Cell Placement Using Constructive and Iterative Improvement Methods

by

Andrew Kennings

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

Waterloo, Ontario, Canada, 1997

0-612-21359-5

Abstract

Modern integrated circuits contain thousands of switching cells making their design an overwhelming task. The design procedure is therefore divided into sequence of design steps. Circuit layout is the design step in which a physical realization of a circuit is obtained from its functional description. Placement is one subproblem of circuit layout which involves positioning cells within a target placement geometry while minimizing the placement area and the total interconnecting wire length. Placement heuristics capable of producing high quality (near optimal) placements with little computational effort are required as integrated circuits increase in size.

In this thesis, we propose and investigate a placement heuristic that combines constructive and iterative improvement methods. The heuristic is both flexible and extensible. A good initial placement is constructed through a combination of relative placements and circuit partitioning. Computational efficiency is achieved by using an interior point method for finding relative placements and cell interchange heuristics for finding circuit partitions. Two formulations for the relative placement problem are proposed and investigated. Iterative rather than direct methods are shown to reduce the computational time required by the interior point method. A clustering heuristic is also proposed for improving the efficiency of the placement heuristic. Subsequently, iterative improvement is applied to further improve the placement. We describe a simple and greedy iterative improvement method which is capable of producing high quality final placements when provided with a good initial placement. Placements generated by our heuristic are shown to compare favourably in terms of quality and computational efficiency to other established placement heuristics on a set of test circuits.

Acknowledgements

Many people have made significant contributions during the completion of this work. It is not possible to sufficiently thank everyone, but special thanks must be extended to several individuals.

I would like to thank my supervisor, Dr. Anthony Vannelli. Without his time, financial support and patience during times of tantrums, this work would have never been completed. Thanks to the Natural Sciences and Engineering Research Council of Canada for making financial support available through a number of scholarships. I would like to thank my examining committee, Dr. George Kesidis, Dr. Henry Wolkowitz and Dr. Stefan Leue. Special thanks to my external examiner, Dr. Tamás Terlaky.

I would like to thank Dr. Victor Quintana, without whom I might never have pursued graduate studies, at least during this lifetime. Thanks to Dr. Shawki Areibi for valuable work related discussions and for making a short sabbatical abroad possible. I would like to thank Phil Regier, our system administator, for keeping my computer healthy. I would also like to thank my fellow graduate students, in particular Hussein Etwail, Tony Savor and William Bishop.

Last, but certainly not least, I would like to thank my family. Thanks to my sister Nancy and her husband Steve. Thanks to my youngest niece, Megen, for reminding me of how happy I must have been when I was two years old. Thanks to my older niece, Stephenie, for teaching me how to count to ten without a calculator. Finally, I would like to thank my parents, Reginald and Marlene Kennings, for their continual support.

To be means to be in competition.

C. S. Lewis

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Modern integrated circuits typically contain many thousands of switching cells. The large number of switching cells makes the design of an integrated circuit an overwhelming task. The design procedure is therefore divided into a sequence of design steps [22], namely *architectural design*, *logical design*, *circuit design*, *circuit layout*, *design verification*, *testing*, and *debugging*. These design steps are illustrated in Figure 1.1, and we briefly describe each of these steps.

Architectural design involves defining the goals and constraints of the system. This includes what the system will do, how the system will be divided into components and how the components will interact. Criteria for the system are also specified during the architectural design and may include specifications such as power requirements, area requirements, speed requirements and so forth.

Logic design involves deciding how each component of the system will be expressed logically. Various components may be implemented as RAM (random access memory), ROM (read only memory) or as PLAs (programmable logic arrays), for example.

Circuit design involves converting the logic design into electronic circuits that implement the desired functions. The result of the circuit design step is a set of functional blocks (cells) which are generally rectangular in shape and connected together via a set of signal paths (nets).

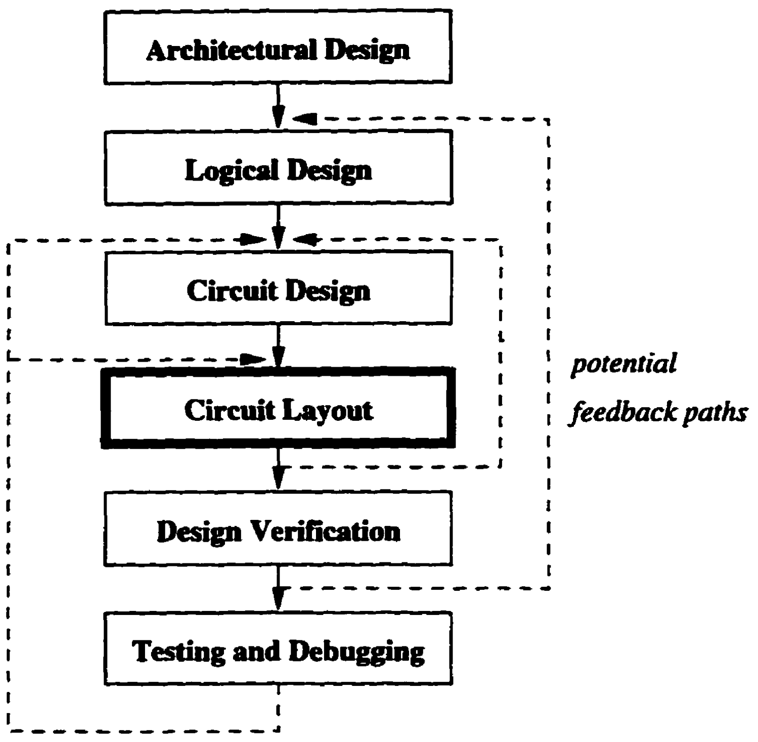


Figure 1.1: Integrated circuit design procedure.

Circuit layout involves converting the functional description of the circuit into a set of *physical masks* which can be used to physically realize the circuit. Since circuit layout is the topic of this thesis, a more detailed description of circuit layout is provided in a subsequent section.

Finally, design verification, testing and debugging involve checking the circuit to ensure all design criteria are satisfied, and that the circuit correctly performs its intend function.

Although the design procedure has been described as a sequence of steps, the individual design steps are not mutually exclusive operations. Each step influences subsequent steps, and the results of any step may be used as feedback information to revise an earlier step. Several iterations of the design procedure may be required to obtain a satisfactory design for any given integrated circuit.

1.1 Circuit Layout

As input to the circuit layout design step, we are given a description of a circuit in the form of a *netlist* which is a description of switching elements, or *cells*, and their interconnecting wires, or *nets*. Nets connect to cells via *pins* which represent electrical connection points to the cells and are located within the boundaries of the cells. Circuit layout involves determining the geometrical coordinates for cells within a two dimensional plane, or in one of a few planar layers, and connecting the cells according to the netlist. In performing this task, several objectives and requirements are satisfied. Typical objectives include (i) area minimization, (ii) wire length minimization, and (iii) minimization of performance driven criteria such as path delays, power consumption and so forth. Requirements represent constraints imposed by the design technology or fabrication technology being used, such as sufficient spacing between cells and wires, and so forth. Circuit layout is therefore an example of a constrained optimization problem.

The circuit layout problem itself is NP-hard [29] and is therefore divided into a sequence of subproblems which are solved one after another. These subproblems are also intractable, but are amendable to *heuristic solution methodologies* which are best described as approximate schemes capable of yielding near optimal solutions to the required problem with reasonable computational effort. The most common division of the circuit layout problem into subproblems is illustrated in Figure 1.2. First, a *partitioning* subproblem may be solved. This subproblem involves dividing a circuit into a small set

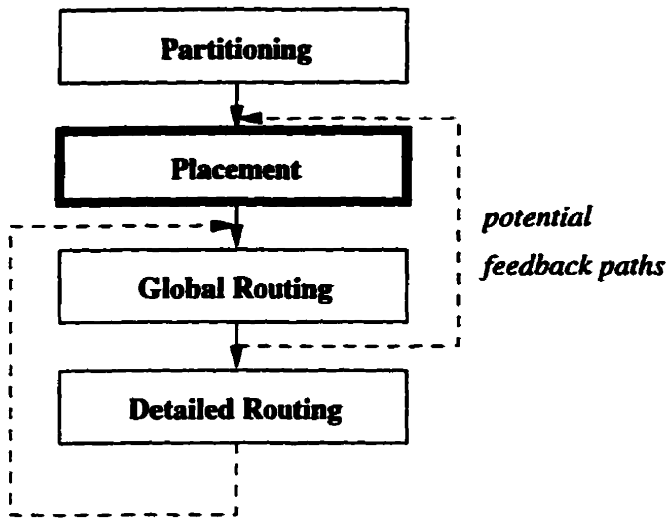


Figure 1.2: Circuit layout subproblems.

of relatively independent subcircuits which can be designed and implemented as separate circuits, and subsequently interconnected. The cell *placement* subproblem is solved (for each partitioned subcircuit) to determine the positions of the cells. Subsequently, the *global routing* and the *detailed routing* subproblems are solved to connect the cells according to the netlist. As illustrated for the circuit design procedure, the circuit layout subproblems are not mutually exclusive. The solution of each subproblem influences subsequent subproblems, and the results of any subproblem may be used as feedback information to revise and influence the solution of a previous subproblem.

