a subgraph which is a tree and contains all the vertices of the graph. In an n-pin net, the algorithm searches the distances between all potential pairs of pins and connects the smallest (n-1) edges without cycles to form the spanning tree.

5. *Steiner tree estimation*: A Steiner tree is a tree in a distance graph which spans a given subset of vertices (Steiner Points) with the minimal total distance on its edges. The so-called Steiner tree problem is to find a minimum Steiner tree, *i.e.*, a Steiner tree of minimum length. This problem is known as the Steiner tree problem and finding the minimum Steiner tree is known to be NP-complete. A Steiner tree is the shortest route when connecting a set of pins in modules (Sait and Youssef, 1995). In an n-pin net, the net is able to start from any pin along its length to connect to other pins of the net to form a Steiner tree.

6. *Half-perimeter wirelength estimation*: The quality of the floorplanning is measured in terms of the half-perimeter wirelength (HPWL). This simple and efficient method for estimating the wirelength measures the half-perimeter of a box which bounds the pins of a corresponding net.

In the past two decades, rectilinear (Manhattan) distance, Steiner tree, and spanning tree have been used as typical wirelength measures in VLSI placement and floorplanning (e.g., Sait and Youssef, 1995; Sarrafzadeh and Wong, 1996). Wire seg-

ments using rectilinear distance connecting the different modules are parallel to the x and y axes. Therefore, it is particularly suitable for two-pin nets. The minimum Steiner tree that connects an n-pin net generates the minimum wirelength. However, constructing the minimum Steiner tree is a difficult task. The rectilinear minimum spanning tree (RMST) is a good choice to implement a multipin net that has minimum wirelength with linear distance. RMST may be resulted from approximation to Steiner tree. Hence, finding the RMST is also an intractable problem as the number of nets in a modern circuit is large (for instance, there are 1893 nets and 4358 pins for the GSRC n300 circuit). Consequently, a simpler and more efficient approximation measure for the wirelength is necessary.

The HPWL is the most popular and efficient approximation of the wirelength. The HPWL is equal to the weighted sum of *half-perimeters* of the bounding boxes that encompass the modules incident on each net. This technique is used extensively becasue its calculation is relatively simple and accurate. An instance that estimates the wirelength of a net as the half-perimeter of the smallest enclosing rectangle is illustrated in Fig. 2.6. The total HPWL $\mathcal{L}_t$ is computed as the sum of the half-perimeter of the individual nets composed in the netlist; this is thus given by

$$\mathcal{L}_t = \sum_{i=1}^{N} (h_i + w_i) \tag{2.1}$$

where h_i and w_i are the horizontal and vertical dimensions respectively of the bounding box of net i.

Another reason for the popularity of the HPWL is that it is correlated with shortest paths being used to route more nets in multilayer over-the-cell routing (Kahng, 2000). There have been a number of studies on the minimization of the HPWL. For example, Jackson and Kuh (1989) and Weis and Mlynski (1987) minimize the HPWL using LP, while Hur and Lillis (1999) use a max-flow computation

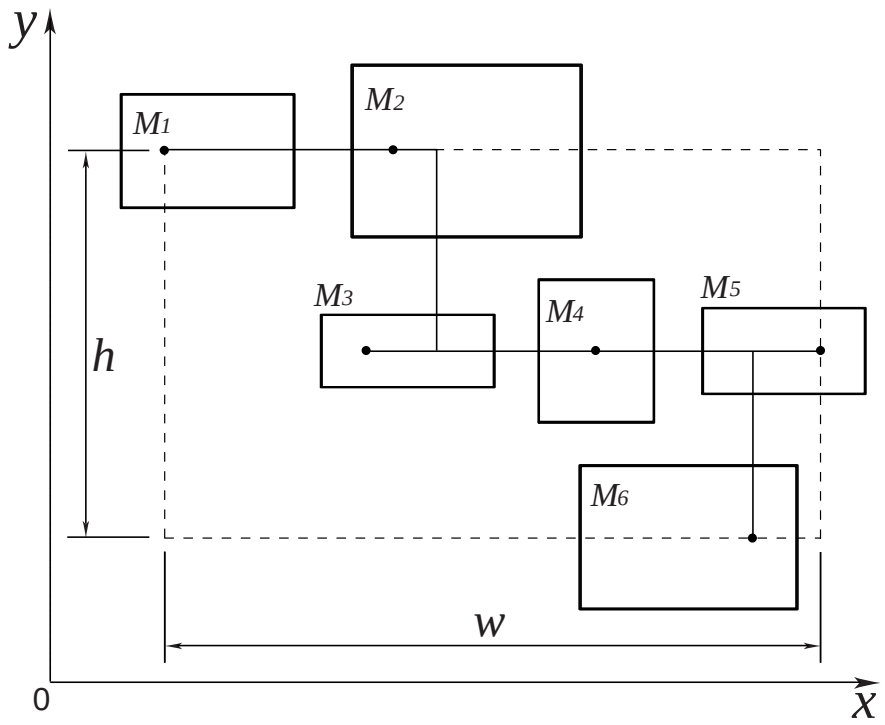


Figure 2.6: *Estimation of the wirelength of a net by the half-perimeter of the minimum rectangle enclosing the modules in the net*, HPWL=h+w.

method. Kennings and Markov (2000) proposed an analytical algorithm based on a convex approximation of the HPWL. Their regularization technique is able to approximate the HPWL with arbitrarily small relative error, and thus to minimize the HPWL using unconstrained convex optimization.

Although we minimize different estimates of the wirelength in our models (i.e., rectilinear distance, quadratic distance, and direct HPWL known as Methods A, B, and C in Section 3.4.2), we always use the HPWL to measure the quality of our final floorplans.

2.8 Benchmarks of Test Circuits

Two benchmarks, Microelectronics Center of North Carolina (MCNC) and Gigascale Systems Research Center (GSRC), are used as test circuits to evaluate the performance of the proposed methodology in this thesis. The circuit characteristics of MCNC (MCNC, 2004) and GSRC (GSRC, 2006) are presented in Table 2.2 and Table 2.3, respectively.

Table 2.2: The standard MCNC benchmark circuits

Circuit	# of modules	# of nets	# of I/O pads	# of pins	area
apte	9	97	73	287	46.56
xerox	10	203	2	698	19.35
hp	11	83	45	309	8.30
ami33	33	123	42	522	1.16
ami49	49	408	22	953	35.4

Table 2.3: The standard GSRC benchmark circuits

Circuit	# of modules	# of nets	# of I/O pads	# of pins	area
n10	10	118	69	248	22.17
n30	30	349	212	723	20.86
n50	50	485	209	1050	19.86
n100	100	885	334	1873	17.95
n200	200	1585	564	3599	17.57
n300	300	1893	569	4358	27.32

Each circuit entry in Tables 2.2 and 2.3 consists of the number of modules, nets, I/O pads, pins, and chip area. The test cases provided by these two benchmarks cover a large spectrum of circuits. A great variety in circuit characteristics qualifies the benchmarks to evaluate the robustness, scalability, efficiency, and effectiveness of our model. Furthermore, results using these benchmarks are available in the literature for comparison.

There are two choices for the locations of the I/O pads when performing experiments. First, the I/O pads are originally fixed at the locations given by the benchmarks. Second, the I/O pads are scaled to the boundary of the floorplans. We will use these two cases to compare our experimental results with others in the literatures.

An illustration of the numbers of nets, I/O pads, and pins corresponding to the number of modules for the MCNC benchmarks is shown in Fig. 2.7 and for the GSRC benchmarks in Fig. 2.8.

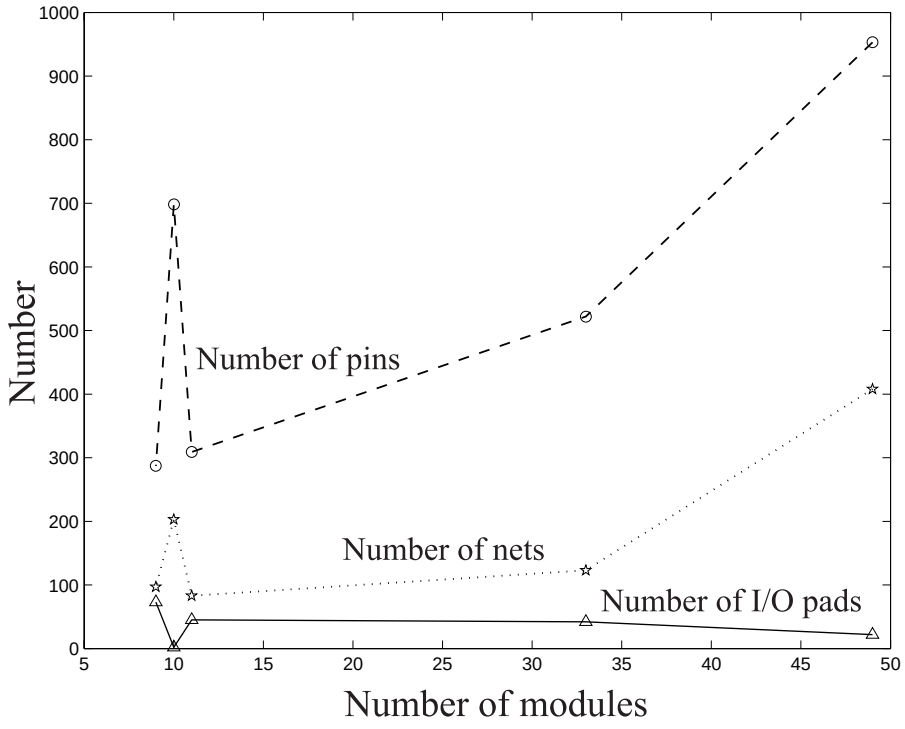


Figure 2.7: *Illustration of the numbers of nets, I/O pads, and pins corresponding to the number of modules for the MCNC benchmarks.*

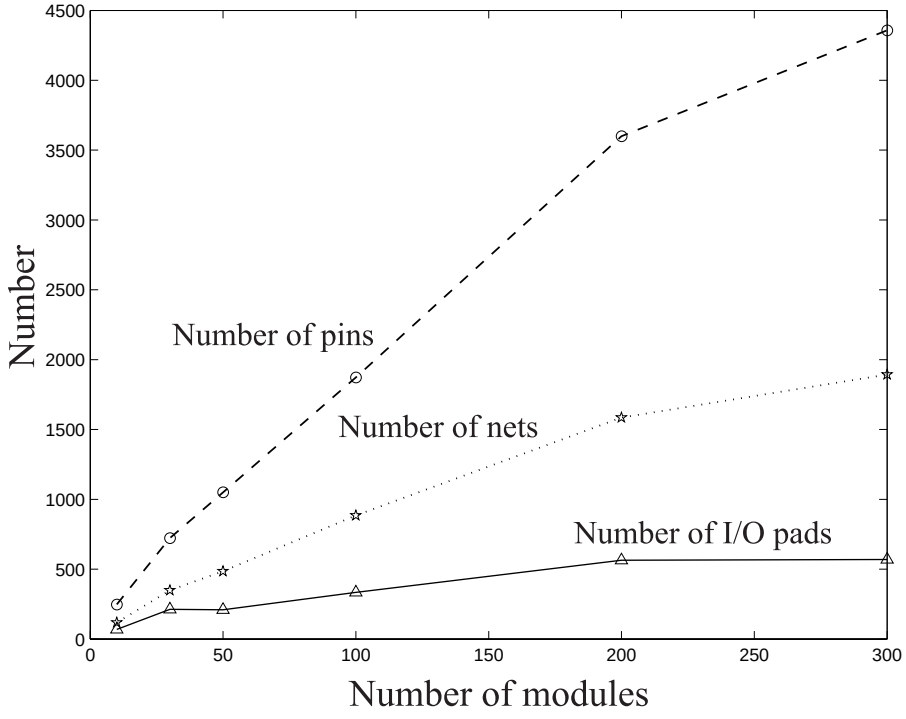


Figure 2.8: *Illustration of the numbers of nets, I/O pads, and pins corresponding to the number of modules for the GSRC benchmarks.*

2.9 Summary

This chapter has shown the importance of placement and floorplanning in layout design. It has also provided a brief survey of previous approaches. These approaches were classified according to their algorithms and features in the placement and floorplanning. Floorplanning is becoming an increasingly important tool. First, floorplanning is the early stage of physical design; it significantly determines the overall chip performance. Second, it is becoming an increasingly important tool for designing flows in the hierarchical design of ASICs and SoC (Kahng, 2000; Adya and Markov, 2003). Third, it is closely related to placement and it is a feasibility study of the placement (Sait and Youssef, 1995).

Classical floorplanning and fixed-outline floorplanning, and zero-deadspace fixed-outline floorplanning were described. Our methodology is motivated by the fixed-die problem, also called fixed-outline floorplanning, but in principle it is applicable to variable-die as well. Fixed-outline floorplanning is significantly more difficult than variable-die floorplanning and it plays an important role in state-of-the-art hierarchical methods for multi-level large-scale circuit design (ASIC and SoC). First, ideally, at the top level of design, all the space resources should be used. Second, in industry, before the use of floorplanning during the chip synthesis process, usually the die size and package have been chosen. Hence, fixed-outline floorplanning with non-overlap and zero-deadspace is an important, realistic, but very difficult problem.

The clique model was introduced to transform a hypergraph into a graph with a weight for each net. Total wirelength is commonly used to compare the quality of different floorplans; various approaches of the estimation of the wirelength were briefly analyzed and the HPWL measure method was introduced in this chapter. In the next chapter, our mathematical programming model will be proposed for floorplanning.

Chapter 3

A Nonlinear Optimization Methodology

We propose a two-stage nonlinear-optimization-based methodology specifically designed to perform fixed-outline floorplanning by minimizing wirelength while simultaneously enforcing aspect ratio constraints on soft modules and handling a zero deadspace situation. In the first stage, a convex optimization globally minimizes an approximate measure of wirelength. This stage provides the relative position of modules without considering their shape and dimension. Modules are allowed to overlap. Using the solution from the first stage as a starting point, the second stage minimizes the wirelength by sizing the modules subject to the prescribed aspect ratios, and ensuring no overlap (Luo *et al.*, 2007).

Another important consideration is the use of soft modules, meaning that the area of each rectangular module is assumed to be fixed while its height and width are allowed to vary subject to given aspect ratio constraints (Wong and Liu, 1986; Sait and Youssef, 1995). At this stage of the design process, the modules have not been laid out in detail yet, and so the floorplanner will perform better if it is allowed to

change module dimensions in a controlled manner. (This control typically comes in the form of aspect ratio constraints.) However, few studies using soft modules have been done in the fixed-outline context. Cong *et al.* (2006) do consider this in their PATOMA package. The PATOMA method requires the presence of deadspace (in particular since it uses a partitioning-based approach), and all the results reported have significant amounts of deadspace. The approach of Zhan *et al.* (2006) also requires the presence of deadspace (the authors use a minimum of 10% deadspace in all the results).

An important feature of our method is that it achieves complete area utilization using soft modules in a fixed-outline situation, something that has not been achieved in the literature to date. Computational results on the standard MCNC benchmark demonstrate that the model is competitive with other approaches.

The first stage model introduced in this chapter will also provide the relative position of modules for the second stage in Chapter 5. This chapter is organized as follows. Previous nonlinear-programming-based models are described in Section 3.1. Rectilinear and quadratic objective functions are discussed in Section 3.2. The first stage model based on convex optimization is introduced in Section 3.3. The second stage model is proposed in Section 3.4. The experimental results and comparisons with other state-of-the-art floorplanners are described in Section 3.5. Finally, this chapter is summarized in Section 3.6.

3.1 Previous Research

There are previous studies on obtaining relative position of modules by facility layout approaches. We start to describe some previous models. Previous studies on nonlinear optimization methods to the layout problem are reviewed. Let us first review the

DISCON model (Drezner, 1980) for the layout problem.

- Each module labelled 1, 2, . . . , N is represented as a circle with radius r_i, i = 1, 2, . . . , N.
- The location of each module 1, 2, . . . , N is given by the coordinates of its centre denoted as (x_i, y_i).
- The non-negative cost per unit distance between modules i and j is expressed by c_{ij}, which is equivalent to the weight between modules.
- The distance between modules i and j measured from centre to centre by Euclidean distance (l_2 norm) is denoted by d_{ij}

$$d_{ij}=\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}. \tag{3.1}$$

Then the DISCON model is formulated as a minimization problem:

$$\begin{aligned} &\min_{(x_i,y_j)} \quad \sum_{1\le i<j\le N} c_{ij}d_{ij} \\ &\text{s.t.} \\ &\qquad r_i + r_j - d_{ij} \le 0, \quad \forall \ \ 1 \le i < j \le N \end{aligned} \tag{3.2}$$

It is obvious that the objective function

$$\sum_{1\le i<j\le N} c_{ij}d_{ij} \tag{3.3}$$

attempts to make distance d_{ij} as small as possible. It is able to attract pairs of circles i and j towards each other and so can act as an attractor. On the other hand, the constraints in the DISCON model

$$r_i + r_j - d_{ij} \le 0, \quad \forall \ \ 1 \le i < j \le N \tag{3.4}$$

push any pair of circles away from each other to prevent them from overlapping.

A nonlinear optimization layout technique (NLT) (van Camp *et al.*, 1991) was proposed to solve the facility layout problem using a three-phase method. Modules are approximated by circles whose radii are proportional to the areas of the modules. The relative positions of the modules are captured in the first two stages by solving a relaxation of the NLT model as formulation (3.5). The centres of the modules inside the floorplan are evenly distributed in the first stage while the second stage decreases the overlap among modules. Finally, the third stage determines the final solution, starting from the result obtained from the second stage. The areas of the modules and the floorplan are fixed and rectangular. The objective of the NLT model is to optimize the height and width of each module.

$$\min_{(x_i,y_j),w_i,h_i,w_F,h_F} \sum_{1\le i<j\le n} c_{ij}d_{ij}$$

s.t.

$$\begin{aligned}
&|x_i - x_j| \ge \frac{1}{2}(w_i + w_j) \quad \text{if} \quad \frac{1}{2}(h_i + h_j) > |y_i - y_j|,\\
&|y_i - y_j| \ge \frac{1}{2}(h_i + h_j) \quad \text{if} \quad \frac{1}{2}(w_i + w_j) > |x_i - x_j|,\\
&\frac{1}{2}w_F \ge x_i + \frac{1}{2}w_i \quad \forall\, i,\\
&\frac{1}{2}h_F \ge y_i + \frac{1}{2}h_i \quad \forall\, i,\\
&\frac{1}{2}w_F \ge \frac{1}{2}w_i - x_i \quad \forall\, i,\\
&\frac{1}{2}h_F \ge \frac{1}{2}h_i - y_i \quad \forall\, i,\\
&\min(w_i, \ h_i) - l_i^{\min} \ge 0 \quad \forall\, i,\\
&\min(w_F, h_F) - l_F^{\min} \ge 0,\\
&l_i^{\max} - \min(w_i, \ h_i) \ge 0 \quad \forall\, i,\\
&l_F^{\max} - \min(w_F, h_F) \ge 0,
\end{aligned} \tag{3.5}$$

where (x_i, y_i) and d_{ij} are as defined previously, and

- w_i, h_i are the width and height of module i;
- w_F, h_F are the width and height of the floorplan;
- $l_i^{\min}$, $l_i^{\max}$ are the minimum and maximum allowable lengths for the shortest side of module i; and
- $l_F^{\min}$, $l_F^{\max}$ are the minimum and maximum allowable lengths for the shortest side of the floorplan.

The Spring Embedding (SE) model for facility layout problem proposed by Castillo and Sim (2004) is inspired by a dynamic spring system. Again let the distance d_{ij} be

$$d_{ij}=\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}, \tag{3.6}$$

then the total energy of springs is formulated in a dynamic spring system

$$E = \sum_{i,j \in M} \frac{1}{2} c_{ij} d_{ij}^2 \tag{3.7}$$

The r_i and r_j are radii of the modules i and j modelled by circles, respectively, $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, N$. If (x_i, y_i) and (x_j, y_j) are the coordinates of the centre points of modules i and j, the generated SE model is:

$$\min_{(x_i,y_j),w_F,h_F} \quad \sum_{i,j \in M} \frac{1}{2} c_{ij} d_{ij}^2 \quad + \sum_{i,j \in M} \max\Big\{0, \omega_{ij}(r_i + r_j - d_{ij})\Big\}$$

s.t.

$$\begin{aligned}
&\frac{1}{2}w_F \geq x_i + r_i \ \text{ and } \ \frac{1}{2}w_F \geq r_i - x_i, \ \text{ for all } i \in M, \\
&\frac{1}{2}h_F \geq y_i + r_i \ \text{ and } \ \frac{1}{2}h_F \geq r_i - y_i, \ \text{ for all } i \in M, \\
&w_F^{up} \geq w_F \geq w_F^{low}, \\
&h_F^{up} \geq h_F \geq h_F^{low},
\end{aligned} \tag{3.8}$$

where ω_{ij} is a constant, $\omega_{ij} > 0$. The mechanism on the relationships of d_{ij} and r_i, r_j of the SE model is similar to that of the AR model in formulation (3.12). If $d_{ij} < r_i + r_j$, so modules i and j overlap, $\omega_{ij}(r_i + r_j - d_{ij})$ is penalized in the total energy function to generate a repulsive force. If $d_{ij} \geq r_i + r_j$, then modules i and j do not overlap. The penalty term thus disappears from the objective function and there is a net attractive force between modules i and j. The use of the max function, which is non-smooth, is detrimental in the nonlinear programming solver. An optimal solution of relative locations of modules will be obtained by solving this problem.

3.2 Rectilinear and Quadratic Objective Functions

We will investigate the influence of rectilinear and quadratic objective functions on floorplanning and placement (Sigl *et al.*, 1991).

The quadratic objective function has the property that it is continuously differentiable and therefore can be minimized by solving a system of linear equations.

By using linear programming with a large number of constraints, the half perimeter can be minimized. In fact, for medium-size circuits (e.g., circuit $ami33$), minimization of the half perimeter wirelength can be achieved in an acceptable time.

We use the following example to demonstrate the difference between rectilinear and quadratic objective functions. Three modules, two fixed and one movable, are to be placed. In Fig. 3.1, two fixed modules A and B are connected to movable module C by three nets with lengths x, y, and z, respectively. The result $x = y = 0$ in Fig. 3.1A arises from minimizing the rectilinear objective function $L_r = x + y + z$, while the result $x = y = \frac{1}{2}z$ in Fig. 3.1B arises from minimizing the quadratic objective function $L_q = x^2 + y^2 + z^2$.

Several observations can be made from the experiments above. A rectilinear

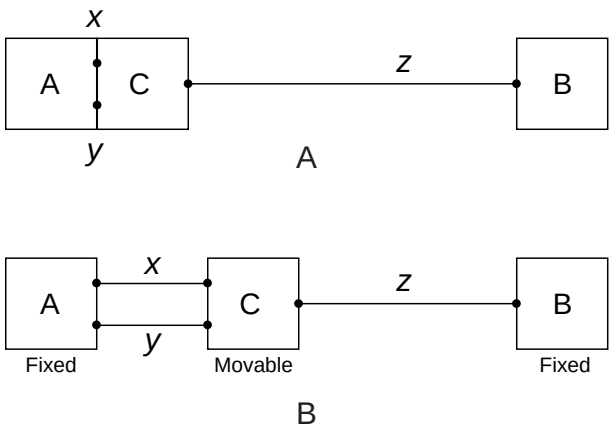


Figure 3.1: *An example of optimal placement based on different objective functions. A: rectilinear; B: quadratic.*

objective function is normally used in placement because more tracks and more feedthroughs are needed; these are not important factors in floorplanning. On the other hand, a quadratic objective function tends to reflect adjacency demands more accurately. It also tends to route shorter lengths for the long nets (net z in Fig. 3.1B), and to increase the length of short nets (nets x and y in Fig. 3.1B). Therefore, the standard deviation of the net lengths is larger for a rectilinear objective function than for a quadratic objective function (Sigl *et al.*, 1991).

3.3 First Stage Model

This is a *relative placement* stage providing the relative position of modules. The dimension and shape of the modules are not considered.

Let (x_i, y_i) and (x_j, y_j) denote the coordinates of the centres of modules i and j. Following Du and Vannelli (1998), the minimization of the quadratic wirelength with

respect to a given target distance may be formulated as a quartic objective function of the form:

$$\min \sum_{i,j \in \mathrm{Net}_k} c_{ij} \left[(x_i - x_j)^2 + (y_i - y_j)^2 - d_{ij}^2 \right]^2, \tag{3.9}$$

where c_{ij} is the connectivity between modules i and j, and d_{ij} denotes the specified target distance between modules i and j. This model is an early use of a target distance. Solving the wirelength minimization problem (3.9) results in a placement without overlap if the distances represented by the d_{ij} terms are chosen appropriately. Further extensions of the concept of target distance to placement were introduced by Anjos and Vannelli (2002), Anjos and Vannelli (2006), and Etawil *et al.* (1999).

We now describe the target distance methodology employed in the first stage of our approach. Let each module i be represented by a circle of radius r_i, where r_i is proportional to $\sqrt{a_i}$, the square root of the area of module i. Following Anjos and Vannelli (2002), we define the target distance for each pair of circles i, j as:

$$t_{ij} := \alpha (r_i + r_j)^2, \tag{3.10}$$

where $\alpha > 0$ is a parameter. To prevent circles from overlapping, the target distance is enforced by introducing a penalty term:

$$f(\frac{D_{ij}}{t_{ij}}),$$

where $f(z) = \frac{1}{z} - 1$ for $z > 0$, and $D_{ij} = (x_i - x_j)^2 + (y_i - y_j)^2$. The objective function is thus given by

$$\sum_{1 \le i < j \le n} c_{ij} D_{ij} + f(\frac{D_{ij}}{t_{ij}}). \tag{3.11}$$

If $r_i + r_j \le d_{ij}$, there is no any overlap between circles (Fig. 3.1A), and the repeller term is zero or becomes a negative value. So, the attractor in the objective function applies an attractive force to the two circles. Conversely, if $r_i + r_j > d_{ij}$, the repeller

term (i.e. the penalty function) holds a positive value that acts as a repulsive force to prevent the circles from overlapping (Fig. 3.2B).