As previously mentioned, circuit layout involves the minimization of several different objectives subject to constraints or restrictions imposed on the problem. However, due to the division of circuit layout into amendable subproblems, these objectives can often only be *estimated* during the solution of each subproblem. For instance, until the circuit routing is actually performed, quantities such as wire lengths and path delays are unknown. It is therefore necessary to have accurate estimates of these quantities during the placement subproblem. The quality of the circuit layout highly depends on the design of the heuristic methods used to solve the placement and routing subproblems.

1.2 Research Motivations and Objectives

In this thesis, we are concerned with the cell placement subproblem. Heuristics for cell placement may be classified as either iterative improvement [41] or as constructive methods [25]. We defer to Chapter 2 for a more detailed description of these heuristics and presently provide only enough description to illustrate the motivation and objectives for additional investigations into the cell placement subproblem.

1.2.1 Motivations

Previously proposed placement heuristics all exhibit certain advantages and disadvantages. Interestingly, the advantages of certain heuristics tend to be the disadvantages of other heuristics. For instance, certain iterative improvement methods produce very high quality placements, but require excessively large computational effort to do so. Conversely, certain constructive methods require little computational effort, and although good, the placements produced by these methods are generally of lesser quality than their iterative improvement counterparts. In designing a placement heuristic, both quality and computational efficiency are issues that must be addressed. The quality of the placement is essential for the performance of the final circuit whereas computational efficiency is essential for shortening the design procedure (this is especially true for large circuits where a "short" design procedure may correspond to weeks, months or years). Finally, the flexibility and robustness of a placement heuristic are also issues which must be addressed as layout styles change and different design objectives are proposed. Ongoing research is therefore necessary to develop newer and better placement heuristics which are: (i) effective, (ii) efficient, and (iii) flexible and robust.

1.2.2 Objectives

In this thesis, our main objective is to investigate and develop a placement methodology which exhibits the aforementioned characteristics through enhancements, alterations and combinations of previously proposed constructive and iterative improvement methods. Essentially, we propose to combine constructive and iterative improvement methods where the constructive method is used to create a good initial placement which is subsequently improved using an iterative improvement method.

Our proposed constructive method requires a combination of mathematical programming and circuit partitioning. When used in combination with circuit partitioning, the solution of the mathematical program (which is known as the relative placement problem) provides relative cell positions (that is, cell adjacencies and proximities to desired positions) throughout the placement area. These cell positions represent useful information for determining cell positions in a good initial placement. We present two formulations of the relative placement problem which facilitate an investigation of the tradeoff in quality of the initial placements created by the constructive method versus the computational effort required to do so. Unlike previous heuristics, we propose an interior point method for solving the relative placement problems which arise during the constructive method. We introduce the interior point method as an enhancement to previously proposed solution methodologies since the interior point method allows a wide variety of constraints to be included into the formulation of the relative placement problem without requiring any changes in the solution methodology. We illustrate that the computational "bottleneck" of the interior point method is the solution of a sequence of systems of linear equations. Therefore, as an additional enhancement, we investigate the application of iterative methods as a means of solving these systems of equations. We illustrate that iterative methods are useful for reducing the computational effort required to solve very large and sparse optimization problems when the degree of accuracy required in the solution (such as that required by the relative placement problem) is low. We also investigate circuit clustering as a means of reducing the computational effort required by the constructive method while improving the quality of the initial placements. Additionally, we illustrate that circuit clustering is applicable to other problems, such as circuit partitioning, which arise during the integrated circuit design procedure.

Our proposed iterative improvement method improves the placements created by the constructive method using a search heuristic to locally rearrange cells. We illustrate that, by taking into account the quality of the initial placement, the iterative improvement method requires little computational effort to create high quality placements. Finally, the objective function used to determine the improvement in any local rearrangement can be arbitrarily flexible when using a search heuristic, provided evaluation of the objective function is not prohibitive.

1.3 Thesis Outline

This thesis is organized as follows. In Chapter 2, we describe cell placement in greater detail. We describe different types of placement topologies encountered in practice. Previous placement heuristics are also described. Finally, our proposed placement heuristic is described as a combination of constructive and iterative improvement methods. The purposes and interactions of the various components of the heuristics are described.

In Chapters 3 through 6, our constructive method for generating initial placements is described. In Chapter 3, the relative placement problem is described as a method for determining relative cell positions throughout the placement area. Two different formulations of the problem are presented to illustrate potential tradeoffs in quality versus speed during the relative placement. An interior point method which is suitable for solving both formulations of the relative placement problem is described in Chapter 4. In Chapter 5, the circuit partitioning problem is presented and its application to cell placement is described. We illustrate that several iterations of relative placement and circuit partitioning results in a procedure for determining a good distribution of cells throughout the placement area. This distribution of cells is shown to be useful in determining good cell positions in an initial legal placement. The creation of an initial legal placement from the cell positions provided by the relative placement and circuit partitioning iterations is described in Chapter 6. Additionally, numerical results are presented in Chapter 6 to illustrate the effectiveness and efficiency of our constructive method using both relative placement formulations.

In Chapter 7, an iterative improvement method is presented for further improving the initial placement provided by the constructive method. Numerical results are presented to demonstrate that final placements generated after the completion of the constructive and iterative improvement methods are comparable or better than existing placement heuristics on a set of test circuits. We consider iterative improvement as a necessary step in the overall placement heuristic.

Circuit clustering is described in Chapter 8. A clustering heuristic for condensing a circuit netlist is proposed. Circuit clustering is proposed in order to reduce the dimensionality of the relative placement problems thereby improving the computational efficiency of the constructive method. Numerical results are presented to demonstrate the effectiveness of circuit clustering when incorporated into the placement heuristic. Clustering is demonstrated to provide comparable placements (to those obtained without clustering) while substantially reducing the computational effort.

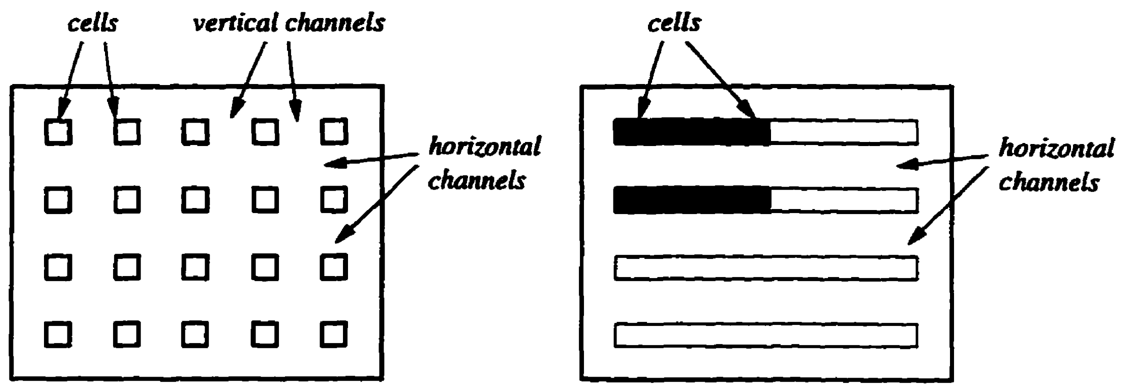
Finally, a summary of this thesis is presented in Chapter 9. The intention is to highlight the contributions of this thesis and to provide a description of future research possibilities for additional enhancements to this work.

Chapter 2

Cell Placement

As previously mentioned, cell placement is a subproblem of the circuit layout design step which involves positioning cells (although a small number of cells, known as *I/O pads* are fixed at positions around the periphery of the placement area) within a specified placement geometry. In performing the placement, several objectives are minimized while satisfying several restrictions or constraints imposed on the positions of the cells. Primarily, the objectives in cell placement are minimization of (i) the placement area, (ii) the wire length, and/or (iii) performance criteria such as path delays and so forth. Restrictions on cell positions are generally due to the type of placement required, which is a function of the technology and layout style begin used in the design. For instance, the technology determines the size of the cells and may influence the required spacing between neigbouring cells. The layout style may require the cells to be positioned within rows or at points arranged in a grid [29].

The type of cell placement we consider is *semi-custom* design [29] which is applicable to the design of Application Specific Integrate Circuits (ASICs) and Field Programmable Gate Arrays (FPGAs). In semi-custom design, typical circuits may contain thousands of cells. Restrictions are imposed on the positions of cells and locations for routing con-



(a) island-styled gate arrays (b) row-oriented standard cells

Figure 2.1: Semi-custom design topologies.

nections between cells. In other words, the physical topology of the circuit is essentially known a priori. Cells implementing different functions can be placed at any of the restricted positions and subsequently implemented by modifying a standard set of physical masks prior to the fabrication of the circuit.

Figure 2.1 illustrates common semi-custom design topologies, namely island-styled gate arrays and row-oriented standard cells/gate arrays. In island-styled gate arrays, cells are typically equal in width and height, and must be positioned at specified grid points within the placement area. Subsequent routing is performed in vertical and horizontal channels. In row-oriented placement, cells are equal in height, but are different in width. Cells must be placed into a prescribed number of rows located within the placement area. Once placed, routing is performed in the horizontal channels in between adjacent rows. Connections between cells in non-adjacent rows may be routed around the end of the rows, or through the insertion of additional cells, known as *feedthroughs*, into the rows. Since identical placement heuristics are typically used for each topology, we consider only row-oriented placement.

In performing cell placement, hand designs are impractical due to the large circuit sizes. Furthermore, the design objectives can only be approached at the expense of ex-

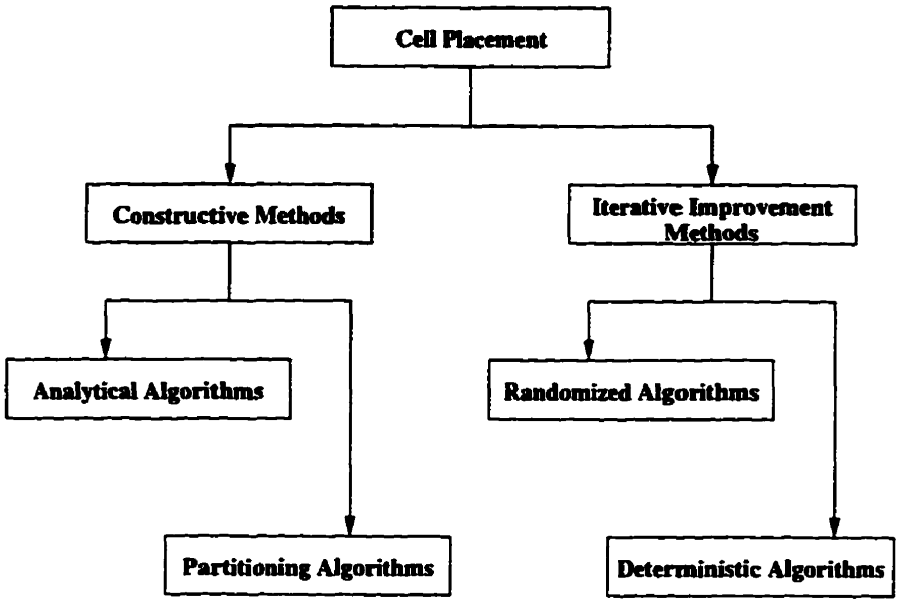


Figure 2.2: Taxonomy of placement heuristics.

cessive computational efforts. Therefore, high quality placements are obtained only by accepting excessive computational effort, or by sacrificing placement qualities. Automated placement heuristics are required to facilitate semi-custom design, and to reach a reasonable tradeoff in quality versus computational effort.

2.1 Previous Approaches

Many different heuristics have been proposed for cell placement. The taxonomy of these placement heuristics is illustrated in Figure 2.2. Placement heuristics may be broadly classified as either constructive or iterative improvement methods [29].

2.1.1 Constructive Methods

Constructive methods produce placements directly from the circuit description (that is, the cell-net interconnections provided by the netlist). These methods may be subdivided into two major categories, namely partitioning algorithms [8, 40] and analytical algorithms [32]. Constructive methods are considered global approaches, since placements are obtained by considering all circuit connections simultaneously. Typically, the combination of partitioning and analytical algorithms has resulted in the most successful constructive methods [25, 49].

Constructive methods have the advantage that they are generally fast and produce reasonably good placements for large circuits due to the globality (that is, the simultaneous consideration of all cell interconnections) of the approach. However, these methods are typically restrictive in the design objectives which can be incorporated into the cost functions and therefore cannot produce the highest quality placements.

2.1.2 Iterative Improvement Methods

Iterative improvement methods do not produce placements directly from the circuit description, but rather begin with an initial placement and search for an improved placement by making local changes to the existing placement. These perturbations are continued until a near optimal placement is obtained. Therefore, these methods are essentially local *search algorithms* which begin at an existing solution and move to neighbouring solutions via small and local perturbations. Iterative improvement methods may be subdivided into two major subclasses, namely randomized or deterministic algorithms. This division depends on whether or not a given perturbation is used to alter an existing placement.

Iterative improvement methods are highly flexibility in the design objectives which can be incorporated into the cost functions. Since the cost of a given placement is a function of the design objectives, it is only necessary to re-evaluate the cost function to determine if one placement is better than another placement.

Randomized algorithms always accept changes which lead to an improved solution, and may also accept changes leading to poorer solutions with a low probability. It is this acceptance of changes leading to poorer solutions which gives these algorithms the ability to escape from local minima, and to approach a global optimum. Randomized iterative improvement methods are traditionally based on Simulated Annealing [41] or on evolution-based algorithms [26, 27]. Randomized algorithms produce very high quality placements (often the best of the heuristic methods), which is a direct consequence of the ability of such approaches to escape from local minima. However, excessive amounts of computational effort are required by randomized algorithms to produce these results.

Unlike randomized algorithms, deterministic algorithms only accept changes which lead to improved solutions. These algorithms require less computational effort than their randomized counterparts, but are generally unable to produce placements of comparable quality due to their inability to escape from local minima. Deterministic algorithms that have the ability to escape local optima have been proposed [37]. Even so, these algorithms are not comparable to randomized algorithms in there ability to escape from local minima.

2.2 A Combination of Methods

Since constructive and iterative improvement methods both exhibit advantages and disadvantages, further progress in developing placement heuristics requires combining and/or enhancing the different methods. Constructive methods, although not as flexible as iterative improvement methods, are capable of producing good placements with low computational effort. Flexibility for the constructive method may be achieved by enhancing existing constructive methods by using different modelling of the problem or by using more flexible solution algorithms. Conversely, iterative improvement methods are very flexible and are capable of producing high quality placements, but at the expense of excessive computational effort. By combining the methods, excessive computational efforts may be avoided. An iterative improvement method can take advantage of a good initial placement generated by a constructive method and less computational effort will be required to reach a near optimal placement. In other words, a deterministic improvement algorithm may be sufficient. Similarily, the quality of the final placement is not restricted by the flexibility of the constructive method since subsequent iterative improvement is performed. The combination of a constructive and an iterative improvement method results in an overall placement heuristic illustrated in Figure 2.3. As illustrated, the placement heuristic requires a combination of relative placements, circuit partitioning and search heuristics.

An initial placement is generated using a constructive method as follows. The circuit description and desired placement geometry is read, and an iterative procedure involving a combination of mathematical programming and circuit partitioning begins. The circuit description and placement geometry is first used to create an initial overlapping cell placement which is described by a mathematical program known as the *relative placement problem.* By allowing several placement violations (cells are allowed to overlap and are not restricted to positions within the rows), the solution of the relative placement problem provides the general positions of cells throughout the placement area while minimizing an estimate of the total wire length. The intention is to determine general cell proximities and adjacencies in the final placement while providing a global view of the circuit interconnections. Ideally, the placement restrictions should be included in the formulation of the relative placement problem. However, these simplifications are required in order to obtain a practical solution methodology for solving the relative placement formulations.

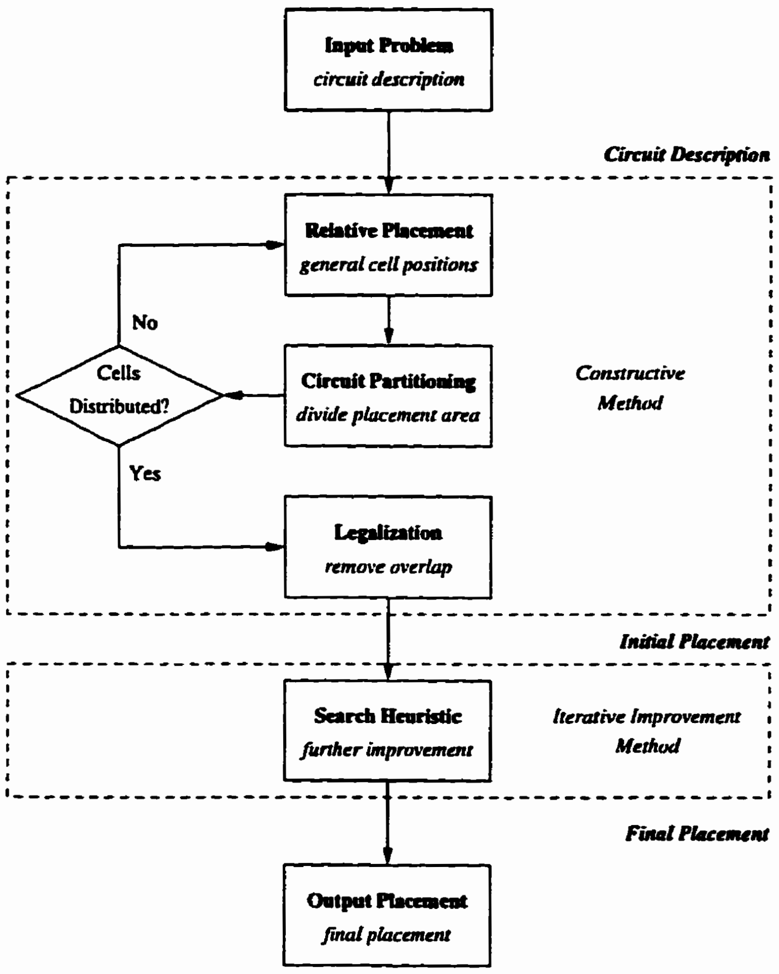


Figure 2.3: A combination of constructive and iterative improvement methods.