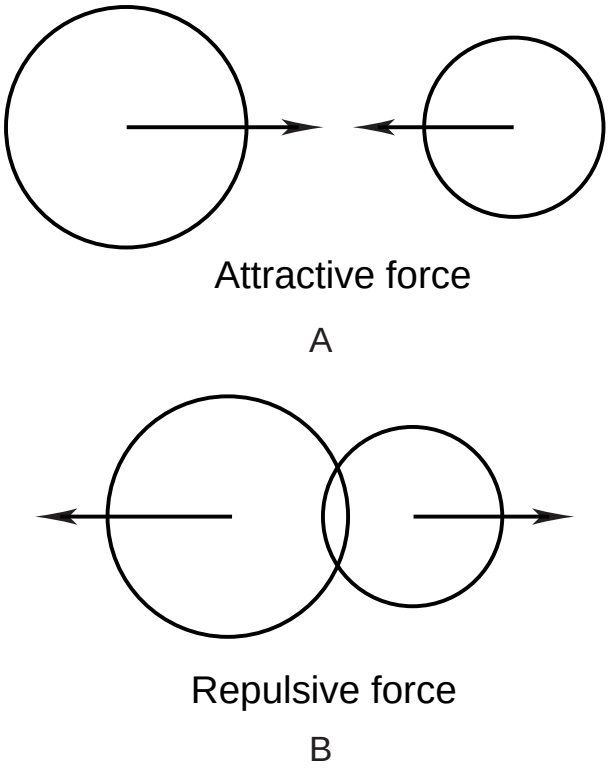


Figure 3.2: *Two circles with attractive and repulsive forces. A: two disconnected circles with attractive force,* $r_i + r_j \leq d_{ij}$*; B: two connected circles with repulsive force,* $r_i + r_j > d_{ij}$.

The interpretation here is that the first term is an attractor that makes the two circles move closer together and pulls them towards a layout where $D_{ij} = 0$, while the second term is a repeller that prevents the circles from overlapping. Indeed, if $D_{ij} \geq t_{ij}$ then there is no any overlap between circles and the repeller term is zero or slightly negative, while the attractor in the objective function applies an attractive

force to the two circles. On the other hand, if $D_{ij} < t_{ij}$ then the repeller term is positive, and it tends to positive infinity as D_{ij} tends to zero, preventing the circles from overlapping completely. Finally, note that there is no force between i and j exactly if $D_{ij}^2 = t_{ij}/c_{ij}$. This observation leads to the definition of the generalized target distance T_{ij} below.

The model aims to ensure that $\frac{D_{ij}}{t_{ij}} = 1$ at optimality, so choosing $\alpha < 1$ sets a target value t_{ij} that allows some overlap of the respective circles, which means that the non-overlap requirement is relaxed. In practice, by properly adjusting the value of α we achieve a reasonable separation between all pairs of circles. Let M and P denote the set of mobile modules and the set of fixed I/O pads, respectively. Target distances are applied only for pairs of mobile modules. The complete attractor-repeller (AR) model as given by Anjos and Vannelli (2002) is:

$$\min_{(x_i,y_i),i\in M,w_F,h_F} \sum_{i,j\in M\cup P} c_{ij}D_{ij} + \sum_{i,j\in M} f(\frac{D_{ij}}{t_{ij}})$$

s.t.

$$\begin{aligned}
& x_i + r_i \leq \frac{1}{2}w_F \ \text{ and } r_i - x_i \leq \frac{1}{2}w_F, \ \text{for all } i \in M, \\
& y_i + r_i \leq \frac{1}{2}h_F \ \text{ and } r_i - y_i \leq \frac{1}{2}h_F, \ \text{ for all } i \in M, \\
& w_F^{low} \leq w_F \leq w_F^{up}, \\
& h_F^{low} \leq h_F \leq h_F^{up},
\end{aligned} \tag{3.12}$$

where (x_i, y_i) are the coordinates of the centre of circle i as previously defined; w_F, h_F are the width and height of the floorplan; and w_F^{low}, w_F^{up}, h_F^{low}, and h_F^{up} are the lower and upper bounds on the width and the height, respectively. The first two sets of constraints require that all the circles be entirely contained within the floorplan, and the remaining two pairs of inequalities bound the width and height of the floorplan. (Note that the geometric centre is at the origin of the $x - y$ plane.) In particular, for

fixed-outline floorplanning, we set $w_F^{low} = w_F^{up} = \bar{w}_F$ and $h_F^{low} = h_F^{up} = \bar{h}_F$, where $\bar{w}_F$ and $\bar{h}_F$ are the fixed width and height of the floorplan.

An important drawback of the AR model is that the objective function is not convex, and hence the overall model is not convex. By modifying it to obtain a convex problem, we expect to obtain a relaxation that better captures global information about the problem. The effectiveness of this convex approach for the closely related facility layout problem has been documented in Anjos and Vannelli (2006).

In real-world applications, the distances D_{ij} between the circles should be inversely proportional to c_{ij} representing the weights on the wirelength, and should be proportional to the relative size of the modules through the value of t_{ij}. Hence, a generalized target distance, namely T_{ij}, is defined such that $D_{ij} \approx T_{ij}$ at optimality (Anjos and Vannelli, 2006). The analysis in Anjos and Vannelli (2002) and Anjos and Vannelli (2006) motivates the definition of the following *generalized target distance* T_{ij}:

$$T_{ij} := \sqrt{\frac{t_{ij}}{c_{ij} + \varepsilon}}, \tag{3.13}$$

where $\varepsilon > 0$ is sufficiently small to ensure that $D_{ij} \approx \sqrt{\frac{t_{ij}}{c_{ij}}}$ if $D_{ij} \approx T_{ij}$. Using T_{ij}, a convex version of the AR model of Anjos and Vannelli (2006) may be stated as:

$$\begin{aligned}
&\min_{(x_i,y_j),w_F,h_F} \sum_{1\le i<j\le n} F_{ij}(x_i, x_j, y_i, y_j) \\
&\text{s.t.} \\
&\qquad x_i + r_i \le \frac{1}{2} w_F \ \text{ and } r_i - x_i \le \frac{1}{2} w_F, \ \text{ for all modules } i, \\
&\qquad y_i + r_i \le \frac{1}{2} h_F \ \text{ and } r_i - y_i \le \frac{1}{2} h_F, \ \text{ for all modules } i, \\
&\qquad w_F^{low} \le w_F \le w_F^{up}, \\
&\qquad h_F^{low} \le h_F \le h_F^{up},
\end{aligned} \tag{3.14}$$

where

$$F_{ij}(x_i, x_j, y_i, y_j) := \begin{cases} c_{ij}z + \frac{t_{ij}}{z} - 1, & z \geq T_{ij} \\ 2\sqrt{c_{ij}t_{ij}} - 1, & 0 \leq z < T_{ij} \end{cases}$$

with $z = (x_i - x_j)^2 + (y_i - y_j)^2$. It was shown by Anjos and Vannelli (2002) that this problem is convex, and that by construction F_{ij} attains its minimum value whenever the positions of circles i and j satisfy $D_{ij} \leq T_{ij}$. This includes the case where $D_{ij} = 0$, i.e., both circles completely overlap. Of course, we do not want such a placement, therefore what we seek is an arrangement of the circles where $D_{ij} \approx T_{ij}$, since for such arrangements the minimum value of F_{ij} is still attained but the resulting overlap is minimized.

Following the approach used in Anjos and Vannelli (2006), we use a slightly modified model whose minima satisfy $D_{ij} \approx T_{ij}$ at optimality. It can be viewed as a compromise between convexity and computational practice, in the sense that we lose the convexity of the model above, but gain a model which can be solved efficiently and still aims to achieve the generalized target distances. The idea is to add to the objective function a term of the form $-\ln(D_{ij}/T_{ij})$ for each pair i, j of circles. (This particular choice of function is inspired by the log-barrier functions in interior-point methods for convex optimization.) Hence, the model solved in the first stage of our

method is:

$$\min_{(x_i,y_j),w_F,h_F} \sum_{1\le i<j\le n} F_{ij}(x_i,x_j,y_i,y_j) - \beta K \ln(\frac{D_{ij}}{T_{ij}})$$

s.t.

$$\begin{aligned}
&x_i + r_i \le \frac{1}{2}w_F \ \text{ and } r_i - x_i \le \frac{1}{2}w_F, \ \text{ for all modules } i,\\
&y_i + r_i \le \frac{1}{2}h_F \ \text{ and } r_i - y_i \le \frac{1}{2}h_F, \ \text{ for all modules } i,\\
&w_F^{low} \le w_F \le w_F^{up},\\
&h_F^{low} \le h_F \le h_F^{up},
\end{aligned} \tag{3.15}$$

where β is a parameter selected empirically. K is chosen to reflect the weights of all the pairs of mobile modules in the objective function:

$$K = \sum_{i<j} c_{ij}. \tag{3.16}$$

The topological relationships between modules are obtained in this first stage. Without the term $-\beta K \ln(D_{ij}/T_{ij})$ in formulation (3.15), this problem is convex (Anjos and Vannelli, 2006). By solving formulation (3.15), the solution of the first stage provides relative positions within the floorplan for all the modules represented by circles.

3.4 Second Stage Model

The solution of the first stage provides relative locations for all the modules. In the second stage, we determine the precise location and dimensions of the modules while minimizing the total wirelength. At this point, classical floorplanning performs a multi-objective minimization which seeks to minimize both the wirelength and the area of the floorplan. We instead focus on fixed-outline floorplanning, and

use a mathematical program with complementarity constraints (MPCC) which minimizes wirelength and yields deadspace-free and overlap-free floorplans. (Note that it is straightforward to incorporate the minimization of the area into the objective function, but we use only the wirelength.)

The non-overlap constraints for each pair of modules can be expressed as

$$\frac{1}{2}(w_i + w_j) \leq |x_i - x_j| \quad \text{or} \quad \frac{1}{2}(h_i + h_j) \leq |y_i - y_j|.$$

However, these constraints are disjunctive, nonlinear and non-convex. They were reformulated in Anjos and Vannelli (2006) by introducing two new variables X_{ij} and Y_{ij} and expressing them as:

$$\begin{cases} X_{ij} \geq \frac{1}{2}(w_i + w_j) - |x_i - x_j| & X_{ij} \geq 0, \\ Y_{ij} \geq \frac{1}{2}(h_i + h_j) - |y_i - y_j| & Y_{ij} \geq 0, \\ X_{ij}Y_{ij} = 0. \end{cases} \tag{3.17}$$

It is straightforward to check that the constraints (3.17) enforce no overlap between modules i and j. We also require that the area requirement $w_i h_i = a_i$ be satisfied for every module i.

Incorporating all these constraints, the problem of minimizing wirelength for fixed-

outline floorplanning is formulated as:

$$\min_{(x_i,y_i),w_i,h_i} \quad \sum_{1\le i<j\le n} c_{ij} L(x_i, x_j, y_i, y_j)$$

s.t.

$$\begin{aligned}
& x_i + \frac{1}{2}w_i \le \frac{1}{2}\bar{w}_F \quad \forall\, i, \\
& y_i + \frac{1}{2}h_i \le \frac{1}{2}\bar{h}_F \quad \forall\, i, \\
& \frac{1}{2}w_i - x_i \le \frac{1}{2}\bar{w}_F \quad \forall\, i, \\
& \frac{1}{2}h_i - y_i \le \frac{1}{2}\bar{h}_F \quad \forall\, i, \\
& w_i h_i = a_i \quad \forall\, i, \\
& w_i^{low} \le w_i \le w_i^{up} \quad \forall\, i, \\
& h_i^{low} \le h_i \le h_i^{up} \quad \forall\, i, \\
& \frac{1}{2}(w_i + w_j) - |x_i - x_j| \le X_{ij} \quad \forall\, 1 \le i < j \le n, \\
& \frac{1}{2}(h_i + h_j) - |y_i - y_j| \le Y_{ij} \quad \forall\, 1 \le i < j \le n, \\
& X_{ij} \ge 0, Y_{ij} \ge 0 \quad \forall\, 1 \le i < j \le n, \\
& X_{ij} Y_{ij} = 0 \quad \forall\, 1 \le i < j \le n,
\end{aligned} \tag{3.18}$$

where $\bar{w}_F$ and $\bar{h}_F$ are the fixed width and height of the floorplan, and $L(x_i, x_j, y_i, y_j)$ is the distance between modules i and j. (Different choices of L lead to different methods as discussed in Section 3.4.2 below.) The above formulation is an instance of an MPCC due to the presence of the complementarity constraints:

$$X_{ij} \ge 0,\ Y_{ij} \ge 0,\ X_{ij} Y_{ij} = 0.$$

One consequence of these constraints is that the problem lacks a strictly feasible point (because at any feasible point $X_{ij} = 0$ or $Y_{ij} = 0$ must be satisfied). To address this

difficulty, we penalize the complementarity constraints $X_{ij}Y_{ij} = 0$ in the objective function and solve the resulting problem. If $X_{ij}Y_{ij} = 0$ is satisfied for all pairs of modules i and j, then the computed solution is feasible for formulation (3.18).

This leads to the model used in the second stage of our method:

$$\min_{(x_i,y_i),w_i,h_i} \quad \sum_{1\le i<j\le n} c_{ij}L(x_i,x_j,y_i,y_j) + \gamma K X_{ij}Y_{ij}$$

s.t.

$$\begin{aligned}
& x_i + \frac{1}{2}w_i \le \frac{1}{2}w_F \quad \forall\, i, \\
& y_i + \frac{1}{2}h_i \le \frac{1}{2}h_F \quad \forall\, i, \\
& \frac{1}{2}w_i - x_i \le \frac{1}{2}w_F \quad \forall\, i, \\
& \frac{1}{2}h_i - y_i \le \frac{1}{2}h_F \quad \forall\, i, \\
& w_i h_i = a_i \quad \forall\, i, \\
& w_i^{low} \le w_i \le w_i^{up} \quad \forall\, i, \\
& h_i^{low} \le h_i \le h_i^{up} \quad \forall\, i, \\
& \delta\left(\frac{1}{2}(w_i + w_j) - |x_i - x_j|\right) \le X_{ij} \quad \forall\, 1 \le i < j \le n, \\
& X_{ij} \ge 0 \quad \forall\, 1 \le i < j \le n, \\
& \delta\left(\frac{1}{2}(h_i + h_j) - |y_i - y_j|\right) \le Y_{ij} \quad \forall\, 1 \le i < j \le n, \\
& Y_{ij} \ge 0 \quad \forall\, 1 \le i < j \le n
\end{aligned} \tag{3.19}$$

where K is as in (3.16), and γ and δ are parameters.

3.4.1 Inclusion of the Aspect Ratio Constraints

We now show how the aspect ratio constraint for each module is easily incorporated into the formulation (3.19). We assume that we are given lower and upper bounds

$R_i^{low} > 0$ and $R_i^{up} > 0$ on the aspect ratio R_i of module i. Aspect ratios restrict modules from becoming excessively narrow in either direction. By definition, the aspect ratio R_i for module i is

$$R_i := \max\{h_i, w_i\}/\min\{h_i, w_i\}.$$

If we set $w_i^{low} = h_i^{low} = \sqrt{a_i/R_i^{up}}$ where $a_i = w_i h_i$, then

$$w_i \geq w_i^{low} \Rightarrow w_i^2 \geq a_i/R_i^{up} \Rightarrow R_i^{up} w_i^2 \geq a_i \Rightarrow R_i^{up} \geq h_i/w_i$$

since $w_i \geq w_i^{low} > 0$. Similarly, $h_i \geq h_i^{low} > 0$ implies $R_i^{up} \geq w_i/h_i$.

By the same line of argument, setting $w_i^{up} = h_i^{up} = \sqrt{a_i R_i^{low}}$ enforces $R_i^{low} \leq w_i/h_i$ and $R_i^{low} \leq h_i/w_i$.

3.4.2 Minimization of Different Wirelengths

The formulations (3.18) and (3.19) allow different distance functions $L(x_i, x_j, y_i, y_j)$ to estimate the wirelength between modules i and j. Common choices of distance function are:

- the rectilinear distance $|x_i - x_j| + |y_i - y_j|$;
- the quadratic distance $(x_i - x_j)^2 + (y_i - y_j)^2$; and
- the HPWL. This method requires additional linear constraints as well as a specific choice of L. This is described in Section 3.4.3 below.

Computational results for these three choices are reported in Section 3.5.3.

3.4.3 Minimization of Half Perimeter Wirelength

We describe here how the formulations (3.18) and (3.19) can be used to minimize the HPWL. We use a four-module net in Fig. 3.3 to illustrate the approach. In Fig. 3.3,

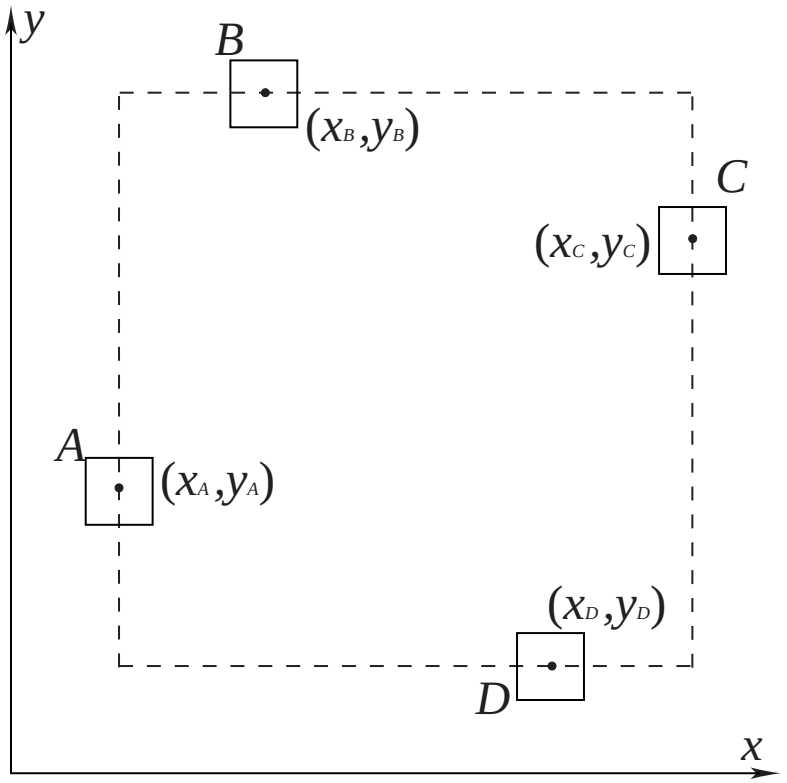


Figure 3.3: *Illustration of the bounding box for a four-module net*

the HPWL is equal to the half-perimeter of the bounding box of the four-module net that includes the modules A, B, C, and D. We introduce two new variables, wl_x and wl_y, representing the components of the HPWL in the x and y directions respectively. So, (x_A,y_A) is the coordinate of the centre point of module A. Clearly,

$$wl_x = \max\{|x_A - x_B|, |x_A - x_C|, |x_A - x_D|, |x_B - x_C|, |x_B - x_D|, |x_C - x_D|\},$$
$$wl_y = \max\{|y_A - y_B|, |y_A - y_C|, |y_A - y_D|, |y_B - y_C|, |y_B - y_D|, |y_C - y_D|\}.$$

Therefore, we can minimize the HPWL for this small example by solving

$$\begin{aligned}
&\min \quad c(wl_x + wl_y) \\
&\text{s.t.} \\
&wl_x \geq x_A - x_B, \qquad wl_x \geq x_B - x_A, \\
&wl_x \geq x_A - x_C, \qquad wl_x \geq x_C - x_A, \\
&wl_x \geq x_A - x_D, \qquad wl_x \geq x_D - x_A, \\
&wl_x \geq x_B - x_C, \qquad wl_x \geq x_C - x_B, \\
&wl_x \geq x_B - x_D, \qquad wl_x \geq x_D - x_B, \\
&wl_x \geq x_C - x_D, \qquad wl_x \geq x_D - x_C, \\
&wl_y \geq y_A - y_B, \qquad wl_y \geq y_B - y_A, \\
&wl_y \geq y_A - y_C, \qquad wl_y \geq y_C - y_A, \\
&wl_y \geq y_A - y_D, \qquad wl_y \geq y_D - y_A, \\
&wl_y \geq y_B - y_C, \qquad wl_y \geq y_C - y_B, \\
&wl_y \geq y_B - y_D, \qquad wl_y \geq y_D - y_B, \\
&wl_y \geq y_C - y_D, \qquad wl_y \geq y_D - y_C, \\
&X_{ij} \geq 0, \quad \forall\, i, \\
&Y_{ij} \geq 0, \quad \forall\, i, \\
&X_{ij}Y_{ij} = 0, \quad \forall\, i,
\end{aligned} \tag{3.20}$$

where c is the connectivity of the four-module net.

Therefore, starting with formulations (3.18) and (3.19), adding a similar set of constraints for each net in the circuit, and setting up the objective function as the sum of the weighted half-perimeters, we can directly minimize the HPWL in the second stage of our method.

We note that due to the large number of linear constraints that HPWL minimization requires, it is computationally expensive to solve the placement problem when there are many modules (Jackson and Kuh, 1989). However, the number of linear constraints is technically acceptable for floorplanning since the number of modules is relatively small.

3.5 Computational Results

In this section, the proposed method is applied to the standard MCNC benchmarks to demonstrate its effectiveness and flexibility. We use the MCNC benchmark problems *apte*, *xerox*, *hp*, *ami*33, and *ami*49 (MCNC, 2004) shown in Table 2.2.

All the modules are taken to be soft modules with fixed areas and variable dimensions, and (as an approximation) all the pins are assumed to be at the centres of the modules. The aspect ratio of every module is constrained to lie between 0.1 and 10, so we set $R_i^{low} = R_i^{up} = 10$ for every i. The multipin nets are transformed into cliques using the clique model discussed in Section 2.6, and taking $W = 1$ for all the nets.

Both stages of our method were solved by the optimization package MINOS (Murtagh and Saunders, 1982; Murtagh and Saunders, 1983) via the modeling language AMPL (Fourer *et al.*, 2003) accessed on the NEOS server (Czyzyk *et al.*, 1998; Ferris *et al.*, 2000).

In the second stage, we apply the proposed optimization model in three different ways:

Method A: Solve (3.19) with the rectilinear distance $|x_i - x_j| + |y_i - y_j|$, and calculate the HPWL of the resulting floorplan;

Method B: Solve (3.19) with the quadratic distance $(x_i - x_j)^2 + (y_i - y_j)^2$, and calculate the HPWL of the resulting floorplan;

Method C: Minimize the HPWL directly. This was described in Section 3.4.3.

Note that we always use the HPWL to compare the quality of the floorplans obtained.

For these instances of fixed-outline floorplanning, we respect the dimensions of the floorplans provided in the MCNC data, so that the fixed layout area is a sum of the areas of all the modules included in a circuit.

The parameters in our models took values in the following ranges:

- $\alpha \in [0.10, 2.90]$;
- $\beta = 10$;
- $\gamma \in [0.10, 2.00]$;
- $\delta \in [0.01, 5.00]$.

3.5.1 Initial Configuration

MINOS is based on an iterative algorithm that requires the user to provide an initial configuration. An ideal initial configuration for MINOS is not generally known *a priori*. Anjos and Vannelli (2006) suggested a method for finding a starting configuration. They distribute the centres of the N modules at regular intervals around a circle. Let the radius of the circle be

$$r = w_F^{up} + h_F^{up} \tag{3.21}$$

then the centres (x_i, y_i) of the modules are initialized by

$$\begin{aligned} x_i &= r\cos\theta_i \\ y_i &= r\sin\theta_i, \end{aligned} \tag{3.22}$$

where $i = 1, ..., N$, and θ_i is defined as

$$\theta_i = \frac{2\pi(i-1)}{N} \tag{3.23}$$

Experiments demonstrated that the *regular* distribution of the centres of the modules around the circle leads to infeasible solutions for some circuits. Therefore, we propose a method for placing the modules at arbitrary intervals around the circle. We modify the radius of the circle as

$$r = (w_F^{up} + h_F^{up})/\phi, \tag{3.24}$$

where ϕ is a constant to control the radius. We choose N uniformly distributed numbers and sort them in ascending order, i.e.,

$$I = \text{ sort (rand } (1, N)), \tag{3.25}$$

where $rand(1, N)$ is a function that produces a $1 \times N$ matrix with random elements chosen from a uniform distribution on the interval (0,1); and sort() sorts elements in ascending order. The centres (x_i, y_i) of the modules are initialized by Equation (3.22). However, θ is modified as

$$\theta = 2\pi \text{ sort (rand } (1, N)). \tag{3.26}$$

The starting configuration of the 9-module circuit *apte* is illustrated in Fig. 3.4, where nine modules represented by circles are arbitrarily distributed around a circle.

When MINOS solves the first stage model formulated in (3.15), it applies a reduced-gradient algorithm associated with a quasi-Newton approach to exploit the

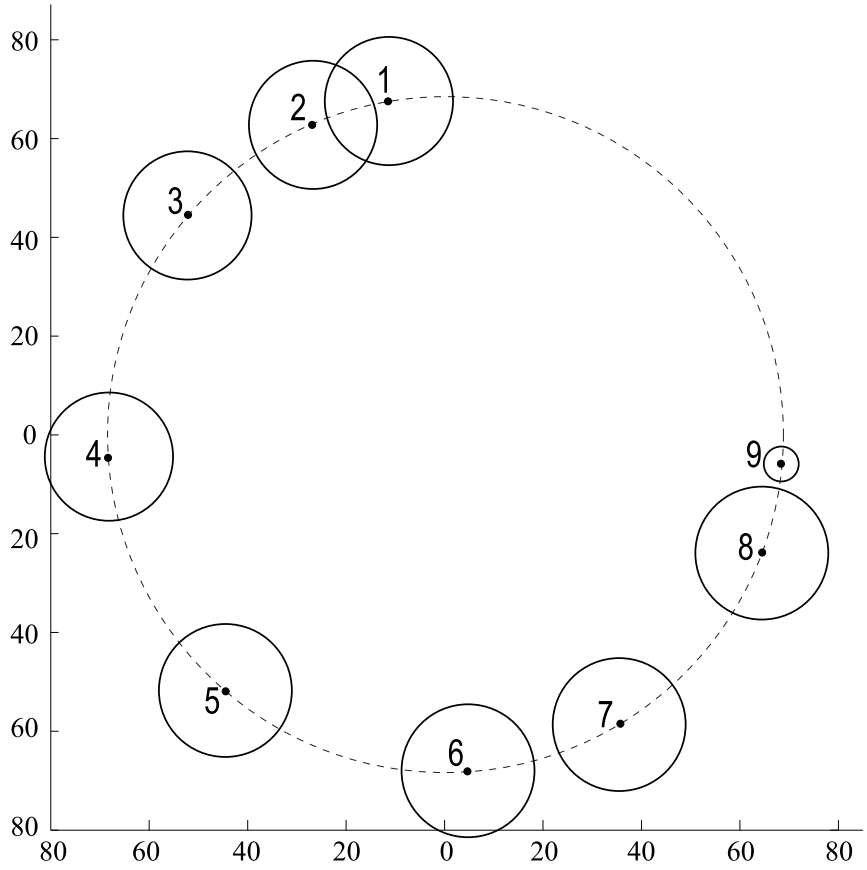


Figure 3.4: *Initial configuration for 9-module circuit apte.*

structure of the model. The model has $2N+2$ variables and $4N+4$ inequality constraints, all of which are linear, and objective function is nonlinear. Because of the structure of this model, MINOS is superlinearly convergent and computationally efficient even for a fairly large number of modules.

3.5.2 Computation of Overlapping Areas of Modules

We address the computation of overlap between modules in this section, as we shall verify whether the layout achieves ZDS and overlap-free floorplan and compute the overlapping areas between modules.

If (x_i,y_i) and (x_j,y_j) are centres of rectangular modules i and j, then the height and width of modules i and j are h_i, w_i, and h_j, w_j, respectively. The geometric relation of the non-overlap case between modules i and j is illustrated in Fig. 3.5A.

Modules may overlap partially (Fig. 3.5B) or completely (Fig. 3.5C and Fig. 3.5D). We use the following notation (see Fig. 3.5):

$$\begin{aligned} \xi_x &= \frac{1}{2}(w_i + w_j) - |x_i - x_j|, \\ \xi_y &= \frac{1}{2}(h_i + h_j) - |y_i - y_j|. \end{aligned} \tag{3.27}$$

The following conditions are satisfied if modules i and j partially overlap along x:

1. $|x_i - x_j| \leq \frac{1}{2}(w_i + w_j)$;
2. $|y_i - y_j| \leq \frac{1}{2}(h_i + h_j)$;
3. $\xi_x < \min(w_i, w_j)$;

where condition (3) indicates that module i overlaps module j *partially* (see Fig. 3.5B). The overlap in x is computed as follows:

$$\varphi_x = \xi_x = \frac{1}{2}(w_i + w_j) - |x_i - x_j|. \tag{3.28}$$

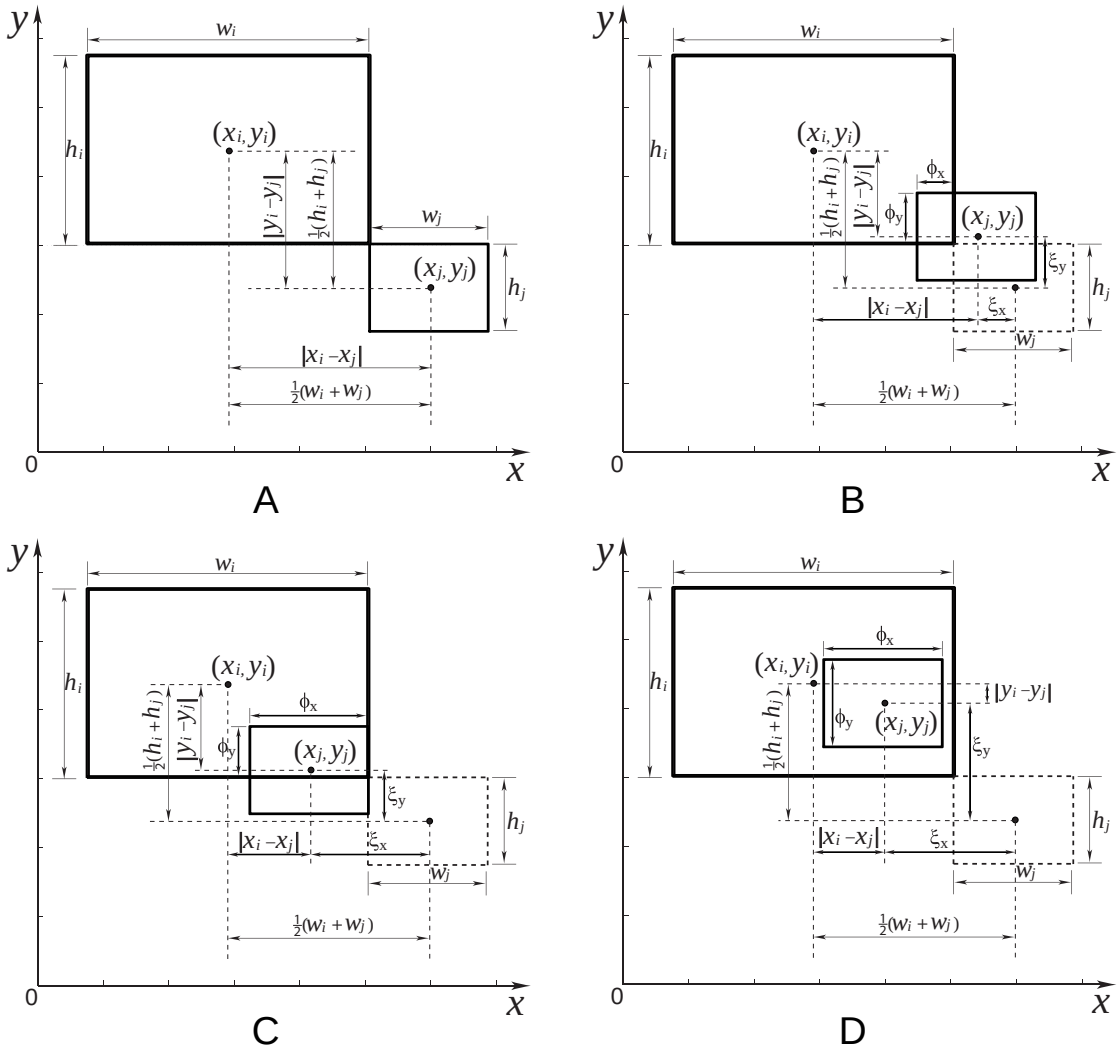


Figure 3.5: *Instance of four-module case for fixed-outline floorplanning. A: no overlap; B: partial overlap; C: complete overlap along x; D: complete overlap.*

Similarly, the overlap in y is computed as follows:

$$\varphi_y = \xi_y = \frac{1}{2}(h_i + h_j) - |y_i - y_j|. \tag{3.29}$$

The following conditions are satisfied if modules i and j completely overlap along x:

1. $|x_i - x_j| \leq \frac{1}{2}(w_i + w_j)$;
2. $|y_i - y_j| \leq \frac{1}{2}(h_i + h_j)$;
3. $\xi_x \geq \min(w_i, w_j)$;

where condition (3) indicates that module i overlaps module j *completely* (see Fig. 3.5C and Fig. 3.5D). If $\xi_x = \min(w_i, w_j)$ as illustrated in Fig. 3.5C, which shows module j overlapping module i but not embedded in module i, the overlap in x is computed as follows:

$$\varphi_x = \min(w_i, w_j), \tag{3.30}$$

Similarly, the overlap in y is computed as follows:

$$\varphi_y = \min(h_i, h_j), \tag{3.31}$$

If module j overlaps module i and is also enveloped in module i, then $\xi_x > \min(w_i, w_j)$ in condition (3). The overlap in x and y is given by equations (3.30) and (3.31), respectively.

In summary, the overlap in x if modules i and j overlap can be computed as follows:

$$\begin{cases} \varphi_x = \frac{1}{2}(w_i + w_j) - |x_i - x_j|, & \text{if partial overlap;} \\ \varphi_x = \min(w_i, w_j), & \text{if complete overlap.} \end{cases} \tag{3.32}$$

The overlap in y if modules i and j overlap can be computed as follows:

$$\begin{cases} \varphi_y = \frac{1}{2}(h_i + h_j) - |y_i - y_j|, & \text{if partial overlap;} \\ \varphi_y = \min(h_i, h_j), & \text{if complete overlap.} \end{cases} \tag{3.33}$$

Algorithms 1 and 2 below calculate overlaps φ_x and φ_y along x and y respectively, and the overlapped area φ_A of modules i and j can be computed as follows:

$$\varphi_A = \varphi_x \times \varphi_y. \tag{3.34}$$

Input: Geometric property of every module

Output: Overlap in x

foreach *Module i= 1 to M* **do**
  foreach *Module j= 1 to N* **do**
    if *Overlapped in y,* $|y_i - y_j| \leq \frac{1}{2}(h_i + h_j)$ **then**
      if *Overlapped in x,* $|x_i - x_j| \leq \frac{1}{2}(w_i + w_j)$ **then**
        if $\xi_x = \frac{1}{2}(w_i + w_j) - |x_i - x_j| < min(w_i, w_j)$ **then**
          Overlap partially in x: $\varphi_x = \frac{1}{2}(w_i + w_j) - |x_i - x_j|$;
        else
          Overlap completely in x: $\varphi_x = \min(w_i, w_j)$.
        end
      end
    end
  end
end

Algorithm 1: Calculation of overlap φ_x along x

Input: Geometric property of every module

Output: Overlap in y

foreach *Module i= 1 to M* **do**

  foreach *Module j= 1 to N* **do**

    if *Overlapped in x,* $|x_i - x_j| \leq \frac{1}{2}(w_i + w_j)$ **then**

      if *Overlapped in y,* $|y_i - y_j| \leq \frac{1}{2}(h_i + h_j)$ **then**

        if $\xi_y = \frac{1}{2}(h_i + h_j) - |y_i - y_j| < min(h_i, h_j)$ **then**

          Overlap partially in y: $\varphi_y = \frac{1}{2}(h_i + h_j) - |y_i - y_j|$;

        else

          Overlap completely in y: $\varphi_y = \min(h_i, h_j)$.

        end

      end

    end

  end

end

Algorithm 2: Calculation of overlap φ_y along y

3.5.3 Computational Results for the MCNC Benchmarks

We compare our methods with two state-of-the-art academic floorplanners (Adya and Markov, 2003; Murata and Kuh, 1998). All reported wirelengths are measured using the HPWL, also used by Adya and Markov (2003) and Murata and Kuh (1998). Methods A, B, and C each use a different measure of wirelength as the objective function, and once the final floorplan is obtained the corresponding HPWL is computed.

Methods A, B, and C were each run 20 times; we report the best floorplan achieved and the average HPWL and runtime.

Table 3.1: Experimental results by the proposed model

MCNC circuit	Total area	Our Method						
		Our area	Runtime			HPWL		
			Method A min/avg	Method B min/avg	Method C min/avg	Method A min/avg	Method B min/avg	Method C min/avg
	(mm^2)	(mm^2)	(s)	(s)	(s)	(mm)	(mm)	(mm)
apte	46.56	46.56	0.093/0.69	0.084/1.04	0.11/0.94	384.30/425.09	386.81/436.59	397.70/438.82
xerox	19.35	19.35	0.34/1.23	0.33/2.03	0.25/0.98	420.11/462.12	433.27/475.87	427.61/469.75
hp	8.30	8.30	0.37/1.17	0.21/1.65	0.42/1.72	131.83/154.84	139.80/149.64	130.50/151.28
ami33	1.16	1.16	8.11/14.16	7.41/10.03	7.53/9.51	60.36/65.31	60.25/62.37	61.40/62.83
ami49	35.4	35.4	37.91/66.09	38.78/55.53	38.90/56.46	684.62/720.65	681.72/706.06	681.70/709.46

3.5.4 Comparison with *MK* Model

First, we compare our approach with the MK model (Murata and Kuh, 1998). Recall that Murata and Kuh used a convex optimization algorithm to optimize the aspect ratios of the soft modules, and iteratively carried out this procedure by an SA algorithm to improve the SP for floorplanning.

Table 3.2: Results reported by *MK*

MCNC circuit	Total area (mm^2)	*MK* (Murata and Kuh, 1998)		
		Area (mm^2)	Runtime (s)	HPWL (mm)
apte	46.56	46.55	789	344.36
xerox	19.35	19.50	1198	401.25
hp	8.30	8.83	1346	118.82
ami33	1.16	1.16	75684	53.39
ami49	35.4	35.58	612103	775.10

In Murata and Kuh's experiment the chip aspect ratio for every benchmark was modified to 1, thus requiring the chip to be square in shape. However, the dimensions of the floorplans in our experiments comply with those provided in the benchmark. Murata and Kuh assumed, as we did, that every module is soft with aspect ratio in the range 0.1 to 10.

Table 3.1 gives the results obtained using our method, while Table 3.2 lists the areas, computation times, and total wirelengths reported by Murata and Kuh. Comparisons of total wirelength are reported in Table 3.3. The results show that the total HPWL is competitive for methods A, B, and C, and in fact we obtain an improve-

ment in total wirelength over *MK* for the largest benchmark problem *ami*49. Our floorplanner takes 66.09 seconds for *ami*49 with 49 modules while *MK* requires seven days using a 250 MHz DEC Alpha. For *ami*33 with 33 modules, our runtime is 14.16 seconds, while that of *MK* is 21 hours.

Table 3.3: Improvements in total wirelength compared with *MK*

MCNC circuit	Our Method vs *MK*		
	Method *A* min	Method *B* min	Method *C* min
apte	-11.60%	-12.33%	-15.49%
xerox	-4.70%	-7.98%	-6.57%
hp	-10.95%	-17.66%	-9.83%
ami33	-13.05%	-12.85%	-15.00%
ami49	+11.67%	+12.05%	+12.05%
Average	-5.73%	-7.75%	-6.97%

3.5.5 Comparison with *AM* Model

We also compare our approach with the *AM* model (Adya and Markov, 2003). Recall that Adya and Markov use sequence pair utilization to represent the topology of a floorplan, together with a moving technique based on slack computation and SA.

Table 3.4 summarizes the areas, computation times, and total wirelengths reported by Adya and Markov. Comparisons of total wirelength are reported in Table 3.5. Our average HPWL is consistently better, and we obtain a better floorplan for several

Table 3.4: Results reported by *AM*

MCNC circuit	Total area (mm^2)	*AM* (Adya and Markov, 2003)		
		Area min/avg (mm^2)	Runtime avg (s)	WL min/avg (mm)
apte	46.56	46.97/48.95	15.4	464/560
xerox	19.35	19.51/20.62	20.1	373/468
hp	8.30	8.96/9.72	15.3	177/214
ami33	1.16	1.18/1.24	31.0	62.5/75.4
ami49	35.4	36.07/37.8	31.9	673/812

benchmarks.

On average, we improved the total wirelength by 14.94% to 16.15% compared to *AM*.

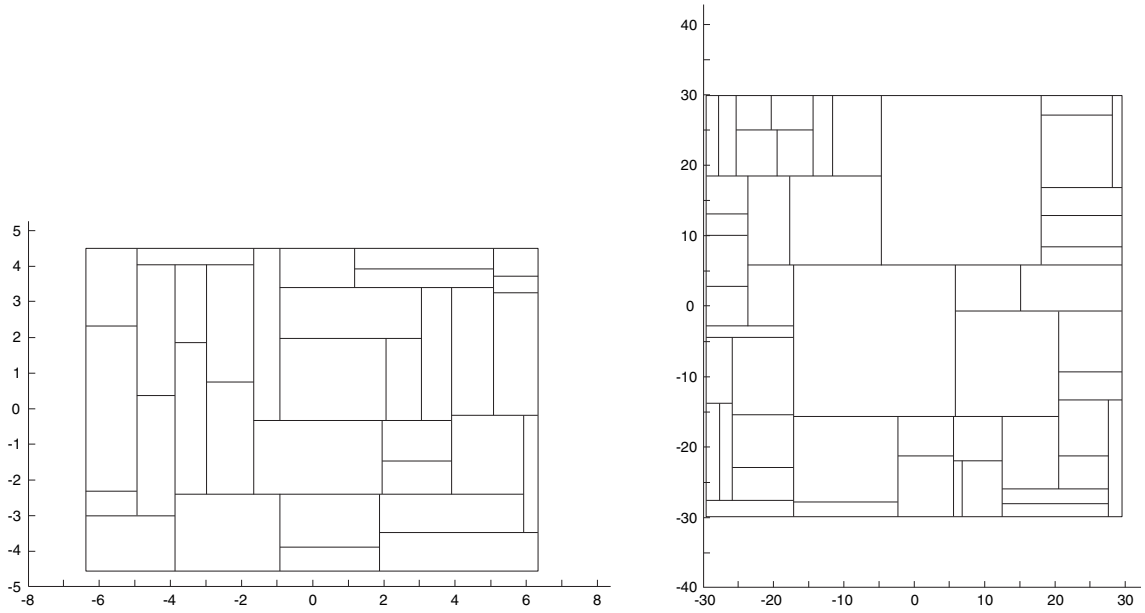
3.5.6 Obtaining Zero-Deadspace Floorplans

The results show that our method is always competitive with, and frequently outperforms, the *MK* and *AM* methods. An important feature of our method is that the dimensions of the floorplans comply with those provided in the MCNC benchmark, and moreover zero-deadspace floorplans were obtained for all five problems, as shown in Table 3.6. In Fig. 3.6, we depict the best zero-deadspace floorplans for *ami*33 and *ami*49 that we obtained. In fact, our floorplans have little overlap, as demonstrated by the average deadspace results in Table 3.6. We illustrate this by presenting in Fig. 3.7 the floorplan with the best HPWL for *ami*49; its deadspace is only 0.044% of the

Table 3.5: Improvements in total wirelength compared with AM

MCNC circuit	Our Methodology vs AM		
	Method A min/avg	Method B min/avg	Method C min/avg
apte	+17.18/+24.09%	+16.64/+22.04%	+14.29/+21.64%
xerox	-12.63/+1.26%	-16.16/-1.68%	-14.64/-0.37%
hp	+25.52/+27.65%	+21.02/+30.07%	+26.27/+29.31%
ami33	+3.42/+12.48%	+3.60/+17.28%	+1.76/+16.67%
ami49	-1.73/+9.22%	-1.29/+13.05%	-1.29/+12.63%
Average	+6.35/+14.94%	+4.76/+16.15%	+5.28/+15.97%

total area. In contrast, *MK* and *AM* both have some deadspace in their floorplans.



Floorplan for *ami*33 with HPWL = 62.65 ($\alpha = 1.02, \beta = 10, \gamma = 1.08, \delta = 1$)

Floorplan for *ami*49 with HPWL = 716.74 ($\alpha = 0.15, \beta = 10, \gamma = 1, \delta = 0.128$)

Figure 3.6: *Best zero-deadspace floorplans for the two largest benchmarks*

3.6 Summary

We proposed a two-stage nonlinear-optimization-based methodology that can be applied to fixed-outline floorplanning. The first stage consists of a convex relaxation of the problem which globally minimizes an approximate measure of wirelength. The second stage minimizes wirelength by sizing the modules subject to the prescribed aspect ratios. Computational results on MCNC benchmarks demonstrate that our method is always competitive with, and frequently outperforms, the results reported in the literature. We obtained ZDS floorplans for all five problems. Thus, our approach guarantees complete area utilization on the fixed-outline floorplan.

Table 3.6: Deadspace comparisons with *MK* and *AM*

MCNC circuit	Total area (mm^2)	*MK*		*AM*		Our Method			
		Area (mm^2)	Deadspace	Area min/avg (mm^2)	Deadspace min/avg	Area (mm^2)	Deadspace min/avg		
							Method *A*	Method *B*	Method *C*
apte	46.56	46.55	-0.02%	46.97/48.95	0.87%/4.88%	46.56	**0%/0%**	**0%/0%**	**0%/0%**
xerox	19.35	19.50	0.77%	19.51/20.62	0.82%/6.16%	19.35	**0%/0%**	**0%/0%**	**0%/0%**
hp	8.30	8.83	6.0%	8.96/9.72	7.40%/14.60%	8.30	**0%/0%**	**0%/0%**	**0%/0%**
ami33	1.16	1.16	0%	1.18/1.24	1.70%/6.45%	1.16	**0%/0.11%**	**0%/0.034%**	**0%/0.013%**
ami49	35.4	35.58	0.5%	36.07/37.8	1.86%/6.35%	35.4	**0%/0.11%**	**0%/0.063%**	**0%/0.094%**

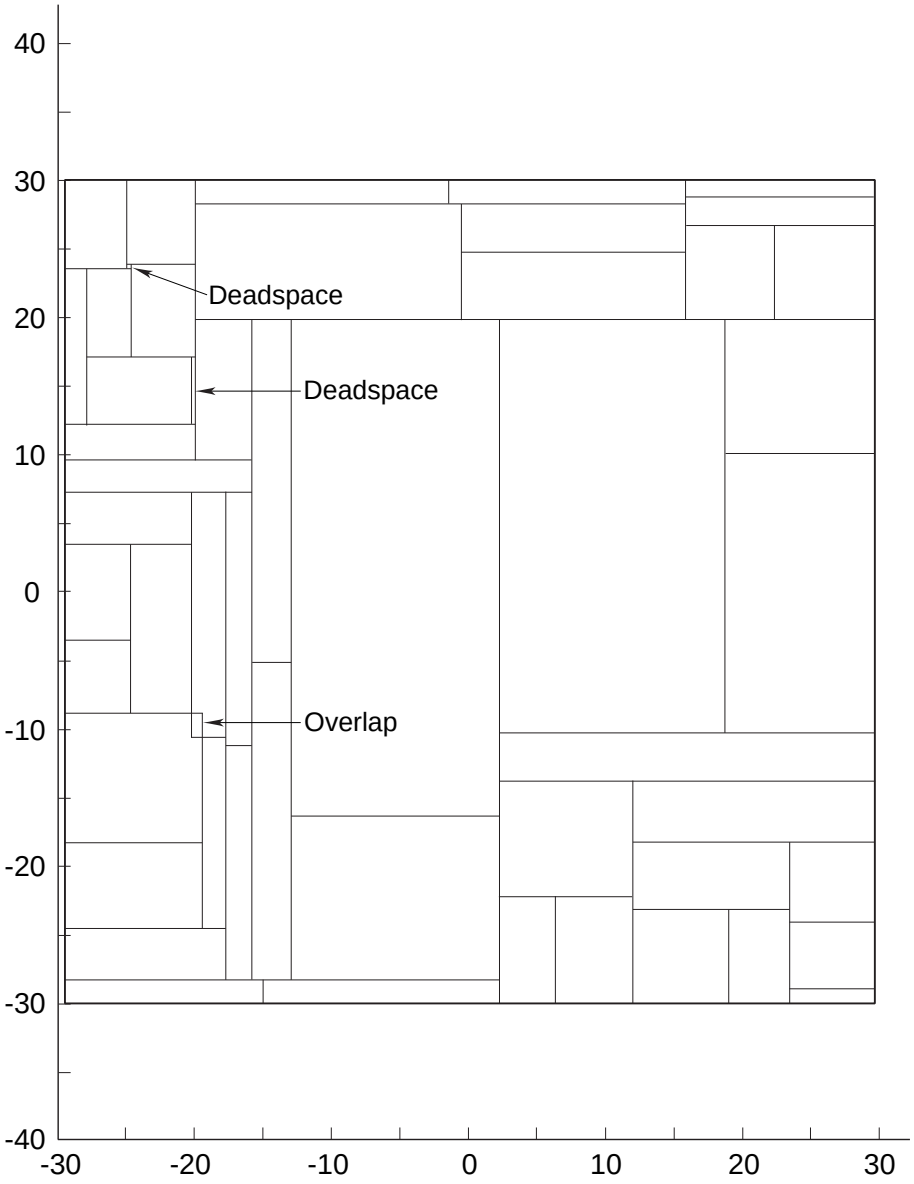


Figure 3.7: *Floorplan for ami*49 *circuit with best HPWL (HPWL = 681.70, deadspace = 0.044%; obtained with* $\alpha = 0.15, \beta = 10, \gamma = 1, \delta = 0.11$*)*

Chapter 4

The Relative Position Matrix Technique

Relative positions of modules are obtained from the first stage and must be enforced at the second stage. In this chapter, we consider a relative position matrix (RPM) technique to encode relative positions. Using this technique the non-overlap constraints that are originally disjunctive, nonlinear and non-convex can be linearized and easily enforced in the second stage model. We consider the Voronoi diagram (VD) method for three main reasons: (i) it spreads out modules in the floorplan, (ii) it transforms the relative position graph into a planar graph, and (iii) it helps to build up a sparse relative position matrix (SRPM). An SRPM approach is developed to decrease computational effort at the second stage. This chapter is organized as follows. Section 4.1 describes the geometrical structure of non-overlap among modules and the non-overlap constraints. In Sections 4.2 and 4.3 we introduce the RPM technique and the VD used. The SRPM approach is presented in Section 4.4. Finally, Section 4.5 summarizes the chapter.

4.1 Geometrical Structure of Non-overlap Among Modules

If (x_i, y_i), (x_j, y_j), w_i, w_j, h_i, and h_j are the coordinates, widths, and heights of rectangular modules i and j, respectively, the two-module case with overlap is illustrated in Fig. 4.1. The non-overlap constraints for each pair of modules can be expressed as

$$\frac{1}{2}(w_i + w_j) \leq |x_i - x_j| \quad \text{if } |y_i - y_j| \leq \frac{1}{2}(h_i + h_j). \tag{4.1}$$

$$\frac{1}{2}(h_i + h_j) \leq |y_i - y_j| \quad \text{if } |x_i - x_j| \leq \frac{1}{2}(w_i + w_j). \tag{4.2}$$

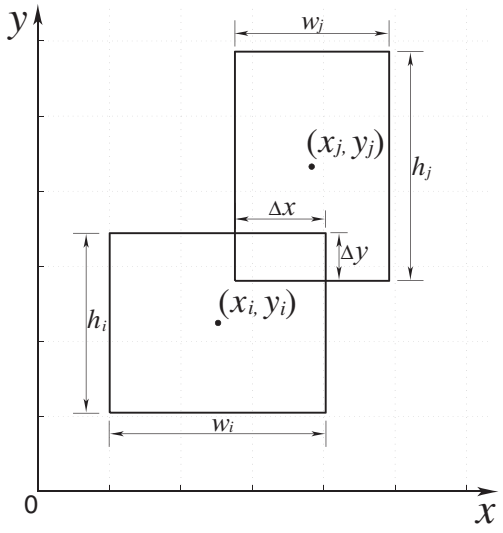


Figure 4.1: *The two-module case with overlap.*

Constraint (4.1), requiring that there is no overlap along the x-axis for one pair of modules, can be expressed as a separation in the x-direction as follows (see Fig.

4.2):

$$0 \geq \frac{1}{2}(w_i + w_j) - |x_i - x_j|. \tag{4.3}$$

Similarly, constraint (4.2), requiring that there is no overlap along y-axis for one pair of modules, can be expressed as a separation in the y-direction as follows (see Fig. 4.3):

$$0 \geq \frac{1}{2}(h_i + h_j) - |y_i - y_j|. \tag{4.4}$$

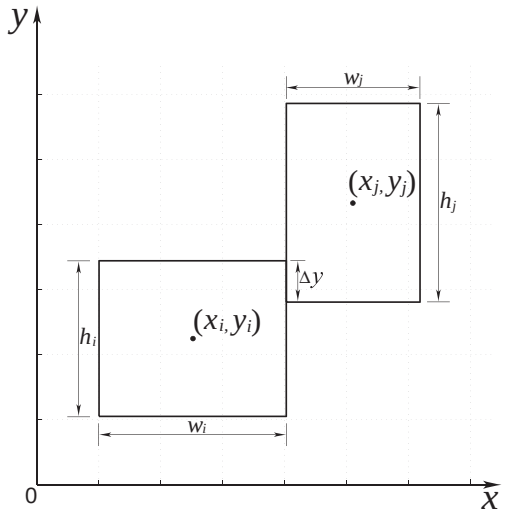


Figure 4.2: *The two-module case without overlap along the x-axis.*

Constraints (4.3) and (4.4) are disjunctive, nonlinear and non-convex. Once relative positions for the modules have been obtained from the first stage, we may remove the absolute values from inequalities (4.3) and (4.4), obtaining linear constraints.