The simplifications made when determining relative cell positions may result in significant cell overlap. Furthermore, cells tend to cluster towards the center of the placement area. Circuit partitioning is applied to divide the cells into disjoint groups and the placement area into regions. By assigning the partitioned groups of cells to disjoint regions within the placement area cell overlap may be reduced (since cells in one regions are prevented from overlapping with cells in other regions) and area utilization is improved (by forcing cells into unoccupied portions of the placement area). By resolving the relative placement problem with additional information (constraints) appended from the circuit partitioning problem, a good (global) distribution of cells throughout the placement area may be obtained. This information aids in determining cell proximities in the final placement.

Therefore, there is an interaction between the relative placement and the circuit partitioning problems. The relative cell positions are used to generate initial partitions of the cells. These initial partitions are subsequently improved using circuit partitioning. Once the partitioning is completed, the division of the cells into disjoint regions throughout the placement area is taken into account during subsequent relative placements by including additional constraints into the relative placement formulations. This alternation between mathematical programming and circuit partitioning is repeated until cells are distributed evenly throughout the entire placement area. The interaction between the relative placement and circuit partitioning problems is described in more detail in Chapter 5. Subsequently, a *legalization* heuristic based on the cell positions can be applied to remove any residual cell overlap and to satisfy any placement restrictions imposed on the cell positions.

Once the initial cell placement is obtained, an iterative improvement method is applied to further improve the placement. Given the effort expended to produce an initial placement, we assume that the initial placement is good (cells are in the vicinity of their final

Circuit	Cells	Nets	Pads	Rows
circuit1	833	983	81	21
circuit2	3014	3136	107	21
biomed	6417	5742	97	27
industry1	2271	2478	580	15
industry2	12142	13419	495	70
industry3	15059	21938	375	50

Table 2.1: Test circuit sizes and placement geometries.

positions in a near optimal placement). A greedy and highly localized search heuristic based on cell moves and swaps is applied to rearrange cells. Localizing the rearrangement of the cells (which is possible as a consequence of the effort spent to produce the initial placement), implies our iterative improvement method requires less computational effort. That is, since the cells are highly restricted in their possible positions, the search for improved cell positions is accomplished in a more timely manner.

2.3 Test Circuits

Throughout this work, numerical results are presented on a set of test circuits to illustrate various aspects of the placement heuristic. All the numerical results presented were produced on a Sun SPARCstation 5/85 with 64 Mbytes of memory.

The test circuits we consider are illustrated in Table 2.1. In Table 2.1, the identifier and the number of cells, nets and I/O pads for each circuit are presented. The first two test circuits are of our own creation and the remaining four circuits are taken from the MCNC benchmark test suite [28]. The number of rows required in the final placement of each circuit are also indicated in Table 2.1. The number of rows determines the desired placement geometry in the following sense. The height of the cells determines the height of the rows and the spacing between the rows. This in turn determines the height of the placement area. The width of the placement area is determined by the length of

Circuit	Cell Deg.		Net Size			
	Avg.	Std.	Avg.	Std.	% ≤ 3	% > 10
circuit1	3.36	1.34	3.12	2.50	82.50	3.46
circuit2	3.66	1.57	3.65	3.77	73.57	4.82
biomed	3.23	1.06	3.66	20.89	84.78	3.48
industry1	3.02	1.38	3.47	8.56	80.79	3.71
industry2	3.81	1.81	3.59	10.97	85.22	4.48
industry3	4.27	1.54	3.00	3.23	79.36	0.77

Table 2.2: Additional test circuit statistics.

the longest row once the cells are placed into the rows. By keeping the rows equal in length (which is generally not possible since the cells differ in widths), the width of the placement area is minimized.

Additional statistics for each circuit are presented in Table 2.2. For each circuit, the average and standard deviation for the number of nets incident on each cell are presented. These statistics indicate that very few nets are incident to each cell and that the variation in the number of incident nets is low. The average and standard deviation for the number of cells connected to each net are also presented in Table 2.2. Finally, the number of nets (expressed as a percentage of the total number of nets) connecting less than or equal to 3 cells and more than 10 cells are presented. These statistics indicate that a very large percentage of the nets are short (connected to only a few cells) and very few nets are long (connected to many cells). Therefore, the circuits are typically sparse. For several circuits, namely *biomed*, *industry1* and *industry2*, the standard deviations for the number of cells connected to each net are quite large. This indicates that for these test circuits, the variation in net connectivity is large and that some of the long nets are very long.

2.4 Summary

In this chapter, semi-custom design has been described. Semi-custom design is especially relevant to ASIC and FPGA design, where the circuits may contain many thousands of cells. Because of the large number of cells, hand design is not possible and optimal cell placements may only be obtained by accepting excessively large computational efforts. Several automated placement heuristics have been proposed and may broadly be classified as either constructive or iterative improvement methods. The classification depends on the approach for finding placements. Primarily, constructive methods are less flexible, but are capable of producing good placements with little computational effort. Conversely, iterative improvement methods are highly flexible, and are capable of producing high quality placements. Unfortunately, those iterative improvement methods capable of producing high quality placements require large computational time and effort.

To improve cell placement, a combination of methods is useful in order to exploit the advantages of the different methods. A constructive method is useful for generating a good initial placement which is subsequently improved using an iterative improvement method. Given that the initial placement is good, the computational effort of the iterative improvement method may be reduced by taking into account the quality of the initial placement. Similarily, the quality of the final placement is not restricted by the constructive method due to the subsequent iterative improvement. The combination of methods results in an overall placement heuristic requiring a combination of mathematical programming and circuit partitioning to construct an initial placement. This is followed by the application of a search heuristic to perform the iterative improvement.

In the next chapter, we begin the description of our proposed constructive method by introducing the relative placement problem.

Chapter 3

Relative Placement

In this chapter, we consider the relative placement problem. Relative placement involves determining the relative positions of cells throughout the placement area while minimizing an estimate to the total interconnecting wire length. In determining the relative cell positions, the relative placement problem takes a *global view* of the cell positions by considering all cell interconnections simultaneously. This global view aids in determining cell proximities and adjacencies in an initial legal (non-overlapping) placement. Several simplifications are made when formulating the relative placement problem in order to make the resulting problem solvable using an efficient algorithm. For instance, cells are permitted to overlap and are not restricted in their positioning provided they fall within the placement area. Due to these simplifications, the solution of the relative placement problem does not produce a legal placement by itself. Relative placement is a heuristic method intended to provide a good "idea" of the general cell positions. These cell positions can be provided to (or used in combination with) other heuristics to remove cell overlap and to satisfy placement restrictions.

The relative placement problem has been formulated as a mathematical program, typically with either linear or quadratic objective functions [3, 5, 20, 21, 23, 25, 37, 39, 38, 42, 49]. The solution methodologies proposed have varied according to the objective function used and the constraints included in the formulation. Example solution methodologies have included linear programming methods [5, 23, 49], eigenvector methods [3, 20], linear systems methods [25, 37] and various lagrangian multiplier methods [21, 39, 38, 42].

In this chapter, we present two relative placement formulations. The first formulation uses a quadratic objective function whereas the second formulation uses a linear objective function. Both formulations are subject to a set of linear constraints and variable bounds. Therefore, one formulation may be solved as a Quadratic Program (QP) and the other as a Linear Program (LP). We consider two formulations since the different objective functions imply a different estimate of wire length and therefore different relative cell placements. Furthermore, the different formulations result in optimization problems which differ in the number of unknowns and the number of constraints. Therefore, the potential for tradeoffs in quality of solution versus computational effort may be investigated. Finally, we consider both formulations since both may be solved using an interior point method [45] (which is the topic of the next chapter).

This chapter is organized as follows. The formulations of the relative placement problem as a QP and a LP are presented in Sections 3.1 and 3.2, respectively. In Section 3.3, we present an analysis of the two formulations. We demonstrate the differences in the size and storage requirements for the two formulations. The potential benefits in quality versus computational effort for each of the formulations is also described to illustrate why both formulations should be considered. Finally, a summary of this chapter is provided in Section 3.4.