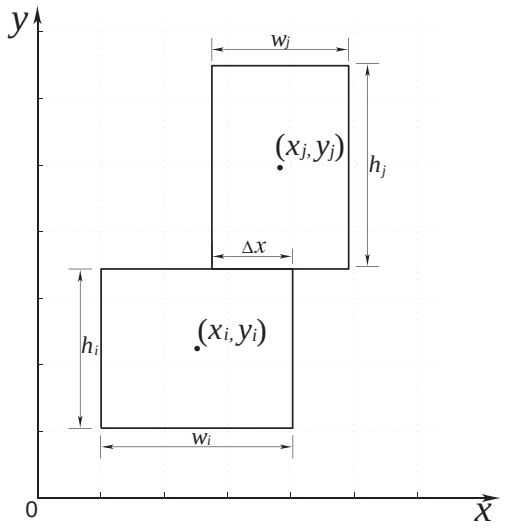


Figure 4.3: *The two-module case without overlap along the y-axis.*

4.2 The Relative Position Matrix

The relative positions of the modules are encoded in a relative position matrix (RPM). An RPM is an $N \times N$ non-negative, symmetric matrix with zeros on the principal diagonal, where N is the number of modules in the floorplan. The information in the upper triangular portion suffices since the RPM is symmetric. The RPM matrix is filled as follows:

- "1" is used to represent this case as an entry in RPM, meaning there is only one constraint separating in the x-direction as inequality (4.3).
  - "11" means that module i is on the left of module j (Fig. 4.4A). The following inequality is satisfied:

$$0 \geq \frac{1}{2}(w_i + w_j) - (x_j - x_i). \tag{4.5}$$

  - "12" means that module i is on the right of module j (Fig. 4.4B). The following inequality is satisfied:

$$0 \geq \frac{1}{2}(w_i + w_j) - (x_i - x_j). \tag{4.6}$$

- "2" means that there is only one constraint separating in the y-direction as inequality (4.4).

  - "21" means that module i is above module j (Fig. 4.4C). The following inequality is satisfied:

$$0 \geq \frac{1}{2}(h_i + h_j) - (y_i - y_j); \tag{4.7}$$

  - "22" means that module i is below module j (Fig. 4.4D). The following inequality is satisfied:

$$0 \geq \frac{1}{2}(h_i + h_j) - (y_j - y_i). \tag{4.8}$$

If we have the relative positions of two modules separated diagonally (Fig. 4.5), a rule is defined to determine how the two modules are separated:

- If $\Delta y \geq \Delta x$, then these two modules should be separated in the y-direction. That is, only the vertical relative positioning is considered.

- If $\Delta x > \Delta y$, then these two modules should be separated in the x-direction. That is, only the horizontal relative positioning is considered.

For instance, RPM(i,j) indicates the relative position between pair of modules i and j. If the relative position of modules i and j is placed as in Fig. 4.4B, then

$$|x_i - x_j| \geq \frac{1}{2}(w_i + w_j) \tag{4.9}$$

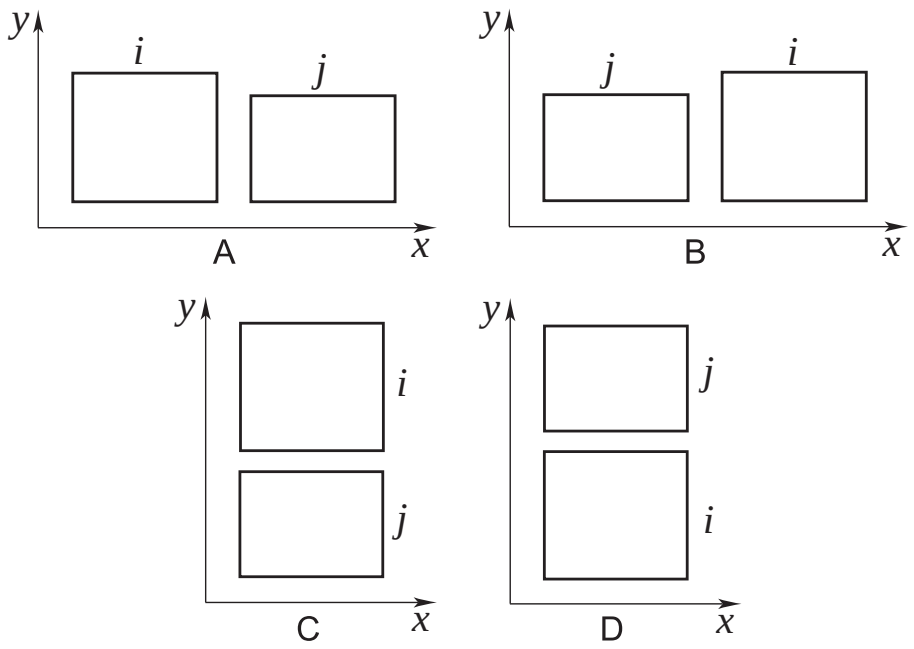


Figure 4.4: *Relative positions of two modules separated horizontally and vertically. A: module i is on the left of module j; B: module i is on the right of module j; C: module i is above module j; D: module i is below module j.*

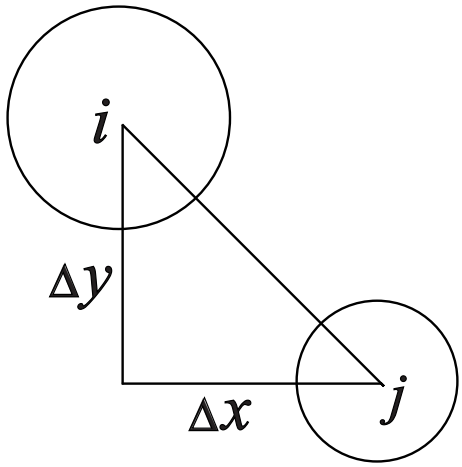


Figure 4.5: *Relative position of two modules separated diagonally.*

becomes

$$x_i - x_j \geq \frac{1}{2}(w_i + w_j), \tag{4.10}$$

where the absolute value is eliminated. Constraint (4.10) is enforced in the second stage model in Chapter 5. Similarly, the objective function describing the cost between modules i and j becomes (note that $|y_i - y_j| = 0$ in this case)

$$c_{ij}(x_i - x_j). \tag{4.11}$$

With the RPM(i,j), the following constraint

$$|y_i - y_j| \geq \frac{1}{2}(h_i + h_j), \tag{4.12}$$

becomes linear (the absolute values are removed). Also the absolute values in the objective function are removed, with the RPM(i,j).

In summary, in the RPM, the following entries represent the relative position relations of two modules:

- "11" with index (i, j): module i is horizontally separated from module j, and module i is on the left of module j;
- "12" with index (i, j): module i is horizontally separated from module j, and module i is on the right of module j;
- "21" with index (i, j): module i is vertically separated from module j, and module i is above module j;
- "22" with index (i, j): module i is vertically separated from module j, and module i is below module j.

An four-module example is used to demonstrate how an RPM can be generated. A relation position graph obtained from the first stage for a four-module case is illustrated in Fig. 4.6A. According to the encoding system, the RPM obtained for this example is a 4×4 non-negative upper triangular matrix with zeros on the principal diagonal shown in (4.13).

$$\begin{pmatrix} 0 & 11 & 21 & 0 \\ 0 & 0 & 0 & 21 \\ 0 & 0 & 0 & 11 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{4.13}$$

Based on this RPM, the final floorplan obtained using second-stage model (described in Chapter 5) is illustrated in Fig. 4.6B.

4.3 The Voronoi Diagram

For the two-dimensional case, a planar graph that can reflect relative positions of the modules is needed to build up an RPM for use in the second stage, inspired by a

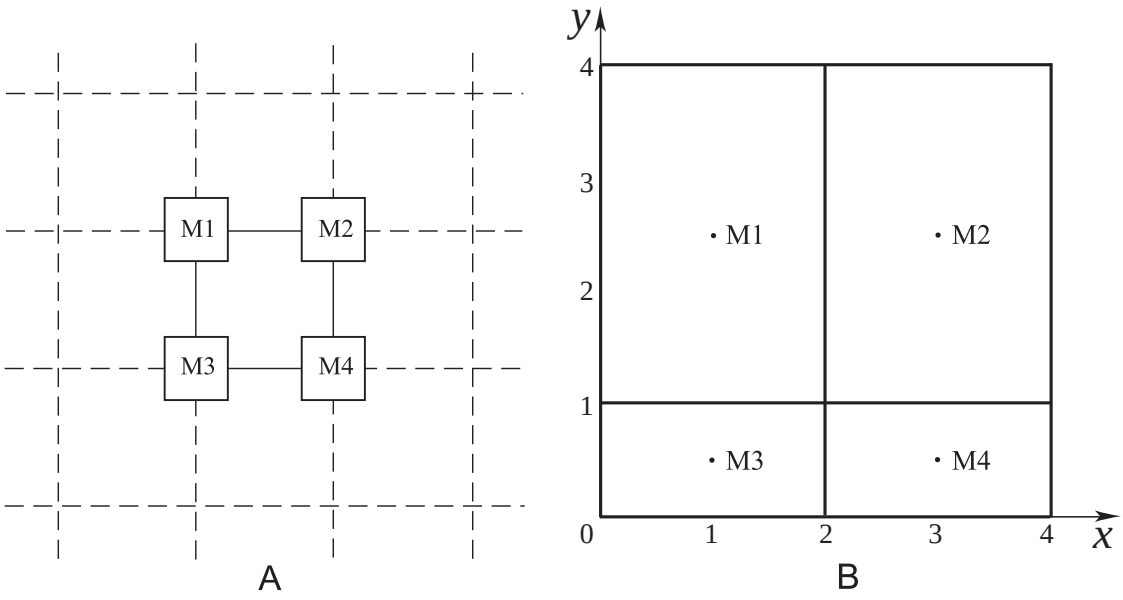


Figure 4.6: *An example of a four-module case. A: the relation position graph; B: the final floorplan.*

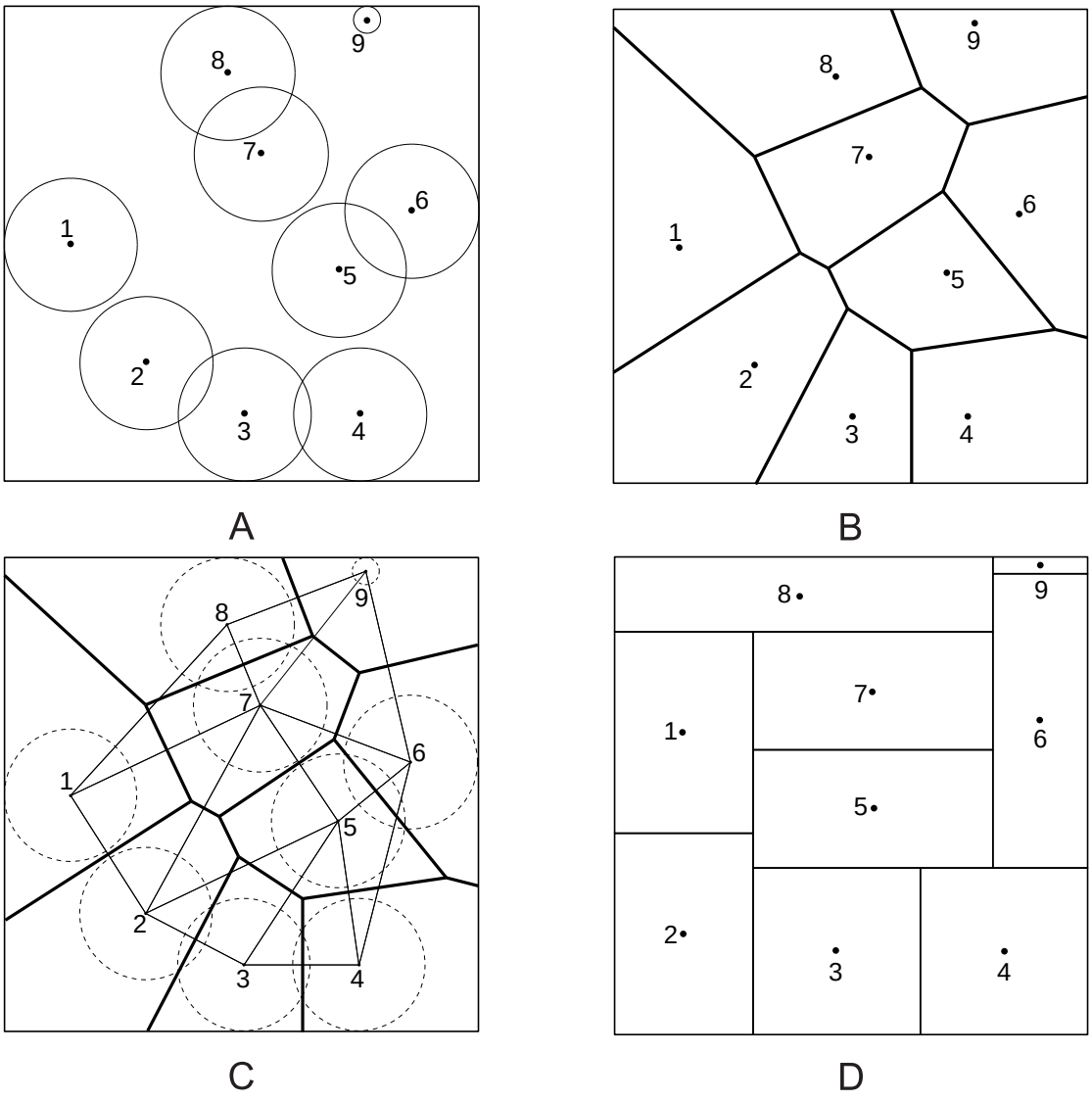


Figure 4.7: *Example for apte circuit. A: the relative position figure; B: the corresponding Voronoi diagram; C: the relative position graph, its Voronoi diagram and Delaunay triangulation; D: the final floorplan.*

navigation system model using the Voronoi diagram (VD) and Delaunay triangulation (DT) for robot path planning (Rao *et al.*, 1991). Jin *et al.* (2006) convert the relative position graph into a planar graph using the VD obtained from DT. The VD and DT can also be used to VLSI placement (Luo *et al.*, 2005; Reda and Chowdhary, 2006). The VD and DT are important topics in computational geometry and have been extensively applied in many engineering fields (Rao *et al.*, 1991; Papadopoulou and Lee, 2004; Luo *et al.*, 2005; Reda and Chowdhary, 2006).

VD is a useful geometric structure that represents distance relationships and partitions a plane into Voronoi cells. The objective is to obtain a relative position planar graph. The VD induces a subdivision of the total area of the layout. The computational complexity of VD and DT algorithms ranges from $O(n \log n)$ to $O(n^2)$ (Jin *et al.*, 2006). Given n circles representing modules, the VD and DT of the centres of the circles provide a planar graph $G = (V, E)$, with vertices set $V = v_1, v_2, ..., v_n$ corresponding to modules $1, 2, ..., n$ and edges $E = e_1, e_2, ..., e_m$ corresponding to the DT. The boundary of the graph is the boundary of the floorplan. A VD induces a subdivision of the total area of the layout. A Voronoi diagram of m sites has at most $2m - 5$ vertices and $3m - 6$ edges (Jin *et al.*, 2006).

The DT of sites S is the geometric dual of the VD of S. The DT is constructed by connecting any two points p, q of S for which a circle C exists without containing any other site of S in its interior when it passes through p and q. In the DT, each site is connected to its nearest neighbour by an edge in the triangulation. Any point to the centre of a given cell in the VD has shorter distance than to any other centres of neighbouring cells to this cell. The central points of the circles in the relative position graph are regarded as sites of the DT. By connecting any two central points, i.e., sites, with a line segment, a DT is formed from the relative position graph. The relative position graph is converted into DT and VD, each of which is a planar graph.

The positions of the centres of the circles obtained from the first stage, illustrated in Fig. 4.7A, is used to partition the plane into the resulting Voronoi diagram.

In Fig. 4.7C, the relative position graph for the *apte* circuit is converted into its corresponding DT, illustrated by fine lines, and corresponding VD, depicted by bold lines. The RPM corresponding to this VD is given in (4.14).

$$\begin{pmatrix} 0 & 21 & 11 & 11 & 11 & 11 & 11 & 22 & 11 \\ 0 & 0 & 11 & 11 & 11 & 11 & 22 & 22 & 22 \\ 0 & 0 & 0 & 11 & 22 & 22 & 22 & 22 & 22 \\ 0 & 0 & 0 & 0 & 22 & 22 & 22 & 22 & 22 \\ 0 & 0 & 0 & 0 & 0 & 11 & 22 & 22 & 22 \\ 0 & 0 & 0 & 0 & 0 & 0 & 12 & 12 & 22 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 22 & 22 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 11 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{4.14}$$

As can been seen, by using the VD (see Fig. 4.7B), the layout is partitioned into nine cells that represent the relative position of the nine modules. After solving the second stage described in Chapter 5, a zero-deadspace fixed-outline floorplan is obtained (Fig. 4.7D).

4.4 The Sparse Relative Position Matrix

The motivation for using a sparse relative position matrix (SRPM) instead of the full RPM is to have fewer constraints and thus faster computation. The one-dimensional

six-module case given in Fig. 4.8 is used to show how the RPM and SRPM are built. The upper triangular RPM for Fig. 4.8 is shown in (4.15).

$$RPM = \begin{pmatrix} 0 & 11 & 11 & 12 & 11 & 11 \\ & 0 & 12 & 12 & 12 & 12 \\ & & 0 & 12 & 12 & 11 \\ & & & 0 & 11 & 11 \\ & & & & 0 & 11 \\ & & & & & 0 \end{pmatrix} \tag{4.15}$$

The distances between modules are obtained from their relative positions and included in a distance matrix (DM), which indicates the distances among modules. We take a one-dimension six-module case in Fig. 4.8 as an example. The DM for this case is shown in (4.16). A parameter δ is defined to discriminate among the distances. Those modules with distance at least δ from any given module can be eliminated from consideration. Considering module 1 as the current module in Fig. 4.8, for instance, and $\delta = 2$, the neighbouring modules within a distance of δ are modules 3, 4, and 5. Within a distance of δ from a current module, neighboring modules associated with the current modules are included to indicate the significance of relative position of modules in the DM. For instance, DM(1,3)=2, DM(1,4)=1, DM(1,5)=1, in (4.16) imply that modules 3, 4, and 5 are significant to module 1. Modules 2 and 6 are located farther than δ from module 1 thus the relative positions between modules 1 and 2, as well as between modules 1 and 6 are considered insignificant (the distance from the current module 1 to module 2 or module 6 exceeds δ, DM(1,2)=4, DM(1,6)=3; note that DM(1,2)>δ, DM(1,6)>δ). The insignificant distances are highlighted in the first row of (4.16).

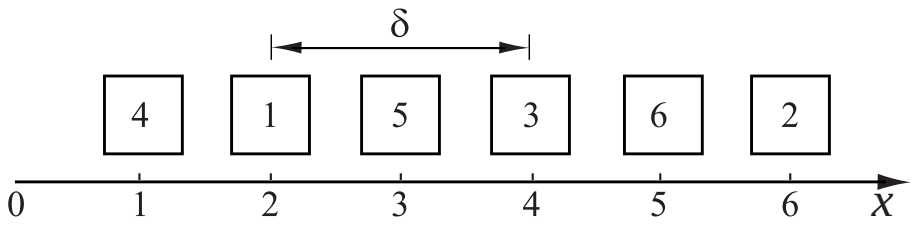


Figure 4.8: *One-dimensional (one-row) six-module example.*

$$DM = \begin{pmatrix} 0 & \mathbf{4} & 2 & 1 & 1 & \mathbf{3} \\ & 0 & 2 & \mathbf{5} & \mathbf{3} & 1 \\ & & 0 & \mathbf{3} & 1 & 1 \\ & & & 0 & 2 & \mathbf{4} \\ & & & & 0 & 2 \\ & & & & & 0 \end{pmatrix} \tag{4.16}$$

By replacing insignificant entries by zeros, we form a sparse DM (SDM) in (4.17). For instance, DM(2,4) = 5 denotes that the distance between modules 2 and 4 is 5. It is only necessary to record the relative position if two modules have distance less than 3. Therefore, the element in the SRPM corresponding to (2,4) is recorded to be zero, i.e., SRPM(2,4):=0. The corresponding entries in the RPM shown in (4.15) are replaced by zeros to form an SRPM in (4.18).

$$SDM = \begin{pmatrix} 0 & \mathbf{0} & 2 & 1 & 1 & \mathbf{0} \\ & 0 & 2 & \mathbf{0} & \mathbf{0} & 1 \\ & & 0 & \mathbf{0} & 1 & 1 \\ & & & 0 & 2 & \mathbf{0} \\ & & & & 0 & 2 \\ & & & & & 0 \end{pmatrix} \tag{4.17}$$

$$SRPM = \begin{pmatrix} 0 & \mathbf{0} & 11 & 12 & 11 & \mathbf{0} \\ & 0 & 12 & \mathbf{0} & \mathbf{0} & 12 \\ & & 0 & \mathbf{0} & 12 & 11 \\ & & & 0 & 11 & \mathbf{0} \\ & & & & 0 & 11 \\ & & & & & 0 \end{pmatrix} \tag{4.18}$$

An efficient approach for building a sparse RPM for the two-dimensional case is illustrated using *ami*33. The underlying idea is similar to the one-dimensional case. A VD is illustrated in Fig. 4.9A; δ now denotes the number of layers of neighbouring modules. The distance δ in the two-dimensional case denotes the number of layer of neighboring modules. For example, δ=1 indicates that the central module is closely related to its *first*-layer neighbouring modules. Similarly, δ=2 indicates that the central module is closely related to its *second*-layer neighbouring modules in Fig. 4.9A. Those modules beyond distance at δ from any given central module are not significant in terms of relative position between modules. In Fig. 4.9, for instance, central module 14 is closely related to its first-layer adjacent modules. The remaining modules need not be considered.

Every module is adjacent to other modules. Module 10, for instance, is adjacent to modules 9, 11, 12, 13, and 19 in the VD. Module 14 is adjacent to underlined modules 1, 6, 15, 16, 19, and 21 (Fig. 4.9A). Module 24 is adjacent to modules 1, 15, 22, 23, 25, and 27. After the layout problem is solved (Fig. 4.9B), module 10 becomes adjacent to modules 9, 12, 13, 19 and 21. Module 14 is adjacent to underlined modules 1, 6, 15, 19, and 21. Module 24 is adjacent to modules 1, 15, 22, 23, and 24.

Entries in the distance matrix (DM) are one if two moduels are adjacent and zero otherwise; (4.19) gives the DM for the *ami*33 as our example. Use of SRPM instead of RPM as constraints in the second stage model improves computational efficiency. Constraints provided by SRPM is fewer than those supplied by RPM.

$$DM = \begin{array}{c} \\ 1 \\ 2 \\ \vdots \\ 6 \\ \vdots \\ 14 \\ \vdots \\ 33 \end{array} \begin{array}{c} \begin{array}{cccccccccccccc} 1 & 2 & \dots & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & \dots & 33 \end{array} \\ \begin{pmatrix} 0 & 0 & \dots & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & & \dots & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & & 0 & & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & \dots & 0 \\ & & & & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ & & & & & & & & 0 & & \dots & \dots & \dots & 0 \end{pmatrix} \end{array} \tag{4.19}$$

4.5 Summary

In this chapter, geometrical structure of non-overlap among modules was described. Various non-overlap constraints were studied. Two modules can be separated hori-

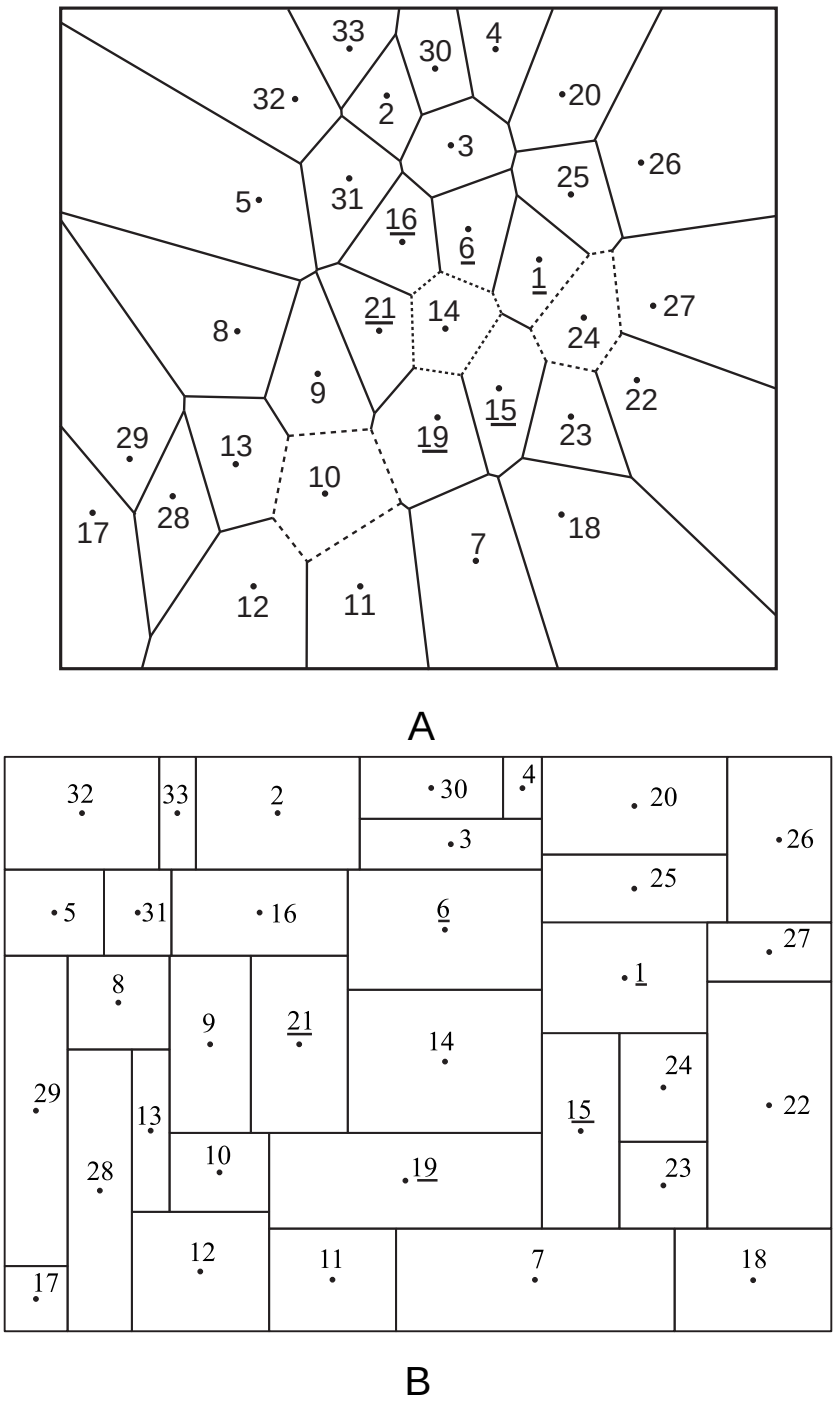


Figure 4.9: *Example for the ami*33 *circuit. A: the Voronoi diagram; B: the corresponding floorplan.*

zontally, vertically and diagonally. Therefore, relative positions of two modules can be represented as horizontal, vertical and diagonal separation.

This chapter presented how non-overlap constraints that are originally disjunctive, nonlinear and non-convex are linearized. RPM was described by introducing the non-overlap constraints and the geometrical structure of overlapped modules. The relative position graph obtained from the first stage is converted to a planar graph by the VD and DT and thus the RPM is encoded for a floorplan problem.

VD is a useful geometric structure that represents distance relationships and partitions a plane into Voronoi cells. Use of VD spreads out modules in the floorplan. VD and DT transform the relative position graph into a planar graph and produce a sparse relative position matrix (SRPM). To accelerate the computation an SRPM technique was proposed. Fewer constraints are required for the second stage formulation by using SRPM.

In the next chapter, we propose convex optimization models to obtain final floorplan solution.

Chapter 5

The Second Stage Convex Optimization Model

In the second stage, we determine the precise locations and dimensions of the modules while minimizing the total wirelength. We focus on fixed-outline floorplanning, and use Semidefinite Programming (SDP) and Second Order Cone Programming (SOCP) formulations, which are both convex optimization problems. The idea behind is that each local minimum of such a problem solved by this convex optimization model must also be a global minimum. In this thesis, these two approaches, particularly, the latter, will be used to minimize wirelength, and yields deadspace-free and overlap-free floorplans. In this chapter, we first review a previous integer linear programming (ILP) model for floorplanning (Sutanthavibul *et al.*, 1991). We describe SDP-based model with its experimental results, then SOCP-based model and its experiments (Luo *et al.*, 2008*b*). We also compare their experimental results in the section of SOCP experiments (Section 5.6).

This chapter is organized as follows. The ILP model is reviewed in Section 5.1. The SDP-based model is described in Section 5.2 and its experimental results are reported

in Section 5.3. The SOCP-based model is discussed in Section 5.4. Relationship of SDP md SOC constraints are described in Section 5.5. Experimental results with the SOCP model is reported in Section 5.6, where experimental results of the SDP and SOCP models are also compared. Finally, this chapter is summarized in Section 5.7.

5.1 Previous ILP Model for Floorplanning

Some previous research on floorplanning used linear programming and mixed integer linear programming (ILP) (Sutanthavibul *et al.*, 1991; Chen and Kuh, 2000; Young *et al.*, 2000*a*; Young *et al.*, 2001; Chu and Young, 2004).

In this section, we present a previous mixed integer linear programming formulation for floorplanning (Sutanthavibul *et al.*, 1991). The objective is to minimize the area of the floorplan. A set of n modules $S = \{1, 2, ..., n\}$ are given, with height h_i and width w_i for ith module. Let (x_i, y_i) and (x_j, y_j) denote the coordinates of the *lower corners* of modules i and j $(1 \leq i, j \leq n)$. The non-overlap constraint for modules i and j is as follows: one of the following linear constraints should be held at any rate (Sutanthavibul *et al.*, 1991).

$$x_i + w_i \leq x_j \qquad \text{if module } i \text{ is on the left of module } j \tag{5.1}$$

$$y_i + h_i \leq y_j \qquad \text{if module } i \text{ is below of module } j \tag{5.2}$$

$$x_i - w_j \geq x_j \qquad \text{if module } i \text{ is on the right of module } j \tag{5.3}$$

$$y_i - h_j \geq y_j \qquad \text{if module } i \text{ is above of module } j \tag{5.4}$$

If there is no overlap along the x-axis for modules i and j, then either constraint (5.1), or constraint (5.3) is satisfied. If there is no overlap along the y-axis for modules i and j, then either constraint (5.2), or constraint (5.4) is satisfied. Therefore, two 0-1 integer variables, x_{ij} and y_{ij}, are introduced for each pair of modules i and j

(Sutanthavibul *et al.*, 1991).

Table 5.1: Interpretation of the integer variables

x_{ij}	y_{ij}	**Interpretation**
0	0	Constraint (5.1) is satisfied
0	1	Constraint (5.2) is satisfied
1	0	Constraint (5.3) is satisfied
1	1	Constraint (5.4) is satisfied

There are four possible choices shown in Table 5.1. If w_F and h_F are width and height on the floorplan respectively, then $|x_i - x_j| \leq w_F$ and $|y_i - y_j| \leq h_F$. Therefore, non-overlap constraints for any pair of modules i and j may be given with x_{ij} and y_{ij} as follows:

$$x_i + w_i \leq x_j + w_F(x_{ij} + y_{ij}) \tag{5.5}$$

$$y_i + h_i \leq y_j + h_F(1 + x_{ij} - y_{ij}) \tag{5.6}$$

$$x_i - w_j \geq x_j - w_F(1 - x_{ij} + y_{ij}) \tag{5.7}$$

$$y_i - h_j \geq y_j - h_F(2 - x_{ij} - y_{ij}) \tag{5.8}$$

It can be verified that one and only one non-overlap constraint from constraints (5.5)-(5.8) above is enforced. Every module is enveloped on the floorplan with width w_F and height H. Hence, $x_i + w_i \leq w_F$ and $y_i + h_i \leq H$, $1 \leq i \leq n$. If the width w_F is fixed, then the area of the floorplan may be minimized by minimizing the height H. Therefore, the area minimization problem for the floorplanning by 0-1 integer linear

programming is formulated as follows (Sutanthavibul *et al.*, 1991):

$$
\begin{aligned}
&\min \quad H \\
&\text{s.t.} \\
&\qquad x_i + w_i \le w_F \qquad \forall\, 1 \le i \le n, \\
&\qquad y_i + h_i \le H \qquad \forall\, 1 \le i \le n, \\
&\qquad x_i + w_i \le x_j + w_F(x_{ij} + y_{ij}) \qquad \forall\, 1 \le i, j \le n, \\
&\qquad y_i + h_i \le y_j + H(1 + x_{ij} - y_{ij}) \qquad \forall\, 1 \le i, j \le n, \\
&\qquad x_i - w_j \ge x_j - w_F(1 - x_{ij} + y_{ij}) \qquad \forall\, 1 \le i, j \le n, \\
&\qquad y_i - h_j \ge y_j - H(2 - x_{ij} - y_{ij}) \qquad \forall\, 1 \le i, j \le n, \\
&\qquad x_i \ge 0 \qquad \forall\, 1 \le i \le n, \\
&\qquad y_i \ge 0 \qquad \forall\, 1 \le i \le n,
\end{aligned}
\tag{5.9}
$$

where all module coordinates are positive, $x_i \ge 0$ and $y_i \ge 0$.

The mixed integer linear programming formulation (5.9) of the floorplanning problem with n modules requires $2n$ continuous variables, $n(n-1)$ integer variables, and $2n^2$ linear constraints. Solving the formulation (5.9) will produce an optimal floorplan solution (Sutanthavibul *et al.*, 1991).

5.2 SDP-based Convex Optimization Model

5.2.1 Semidefinite Programming

SDP is a generalization of linear programming (LP) and is a convex optimization problem that can be effectively solved by interior-point algorithms (Helmberg *et al.*, 1996; Wolkowicz *et al.*, 2000). In recent years it has become one of the most exciting and active research areas in optimization. It greatly attracts our attention

as we can take advantage of the property of SDP that any local minimum is a global solution of the wirelength minimization problem (Helmberg *et al.*, 1996). In this section, area and aspect ratio constraints are formulated by applying semidefinite optimization techniques. Additionally, the disjunctive non-overlap constraints in the previous model in Chapter 3 are replaced by linear constraints due to already have the relative positions of modules.

Definition 5.2.1 (*Positive Semidefinite*) *A matrix $X \in S^n$, where S^n denotes the set of $n \times n$ real symmetric matrices, is said to be Positive Semidefinite (psd) if for all $y \in R^n$:*

$$y^T X y = \sum_{i,j} X_{i,j} y_i y_j \geq 0 \tag{5.10}$$

Definition 5.2.2 (*Semidefinite Programming*) *In $\mathbb{S}_n$, a semidefinite programming (SDP) is described as:*

$$\begin{aligned} &\min \quad C \bullet X \\ &\text{s.t.} \\ &\qquad A_i \bullet X = b_i, \text{ for } i = 1, 2, ...m, \\ &\qquad X \succeq 0, \end{aligned} \tag{5.11}$$

where $A \bullet X = \sum_{i,j} A_{i,j} B_{i,j} = \text{trace}(B^T A)$; X is psd.

Definition 5.2.3 (*Principal Minor*) *Given $X \in \mathbb{S}_n$ and a subset $I \subseteq \{1...n\}$, the principal submatrix of X corresponding to I is the submatrix with rows and columns indexed by I. Its determinant is called the principal minor.*

Definition 5.2.4 (*Primal and Dual Problems of Semidefinite Programming*) *The primal and dual problems of SDP in the standard form are as follows:*

Primal problem	Dual problem
min $C \bullet X$	max $b^T y$
s.t.	s.t.
$A_i \bullet X = b_i, \forall\, i = 1...m,$	$Z = \sum_{i=1}^{m} y_i A_i - C,$
$X \succeq 0.$	$Z \succeq 0.$

where $A \bullet X = \sum_{i,j} A_{i,j} B_{i,j} = \text{trace}(B^T A)$, X is psd; b and y (as well as other small alphabets) denote column vectors, while y_i's denote y's ith component.

5.2.2 Area Constraints

The area constraint, $w_i h_i = a_i$, for each soft module can be relaxed as

$$w_i h_i \geq a_i. \tag{5.12}$$

If the space allocated for a module exceeds the required amount, it is still feasible for layout implementation. According to the property of positive semidefinite matrices that all principal minors are non-negative, and $a_i > 0$, a semidefinite constraint for this relaxed area constraint can be written as

$$\begin{pmatrix} w_i & \sqrt{a_i} \\ \sqrt{a_i} & h_i \end{pmatrix} \succeq 0. \tag{5.13}$$

Theorem 1 *If X is Positive Semidefinite (psd), then all its principal minors are non-negative (Vandenberghe and Boyd, 1996; Wolkowicz* et al.*, 2000).*

Example Given a symmetric matrix A as

$$A = \begin{pmatrix} 8 & -2 & 2 \\ -2 & 5 & 4 \\ 2 & 4 & 5 \end{pmatrix}$$

$I = \{1\}$, the principal submatrix is $A(I) = 8, det(A) = 8.$

$I = \{2\}$, the principal submatrix is $A(I) = 5, det(A) = 5.$

$I = \{3\}$, the principal submatrix is $A(I) = 5, det(A) = 5.$

$I = \{1, 2\}$,

$$A = \begin{pmatrix} 8 & -2 \\ -2 & 5 \end{pmatrix}, \quad det(A) = 36.$$

$I = \{1, 3\}$,

$$A = \begin{pmatrix} 8 & 2 \\ 2 & 5 \end{pmatrix}, \quad det(A) = 36.$$

$I = \{2, 3\}$,

$$A = \begin{pmatrix} 5 & 4 \\ 4 & 5 \end{pmatrix}, \quad det(A) = 9.$$

$I = \{1, 2, 3\}$,

$$A = \begin{pmatrix} 8 & -2 & 2 \\ -2 & 5 & 4 \\ 2 & 4 & 5 \end{pmatrix}, \quad det(A) = 0.$$

As all the principal minors of A are non-negative, A is *psd*.

Theorem 2 SDP for Area *For each soft module with height h_i, width w_i, and area a_i. If $a_i > 0$, then the following is satisfied:*

$$w_i h_i \geq a_i \Longleftrightarrow \begin{pmatrix} w_i & \sqrt{a_i} \\ \sqrt{a_i} & h_i \end{pmatrix} \succeq 0, \ \forall \ i \tag{5.14}$$

Proof

(1). To show necessity, consider the following inequalities:

$$\begin{aligned} w_i &> 0 \\ h_i &> 0 \\ w_i h_i &\geq a_i \end{aligned} \tag{5.15}$$

then, we have

$$\begin{aligned} & w_i h_i \geq (\sqrt{a_i})^2 > 0 \\ \Rightarrow & \ w_i h_i - (\sqrt{a_i})^2 \geq 0 \\ \Rightarrow & \begin{pmatrix} w_i & \sqrt{a_i} \\ \sqrt{a_i} & h_i \end{pmatrix} \succeq 0 \end{aligned} \tag{5.16}$$

(2). To show sufficiency, since

$$X = \begin{pmatrix} w_i & \sqrt{a_i} \\ \sqrt{a_i} & h_i \end{pmatrix} \succeq 0 \ \text{ is psd}$$

all the principal minors of X are non-negative, e.g.

$$\begin{aligned} w_i &\geq 0 \\ h_i &\geq 0 \\ w_i h_i - (\sqrt{a_i})^2 &\geq 0, \end{aligned} \tag{5.17}$$

therefore we obtain

$$w_i h_i \geq a_i.$$

5.2.3 Aspect Ratio Constraints

Suppose a list of the aspect ratios β_i of module i is given. R_i^{low} and R_i^{up} are lower and upper bounds on the aspect ratio of module i. To avoid excessively narrow modules in either direction in the floorplan, the aspect ratio β_i for module i is defined as

$$\beta_i = \max\{h_i, w_i\} / \min\{h_i, w_i\}. \tag{5.18}$$

Assume that the aspect ratio of module i must be bounded above by a given value $\beta_i^* > 0$. So, $\beta_i^* \geq \beta_i$.

Theorem 3 (SDP for Aspect Ratio for height) *Given* $w_i^{low} = h_i^{low} = \sqrt{a_i/\beta_i^*}$ *where* $a_i = w_i h_i$,

(1)

$$a_i \beta_i^* \geq h_i^2$$

(2)

$$a_i \beta_i^* \geq h_i^2 \Longleftrightarrow \begin{pmatrix} \beta_i^* & h_i \\ h_i & a_i \end{pmatrix} \succeq 0, \ \forall \ i$$

Proof.

(1). To show the SDP for Aspect Ratio for height, since

$$w_i \geq w_i^{low} = \sqrt{a_i/\beta_i^*} > 0, \tag{5.19}$$

$$w_i^2 \geq a_i/\beta_i^*, \tag{5.20}$$

$$\beta_i^* w_i^2 \geq a_i, \tag{5.21}$$

$$\beta_i^* \geq h_i / w_i. \tag{5.22}$$

With Inequality (5.22) and $w_i h_i = a_i$, we obtain

$$\beta_i^* \geq h_i^2 / a_i. \tag{5.23}$$

Further,

$$a_i \beta_i^* \geq h_i^2 \tag{5.24}$$

(2).

(2.1). To show necessity, consider the following inequalities:

$$\begin{aligned} a_i &> 0 \\ \beta_i^* &> 0 \\ a_i \beta_i^* &\geq h_i^2 \end{aligned} \tag{5.25}$$

then we have

$$\begin{aligned} & a_i \beta_i^* \geq h_i^2 > 0 \\ \Rightarrow\ & a_i \beta_i^* - h_i^2 \geq 0 \\ \Rightarrow\ & \begin{pmatrix} \beta_i^* & h_i \\ h_i & a_i \end{pmatrix} \succeq 0 \end{aligned} \tag{5.26}$$

(2.2). To show sufficiency, since

$$X = \begin{pmatrix} \beta_i^* & h_i \\ h_i & a_i \end{pmatrix} \succeq 0 \text{ is psd}$$

all the principal minors of X are non-negative, e.g.

$$\begin{aligned} a_i &\geq 0 \\ \beta_i^* &\geq 0 \\ a_i\beta_i^* - h_i^2 &\geq 0, \end{aligned} \tag{5.27}$$

therefore we obtain

$$a_i\beta_i^* \geq h_i^2$$

Theorem 4 (SDP for Aspect Ratio for width) *Given* $w_i^{low} = h_i^{low} = \sqrt{a_i/\beta_i^*}$ *where* $a_i = w_i h_i$,

(1)

$$a_i\beta_i^* \geq w_i^2$$

(2)

$$a_i\beta_i^* \geq w_i^2 \Longleftrightarrow \begin{pmatrix} \beta_i^* & w_i \\ w_i & a_i \end{pmatrix} \succeq 0, \ \forall \ i$$

Proof.

(1). To show SDP for Aspect Ratio for width, since

$$h_i \geq h_i^{low} = \sqrt{a_i/\beta_i^*} > 0, \tag{5.28}$$

$$h_i^2 \geq a_i/\beta_i^*, \tag{5.29}$$

$$\beta_i^* h_i^2 \geq a_i, \tag{5.30}$$

$$\beta_i^* \geq w_i/h_i. \tag{5.31}$$

With Inequality (5.31) and $w_i h_i = a_i$, we obtain

$$\beta_i^* \geq w_i^2/a_i. \tag{5.32}$$

Further,

$$a_i \beta_i^* \geq w_i^2 \tag{5.33}$$

(2).

(2.1). To show necessity, consider the following inequalities:

$$\begin{aligned} a_i &> 0 \\ \beta_i^* &> 0 \\ a_i \beta_i^* &\geq w_i^2 \end{aligned} \tag{5.34}$$

then we have

$$\begin{aligned} & a_i \beta_i^* \geq w_i^2 > 0 \\ \Rightarrow\ & a_i \beta_i^* - w_i^2 \geq 0 \\ \Rightarrow\ & \begin{pmatrix} \beta_i^* & w_i \\ w_i & a_i \end{pmatrix} \succeq 0 \end{aligned} \tag{5.35}$$

(2.2). To show sufficiency, since

$$X = \begin{pmatrix} \beta_i^* & w_i \\ w_i & a_i \end{pmatrix} \succeq 0 \ \text{ is psd}$$

all the principal minors of X are non-negative, e.g.

$$\begin{aligned} a_i &\geq 0 \\ \beta_i^* &\geq 0 \\ a_i \beta_i^* - w_i^2 &\geq 0, \end{aligned} \tag{5.36}$$

therefore we obtain

$$a_i \beta_i^* \geq w_i^2.$$

In summary, the area and aspect ratios for height and width can be expressed as follows, respectively:

$$w_i h_i \geq a_i \iff \begin{pmatrix} w_i & \sqrt{a_i} \\ \sqrt{a_i} & h_i \end{pmatrix} \succeq 0, \ \forall \ i \tag{5.37}$$

$$a_i \beta_i^* \geq h_i^2 \iff \begin{pmatrix} \beta_i^* & h_i \\ h_i & a_i \end{pmatrix} \succeq 0, \ \forall \ i \tag{5.38}$$

$$a_i \beta_i^* \geq w_i^2 \iff \begin{pmatrix} \beta_i^* & w_i \\ w_i & a_i \end{pmatrix} \succeq 0, \ \forall \ i \tag{5.39}$$

Incorporating these constraints, the problem of minimizing the total wirelength for fixed-outline floorplanning can be formulated as:

$$\min_{(x_i,y_i),w_i,h_i} \quad \sum_{1\le i<j\le n} c_{ij}L(x_i,x_j,y_i,y_j)$$

s.t.

$$\begin{aligned}
& x_i + \frac{1}{2}w_i \le \frac{1}{2}\bar{w}_F \quad \forall\, i \\
& y_i + \frac{1}{2}h_i \le \frac{1}{2}\bar{h}_F \quad \forall\, i \\
& \frac{1}{2}w_i - x_i \le \frac{1}{2}\bar{w}_F \quad \forall\, i, \\
& \frac{1}{2}h_i - y_i \le \frac{1}{2}\bar{h}_F \quad \forall\, i, \\
& w_i^{low} \le w_i \le w_i^{up} \quad \forall\, i, \\
& h_i^{low} \le h_i \le h_i^{up} \quad \forall\, i, \\
& \begin{pmatrix} w_i & \sqrt{a_i} \\ \sqrt{a_i} & h_i \end{pmatrix} \succeq 0 \quad \forall\, i, \\
& \begin{pmatrix} \beta_i^* & h_i \\ h_i & a_i \end{pmatrix} \succeq 0, \quad \forall\, i, \\
& \begin{pmatrix} \beta_i^* & w_i \\ w_i & a_i \end{pmatrix} \succeq 0, \quad \forall\, i
\end{aligned} \tag{5.40}$$

where $1 \le i < j \le n$, $\bar{w}_F$ and $\bar{h}_F$ are the fixed width and height of the floorplan, and $L(x_i, x_j, y_i, y_j)$ is the *rectilinear* distance, $|x_i - x_j| + |y_i - y_j|$, between modules i and j. Note that the non-overlap constraints are absent in formulation (5.40). They are enforced in the form of linear constraints if the RPM is available from the first stage.

5.3 Experiments with the SDP Model

The SDP formulation as the second stage is solved using SeDuMi 1.1 (Sturm, 1999) and CSDP 5.0 (Borchers, 1999*a*; Borchers and Young, 2007). Results for the MCNC benchmark are shown in Table 5.2.

Table 5.2: MCNC results for SDP model

Circuit	Total area (mm^2)	Our Method			
		Our area (mm^2)	Runtime (SeDuMi) (s)	Runtime (CSDP) (s)	HPWL (mm)
apte	46.56	46.56	2.464	1.719	397
xerox	19.35	19.35	6.319	4.635	411
hp	8.30	8.30	4.287	2.763	142
ami33	1.16	1.16	470	159	50.08
ami49	35.4	35.4	9465	1236	699

SeDuMi (Sturm, 1999), standing for Self-Dual-Minimization, is a Matlab software package for solving optimization problems with linear, quadratic, and semidefinite constraints using the *self-dual* embedding technique over self-dual homogeneous cones.

CSDP (Borchers, 1999*a*) is a predictor-corrector version of the primal-dual barrier method (Helmberg *et al.*, 1996) that solves optimization problems with LP and SDP constraints (Borchers, 1999*b*). CSDP is written in C and can run in parallel on systems with multiple processors and shared memory (Borchers and Young, 2007). It takes advantage of constraint matrices with symmetric and sparse structure for efficiency. In addition to its default termination criteria, CSDP allows the user to terminate the

solution process after any iteration. It is coded in C whereas SeDuMi is in Matlab and C. CSDP has some features described above that improve computational efficacy. Therefore, it is faster than SeDuMi for our SDP problem. Table 5.2 lists the CPU times for solving the MCNC circuits by CSDP and SeDuMi. The table shows that CSDP is about three times faster than SeDuMi for ami33 and is up to eight times faster for ami49. There are memory limitation issues when SeDuMi is applied to large (e.g., 100 modules) problems.

5.4 SOCP-based Convex Optimization Model

Second Order Cone Programming (SOCP) is a special class of convex programming problems, in which a linear objective function is minimized subject to Second Order Cone (SOC) constraints. As a special case of convex optimization, SOCPs have attracted much attention; They can be efficiently solved by specialized interior-point methods (Lobo *et al.*, 1998; Andersen *et al.*, 2003).

A *standard* SOC (also known as quadratic or Lorentz cone) of dimension n is expressed as (Lobo *et al.*, 1998):

$$\mathcal{C}_n = \left\{ \left[\begin{array}{c} u \\ t \end{array} \right] \middle| u \in \mathbf{R}^{n-1}, t \in \mathbf{R}, \|u\| \leq t \right\}, \tag{5.41}$$

where $u \in \mathbf{R}^{n-1}, t \in \mathbf{R}$, and the SOC defines a convex set. If n=1, the SOC degenerates to a ray on the t-axis with origin at t=0; for n=3, the SOC is illustrated in Fig. 5.1.

A SOC constraint of dimension n is defined as

$$\|Ax + b\| \leq c^T x + d \Longleftrightarrow \left[\begin{array}{c} A \\ c^T \end{array} \right] x + \left[\begin{array}{c} b \\ d \end{array} \right] \in \mathcal{C}_n, \tag{5.42}$$

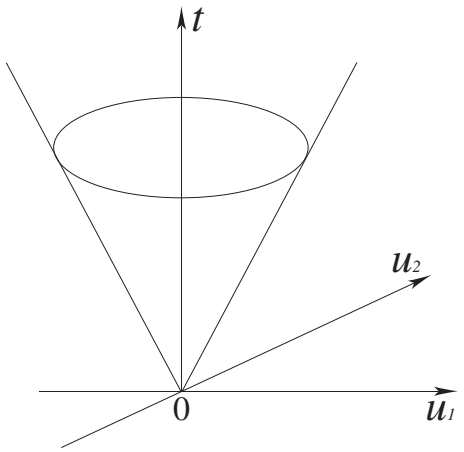


Figure 5.1: *Second-order cone of dimension 3 (n=3 in equation (5.41)).*

where $x \in \mathbf{R}^k$, parameter $A \in \mathbf{R}^{(n-1)\times k}$, $b \in \mathbf{R}^{n-1}$, $c \in \mathbf{R}^k$, $d \in \mathbf{R}$. Therefore, an SOCP is a convex optimization problem of the form:

$$\begin{aligned} &\min \quad g^T x \\ &\text{s.t.} \quad \|A_i x + b_i\| \leq c_i^T x + d_i, \quad \forall\, i = 1, ..., L, \end{aligned} \tag{5.43}$$

where the optimization variable is $x \in \mathbf{R}^n$, $g \in \mathbf{R}^n$ is the objective function, $A_i \in \mathbf{R}^{(n_i-1)\times n}$, $b_i \in \mathbf{R}^{n_i-1}$, $c_i \in \mathbf{R}^n$, and $d_i \in \mathbf{R}$. This SOCP problem may be converted into standard form by introducing variables $u_i \in \mathbf{R}^{n_i-1}$ and $t_i \in \mathbf{R}$ as follows:

$$\begin{aligned} &\min \quad g^T x \\ &\text{s.t.} \\ &\qquad \|u_i\| \leq t_i, \quad \forall\, i = 1, ..., L \\ &\qquad u_i = A_i x + b_i, \quad \forall\, i = 1, ..., L \\ &\qquad t_i = c_i^T x + d_i, \quad \forall\, i = 1, ..., L, \end{aligned} \tag{5.44}$$

where $\|u\| \leq t$ is the standard SOC constraint ($\|u\| = (u^T u)^{\frac{1}{2}}$). In the following

sections, we formulate area and aspect ratio constraints as SOC constraints. The convex optimization package MOSEK can solve SOCP problems (MOSEK, 2006).