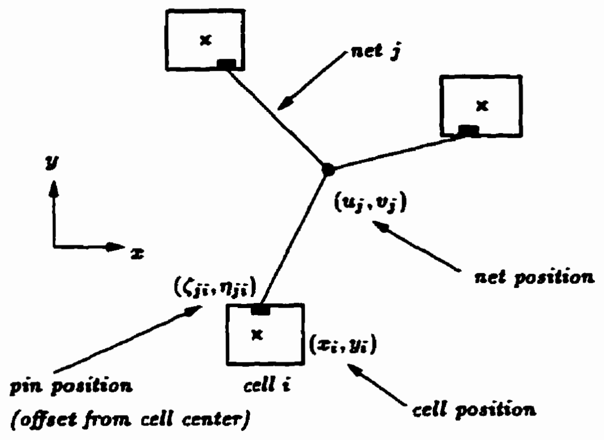


Figure 3.1: Quadratic estimate of wire length.

3.1 A Quadratic Programming Formulation

3.1.1 The Objective Function

Quadratic program formulations are a result of the estimate used for wire length and the constraints included in the problem formulation [25]. As an estimate of wire length for net j, we use the sum of the squared distance from all pins to the net position, which is taken as the mean value of the coordinates of the connected pins. This estimate of wire length is illustrated in Figure 3.1. The resulting estimate of wire length for net j, denoted by l_j, is given by

$$l_j = \sum_{i \in C_j} \left[(x_i + \zeta_{ji} - u_j)^2 + (y_i + \eta_{ji} - v_j)^2 \right], \tag{3.1}$$

where C_j denotes the set of cells connected to net j, (x_i, y_i) denotes the position of cell i, (u_j, v_j) represents the location of net j, and (ζ_{ji}, η_{ji}) denotes the offset for the pin connecting cell i to net j. This estimate of wire length is separable in the x and y directions, and minimization can be performed independently in both directions. The following description involves only the x direction, but extends to the y direction without any loss of generality.

The total estimate of the wire length is given by summing the estimates of wire length for all the nets. Performing this summation and substituting the coordinates for each net and the coordinates of all fixed I/O pads results in a total estimate of wire length (ignoring constant terms) that can be written in matrix form as

$$L_{QP} = \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{c}^T\mathbf{x} \tag{3.2}$$

where $\mathbf{Q}$ is a symmetric positive definite matrix ($\mathbf{Q} > 0$), $\mathbf{c}$ is a cost vector, and $\mathbf{x}$ is the vector of unknown cell positions. This estimate of wire length results in a symmetric positive definite quadratic objective function.

3.1.2 The Problem Constraints

We now consider constraints imposed on the cell positions. Although we allow cells to overlap, we still include constraints in the formulation which tend to reduce the amount of cell overlap and provide an even distribution of cells throughout the placement area. We assume that the placement area has been partitioned into disjoint regions, and that each free cell has been assigned to one region (the method for obtaining a partitioning of the cells is the subject of a subsequent chapter). The partitioning of the placement area and the restriction of cells to regions is illustrated in Figure 3.2.

For each region j, let R_j denote the cells assigned to this region. Additionally, let L_j and U_j denote the lower and upper boundaries for region j, respectively. Cells assigned to region j are restricted to positions within the region by including the variable bounds given by

$$L_j \leq x_i \leq U_j, \quad \forall i \in R_j. \tag{3.3}$$

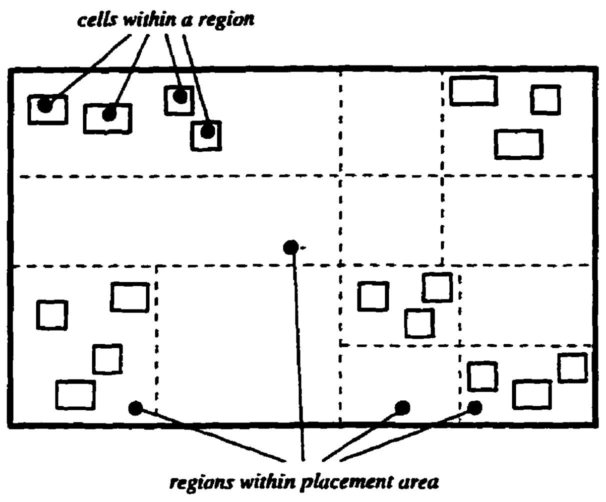


Figure 3.2: Restriction of cells to regions.

The consequence of the variable bounds is a reduction in the total amount of cell overlap since cells in different regions are prevented from overlapping with each other.

To improve the distribution of cells throughout each region, we include first moment constraints. Let X_j denote the centre of region j. For each region j, we include the inequality constraints

$$(1-\alpha)X_j \leq \frac{1}{F_j}\sum_{i\in R_j} A_i x_i \leq (1+\alpha)X_j, \tag{3.4}$$

where F_j denote the total area of the cells assigned to region j and $\alpha \in (0,1)$ is a parameter included to permit some (small) amount of flexibility on the value of the first moment for the cells in region j. The first moment constraints improve the distribution of cells within each region by requiring a balancing of cell area throughout the region. That is, if a cell tends to the left of a region due to the minimization of wire length, then other cells within the region will be forced to the right of the region in order to satisfy the first moment constraints. The result is a better usage of the area within each region.

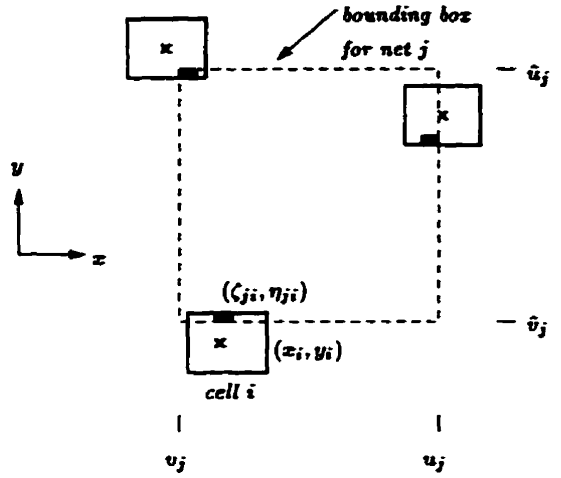


Figure 3.3: Linear estimate of wire length.

The constraints and variable bounds can be expressed in matrix form as

$$\mathbf{Ax} \leq \mathbf{b}, \quad \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}. \tag{3.5}$$

Equations (3.2) and (3.5) result in a QP formulation for the relative placement problem.

3.2 A Linear Programming Formulation

3.2.1 The Objective Function

Like quadratic program formulations, linear program formulations are a result of the wire length estimate and the constraints included during the problem formulation [49]. As an estimate of wire length for each net, we use the half perimeter wire length (HPWL) as shown in Figure 3.3. For each net j, we enclose all cells connected to net j by a *bounding box*, and introduce the variables pairs (u_j, v_j) and $(\hat{u}_j, \hat{v}_j)$ to denote the upper and lower limits on the bounding box in the x and y directions, respectively. The resulting estimate of wire length for net j is given by

$$l_j = (u_j - v_j) + (\hat{u}_j - \hat{v}_j). \tag{3.6}$$

This estimate of wire length is separable in the x and y directions and minimization can be performed independently in both directions. The rest of the discussion involves only the x direction, but extends to the y direction without any loss of generality.

The total estimate of the wire length is given by summing the estimates of wire length for all the nets. Performing this summation results in a total estimate of wire length that can be written in matrix form as

$$L_{LP} = \mathbf{c}^T(\mathbf{u} - \mathbf{v}) + \mathbf{0}^T\mathbf{x}, \tag{3.7}$$

where $\mathbf{u}$ and $\mathbf{v}$ are vectors representing the net variables (that is, the vectors representing the upper and lower limits on the bounding boxes for all nets, respectively), $\mathbf{x}$ represents the unknown cell positions, $\mathbf{0}$ is the zero vector, and $\mathbf{c}$ is a cost vector (equal to $\mathbf{e}$, where $\mathbf{e}$ denotes a vector of ones). This estimate of the total wire length is a linear objective function.

3.2.2 The Problem Constraints

As for the QP formulation, variable bounds and first moment constraints are included in the LP formulation to eliminate overlap between cells in different regions and to improve cell distribution within each region, respectively.