An SDP problem shown in definition (5.2.2) can be solved efficiently using interior-point algorithms in $O(\sqrt{\sum_i n_i})$ iterations of complexity $O(n^2 \sum_i n_i^2)$. However, using the same algorithms, an SOCP problem takes $O(\sqrt{L})$ iterations, each of complexity $O(n^2 \sum_i n_i)$, where n_i is the dimension number of second order cone for ith constraint; n is the dimension number of second order cone; L is the number of variables. Therefore, SOCPs can be solved more efficiently than SDPs (Lobo *et al.*, 1998). SOCPs have been found to highly outperform SDP problems in terms of computational efficiency (Lobo *et al.*, 1998). Hence, an SOCP is applied to fixed-outline floorplanning motivated by its simpler structure and its potential to be solved more efficiently than SDP. For LP, QP(Quadratic Programming), QCQP (Quadratically Constrained Quadratic Program), SOCP, and SDP problems, the model generality and solution difficulty are shown in Fig. 5.2.

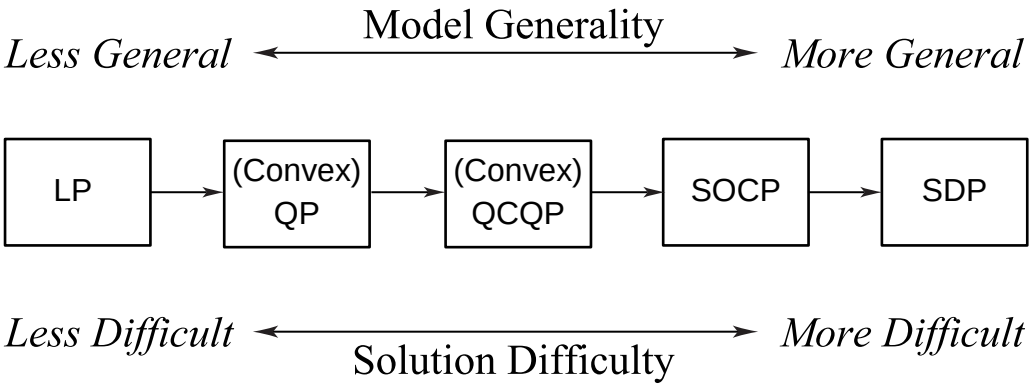


Figure 5.2: *The model generality and solution difficulty of several convex optimization problems.*

5.4.1 Area Constraints

The area constraint, $w_i h_i = a_i$, for each soft module can be relaxed as $w_i h_i \geq a_i$. For $a_i > 0$, this can be formulated as an SOC constraint:

$$\begin{aligned} & w_i h_i \geq a_i \\ \Longleftrightarrow\ & h_i^2 + 2h_i w_i + w_i^2 \geq (h_i^2 - 2h_i w_i + w_i^2) + 4a_i \\ \Longleftrightarrow\ & (h_i + w_i)^2 \geq (h_i - w_i)^2 + 4a_i \geq 0 \\ \Longleftrightarrow\ & h_i + w_i \geq \sqrt{(h_i - w_i)^2 + (2\sqrt{a_i})^2} \\ \Longleftrightarrow\ & h_i + w_i \geq \left\| \begin{pmatrix} h_i - w_i \\ 2\sqrt{a_i} \end{pmatrix} \right\|_2, \ \ \forall\, i. \end{aligned} \tag{5.45}$$

5.4.2 Aspect Ratio Constraints

A list of aspect ratios β_i of module i is defined in (5.18) to avoid excessively narrow modules in either direction in the floorplan. The aspect ratio of module i must be bounded above by a given value $\beta_i^* > 0$. Given $w_i^{low} = h_i^{low} = \sqrt{a_i/\beta_i^*}$, where $a_i = w_i h_i$, then $w_i \geq w_i^{low} > 0$, $w_i^2 \geq a_i/\beta_i^*$, $\beta_i^* w_i^2 \geq a_i$, $\beta_i^* \geq h_i/w_i$. Similarly, since $h_i \geq h_i^{low} > 0$, $\beta_i^* \geq w_i/h_i$. With inequality $\beta_i^* \geq h_i/w_i$ and $w_i h_i = a_i$, we obtain $\beta_i^* \geq h_i^2/a_i$. Further, we obtain $a_i \beta_i^* \geq h_i^2$. Combining inequality $\beta_i^* \geq w_i/h_i$ and $w_i h_i = a_i$ yields $\beta_i^* \geq w_i^2/a_i$. Therefore, $a_i \beta_i^* \geq w_i^2$.

The aspect ratio constraint for the height of a module denoted by inequality

$a_i\beta_i^* \geq h_i^2$ can be formulated by an SOC constraint as follows:

$$\begin{aligned}
& a_i\beta_i^* \geq h_i^2 \\
\Longleftrightarrow\ & a_i^2 + 2a_i\beta_i^* + {\beta_i^*}^2 \geq a_i^2 - 2a_i\beta_i^* + {\beta_i^*}^2 + 4{h_i}^2 \\
\Longleftrightarrow\ & (a_i + \beta_i^*)^2 \geq (a_i - \beta_i^*)^2 + (2h_i)^2 \geq 0 \\
\Longleftrightarrow\ & a_i + \beta_i^* \geq \sqrt{(a_i - \beta_i^*)^2 + (2h_i)^2} \\
\Longleftrightarrow\ & a_i + \beta_i^* \geq \left\| \begin{pmatrix} a_i - \beta_i^* \\ 2h_i \end{pmatrix} \right\|_2, \quad \forall\, i.
\end{aligned} \tag{5.46}$$

Similarly, the aspect ratio constraint for the width of a module can be formulated as follows:

$$\begin{aligned}
& a_i\beta_i^* \geq w_i^2 \\
\Longleftrightarrow\ & a_i^2 + 2a_i\beta_i^* + {\beta_i^*}^2 \geq a_i^2 - 2a_i\beta_i^* + {\beta_i^*}^2 + 4{w_i}^2 \\
\Longleftrightarrow\ & (a_i + \beta_i^*)^2 \geq (a_i - \beta_i^*)^2 + (2w_i)^2 \geq 0 \\
\Longleftrightarrow\ & a_i + \beta_i^* \geq \sqrt{(a_i - \beta_i^*)^2 + (2w_i)^2} \\
\Longleftrightarrow\ & a_i + \beta_i^* \geq \left\| \begin{pmatrix} a_i - \beta_i^* \\ 2w_i \end{pmatrix} \right\|_2, \quad \forall\, i.
\end{aligned} \tag{5.47}$$

Incorporating these SOCP constraints, the complete problem of minimizing the

total wirelength for fixed-outline floorplanning can be formulated as:

$$\min_{(x_i,y_i),w_i,h_i} \quad \sum_{1\le i<j\le n} c_{ij}L(x_i,x_j,y_i,y_j)$$

s.t.

$$\begin{aligned}
& x_i + \frac{1}{2}w_i \le \frac{1}{2}\bar{w}_F \quad \forall\, i \\
& y_i + \frac{1}{2}h_i \le \frac{1}{2}\bar{h}_F \quad \forall\, i \\
& \frac{1}{2}w_i - x_i \le \frac{1}{2}\bar{w}_F \quad \forall\, i, \\
& \frac{1}{2}h_i - y_i \le \frac{1}{2}\bar{h}_F \quad \forall\, i, \\
& w_i^{low} \le w_i \le w_i^{up} \quad \forall\, i, \\
& h_i^{low} \le h_i \le h_i^{up} \quad \forall\, i, \\
& \left\| \begin{pmatrix} h_i - w_i \\ 2\sqrt{a_i} \end{pmatrix} \right\|_2 \le h_i + w_i, \forall\, i, \\
& \left\| \begin{pmatrix} a_i - \beta_i^* \\ 2h_i \end{pmatrix} \right\|_2 \le a_i + \beta_i^*, \forall\, i \\
& \left\| \begin{pmatrix} a_i - \beta_i^* \\ 2w_i \end{pmatrix} \right\|_2 \le a_i + \beta_i^*, \forall\, i
\end{aligned} \tag{5.48}$$

where $1 \le i < j \le n$, $\bar{w}_F$ and $\bar{h}_F$ are the fixed width and height of the floorplan, and $L(x_i, x_j, y_i, y_j)$ is the *rectilinear* distance, $|x_i - x_j| + |y_i - y_j|$, between modules i and j.

Obviously L is not a linear function, so we need to linearize it. By defining $u_{ij} = |x_i - x_j|$ and $v_{ij} = |y_i - y_j|$ and substituting for L using u_{ij} and v_{ij}, the objective function of (5.48) becomes $\sum_{1\le i<j\le n} c_{ij}(u_{ij} + v_{ij})$. The following four constraints are enforced in (5.48): $u_{ij} \ge x_i - x_j$, $u_{ij} \ge x_j - x_i$, $v_{ij} \ge y_i - y_j$ and $v_{ij} \ge y_j - y_i$. The model formulation of the fixed-outline floorplanning with only linear and SOC

constraints becomes:

$$\min_{(x_i,y_i),w_i,h_i} \quad \sum_{1\le i<j\le n} c_{ij}(u_{ij}+v_{ij})$$

s.t.

$$\begin{aligned}
& u_{ij} \ge x_i - x_j, \\
& u_{ij} \ge x_j - x_i, \\
& v_{ij} \ge y_i - y_j, \\
& v_{ij} \ge y_j - y_i, \\
& x_i + \frac{1}{2}w_i \le \frac{1}{2}\bar{w}_F \quad \forall\, i, \\
& y_i + \frac{1}{2}h_i \le \frac{1}{2}\bar{h}_F \quad \forall\, i, \\
& \frac{1}{2}w_i - x_i \le \frac{1}{2}\bar{w}_F \quad \forall\, i, \\
& \frac{1}{2}h_i - y_i \le \frac{1}{2}\bar{h}_F \quad \forall\, i, \\
& w_i^{low} \le w_i \le w_i^{up} \quad \forall\, i, \\
& h_i^{low} \le h_i \le h_i^{up} \quad \forall\, i, \\
& \left\| \begin{pmatrix} h_i - w_i \\ 2\sqrt{a_i} \end{pmatrix} \right\|_2 \le h_i + w_i \ \ \forall\, i, \\
& \left\| \begin{pmatrix} a_i - \beta_i^* \\ 2h_i \end{pmatrix} \right\|_2 \le a_i + \beta_i^* \ \ \forall\, i, \\
& \left\| \begin{pmatrix} a_i - \beta_i^* \\ 2w_i \end{pmatrix} \right\|_2 \le a_i + \beta_i^* \ \ \forall\, i,
\end{aligned} \tag{5.49}$$

where $1 \le i < j \le n$, $\bar{w}_F$ and $\bar{h}_F$ are the fixed width and height of the floorplan, and $(u_{ij} + v_{ij})$ is the rectilinear distance $|x_i - x_j| + |y_i - y_j|$ between modules i and j.

Now we have a convex SOCP model that can be efficiently solved using the com-

mercially available conic software package MOSEK (MOSEK, 2006). Note that the non-overlap constraints are not shown in (5.49). They are included in this formulation in the form of linear constraints derived from the SRPM.

5.5 Relationship of SDP and SOC Constraints

An SOC constraint is equivalent to a linear matrix inequality (Lobo *et al.*, 1998). The SOC may be embedded in the cone of positive semidefinite matrices as

$$\|u\| \leq t \Longleftrightarrow \begin{pmatrix} tI & u \\ u^T & t \end{pmatrix} \succeq 0, \quad \forall\, i \tag{5.50}$$

The area and aspect ratio constraints can be expressed as SDP constraints that can also be formulated to SOC constraints as follows:

For area:

$$\begin{pmatrix} w_i & \sqrt{a_i} \\ \sqrt{a_i} & h_i \end{pmatrix} \succeq 0 \Longleftrightarrow \left\| \begin{pmatrix} h_i - w_i \\ 2\sqrt{a_i} \end{pmatrix} \right\|_2 \leq h_i + w_i, \quad \forall\, i \tag{5.51}$$

For aspect ratio (height):

$$\begin{pmatrix} \beta_i^* & h_i \\ h_i & a_i \end{pmatrix} \succeq 0 \Longleftrightarrow \left\| \begin{pmatrix} a_i - \beta_i^* \\ 2h_i \end{pmatrix} \right\|_2 \leq a_i + \beta_i^*, \quad \forall\, i \tag{5.52}$$

For aspect ratio (width):

$$\begin{pmatrix} \beta_i^* & w_i \\ w_i & a_i \end{pmatrix} \succeq 0 \Longleftrightarrow \left\| \begin{pmatrix} a_i - \beta_i^* \\ 2w_i \end{pmatrix} \right\|_2 \leq a_i + \beta_i^*, \quad \forall\, i \tag{5.53}$$

5.6 Experimental Results with the SOCP Model

We use the clique model for transforming hypergraphs to two-pin nets and the half-perimeter wirelength (HPWL) to measure the quality of all our floorplans. For a set of modules with total area $\mathcal{A}$ and maximum whitespace fraction γ, as well as given aspect ratio ζ for a fixed outline, the height H_F and width W_F of the chip can be given as follows (Adya and Markov, 2003):

$$H_F = \sqrt{(1+\gamma)\mathcal{A}\zeta} \qquad W_F = \sqrt{(1+\gamma)\mathcal{A}/\zeta}. \tag{5.54}$$

Our method is applied to the standard MCNC and GSRC benchmarks and compared with several state-of-the-art academic floorplanners (Murata and Kuh, 1998; Adya and Markov, 2003; Adya *et al.*, 2004; PARQUET, 2006; Chen *et al.*, 2005; Sechen and Sangiovanni-Vincentelli, 1986; Cong *et al.*, 1999). All reported wirelengths are measured using the HPWL, also used by Murata and Kuh (1998), Adya and Markov (2003), Chen *et al.* (2005), and Cong *et al.* (1999). All the modules are chosen to be soft with fixed areas and variable dimensions, and (as an approximation) all the pins are assumed to be at the centres of the modules.

The second stage was solved by solver MOSEK (2006) on a Linux SUN Fire-V890 server with 16 1200-MHz processors and 32 GB RAM. The GAMS was compiled on this computer then the problem was solved by one CPU. We report only CPU time for the second stage. Solving the first stage takes significantly less time than the second stage. Therefore, the CPU time for the first stage is negligible. For these instances of fixed-outline floorplanning, we respect the dimensions of the floorplans provided in the MCNC and GSRC benchmarks, so that the fixed layout area is a sum of the areas of all of the modules included in a circuit if there is no whitespace. We choose ζ=1 in (5.54). (Note that the experimental results of SDP model was described in Section 5.3).

5.6.1 Experiments for MCNC and GSRC Benchmarks

We solved the first stage formulation (3.15) and second stage formulation (5.49) bridged by the VD. Each circuit was run once, and we report the resulting CPU time and HPWL wirelength in Table 5.3 for MCNC and Table 5.4 for GSRC. Compared with Table 5.3 and Table 5.2, the SOCP model is 26 times faster than CSDP with SDP and 78 times faster than SeDuMi with SDP for ami33; the SOCP model is 120 times faster than CSDP with SDP and 903 times faster than SeDuMi with SDP for ami49 (see Fig. 5.3 for comparison). Therefore, the SOCP model is significantly more efficient than the SDP model. A comparison between Table 5.2 and Table 5.3 shows that the total wirelengths produced by SDP and SOCP models are completely identical because both are convex optimization models.

Table 5.3: MCNC experimental results with our SOCP model

circuit	Total area (mm^2)	Our Method		
		Our area (mm^2)	Runtime (s)	HPWL (mm)
apte	46.56	46.56	1.03	397
xerox	19.35	19.35	1.12	411
hp	8.30	8.30	1.29	142
ami33	1.16	1.16	6.06	50.08
ami49	35.4	35.4	9.96	699

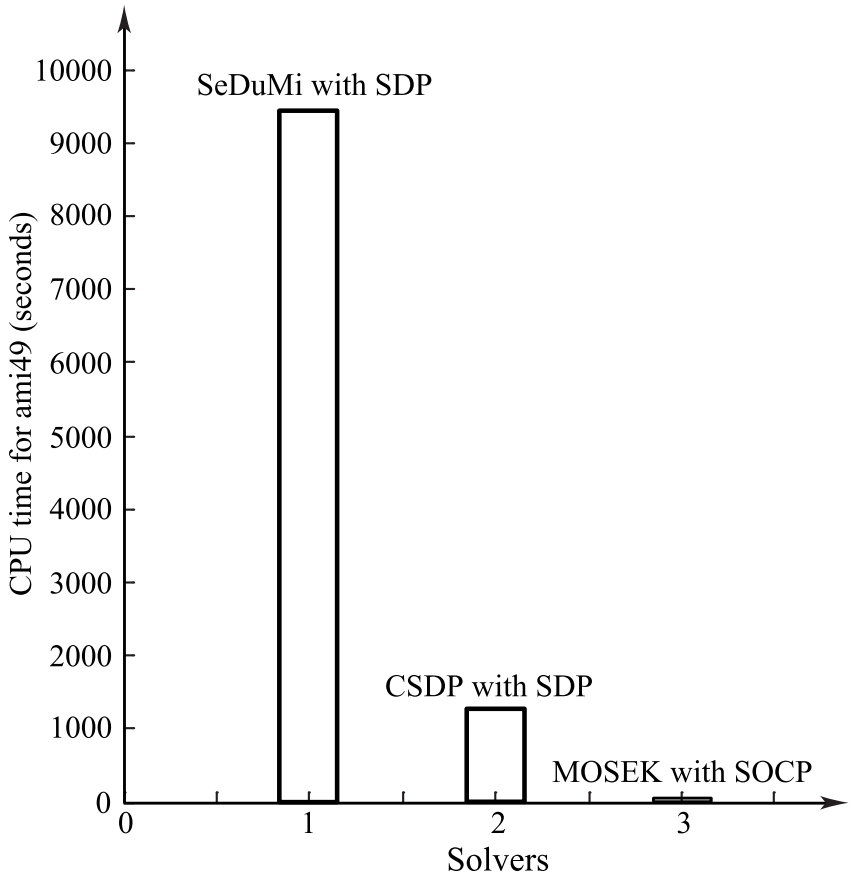


Figure 5.3: *Illustration of comparison of various solvers with different constraints for ami*49 *circuit.*

Table 5.4: GSRC experimental results with our SOCP model

circuit	Total area (mm^2)	Our Method		
		Our area (mm^2)	Runtime (s)	HPWL (mm)
n10a	22.17	22.17	1.21	37090
n30a	20.86	20.86	5.96	117236
n50a	19.86	19.86	10.08	123860
n100a	17.95	17.95	55.89	197490
n200a	17.57	17.57	1026.43	356520
n300a	27.32	27.32	1654.70	477800

5.6.2 Comparison with *MK* Model

First, in Table 5.5 we compare our approach with the *MK* model (Murata and Kuh, 1998). The results show that our total wirelength is competitive and in fact we obtain an improvement in wirelength over *MK* for the two largest benchmark problems, *ami*33 and *ami*49 (but 4.24% worse on average).

Table 5.5: Improvements in total wirelength compared with *MK*

MCNC circuit	Our Method vs *MK* Wirelength
apte	-15.29%
xerox	-2.43%
hp	-19.51%
ami33	+6.20%
ami49	+9.82%
Average	-4.24%

5.6.3 Comparison with *AM* Model

In Table 5.6, we compare our approach with *AM* model (Adya and Markov, 2003) shown in Table 3.4. The results show that our average HPWL is consistently better, and we obtain a better floorplan for several benchmarks. On average, our HPWL wirelength is 24.49% better than theirs. The significant impact of this model is on the quality of the total wirelength on the floorplan.

Table 5.6: Improvements in total wirelength compared with *AM*

MCNC circuit	Our Methodology vs *AM* Wirelength (min/avg)
apte	+14.44/+29.11%
xerox	-10.19/+12.18%
hp	+19.77/+33.64%
ami33	+19.87/+33.58%
ami49	-3.86/+13.92%
Average	+8.01/+24.49%

5.6.4 Comparison with *TimberWolf*

Our floorplaner does not currently take routing space requirements into account. In Table 5.8 we compare our approach with a commercial tool, *TimberWolf 1.3.3*, which is based on SA. Results for MCNC using *TimberWolf 1.3.3* (Sechen and Sangiovanni-Vincentelli, 1986) were reported by Cong *et al.* (1999). Cong *et al.* (1999) ignored the routing space, as did our method.

Table 5.7 gives the *TimberWolf* results including areas, computation times, and total wirelengths. Our method in the total wirelength is competitive, obtaining an average improvement in HPWL of 21.8%.

5.6.5 Obtaining Zero-Deadspace Floorplans

The results show that our method is constantly competitive with, and frequently outperforms, the *MK*, *AM*, and *TimberWolf* models. Moreover, we obtained zero-

Table 5.7: Results reported by *TimberWolf*

MCNC circuit	Total area (mm^2)	*TimberWolf*		
		Area (mm^2)	Runtime (s)	HPWL (mm)
apte	46.56	48.50	66.00	487.71
xerox	19.35	22.64	101.20	526.92
hp	8.30	9.58	91.40	186.76
ami33	1.16	1.27	221.00	71.80
ami49	35.4	40.81	472.80	814.20

Table 5.8: Improvements in total wirelength compared with *TimberWolf*

MCNC circuit	Our Model vs *TimberWolf* Wirelength
apte	+18.60%
xerox	+22.00%
hp	+23.97%
ami33	+30.25%
ami49	+14.15%
Average	+21.79%

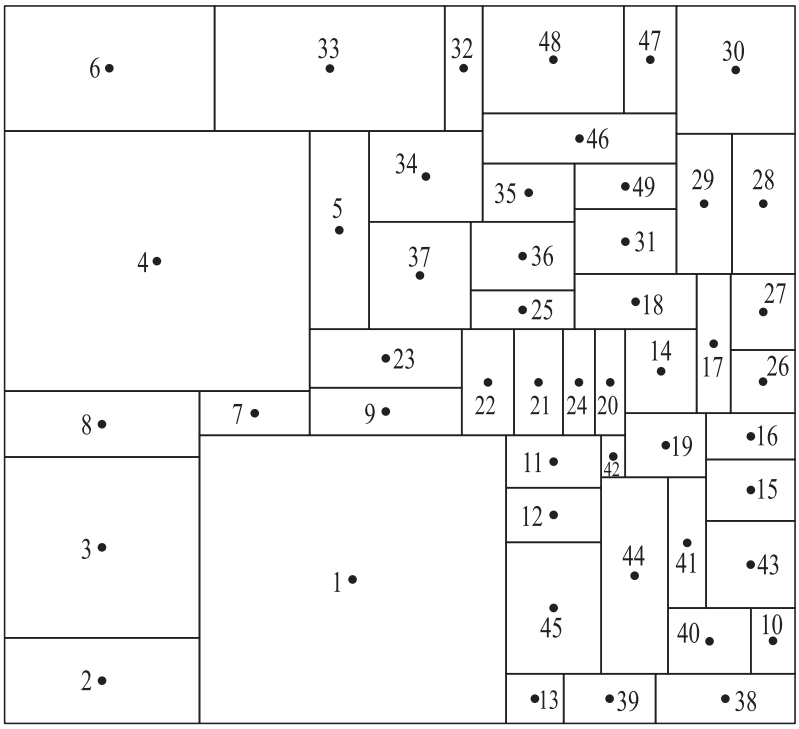


Figure 5.4: *Final layout for ami*49 *circuit.*

deadspace floorplans for all five circuits as shown in Table 5.9. Zero-deadspace floorplans for the circuits *apte*, *ami*33 and *ami*49 obtained using our method are shown in Fig. 4.7D, Fig. 4.9B and Fig. 5.4, respectively. Floorplanners *MK*, *AM*, and *TimberWolf* all have some deadspace in their floorplans. In contrast, our approach can guarantee complete area utilization in a fixed-outline floorplan.

Table 5.9: Deadspace comparisons with *MK*, *AM*, and *TimberWolf*

MCNC circuit	Total area (mm^2)	*MK*		*AM*		*TimberWolf*		Our Methodology	
		Area (mm^2)	Deadspace	Area min/avg (mm^2)	Deadspace min/avg	Area (mm^2)	Deadspace	Area (mm^2)	Deadspace
apte	46.56	46.55	-0.02%	46.97/48.95	0.87/4.88%	48.50	4.00%	46.56	**0%**
xerox	19.35	19.50	0.77%	19.51/20.62	0.82/6.16%	22.64	14.53%	19.35	**0%**
hp	8.30	8.83	6.0%	8.96/9.72	7.40/14.60%	9.58	13.36%	8.30	**0%**
ami33	1.16	1.16	0%	1.18/1.24	1.70/6.45%	1.27	8.66%	1.16	**0%**
ami49	35.4	35.58	0.5%	36.07/37.8	1.86/6.35%	40.81	13.26%	35.4	**0%**

5.6.6 Comparison with Parquet, Capo, IMF, and IMFAFF Models

For large GSRC circuits, we compare our model with Parquet (Adya and Markov, 2003; PARQUET, 2006), Capo (Adya *et al.*, 2004), IMF and IMFAFF (Chen *et al.*, 2005), where I/O pads were shifted to the boundary of the chips. In our experiment, I/O pads also were fixed on the boundary of the chips for comparison purposes. We use the three largest GSRC circuits n100, n200, and n300. All reported wirelengths were measured using the HPWL as the Parquet, Capo, IMF, and IMFAFF did. Each circuit was run once and the floorplans obtained are reported in Table 5.10. The whitespace in the Parquet, Capo, IMF, and IMFAFF is constrained to be less than 15% (Chen *et al.*, 2005). We chose our 10% whitespace case for the comparison in Table 5.11. Our model gives an improvement of 0.33% to 23.88% in terms of the total wirelength. Parquet and Capo can handle both hard and soft modules. Our method is currently focused on soft modules but in principle is applicable to hard modules. It is not clear whether IMF and IMFAFF used hard or soft modules.

5.6.7 Comparison with Parquet, *CC*, and *ZFS* Models

We now compared our model with the Chen and Chang (CC) model (Chen and Chang, 2006) and Parquet (Adya and Markov, 2003; PARQUET, 2006), with the I/O pads fixed at the locations originally given by the benchmarks.

Our results for 0% and 5% whitespace are reported in Table 5.12, and the comparisons with CC and Parquet for 10% and 15% whitespace are given in Tables 5.13 and 5.14. (The data of zero-deadspace and 5% whitespace cases for CC and Parquet are not available in their papers). The CC and Parquet results were obtained by defining the maximum percentage of whitespace to be 10% and 15%. However, the

specific percentage was not reported. Therefore, we compare our experimental results under both percentages of whitespace, 10% and 15%. Our model produces a greater improvement in terms of the total wirelength as the number of modules increases.

Our method is also compared with the ZFS (Zhan *et al.*, 2006) floorplan with 15% whitespace obtained by ZFS and Parquet 4 dealing with soft modules reported by Zhan *et al.* (2006). Table 5.15 gives the comparison; it shows that for soft modules, we again produce greater total wirelength improvement as the number of modules increases (Luo *et al.*, 2008*a*).

Table 5.10: Results for our model with I/O pads fixed at the boundary of the chips

Circuit	Zero-whitespace		5% whitespace		10% whitespace		15% whitespace	
	CPU(s)	HPWL	CPU(s)	HPWL	CPU(s)	HPWL	CPU(s)	HPWL
n100	55.89	197490	55.32	200490	55.96	203700	55.03	207180
n200	1026.43	356520	1129.14	362070	990.79	367880	881.39	373840
n300	1654.70	477800	1669.590	485180	1428.92	492830	1461.95	500910

5.7 Summary

This chapter introduces two convex optimization models as the second stage. Associated with the first stage model introduced in Chapter 3, a two-stage convex optimization methodology for floorplanning is formed that can be applied to both classical floorplanning and fixed-outline floorplanning. Area, aspect ratio constraints formulated to SDP and SOC problems are described in this chapter. The solvers used have varied according to the constraints included in the formulations in this

Table 5.11: Comparisons for Capo, Parquet(Pq), IMF, and IMFAFF; 10% whitespace and I/O pads fixed at the boundary of the chips

Circuit	Our	Parquet		Capo		IMF		IMFAFF	
	HPWL	HPWL	Imprv	HPWL	Imprv	HPWL	Imprv	HPWL	Imprv
n100	203700	242050	15.84%	224390	9.22%	207852	2.00%	208772	2.43%
n200	367880	432882	15.02%	385594	4.59%	369888	0.54%	372845	1.33%
n300	492830	647452	23.88%	522968	5.76%	489868	-0.60%	494480	0.33%

chapter. Computational results on the MCNC and GSRC benchmarks demonstrate that our methodology clearly outperforms the results reported in the literature. Our methodology guarantees complete area utilization in a fixed-outline situation, and it also produces floorplans with any specified percentage of whitespace. Our model provides **flexibility** to allow zero-whitespace or any specific percentage of whitespace (for buffer insertion, for example). Most importantly, our methodology provides greater improvement over other floorplanners as the number of modules increases.

Table 5.12: Results for our model; I/O pads fixed at the locations given by the benchmark

Circuit	Zero-whitespace	5% whitespace
n100	285070	287560
n200	506070	510700
n300	584640	591320

Table 5.13: Comparison for *Parquet* 4.5 and *CC*; 10% whitespace and I/O pads fixed at the locations given by the benchmark

Circuit	Our	Parquet		CC	
	HPWL	HPWL	Imprv	HPWL	Imprv
n100	290010	335600	13.58%	320600	9.55%
n200	515320	635500	18.91%	583300	11.65%
n300	597900	760500	21.38%	710000	15.79%

Table 5.14: Comparison for *Parquet* 4.5 and *CC*; 15% whitespace and I/O pads fixed at the locations given by the benchmark

Circuit	Our	Parquet		CC	
	HPWL	HPWL	Imprv	HPWL	Imprv
n100	292430	335600	12.86%	320600	8.79%
n200	519890	635500	18.19%	583300	10.87%
n300	604420	760500	20.52%	710000	14.87%

Table 5.15: Comparison for *ZFS* and *Parquet* 4 with 15% whitespace; soft modules and I/O pads fixed at the locations given by the benchmark

Circuit	Our	ZFS		Parquet4	
	HPWL	HPWL	Imprv	HPWL	Imprv
n100	292430	291628	-0.28%	342103	14.52%
n200	519890	572145	9.13%	630014	17.48%
n300	604420	702822	14.00%	770354	21.54%

Chapter 6

Interchange-Free Local Improvement

The second stage model may fail to obtain a feasible solution. In this chapter, we propose an automatic interchange-free strategy to perform local improvements to the floorplan. The underlying idea is to locally relax the relationships of relative positions for those modules whose aspect ratios exceed the required upper bounds.

Handling the aspect ratios of soft modules is important and difficult. Previous approaches to optimize the aspect ratios use convex optimization (e.g., Murata and Kuh, 1998; Takouda *et al.*, 2005), stochastic optimization (Kang and Dai, 1997), Lagrangian relaxation (Chu and Young, 2004), bi-partitioning (Cong *et al.*, 2006), and a graph-based methodology (Itoga *et al.*, 2005).

Kang and Dai (1997) considered a stochastic optimization method combining simulated annealing (SA) and Genetic Algorithm (GA) approaches for optimal floorplans while achieving the desired aspect ratios. Ma *et al.* (2001) suggested a simple method based on an SA framework arbitrarily changes aspect ratios within a given discrete range.

Several convex optimization techniques are also proposed to obtain desired aspect ratios of soft modules. Takouda *et al.* (2005) formulated an SDP problem that obtains required aspect ratios of modules. Murata and Kuh (1998) use SA and SP representation while Chu and Young (2004) use Lagrangian relaxation (LR) methodology to formulate the constraints of aspect ratio.

Recently, a two-phase graph-based methodology was proposed to meet the requirement of aspect ratio constraints in floorplanning (Itoga *et al.*, 2005). In the first phase, they use a graph-based minimal cut problem to adjust aspect ratios by choosing a minimal set of soft modules. The second phase determines the aspect ratio of each soft module using the Newton-Raphson method. A bi-partitioning based approach where the wirelength is minimized by SA is used by Cong *et al.* (2006). The floorplan is bi-partitioned into subregions whose aspect ratio bounds determine the aspect ratio bounds of the soft modules.

Most approaches are based on interchange operations that require more computational efforts. We develop an interchange-free local improvement algorithm that improves the aspect ratio constraints. This approach can satisfy the required aspect ratio constraints while maintaining the topology quality and the optimal wirelength.

6.1 Algorithm for Interchange-Free Local Improvement

If formulation (5.49) is infeasible, we relax the aspect ratio constraints. We then find the modules with violated aspect ratios and locally relax the non-overlap constraints for those modules which we call central modules. More precisely, we select two-layer modules surrounding the central module and relax the relationships of relative positions between the central and surrounding modules by entering zeros in the RPM.

We re-solve the SOCP problem iteratively until the upper bound on the aspect ratios is satisfied (see Algorithm 1).

Definition 6.1.1 (*Central Modules* M_i) *The Central Modules $M_i(i = 1, 2, ..., p)$ are those modules whose aspect ratios exceed the required upper bounds.*

Definition 6.1.2 (*First Layer Modules* M_{ij}) *The First Layer Modules $M_{ij}(i = 1, 2, ..., p;\ j = 1, 2, ..., q)$ are those modules adjacent to Central Module M_i.*

Definition 6.1.3 (*Second Layer Modules* M_{ijk}) *Similarly, the Second Layer Modules $M_{ijk}(i = 1, 2, ..., p; j = 1, 2, ..., q; k = 1, 2, ..., r)$ are those modules adjacent to first layer module M_{ij}.*

Definition 6.1.4 (*Relaxed RPM*) *The relaxed RPM is the RPM whose entries corresponding to modules M_i, M_{ij}, and M_{ijk}, are set to zero (it relaxes the relationship of the relative positions between M_i and M_{ij}/M_{ijk}).*

As an example we solved (5.49) for ami49 circuit and obtain the layout shown in Fig. 6.1. All the aspect ratios are satisfied except that the aspect ratio for Module 21 is 19.0408.

The first layer modules adjacent to central module M_{21} are M_1, M_{13}, M_{22}, M_{24}, M_{39}, and M_{45} (they are represented as $M_{21,1}$, $M_{21,13}$, $M_{21,22}$, $M_{21,24}$, $M_{21,39}$, and $M_{21,45}$, respectively). We relax the relationships of relative position between the central module M_{21} and surrounding modules and then we obtain the layout by solving the SOCP problem (see Fig. 6.2). Based on updated SRPM, the final layout, shown in Fig. 6.3, is found by re-solving the SOCP problem.

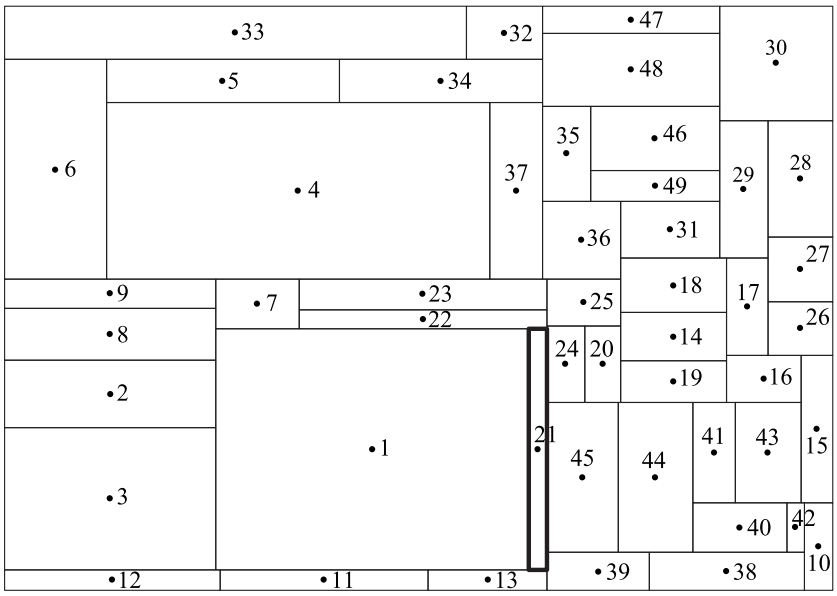


Figure 6.1: *First round layout for ami*49 *circuit.*

6.2 Summary

In this chapter, an interchange-free algorithm was developed to perform local improvements to satisfy the requirement of aspect ratios of soft modules. The reasonable aspect ratio constraints are met by this interchange-free algorithm while maintaining the topology quality and optimal total wirelength.

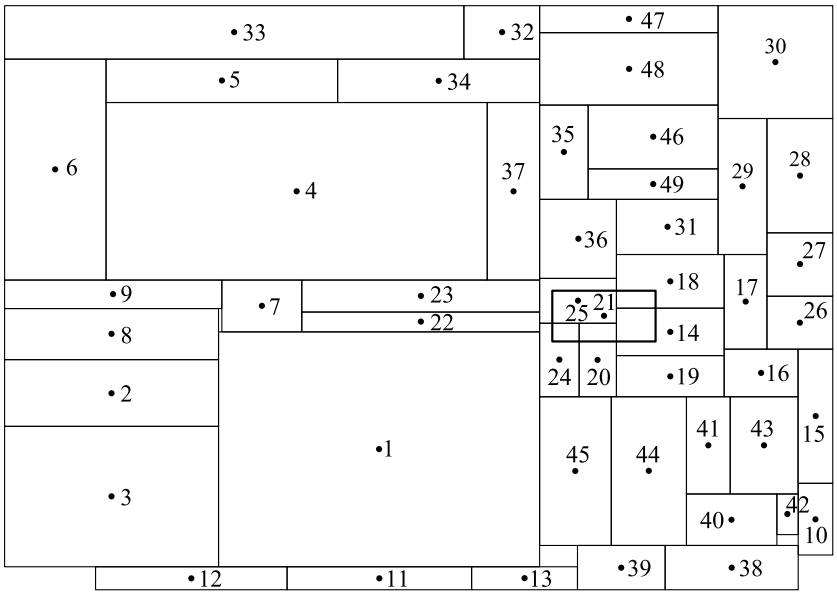


Figure 6.2: *Second round layout for ami*49 *circuit, where the relative positions of the first and second layer modules are locally relaxed.*

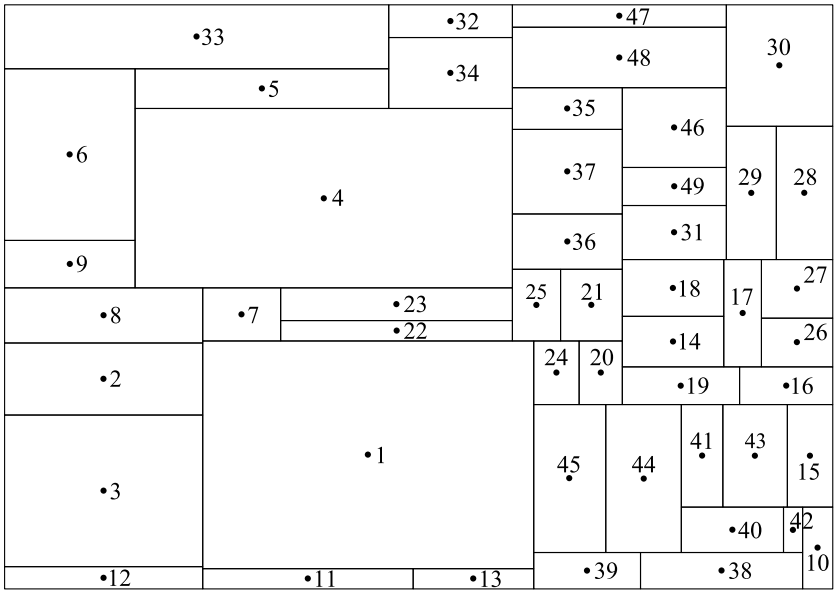


Figure 6.3: *Final layout for ami*49 *circuit.*

Algorithm 1 Algorithm for Interchange-free Local Improvement

Input: $SRPM$

Output: Aspect ratios, module dimensions

1. Solve SOCP model without aspect ratio constraints;

2. If all the aspect ratios are satisfactory, goto Step 9;

otherwise, goto Step 3;

3. Select all the Central Modules M_i;

4. Select and compute all the First Layer Modules M_{ij};

5. Select and compute all the Second Layer Modules M_{ijk};

6. Set up the relaxed SRPM;

7. Solve the SOCP model with relaxed SRPM without aspect ratio constraints to obtain a layout with overlaps; based on this result to update the SRPM;

8. Re-solve SOCP model with aspect ratio constraints;

9. End.

Chapter 7

Conclusion and Future Work

In this chapter, the results obtained in this thesis research will be summarized in the conclusion section. It also suggests suitable improvements and extension for further work.

7.1 Conclusion

In this thesis, a two-stage optimization methodology was proposed to solve the fixed-outline floorplanning problem that is a global optimization problem for wirelength minimization. Some of the important points are listed below.

- This thesis presented a two-stage fixed-outline floorplanning scheme. In the first stage, we transformed the original fixed-die floorplanning problem into a convex model based on a facility layout problem (in Chapter 3). This global optimization problem provided the relative positions of the soft modules on the floorplan while the total wirelength was minimized. In the second stage, a fixed-die floorplanning problem was formulated to nonlinear optimization (in Chapter 3), SDP, and SOCP problems (in Chapter 5), respectively.

- An RPM was used to encode the relative positions of all modules (in Chapter 4). The RPM can avoid the operations of absolute values in the non-overlap constraints.

- VD and DT techniques were used to spread out modules in the floorplan obtained from the first stage and converted it into a planar graph. Sparse relative position matrices (SRPM) further improved computational efficiency (in Chapter 4).

- An interchange-free algorithm for local improvement of the floorplan was presented that achieves desired aspect ratio constraints on soft modules (in Chapter 6).

- Experimental results on the MCNC and GSRC benchmarks demonstrated that our method substantially improved wirelength compared with several floorplnners. The significant impact of this thesis research is on the quality of the total wirelength. Minimizing the total wirelength is the principal objective of most existing floorplanners. Minimization of the total wirelength is helpful to minimize chip size, and thus cost, but also minimizes power and delay, which are proportional to the wirelength and wirelength squared, respectively. Therefore, the considerably improved quality in the total wirelength of our floorplanner is one of our most important contributions. This is the first time that a convex-optimization-based method is used for fixed-outline floorplanning. The proposed two-stage method is particularly suitable for fixed-outline floorplanning and soft module case, and can also be extended to classical floorplanning as well as hard module case.

7.2 Future Work

Although promising results were achieved for fixed-outline floorplaning using the proposed approach, further work is necessary.

- Area Minimization: With advancements in nanotechnology devices, area minimization technique in terms of circuit size is of paramount importance. We have successfully solved wirelength minimization problem of fixed-outline floorplanning by convex optimization approaches. We will develop novel optimization algorithms based on SDP and SOCP techniques to tackle area minimization problem.

- Timing Driven Floorplanning: Floorplanning is important for a VLSI chip performance in terms of not only floorplan size but also timing constraint. Area and wirelength minimization used to be the most important objective in floorplanning design. However, as technology of the deep-submicron has been developed rapidly, interconnect issues such as delay, routability, congestion have become a major concern in floorplanning. It is essential for the floorplan to meet time constraints in a very deep sub-micron design. Previous research into timing-driven floorplanning can be classified into path-based and net-based models.

  The path-based method models the timing problem more accurately (Mo *et al.*, 2001; Donath *et al.*, 1990; Donath *et al.*, 1999). However, it is difficult to incorporate it into floorplanning dominated by traditional techniques. Mo *et al.* (2001) use the star model in a force-directed placement algorithm to deal with path delay constraints. Jackson and Kuh (1989) suggested a linear programming model for timing driven placement. RITUAL, developed by Srinivasan *et al.* (1991), was more efficient, modelling the net length using a quadratic objective function. The number of timing constraints over the nets is decreased by

transforming the quadratic programming problem into a Lagrangian problem. The net-based method assigns different weights to different nets according to the criticality of the paths on which they lie (Brasen and Bushnell, 1990; Riess and Ettelt, 1995; Tellez *et al.*, 1996; Sarrafzadeh and Wang, 1997). The net-based approach meets timing requirements by converting timing constraints into upper bounds over the net delays. Consequently, the main tasks of the net-based model are to choose the weights and bounds.

The proposed preliminary timing-driven floorplanning method consists of two stages. In the first stage, timing-driven floorplanning is performed while minimizing wirelength using convex programming. The second stage deals with the floorplan sizing problem stated previously. However, the sizes, shapes, and aspect ratios of the modules are determined based on the topological arrangement achieved in the first stage by considering timing delay over nets.

- Multilevel Hierarchical Algorithm: As VLSI chip complexity in terms of the number of components increases rapidly, some useful algorithms such as mixed-size (Adya and Markov, 2002; Adya and Markov, 2003; Adya *et al.*, 2004; Cong *et al.*, 2006; Sechen, 1988*a*; Khatkhate *et al.*, 2004; Bourbakis, 2008; Kahng and Wang, 2004*a*) and multilevel (Chan *et al.*, 2003; Kahng and Wang, 2004*b*; Chan *et al.*, 2005; Cong *et al.*, 2006) in placement are developed to process floorplaning problem in a rough level to generate an initial coarse placement. Therefore, it is necessary to pack a large number of standard cells mixed with many macros such as RAMs, ROMs, and IP blocks. The macros are regarded as hard modules while clusters of standard cells are packed as soft modules. In modern VLSI circuit design, a floorplanner is typically required to pack more than 300 modules. In our model, computational time increase rapidly as the

number of modules increases. Therefore, efficient partitioning, top-down and multilevel hierarchical algorithms will be studied.

- Bus-driven Floorplanning: The bus driven floorplan is described as follows (Xiang *et al.*, 2003; Chen and Chang, 2006): Given a collection of p rectangular modules $M = \{m_i | i = 1, 2, ..., p\}$ and q buses $B = \{b_i | i = 1, ..., q\}$, each bus b_i with a width k_i passes through a set of modules M_i, where $M_i \subseteq M$ and $|M_i| = n_i$. Bus driven floorplanning seeks the positions of modules and buses so as to minimize the chip area and the bus area, while each bus b_i passes through all of its n_i modules without any overlap between two modules and two vertical (horizontal) buses. As the bus constraints are linear, it is possible to enforce them in our two-stage convex optimization model to form a bus-driven floorplanning framework.

- Partitioning-based Fixed-outline Floorplanning: Convex optimization techniques such as SDP and SOCP effectively solve VLSI floorplanning problem by interior-point algorithm. However, for larger floorplanning problem, in particular, placement problem, the computation efficiency by SDP and SOCP needs to be improved. Partitioning-based approach is a good candidate. Circuit partitioning decomposes a very large circuit into several relatively independent sub-circuits such that their sizes are small enough to be handled by the existing physical design process. The floorplanning and placement problems of every sub-circuit can be formulated by convex optimization models with less constraints thus less computational effort in total.

- Improvement of Computation Time of Floorplanning: Computation time to solve the proposed model determines the time of design-to-market. Our existing method achieves promising computation time and high-quality floorplans.

The computation time can still be improved. First, the advanced and effective algorithm for solvers will be helpful for the acceleration of solving the model. Second, effective partitioning-based approach and clustering techniques will contribute great improvement in computation time.

Bibliography

Ackland, B. D. (1988). Knowledge-based physical design automation. In: *Physical Design Automation of VLSI Systems* (B. T. Preas and M. J. Lorenzetti, Eds.). pp. 409–454. The Benjamin/Cummings Publishing, Inc.. Menlo Park, California, USA.

Adya, S. N. and I. L. Markov (2002). Consistent placement of macro-blocks using floorplanning and standard-cell placement. In: *Proc. of ACM Intl. Symp. on Physical Design*. pp. 12–17.

Adya, S. N. and I. L. Markov (2003). Fixed-outline floorplanning: Enabling hierarchical design. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems* **11**(6), 1120–1135.

Adya, S. N., I. L. Markov and P. G. Villarrubiac (2006). On whitespace and stability in physical synthesis. *Integration, the VLSI Journal* **39**, 340–362.

Adya, S. N., S. Chaturvedi, J. A. Roy, D. A. Papa and I. L. Markov (2004). Unification of partitioning, placement and floorplanning. In: *Proc. of IEEE/ACM Intl. Conf. on Computer-Aided Design*. pp. 550–557.

Alpert, C. J. and A. B. Kahng (1995). Recent developments in netlist partitioning: A survey. *Integration: the VLSI Journal* **19**(1-2), 1–81.

Alupoaei, S. and S. Katkoori (2002). Net-based force-directed macrocell placement for wirelength optimization. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems* **10**(6), 824–835.

Andersen, E. D., C. Roos and T. Terlaky (2003). On implementing a primal-dual interior-point method for conic quadratic optimization. *Mathematical Programming* **95**(2), 249–277.

Anjos, M. F. and A. Vannelli (2002). An attractor-repeller approach to floorplanning. *Mathematical Methods of Operations Research* **56**, 3–27.

Anjos, M. F. and A. Vannelli (2006). A new mathematical-programming framework for facility-layout design. *INFORMS Journal on Computing* **18**(1), 111–118.

Baker, B. S., E. G. Coffman Jr. and R. L. Rivest (1980). Orthogonal packing in two dimensions. *SIAM Journal on Computing* **9**, 846–855.

Ball, C. F., P. V. Kraus and D. A. Mlynski (1994). Fuzzy partitioning applied to VLSI-floorplanning and placement. In: *Proc. of IEEE Intl. Symp. on Circuits and Systems*. pp. 177–180.

Behjat, L. (2002). *New Modelling and Optimization Techniques for the Global Routing Problem*. PhD thesis. University of Waterloo, Waterloo, ON, Canada.

Borchers, B. (1999*a*). CSDP 2.3 User's Guide. *Optimization Methods and Software* **11**(1), 597–611.

Borchers, B. (1999*b*). CSDP, a C library for semidefinite programming. *Optimization Methods and Software* **11**(1), 613–623.

Borchers, B. and J. G. Young (2007). Implementation of a primal-dual method for SDP on a shared memory parallel architecture. *Computational Optimization and Applications* **37**(3), 355–369.

Bourbakis, N. G. (2008). A generic, formal language-based methodology for hierarchical floorplanning-placement. *Computer Languages, Systems and Structures* **34**(1), 25–42.

Brasen, D. R. and M. L. Bushnell (1990). MHERTZ: A new optimization algorithm for floorplanning and global routing. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 107–110.

Castillo, I. and T. Sim (2004). A spring-embedding approach for the facility layout problem. *Journal of the Operational Research Society* **55**, 73–81.

Chan, T. F., J. Cong and K. Sze (2005). Multilevel generalized force-directed method for circuit placement. In: *Proc. of ACM Intl. Symp. on Physical Design.* pp. 185–192.

Chan, T. F., J. Cong, T. Kong and J. R. Shinnerl (2003). Multilevel circuit placement. In: *Multilevel Optimization in VLSICAD* (J. Cong and J. R. Shinnerl, Eds.). Kluwer. Boston, MA, USA.

Chang, Y.-C., Y.-W. Chang, G.-M. Wu and S.-W. Wu (2000). B*-trees: A new representation for nonslicing floorplans. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 458–463.

Chen, D.-S., C.-T. Lin, Y.-W. Wang and C.-H. Cheng (2007). Fixed-outline floorplanning using robust evolutionary search. *Engineering Applications of Artificial Intelligence* **20**, 821–830.

Chen, P. and E. S. Kuh (2000). Floorplan sizing by linear programming approximation. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 468–471.

Chen, S. and T. Yoshimura (2007). A stable fixed-outline floorplanning method. In: *Proc. of ACM Intl. Symp. on Physical Design.* pp. 119–126.

Chen, T. and M. K. H. Fan (1998). On convex formulation of the floorplan area minimization problem. In: *Proc. of ACM Intl. Symp. on Physical Design.* pp. 124–128.

Chen, T.-C. and Y.-W. Chang (2006). Modern floorplanning based on B*-tree and fast simulated annealing. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* **25**(4), 637–650.

Chen, T.-C., Y.-W. Chang and S.-C. Lin (2005). IMF: interconnect-driven multilevel floorplanning for large-scale building-module designs. In: *Proc. of IEEE/ACM Intl. Conf. on Computer-Aided Design.* pp. 159–164.

Chen, T.-C., Y.-W. Chang and S.-C. Lin (2008). A new multilevel framework for large-scale interconnect-driven floorplanning. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* **27**(2), 286–294.

Cheng, C. K. and E. S. Kuh (1984). Module placement based on resistive network optimization. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* **3**(3), 218–225.

Choi, S.-G. and C.-M. Kyung (1991). A floorplanning algorithm using rectangular Voronoi diagram and force-directed block shaping. In: *Proc. of IEEE/ACM Intl. Conf. on Computer-Aided Design.* pp. 56–59.

Chu, C. C. N. and E. F. Y. Young (2004). Non-rectangular shaping and sizing of soft modules for floorplan-design improvement. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* **23**(1), 71–79.

Cohoon, J. P., S. U. Hegde, W. N. Martin and D. S. Richards (1991). Distributed genetic algorithms for the floorplan design problem. *IEEE Trans. on Computer-Aided Design* **10**(4), 483–492.