For the QP formulation, nets are considered as points that can be expressed directly in terms of the cell positions (that is, by equality relationships) and therefore eliminated from the problem. Since the net variables represent only bounds on the surrounding bounding boxes, they cannot be expressed directly in terms of the cell positions and cannot be eliminated from the LP formulation. The LP formulation requires additional constraints to relate the net and cell positions. The net variables $\mathbf{u}$ and $\mathbf{v}$ must remain above and below all pin positions on the cells in the respective nets. This restriction can be accomplished using inequality constraints and variable bounds. Consider net j and its connected cells C_j. For each net j, u_j is restricted above all pins on the cells in net j using the inequalities

$$u_j \geq x_i + \zeta_{ji}, \quad \forall i \in C_j. \tag{3.8}$$

(Recall that nets connect to pins. For instance, for a net j which connects to a pin of cell i, the pin position is $x_i + \zeta_{ji}$, where ζ_{ji} is a *constant* expressed in the circuit description indicating the offset of the pin from the center of the cell). For free cells, these inequalities represent constraints since the cell positions are unknown. For fixed I/O pads, these inequalities represent variable bounds on the net variables $\mathbf{u}$ since the fixed I/O pad positions are known. By replacing u_j with v_j and changing the direction of the inequality, similar constraints can be included to restrict v_j below all cells in net j.

The problem constraints and variable bounds can be written in matrix form as

$$\mathbf{A}\begin{bmatrix}\mathbf{u}\\ \mathbf{v}\\ \mathbf{x}\end{bmatrix} \leq \mathbf{b}, \quad \mathbf{l} \leq \begin{bmatrix}\mathbf{u}\\ \mathbf{v}\\ \mathbf{x}\end{bmatrix} \leq \mathbf{u}. \tag{3.9}$$

Equations (3.7) and (3.9) result in a LP formulation for the relative placement problem.

3.3 Analysis of the Formulations

Both the QP and LP formulations can be expressed as optimization problems in the form given by

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}^T\mathbf{z} + \tfrac{1}{2}\mathbf{z}^T\mathbf{Q}\mathbf{z} \\ \text{subject to} \quad & \mathbf{A}\mathbf{z} \le \mathbf{b}, \\ & \mathbf{0} \le \mathbf{z} \le \mathbf{u} \end{aligned} \tag{3.10}$$

where $\mathbf{z}$ represents the vector of unknowns. When $\mathbf{Q} \neq \mathbf{0}$ the QP formulation is implied whereas when $\mathbf{Q} = \mathbf{0}$ the LP formulation is implied. This optimization problem can be solved in polynomial-time using the same primal-dual interior point method [45]. It is important to emphasize that the LP formulation is *not* solved using a quadratic program solver simply by setting $\mathbf{Q} = 0$. The structure (that is, the sequence of steps and computational requirements at each step) of the interior point method is essentially the *same* regardless of whether or not the optimization problem is a QP or a LP. This represents one benefit of using the interior point method for solving the relative placement formulations in that both formulations are essentially solved using the same algorithm, with only minor changes required to implement the algorithm.

The question arises as to the advantages and disadvantages of both formulations. The complexity of the relative placement problem is a function of the solution method used (in our case the interior point method). In turn, the computational efficiency of the method is a function of the number of mathematical operations required during the solution method. It is therefore useful to consider the optimization problem sizes and storage requirements for both formulations.

Let p represent the number of pins in the circuit, n the number of nets, c the number of cells, and r the number of regions into which the placment area has been partitioned. When considering the optimization problem sizes and storage requirements, the quantities of interest are the number of variables and constraints, and the number of nonzeros in any matrices required in the formulation.

For the QP formulation the number of variables equals c and the number of constraints equals $2r$. The number of nonzeros in the constraint matrix **A** (due to the first moment constraints) is $2c$ since each cell is involved in only 2 constraints. Consider next the matrix **Q**. If we consider one net connecting q cells, then we find that the quadratic estimate of wire length causes a $q \times q$ dense block of nonzero entries (a clique) to appear in **Q**. Therefore, in the worst-case, where we find a net connecting every cell, the matrix **Q** will be totally dense and require storage for c^2 nonzero entries. Considering both matrices **Q** and **A**, the storage requirements for the QP formulation are $c^2 + 2c$. We note, however, that for typical circuits the nonzero entries in **Q** is much lower than c^2 due to the sparsity of the circuit (the circuit statistics presented in Chapter 2 support this observation).

For the LP formulation the resulting number of variables is equal to $c + 2n$ due to the inclusion of the net variables within the formulation. For each pin, two constraints are required to represent the bounding box relationships. Therefore, the total number of bounding box constraints in $2p$. Including the first moment constraints, the total number of constraints for the LP formulation is $2p + 2r$. Since the matrix **Q** does not appear in the LP formulation, it is necessary to consider only the constraint matrix **A**. For each bounding box constraint, two nonzeros are introduced, resulting in a total of $4p$ nonzero entries. Considering the first moment constraints, this implies a total of $4p + 2c$ nonzeros in **A**.

The QP formulation results in optimization problems which have substantially fewer variables and constraints than the LP formulation. Comparision of the storage requirements is slightly more difficult since the storage requirements for the QP formulation can only be bounded and not calculated explicitly. Although it appears that the QP formulation requires more storage (due to the quadratic term c^2), in practical situations the storage requirements for the QP formulation are less than the LP formulation. This observation is highlighted in the next chapter, when we introduce the interior point method and solve several relative placement problems using both formulations. We briefly note that the optimization problem sizes and storage requirements for the LP formulation may be reduced somewhat by applying standard LP preprocessing techniques [16] whereas the QP formulation does not benefit from preprocessing due to the nature of the first moment constraints.

The analysis just presented, along with the empircal results presented in the next chapter for storage requirements, indicate that QP formulation is preferable to the LP formulation: It results in smaller optimization problems requiring less storage which implies lower computational efforts. However, the LP formulation cannot be ignored. Empirical investigations have shown that linear objective functions may produce better placements in terms of wire length than quadratic objective functions [37]. Quadratic objective functions over emphasize the minimization of long nets (nets connecting many cells) at the expense of short nets (nets connecting few cells), whereas linear objective functions consider all nets equally during the minimization of the wire length. Furthermore, linear objective functions better approximate the actual wiring strategies used to connect cells during the actual routing of the circuit. Therefore, the LP formulation offers the possibility of better final placements at the expense of increased computational effort. We consider the LP formulation for this reason and to faciliate an investigation of the tradeoff in computational effort versus quality of solution for different relative placement formulations.

3.4 Summary

In this chapter, the relative placement problem has been formulated as both a quadratic program and as a linear program. The formulation was shown to depend on the estimate of wire length. In terms of the resulting optimization problem sizes and storage requirements, the QP formulation is preferable to the LP formulation. Less variables and constraints are required for the QP formulation. Additionally, for practical circuits the storage requirements are lower for the QP formulation. The smaller problem sizes for the QP formulation were shown to be a consequence of representing the nets as points. When represented as points, the net variables can be expressed in terms of the cell positions and eliminated from the formulation. The elimination of the net variables was not possible for the LP formulation.

The optimization problem sizes and storage requirements do not necessarily make the LP formulation unattractive. Empirical results by other researchers have indicate that the LP formulation may result in better final placements [37]. This improvement in quality stems from the observation that the LP formulation considers all nets equally during the minimization of wire length and better represents the wiring strategies actually used when routing a circuit. The result is two different relative placement formulations, each with advantages and disadvantages. By considering both formulations, the tradeoff in quality versus computational effort may be investigated.

In the next chapter, we consider an interior point method as an efficient solution methodology for both the QP and LP formulations presented in this chapter.

Chapter 4

An Interior Point Method for Relative Placement

In this chapter, we consider a primal-dual interior point method [31, 45, 46] suitable for solving both the QP and LP relative placement formulations. We consider an interior point method for several reasons: (i) it has a worst-case polynomial-time complexity, (ii) it is simple to implement, and (iii) it exhibits effective solution times on large and sparse optimization problems. These characteristics make an interior point method attractive since the relative placement formulations result in large and sparse optimization problems.

As previously illustrated in Chapter 3, the QP and LP formulations presented are similar to those used by other researchers. However, the interior point method represents a different solution methodology than those previously proposed. For instance, previous LP formulations [23, 49] used simplex-based approaches [30] to obtain a solution (Although, more recently, a primal-dual interior point method specifically for solving an LP formulation has been proposed [5]). Computationally, the interior point method was demonstrated to be superior to the simplex-based approaches. For the QP formulation, various solution methodologies have been proposed and rely on the constraints imposed on the problem. For wire length minimization, linear systems approaches have been used [25, 37]. However, these approaches are highly dependent on the structure of the constraints included in the problem formulation. Similarly, other QP approaches, such as eigenvector approaches [3, 20], also rely heavily on the formulation. When these approaches are used, careful consideration must be given when adding constraints into the formulation. Therefore, an interior point method suitable for the QP formulation offers more flexibility than the aforementioned linear system and eigenvector approaches. Other more flexible approaches for QP formulations have been based on Lagrangian multipliers [21, 39, 38, 42], where convex (and sometimes non-differentiable) constraints have been included into the formulation for various reasons (specifically, for inclusion of performance driven criteria). In the context of previously proposed approaches, the interior point method offers computational efficiency when compared to other methods (for instance, simplex-based approaches). In many cases, the interior point method allows additional constraints to be included into the formulation (specifically linear equality/inequality constraints and/or variable bounds) without affecting the solution methodology. Furthermore, the same interior point method is sufficient for both QP and LP formulations. Although the QP and LP formulations do not include performance driven constraints [23, 39] as presented, the interior point method is easily extensible to handle the inclusion of performance constraints used by many other researchers. Therefore, we consider the interior point method to be an efficient and more effective approach than many of those previously proposed. Additionally, it appears to be extensible to more difficult formulations.