Cong, J., M. Romesis and J. R. Shinnerl (2006). Fast floorplanning by look-ahead enabled recursive bipartitioning. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* **25**(9), 1719–1732.

Cong, J., T. Kong, D. Xu, F. Liang, J. S. Liu and W. H. Wong (1999). Relaxed simulated tempering for VLSI floorplan designs. In: *Proc. of Asia and South Pacific Design Automation Conf.* pp. 13–16.

Czyzyk, J., M. Mesnier and J. Moré (1998). The NEOS server. *IEEE J. on Computational Science and Engineering* **5**, 68–75.

Dickinson, A. (1986). Floyd: a knowledge-based floor plan designer. In: *Proc. of IEEE Intl. Conf. on Computer Design.* pp. 176–179.

Donath, W. E., R. J. Norman, B. K. Agrawal, S. E. Bello, S. Y. Han, J. M. Kurtzberg, P. Lowy and R. I. McMillan (1990). Timing driven placement using complete path delays. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 84–89.

Donath, W., P. Kudva and L. Reddy (1999). Performance driven optimization of network length in physical placement. In: *Proc. of IEEE Intl. Conf. on Computer Design.* pp. 258–265.

Drezner, Z. (1980). Discon: A new method for the layout problem. *Operations Research* **28**(6), 1375–1384.

Du, Y. and A. Vannelli (1998). A nonlinear programming and local improvement method for standard cell placement. In: *Proc. of IEEE Custom Integrated Circuit Conf.* pp. 1–4.

Dunlop, A. E. and B. W. Kernighan (1985). A procedure for placement of standard cell VLSI circuits. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* **4**(1), 92–98.

Eisenmann, H. and F. Johannes (1998). Generic global placement and floorplanning. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 269–274.

Etawil, H. (1999). *Convex Optimization and Utility Theory: New Trends in VLSI Circuit Layout*. PhD thesis. University of Waterloo, Waterloo, ON, Canada.

Etawil, H., S. Areibi and A. Vannelli (1999). Attractor-repeller approach for global placement. In: *Proc. of IEEE/ACM Intl. Conf. on Computer-Aided Design.* pp. 20–24.

Feng, Y. and D. Mehta (2004). Constrained floorplanning with whitespace. In: *Proc. of the 17th Intl. Conf. on VLSI Design.* pp. 969–974.

Feng, Y. and D. P. Mehta (2006). Module relocation to obtain feasible constrained floorplans. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* **25**(5), 856–866.

Feng, Y., D. P. Mehta and H. Yang (2004). Constrained floorplanning using network flows. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* **23**(4), 572–580.

Fernando, P. and S. Katkoori (2008). An elitist non-dominated sorting based genetic algorithm for simultaneous area and wirelength minimization in VLSI floorplanning. In: *Proc. of IEEE Intl. Conf. on VLSI Design*. pp. 337–342.

Ferris, M., M. Mesnier and J. Moré (2000). NEOS and Condor: Solving optimization problems over the Internet. *ACM Trans. on Math. Softw.* **26**(1), 1–18.

Fourer, R., D. M. Gay and B. W. Kernighan (2003). *AMPL: A modeling language for mathematical programming*. Pacific Grove, CA: Thomson/Brooks/Cole.

Gamal, A. A. E. (1981). Two-dimensional stochastic model for interconnections in master slice integrated circuits. *IEEE Trans. on Circuits and Systems* **28**, 127–138.

GSRC, Floorplan Benchmarks (2006). http://www.cse.ucsc.edu/research/surf/GSRC/gsrcbench.html.

Guo, P.-N., C.-K. Cheng and T. Yoshimura (1999). An O-tree representation of non-slicing floorplan and its applications. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 268–273.

Hamada, T., C. K. Cheng and P. M. Chau (1996). A wire length estimation technique utilizing neighborhood density equations. *IEEE Trans. on Computer-Aided Design* **15**, 912–922.

He, X. (1997). On floorplans of planar graphs. In: *Proc. of ACM Symp. on Theory of Computing*. pp. 426–435.

Hebgen, W. and G. Zimmermann (1996). Hierarchical netlength estimation for timing prediction. In: *Proc. of Physical Design Workshop*. pp. 118–125.

Helmberg, C., F. Rendl, R. J. Vanderbei and H. Wolkowicz (1996). An interior–point method for semidefinite programming. *SIAM Journal on Optimization* **6**(2), 342–361.

Herrigel, A. and W. Fichtner (1989). An analytic optimization technique for placement of macro-cells. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 376–381.

Ho, S.-Y., S.-J. Ho, Y.-K. Lin and W. C.-C. Chu (2004). An orthogonal simulated annealing algorithm for large floorplanning problems. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems* **12**(8), 874–877.

Hong, X., G. Huang, Y. Cai, J. Gu, S. Dong, C.-K. Cheng and J. Gu (2000). Corner block list: An effective and efficient topological representation of nonslicing floorplan. In: *Proc. of IEEE/ACM Intl. Conf. on Computer-Aided Design.* pp. 8–12.

Hu, B. and M. Marek-Sadowska (2002). FAR: Fixed-points addition and relaxation based placement. In: *Proc. of ACM Intl. Symp. on Physical Design.* pp. 161–166.

Hu, T. C. and E. S. Kuh (1985). *VLSI circuit layout: Theory and design.* IEEE Press. New York, USA.

Huang, D. J. H. and A. B. Kahng (1997). Partitioning-based standard-cell global placement with an exact objective. In: *Proc. of ACM Intl. Symp. on Physical Design.* pp. 18–25.

Hur, S. W. and J. Lillis (1999). Relaxation and clustering in a local search framework: Application to linear placement. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 360–366.

Hur, S.-W., T. Cao, K. Rajagopal, Y. Parasuram, A. Chowdhary, V. Tiourin and B. Halpin (2003). Force directed mongrel with physical net constraints. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 214–219.

Itoga, H., C. Kodama and K. Fujiyoshi (2005). A graph based soft module handling in floorplan. *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences* **12**(E88-A), 3390–3397.

Jackson, M. A. B. and E. S. Kuh (1989). Performance-driven placement of cell based IC's. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 370–375.

Jankovits, I., C. Luo, M. F. Anjos and A. Vannelli (2007). A convex optimization framework for the unequal-areas facility layout problem. *submitted to European Journal of Operational Research*.

Jin, L., D. Kim, L. Mu, D.-S. Kim and S.-M. Hu (2006). A sweepline algorithm for Euclidean Voronoi diagram of circles. *Computer-Aided Design* **38**, 260–272.

Johannes, F. M. (1996). Partitioning of VLSI circuits and systems. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 83–87.

Kahng, A. B. (2000). Classical floorplanning harmful. In: *Proc. of ACM Intl. Symp. on Physical Design*. pp. 207–213.

Kahng, A. B. and Q. Wang (2004*a*). An analytic placer for mixed-size placement and timing-driven placement. In: *Proc. of IEEE/ACM Intl. Conf. on Computer-Aided Design*. pp. 565–572.

Kahng, A. B. and Q. Wang (2004*b*). Implementation and extensibility of an analytical placer. In: *Proc. of ACM Intl. Symp. on Physical Design*. pp. 18–25.

Kang, M. and W. Dai (1997). General floorplanning with L-shaped, T-shaped and soft blocks based on bounded slicing grid structure. In: *Proc. of Asia and South Pacific Design Automation Conf.* pp. 265–270.

Kennings, A. (1994). *A Parallel Dual Affine Scaling Algorithm Using Netlist Partitioning*. Master's thesis. University of Waterloo, Waterloo, ON, Canada.

Kennings, A. (1997). *Cell Placement Using Constructive and Iterative Improvement Methods*. PhD thesis. University of Waterloo, Waterloo, ON, Canada.

Kennings, A. and I. Markov (2000). Analytical minimization of half-perimeter wirelength. In: *Proc. of Asia and South Pacific Design Automation Conf.* pp. 179–184.

Khatkhate, A., C. Li, A. R. Agnihotri, S. Ono, M. C. Yildiz, C.-K. Koh and P. H. Madden (2004). Recursive bisection based mixed block placement. In: *Proc. of ACM Intl. Symp. on Physical Design*. pp. 84–89.

Kim, J.-G. and Y.-D. Kim (2003). A linear programming-based algorithm for floorplanning in VLSI design. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* **2**(5), 584–592.

Kleinhans, J., G. Sigl, F. Johannes and K. Antreich (1991). Gordian: VLSI placement by quadratic programming and slicing optimization. *IEEE Trans. on Computer-Aided Design* **10**(3), 356–365.

Kling, R. M. and P. Banerjee (1991). Empirical and theoretical studies of the simulated evolution method applied to standard cell placement. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* **10**(10), 1303–1315.

Kozminski, K. and E. Kinnen (1984). An algorithm for finding a rectangular dual of a planar graph for use in area planning for VLSI integrated circuits. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 655–656.

Lai, Y. and S. M. Leinwand (1988). Algorithms for floorplan design via rectangular dualization. *IEEE Trans. on Computer-Aided Design* **7**(12), 1278–1289.

Law, J. H. Y. and E. F. Y. Young (2008). Multi-bend bus driven floorplanning. *Integration, the VLSI Journal* **41**(2), 306–316.

Lee, H.-C., Y.-W. Chang and H. H. Yang (2007). MB*-tree: a multilevel floorplanner for large-scale building-module design. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* **26**(8), 1430–1444.

Lee, H.-C., Y.-W. Chang, J.-M. Hsu and H. H. Yang (2003). Multilevel floorplanning/placement for large-scale modules using B*-trees. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 812–817.

Lin, C.-T., D.-S. Chen and Y.-W. Wang (2004). Robust fixed-outline floorplanning through evolutionary search. In: *Proc. of Asia and South Pacific Design Automation Conf.* pp. 42–44.

Lin, C.-T., D.-S. Chen and Y.-W. Wang (2006). Modern floorplanning with boundary and fixed-outline constraints via genetic clustering algorithm. *Journal of Circuits, Systems, and Computers* **15**(1), 107–128.

Lin, J.-M. and Y.-W. Chang (2001). TCG: A transitive closure graph-based representation for nonslicing floorplans. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 764–769.

Lin, J.-M., Y.-W. Chang and S.-P. Lin (2003). Corner sequence – a P-admissible floorplan representation with a worst case linear-time packing scheme. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems* **11**(4), 679–686.

Liu, R., S. Dong and X. Hong (2005). An efficient algorithm to fixed-outline floorplanning based on instance augmentation. In: *Proc. of the Ninth Intl Conf. on Computer Aided Design and Computer Graphics*. pp. 210–215.

Lobo, M. S., L. Vandenberghe, S. Boyd and H. Lebret (1998). Applications of second order cone programming. *Linear Algebra and its Applications* **284**, 193–228.

Lokanathan, B. and E. Kinne (1989). Performance optimized floor planning by graph planarization. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 116–121.

Lu, C.-H., H.-M. Chen and C.-N. Liu (2008). An effective decap insertion method considering power supply noise during floorplanning. *Journal of Information Science and Engineering* **24**(1), 115–127.

Luo, C., M. F. Anjos and A. Vannelli (2007). A nonlinear optimization methodology for VLSI fixed-outline floorplanning. *Special issue of Journal of Combinatorial Optimization, to appear.*

Luo, C., M. F. Anjos and A. Vannelli (2008*a*). Large-scale fixed-outline floorplanning design using convex optimization. *submitted to IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems.*

Luo, C., M. F. Anjos and A. Vannelli (2008*b*). Large-scale fixed-outline floorplanning design using convex optimization techniques. In: *Proc. of Asia and South Pacific Design Automation Conf.* pp. 198–203.

Luo, T., H. Ren, C. J. Alpert and D. Z. Pan (2005). Computational geometry based placement migration. In: *Proc. of IEEE/ACM Intl. Conf. on Computer-Aided Design*. pp. 41–47.

Ma, Y., X. Hong, S. Dong, Y. Cai, C.-K. Cheng and J. Gu (2001). Floorplanning with abutment constraints based on corner block list. *Integration, the VLSI Journal* **31**(1), 65–77.

Maruvada, S. C., K. Krishnamoorthy and F. Balasa (2002). Block placement using the segment tree data structure from computational geometry. In: *Proc. of IEEE Midwest Symp. on Circuits and Systems*. pp. 111–114.

MCNC, Floorplan Benchmarks (2004). http://www.cse.ucsc.edu/research/surf/gsrc /MCNCbench.html.

Mehta, D. and N. Sherwani (2000). On the use of flexible, rectilinear blocks to obtain minimum-area floorplans in mixed block and cell designs. *ACM Trans. on Design Automation of Electronic Systems* **5**, 82–97.

Mo, F., A. Tabbara and R. K. Brayton (2001). A timing-driven macro-cell placement algorithm. In: *Proc. of IEEE Intl. Conf. on Computer Design*. pp. 322–327.

Mogaki, M., C. Miura and H. Terai (1987). Algorithm for block placement with size optimization technique by the linear programming approach. In: *Proc. of IEEE/ACM Intl. Conf. on Computer-Aided Design*. pp. 80–83.

Moh, T.-S., T.-S. Chang and S. L. Hakimi (1996). Globally optimal floorplanning for a layout problem. *IEEE Trans. on Circuits and Systems* **43**(9), 713–720.

Mohamood, F., M. B. Healy, S. K. Lim and H.-H. S. Lee (2007). Noise-direct: a technique for power supply noise aware floorplanning using microarchitecture

profiling. In: *Proc. of Asia and South Pacific Design Automation Conf.* pp. 786–791.

MOSEK (2006). http://www.mosek.com/documentation.html.

Murata, H. and E. S. Kuh (1998). Sequence-pair based placement method for hard/soft/pre-placed modules. In: *Proc. of ACM Intl. Symp. on Physical Design.* pp. 167–172.

Murata, H., K. Fujiyoshi, S. Nakatake and Y. Kajitani (1995). Rectangle-packing based module placement. In: *Proc. of IEEE/ACM Intl. Conf. on Computer-Aided Design.* pp. 472–479.

Murata, H., K. Fujiyoshi, S. Nakatake and Y. Kajitani (1998). VLSI/PCB placement with obstacles based on sequence pair. *IEEE Trans. on Computer-Aided Design* **17**(1), 61–68.

Murofushi, M., M. Yamada and T. Mitsuhashi (1990). FOLM-planner: A new floorplanner with a frame overlapping floorplan model suitable for sog (sea-of-gates) type gate arrays. In: *Proc. of IEEE/ACM Intl. Conf. on Computer-Aided Design.* pp. 140–143.

Murtagh, B. A. and M. A. Saunders (1983). MINOS 5.0 User's Guide. Technical Report SOL 83-20. Department of Operations Research, Stanford University. Revised as MINOS 5.1 User's Guide, Report SOL 83-20R, 1987.

Murtagh, B.A. and M.A. Saunders (1982). A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints. *Math. Programming Stud.* (16), 84–117.

Nakatake, S., K. Fujiyoshi, H. Murata and Y. Kajitani (1996). Module placement on BSG-structure and IC layout applications. In: *Proc. of IEEE/ACM Intl. Conf. on Computer-Aided Design*. pp. 484–491.

Odawara, G., K. Iijima and K. Wakabayashi (1985). Knowledge-based placement technique for printed wiring boards. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 616–622.

Onodera, H., Y. Taniguchi and K. Tamaru (1991). Branch-and-bound placement for building block layout. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 433–439.

Otten, R. H. J. M. (1982). Automatic floorplan design. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 261–267.

Pan, D. (2004). *ECE 382V Lecture Notes, Optimization Issues in VLSI CAD* **Lecture 1**, 10. The University of Texas at Austin, USA.

Pang, Y., C.-K. Cheng and T. Yoshimura (2000). An enhanced perturbing algorithm for floorplan design using the O-tree representation. In: *Proc. of ACM Intl. Symp. on Physical Design*. pp. 168–173.

Papadopoulou, E. and D. T. Lee (2004). The Hausdorff Voronoi diagram of polygonal objects: a divide and conquer approach. *Intl. Journal of Computational Geometry and Applications* **14**(6), 421–452.

PARQUET (2006). http://vlsicad.eecs.umich.edu/bk/parquet/.

Pedram, M. and B. Preas (1989). Interconnection length estimation for optimized standard cell layouts. In: *Proc. of IEEE/ACM Intl. Conf. on Computer-Aided Design*. pp. 390–393.

Peixoto, H., M. Jacome, A. Royo and J. Lopez (2000). A tight upper bound for slicing floorplans. In: *Proc. of IEEE Intl. Symp. on Circuits and Systems*. pp. 280–285.

Prasitjutrakul, S. and W. J. Kubitz (1989). Path-delay constrained floorplanning: A mathematical programming approach for initial placement. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 364–369.

Qi, X., Z. Feng and X. Yan (1994*a*). An algorithm of timing driven floorplanning for VLSI layout design. In: *Proc. of Intl. Conf. on Computer-Aided Drafting, Design and Manufacturing Technology*. pp. 642–646.

Qi, X., Z. Feng and X. Yan (1994*b*). An algorithm of timing driven floorplanning for VLSI layout design. In: *Proc. of the 4th Intl. Conf. on Computer - Aided Drafting, Design and Manufacturing Technology*. pp. 642–646.

Rajagopal, K., T. Shaked, Y. Parasuram, T. Cao, A. Chowdhary and B. Halpin (2003). Timing driven force directed placement with physical net constraints. In: *Proc. of ACM Intl. Symp. on Physical Design*. pp. 60–66.

Ranjan, A., K. Bazargan, S. Ogrenci and M. Sarrafzadeh (2001). Fast floorplanning for effective prediction and construction. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems* **9**(2), 341–351.

Rao, N. S. V., N. Stoltzfus and S. S. Iyengar (1991). A 'retraction' method for learned navigation in unknown terrains for a circular robot. *IEEE Trans. on Robotics and Automation* **7**(5), 699–707.

Reda, S. and A. Chowdhary (2006). Effective linear programming based placement methods. In: *Proc. of ACM Intl. Symp. on Physical Design*. pp. 186–191.

Riess, B. M. and G. G. Ettelt (1995). SPEED: Fast and efficient timing driven placement. In: *Proc. of IEEE Intl. Symp. on Circuits and Systems*. pp. 377–380.

Rosenberg, E. (1989). Optimal module sizing in VLSI floorplanning by nonlinear programming. *Methods and Models of Operations Research* **33**, 131–143.

Sait, S. M. and H. Youssef (1995). *VLSI physical design automation : Theory and practice*. IEEE Press. New York, USA.

Sarrafzadeh, M. and C. K. Wong (1996). *An introduction to VLSI physical design*. The McGraw-Hill Companies, Inc. New York, USA.

Sarrafzadeh, M. and M. Wang (1997). NRG: Global and detailed placement. In: *Proc. of IEEE/ACM Intl. Conf. on Computer-Aided Design*. pp. 532–537.

Sassone, P. and S. K. Lim (2006). Traffic: A novel geometric algorithm for fast wire-optimized floorplanning. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* **25**(6), 1075–1086.

Sechen, C. (1988*a*). Chip planning, placement, and global routing of macro/custom cell integrated circuits using simulated annealing. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 73–80.

Sechen, C. (1988*b*). *VLSI placement and global routing using simulated annealing*. Kluwer Academic Publishers. Boston, USA.

Sechen, C. and A. L. Sangiovanni-Vincentelli (1986). TimberWolf 3.2: A new standard cell placement and global routing package. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 432–439.

Shahookar, K. and P. Mazumder (1991). VLSI cell placement techniques. *ACM Computing Surveys* **23**(2), 143–220.

Sherwani, N. A. (1999). *Algorithms for VLSI physical design automation*. Kluwer Academic Publishers. Boston, USA.

Sigl, G., K. Doll and F. M. Johannes (1991). Analytical placement: A linear or quadratic objective function. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 427–432.

Srinivasan, A., K. Chaudhary and E. S. Kuh (1991). RITUAL: A performance driven placement algorithm for small cell ICs. In: *Proc. of IEEE/ACM Intl. Conf. on Computer-Aided Design*. pp. 48–51.

Sturm, J. F. (1999). Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software* **11-12**, 625–653.

Suaris, P. R. and G. Kedem (1988). An algorithm for quadrisection and its application to standard cell placement. *IEEE Trans. on Circuits and Systems* **35**(3), 294–303.

Suaris, P. R. and G. Kedem (1989). A quadrisection-based combined place and route scheme for standard cells. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* **8**(3), 234–244.

Sutanthavibul, S., E. Shragowitz and J. B. Rosen (1991). An analytical approach to floorplan design and optimization. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* **10**(6), 761–769.

Takahashi, K., K. Nakajima, M. Terai and K. Sato (1995). Min-cut placement with global objective functions for large scale sea-of-gates arrays. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* **14**(4), 434–446.

Takouda, P. L., M. F. Anjos and A. Vannelli (2005). Global lower bounds for the VLSI macrocell floorplanning problem using semidefinite optimization. In: *Proc. of the Fifth Intl. Workshop System-on-Chip for Real-Time Applications*. pp. 275–280.

Tang, X., R. Tian and D. F. Wong (2001). Fast evaluation of sequence pair in block placement by longest common subsequence computation. *IEEE Trans. on Computer-Aided Design* **20**(12), 1406–1413.

Tang, X., R. Tian and M. D. F. Wong (2006). Minimizing wire length in floorplanning. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* **25**(9), 1744–1753.

Tani, K., S. Tsukiyama, I. Shirakawa and H. Ariyoshi (1988). Area-efficient drawings of rectangular duals for VLSI floor-plan. In: *Proc. of IEEE Intl. Symp. on Circuits and Systems*. pp. 1545–1548.

Tellez, G. E., D. A. Knol and M. Sarrafzadeh (1996). A performance-driven placement technique based on a new budgeting criterion. In: *Proc. of IEEE Intl. Symp. on Circuits and Systems*. pp. 504–507.

Tsay, R. S., E. Kuh and C. P. Hsu (1988). PROUD: A sea-of-gates placement algorithm. *IEEE Design and Test of Computers* **5**(6), 44–56.

van Camp, D. J., M. W. Carter and A. Vannelli (1991). A nonlinear optimization approach for solving facility layout problems. *European Journal of Operational Research* **57**, 174–189.

Vandenberghe, L. and S. Boyd (1996). Semidefinite programming. *SIAM Review* **38**(1), 49–95.

Viswanathan, N. and C. C.-N. Chu (2004). FastPlace: Efficient analytical placement using cell shifting, iterative local refinement and a hybrid net model. In: *Proc. of ACM Intl. Symp. on Physical Design*. pp. 26–33.

Vorwerk, K., A. Kennings and A. Vannelli (2004). Engineering details of a stable force-directed placer. In: *Proc. of IEEE/ACM Intl. Conf. on Computer-Aided Design*. pp. 573–580.

Vygen, J. (1997). Algorithms for large-scale flat placement. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 746–751.

Wang, B., M. Chrzanowska-Jeske and M. Jeske (2003). Methods for efficient use of lagrangian relaxation for soc soft-module floorplanning. In: *Proc. of IEEE Intl. Conf. on Systems-on-Chip*. pp. 293–294.

Wang, K. and W. K. Chen (1993). A class of zero wasted area floorplan for VLSI design. In: *Proc. of IEEE Intl. Symp. on Circuits and Systems*. pp. 1762–1765.

Wang, M., X. Yang and M. Sarrafzadeh (2000). Dragon2000: Standard-cell placement tool for large industry circuits. In: *Proc. of IEEE/ACM Intl. Conf. on Computer-Aided Design*. pp. 260–263.

Weis, B. X. and D. A. Mlynski (1987). A new relative placement procedure based on MSST and linear programming. In: *Proc. of IEEE Intl. Symp. on Circuits and Systems*. pp. 564–567.

Wimer, S., I. Koren and I. Cederbaum (1988). Floorplans, planar graphs, and layouts. *IEEE Trans. on Circuits and Systems* **35**(3), 267–278.

Wimer, S., I. Koren and I. Cederbaum (1989). Optimal aspect ratios of building blocks in VLSI. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* **8**(2), 139–145.

Wipfler, G. J., M. Wiesel and D. A. Mlynski (1983). A combined force and cut algorithm for hierarchical VLSI layout. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 124–125.

Wolkowicz, H., L. Vandenberghe and R. Saigal (Eds.) (2000). *Handbook of Semidefinite Programming: Theory, Algorithms, and Applications*. Kluwer Academic Publishers, Dordrecht. The Netherlands.

Wong, D. F. and C. L. Liu (1986). A new algorithm for floorplan design. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 101–107.

Wong, D. F. and C. L. Liu (1989). Floorplan design of VLSI circuits. *Algorithmica* (4), 263–291.

Wong, D. F., H. W. Leong and C. L. Liu (1988). *Simulated annealing for VLSI design*. Kluwer Academic Publishers. Norwell, MA, USA.

Xiang, H., X. Tang and M. D. F. Wong (2003). Bus-driven floorplanning. In: *Proc. of IEEE/ACM Intl. Conf. on Computer-Aided Design*. pp. 66–73.

Xu, N., Z. Jiang and F. Huang (2006). Fuzzy logic for low power driven floorplanning. In: *Proc. of IEEE Intl. Conf. on Solid-State and Integrated Circuit Technology*. pp. 23–26.

Yildiz, M. C. and P. H. Madden (2001). Improved cut sequences for partitioning based placement. In: *Proc. of ACM/IEEE Design Automation Conf.* pp. 776–779.

Ying, C.-S. and J. S.-L. Wong (1989). An analytical approach to floorplanning for hierarchical building blocks layout. *IEEE Trans. on Computer-Aided Design* **8**(4), 403–412.

Young, F. Y. and D.F. Wong (1997). How good are slicing floorplans. *Integration, the VLSI Journal* **23**, 61–73.

Young, F. Y., C. C. N. Chu, W. S. Luk and Y. C. Wong (2000*a*). Floorplan area minimization using lagrangian relaxation. In: *Proc. of ACM Intl. Symp. on Physical Design*. pp. 174–179.

Young, F. Y., C. C. N. Chu, W. S. Luk and Y. C. Wong (2001). Handling soft modules in general nonslicing floorplan using lagrangian relaxation. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* **20**(5), 687–692.

Young, F. Y., D. F. Wong and H. H. Yang (2000*b*). Slicing floorplans with range constraint. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* **19**(2), 272–278.

Youssef, H., S. M. Sait and K. J. Al-Farra (1995). Timing influenced force directed floorplanning. In: *Proc. of Design Automation Conf. with EURO-VHDL*. pp. 156–161.

Zeng, Y., S. Dong and X. Hong (2003). Floorplanning with soft rectilinear blocks using corner block list. In: *Proc. of IEEE Intl. Conf. on ASIC*. pp. 356–359.

Zhan, Y., Y. Feng and S. S. Sapatnekar (2006). A fixed-die floorplanning algorithm using an analytical approach. In: *Proc. of Asia and South Pacific Design Automation Conf.* pp. 771–776.