This chapter is organized as follows. In Section 4.1, we describe the basic idea behind our primal-dual interior point method for solving the QP and LP relative placement formulations as previous described in Chapter 3. We demonstrate that the main computational burden of the interior point method is the solution of a sequence of large and sparse symmetric indefinite systems of linear equations. Therefore, in practical implementations efficient solution techniques for these systems of equations are required. In Section 4.2, we describe direct and iterative methods for solving these systems of equations. Iterative methods require more consideration than direct methods when implemented. Specifically, consideration is given to the generation of an effective *preconditioning matrix* when using an iterative method. Generation of a preconditioning matrix using established drop tolerance techniques is described in Section 4.3. In Section 4.4, numerical results are presented to demonstrate several aspects of the QP and LP relative placement formulations. The differences in the problem sizes and storage requirements for the QP and LP formulations are presented. These results confirm the observations made previously in Chapter 3, namely that the QP results in smaller problem sizes and require less storage than the LP formulation. Additionally, the QP formulation is solved more efficiently than the LP formulation. Finally, the potential benefits of iterative versus direct methods for solving the systems of equations arising during the interior point method are illustrated. A summary of this chapter is provided in Section 4.5.

4.1 A Primal-Dual Interior Point Method

As previously demonstrated, both relative placement formulations can be expressed in the form of the *primal* problem

$$\begin{array}{rrcl} \text{minimize} & \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} & & \\ \text{subject to} & \mathbf{A}\mathbf{x} + \mathbf{p} & = & \mathbf{b}, \\ & \mathbf{x} + \mathbf{s} & = & \mathbf{u}, \\ & \mathbf{x}, \mathbf{s}, \mathbf{p} & \geq & \mathbf{0}, \end{array} \tag{4.1}$$

which has the associated *dual* problem

$$\begin{array}{lrcl} \text{maximize} & \mathbf{b}^T\mathbf{y} + \mathbf{u}^T\mathbf{w} - \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} & & \\ \text{subject to} & \mathbf{A}^T\mathbf{y} + \mathbf{w} + \mathbf{r}_1 - \mathbf{Q}\mathbf{x} & = & \mathbf{c}, \\ & \mathbf{w} + \mathbf{r}_2 & = & \mathbf{0}, \\ & \mathbf{y} + \mathbf{r}_3 & = & \mathbf{0}, \\ & \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3 & \geq & \mathbf{0}, \end{array} \tag{4.2}$$

where $\mathbf{Q}$ is an $n \times n$ symmetric semi-definite matrix ($\mathbf{Q} \geq \mathbf{0}$), $\mathbf{A}$ is an $m \times n$ constraint matrix, $\mathbf{c}$ and $\mathbf{u}$ are n-vectors, $\mathbf{b}$ is an m-vector, and $\mathbf{x}$ is an n-vector of unknowns. For this primal-dual pair of problems, $\mathbf{Q} \neq \mathbf{0}$ implies the QP formulation whereas $\mathbf{Q} = \mathbf{0}$ implies the LP formulation of the relative placement problem.

The primal-dual interior point method is derived by applying a logarithmic barrier function to either (or both) the primal or dual problems in order to eliminate the non-negativity constraints. Assuming a solution that satisfies

$$\{(\mathbf{x}, \mathbf{s}, \mathbf{p}, \mathbf{y}, \mathbf{w}, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3) : \mathbf{x}, \mathbf{s}, \mathbf{p}, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3 > 0\} \tag{4.3}$$

is provided, the first order conditions for simultaneous optimality for the primal and dual barrier problems are:

$$\begin{aligned}
\mathbf{Ax}+\mathbf{p} &= \mathbf{b} \\
\mathbf{x}+\mathbf{s} &= \mathbf{u} \\
\mathbf{A}^T\mathbf{y}+\mathbf{w}+\mathbf{r}_1-\mathbf{Qx} &= \mathbf{c} \\
\mathbf{w}+\mathbf{r}_2 &= \mathbf{0} \\
\mathbf{y}+\mathbf{r}_3 &= \mathbf{0} \\
\mathbf{XR}_1 &= \mu\mathbf{e} \\
\mathbf{SR}_2 &= \mu\mathbf{e} \\
\mathbf{PR}_3 &= \mu\mathbf{e}
\end{aligned} \tag{4.4}$$

where e denotes a vector of ones, and $\mu > 0$ is the penalty parameter for the logarithmic barrier problems. The first two equations represent primal feasibility and the following three equations represent dual feasibility. The final three equations represent the μ-complementarity conditions, and in these equations an upper case letter denotes a diagonal matrix with components of the corresponding lower case vector on its diagonal.

Assuming an initial solution satisfying (4.3) is provided, one step of Newton's method is used to to find a solution closer the solution of the first order optimality conditions. This new solution becomes the current solution and the penalty parameter μ is reduced appropriately. This procedure is continued until μ is sufficiently close to zero. When μ is reduced sufficiently, it follows from the first order optimality conditions that the final solution is both primal feasible, dual feasible, and optimal for the primal-dual pair of problems in (4.1)–(4.2).

The primal-dual algorithm is therefore an iterative procedure where each iteration requires application of Newton's method to determine a search direction for updating the solution. To determine the search direction, Newton's method requires the solution of the augmented equations (the derivation of this system of equations is provided in Appendix

A) given by

$$\begin{bmatrix} -\mathbf{E} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{F} \end{bmatrix} \begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{bmatrix} = \begin{bmatrix} \zeta_1 \\ \zeta_2 \end{bmatrix}. \tag{4.5}$$

The matrices $\mathbf{E}$ and $\mathbf{F}$ are symmetric positive definite matrices, $[\mathbf{s}_1, \mathbf{s}_2]^T$ represents the search direction, and $[\zeta_1, \zeta_2]^T$ is some right-hand side vector. This system of equations is sparse, symmetric and indefinite. The sparsity pattern of $\mathbf{E}$ is identical to the sparsity pattern of $\mathbf{Q}$ when $\mathbf{Q} \neq \mathbf{0}$, otherwise it is diagonal. In both cases, $\mathbf{F}$ is diagonal. The solution of the augmented equations represents the main computational burden of each iteration of an interior point method. Efficient solution techniques for the augmented equations are required for efficient implementations of an interior point method. Further details regarding interior point methods, including methods for reducing the penalty parameter, termination criteria, and so forth can be found in [45].

4.2 Solving the Augmented Equations

Since each iteration of the interior point method is dominated by the solution of the augmented equations, it is necessary to consider efficient methods for their solution. We briefly note that it is not necessary to work with the augmented equations directly. We can either eliminate $\mathbf{s}_1$ from (4.5) and solve the reduced system of equations

$$(\mathbf{A}\mathbf{E}^{-1}\mathbf{A}^T + \mathbf{F})\mathbf{s}_2 = \zeta_2 + \mathbf{A}\mathbf{E}^{-1}\zeta_1 \tag{4.6}$$

for $\mathbf{s}_2$, or eliminate $\mathbf{s}_2$ and solve the reduced system of equations

$$(\mathbf{A}^T\mathbf{F}^{-1}\mathbf{A} + \mathbf{E})\mathbf{s}_1 = \mathbf{A}^T\mathbf{F}^{-1}\zeta_2 - \zeta_1 \tag{4.7}$$

for $\mathbf{s}_1$. Both of these reduced systems of equations are symmetric and positive-definite, and can be solved using any appropriate direct or iterative method [19]. In many cases, the reduced systems of equations suffer more fill than the augmented equations during a matrix factorization. Moreover, it is necessary to explicitly form the reduced systems of equations, If one uses the reduced system of equations in (4.6), it is necessary calculate the inverse of $\mathbf{E}$ which may not be easy when it has a non-diagonal sparsity pattern. If one uses the reduced system of equations in (4.7), the resulting matrix has a sparsity pattern identical to that of $\mathbf{A}^T\mathbf{A}$. For the relative placement problem, this matrix is very dense (totally dense at the beginning of the algorithm) since the first moment constraints become dense columns in the matrix $\mathbf{A}^T$. Hence, we prefer to work directly with the augmented equations.

4.2.1 Direct Methods

Typically, the augmented equations are solved using direct factorizations. Since the system is symmetric indefinite, a Bunch-Parlett factorization [4] has been advocated [10]. Although such a factorization is guaranteed to exist and is stable, the pivot order cannot be computed a priori, since the pivot choice is based on numerical values. Thus, the disadvantage of a Bunch-Parlett factorization is the large overhead required to implement the approach. Recently, it has been shown [44, 46] that an appropriate selection of an a priori pivot order is possible such that an $\mathbf{LDL}^T$ factorization of the augmented equations exists. In this case, $\mathbf{D}$ remains diagonal and non-singular, but contains both positive and negative values on its diagonal. Moveover, since the matrix $\mathbf{K}$ is *quasi-definite* [44], the $\mathbf{LDL}^T$ factorization exists under any symmetric pertubation. An ordering scheme such as minimum degree ordering [13] may be used to reduce the fill which occurs during the factorization process.

4.2.2 Iterative Methods

Generally, direct factorizations tend to perform well on small problems, or larger problems which do not incur large amounts of fill during the factorization. For large problems, and for those where fill becomes substantial, it is necessary to consider iterative methods. Iterative methods offer several potential benefits versus direct methods, namely that storage requirements are typically lower than for direct methods and solution times are often lower (due to the inexact nature of the resulting solution). However, one disadvantage of iterative methods is the failure to converge to an accurate solution.

An important aspect of an iterative method is the ability to determine a preconditioning matrix that closely approximates the original matrix. That is, we want to compute an approximate $\mathbf{LDL}^T$ factorization such that

$$\mathbf{K} = \mathbf{LDL}^T + \mathbf{R}, \tag{4.8}$$

where $\mathbf{R}$ is an error matrix. The approximate factorization should be easier to compute and require less storage space than a direct factorization of $\mathbf{K}$. Good preconditioners tend to significantly reduce the iteration counts for iterative solvers. For interior point methods and the augmented equations, iterative solvers have not been widely applied. SYMMLQ [33] has been proposed for solving the augmented equations arising at each iteration of an interior point method [14]. In this work, several methods for generating preconditioners were proposed. However, in all cases the preconditioners were formed by using a Bunch-Parlett factorization on a simpler matrix which closely approximated the original matrix. A positive definite preconditioner (a positive definite preconditioner is required by SYMMLQ) was then obtained by replacing the eigenvalues of the Bunch-Parlett factorization with their absolute values. Level of fill precondioners [6] have also been proposed for solving the augmented equations at each iteration of an interior point method designed specifically for linear programs [5]. In [5], rather than using SYMMLQ, the iterative solver BiCGSTAB [43] is proposed. The advantage of BiCGSTAB is that a positive definite preconditioner is not required, since BiCGSTAB is designed for non-symmetric matrices.

Motivated by the a priori orderings and $\mathbf{LDL}^T$ factorizations introduced in [44, 46], we consider generating incomplete $\mathbf{LDL}^T$ factorizations directly from the original quasi-definite matrix without resorting to a complete Bunch-Parlett factorization on a matrix approximation. We consider generating preconditioners by retaining (or rejecting) entries in $\mathbf{L}$ based on numerical values. Preconditioners generated in this manner are referred to as *drop tolerance* preconditioners [6] and tend to yield more accurate approximations than those based on level of fill. Furthermore, the preconditioning technique is applicable for the augmented equations arising for both QP and LP optimization problems. As an iterative solver, we follow the approach in [5] and use BiCGSTAB since a positive definite preconditioner is not required.

One important issue which must be addressed is the existence of an incomplete $\mathbf{LDL}^T$ factorization. That is, zero pivots must be avoided due to the dropping of terms during the factorization process since zero pivots result in failure of the method. We illustrate how this issue may be resolved in the next section.

4.3 Drop Tolerance Preconditioning

We now consider incomplete factorizations of the augmented equations based on drop tolerance techniques, as well as techniques for avoiding zero pivots during the incomplete factorization. Recall we are interested in computing an approximate $\mathbf{LDL}^T$ factorization of the augmented equations such that

$$\mathbf{K} = \mathbf{LDL}^T + \mathbf{R}, \tag{4.9}$$

where $\mathbf{R}$ is an error matrix. Drop tolerance preconditioning refers to the retention (or rejection) of nonzero entries in the approximate factorization based on their numerical value. At the p-th stage of the elimination process, the entire column p of $\mathbf{L}$ is calculated. If any entry in column p is small compared to the current diagonal entries it is dropped, otherwise it is kept. That is, entry l^p_{jp} is dropped if

$$(l^p_{jp})^2 < \epsilon \mid d^p_{pp} d^p_{jj} \mid, \tag{4.10}$$

where d^p_{pp} and d^p_{jj} are the current diagonal entries in the matrix $\mathbf{D}$ at the p-th stage of the elimination, and ϵ is a small positive number. For interior point methods, where the augmented equations change at each iteration, it is necessary to update the data structures for drop tolerance preconditioning at each iteration. However, due to the increased sparsity of the incomplete factorization, some of the additional effort to update the data structures is hopefully offset by the reduced factorization time.

4.3.1 The Augmented Equations and Zero Pivots

Positive Definite Matrices

We briefly consider the case when the matrix $\mathbf{K}$ is symmetric positive definite. One difficulty that may be encountered when generating an incomplete $\mathbf{LDL}^T$ factorization for a positive definite matrix is the loss of positive definiteness due to the dropping of terms. A positive definite preconditioner is required, since for symmetric positive definite systems the preconditioned conjugate gradient algorithm is the iterative solver of choice

[19], and this algorithm requires a positive definite preconditioner.

Fortunately, for positive definite systems, it is possible to guarantee the preconditioner remains positive definite by adding fractions of the absolute values of dropped terms to the diagonals of $\mathbf{D}$ during the incomplete factorization [1]. Let l^p_{jp} denote the entry which is being dropped from column p of the incomplete factorization and introduce the quantities

$$c_{pp} = \left(\frac{d^p_{pp}}{d^p_{jj}}\right)^{1/2} \mid l^p_{pj} \mid \quad \text{and} \quad c_{jj} = \left(\frac{d^p_{jj}}{d^p_{pp}}\right)^{1/2} \mid l^p_{kj} \mid . \tag{4.11}$$

By modifying these diagonal entries according to the rule

$$d^p_{pp} \leftarrow d^p_{pp} + c_{pp} \quad \text{and} \quad d^p_{jj} \leftarrow d^p_{jj} + c_{jj}, \tag{4.12}$$

the incomplete factorization will remain positive definite. Although promising, it has been observed that when many terms are dropped, the amount added to the diagonals is often an overestimation of the amount actually required to ensure positive definiteness [6], leading to an increase in the iteration count of the iterative solver. A more heuristic approach that performs well in practice is to simply abort the factorization process when a negative or zero pivot is encountered, scale the diagonal elements and re-attempt to compute the incomplete factorization using the scaled matrix [6].

Quasi-Definite Matrices

We now consider the case when $\mathbf{K}$ is a quasi-definite matrix. It should be observed that the matrix $\mathbf{D}$ contains exactly n negative entries and m positive entries (when $\mathbf{E}$ is an $n \times n$ matrix and $\mathbf{F}$ is an $m \times m$ matrix). For an incompete $\mathbf{LDL}^T$ factorization, we show that the same property can be guaranteed, implying that no zero pivots occur during the factorization process. To do so, we require the following assumption.

Assumption 4.1 *For a symmetric positive definite matrix, zero and negative pivots can be avoided by modifying the diagonal elements during an incomplete factorization. Furthermore, the resulting incomplete factorization will be positive definite.*

We note that this assumption is trivial from the discussion of positive definite matrices in the previous section. We now consider the case of a quasi-definite matrix.

Proposition 4.1 *For a quasi-definite matrix, zero pivots can be avoided by appropriate modification of the diagonal elements during an incomplete factorization.*

Proof Consider the elimination of the first column of a quasi-definite matrix $\mathbf{K}$ during a direct $\mathbf{LDL}^T$ factorization. The result is as follows:

$$\begin{bmatrix} -\mathbf{E} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{F} \end{bmatrix} = \begin{bmatrix} -e_1 & -e_3^T & a_1^T \\ -e_3 & -\mathbf{E}_2 & \mathbf{A}_2^T \\ a_1 & \mathbf{A}_2^T & \mathbf{F} \end{bmatrix} \tag{4.13}$$

$$\begin{aligned} &= \begin{bmatrix} 1 & 0 & 0 \\ e_3/e_1 & \mathbf{I} & 0 \\ -a_1/e_1 & 0 & \mathbf{I} \end{bmatrix} \\ &\times \begin{bmatrix} -e_1 & 0 & 0 \\ 0 & -(\mathbf{E}_2 - e_3 e_3^T/e_1) & \mathbf{A}_2^T - e_3 a_1^T/e_1 \\ 0 & \mathbf{A}_2 - a_1 e_3^T/e_1 & \mathbf{F} + a_1 a_1^T/e_1 \end{bmatrix} \\ &\times \begin{bmatrix} 1 & e_3^T/e_1 & -a_1^T/e_1 \\ 0 & \mathbf{I} & 0 \\ 0 & 0 & \mathbf{I} \end{bmatrix}. \end{aligned} \tag{4.14}$$

If we consider the lower diagonal submatrix $\mathbf{F} + a_1 a_1^T/e_1$, any terms in the rank 1 update $a_1 a_1^T/e_1$ can be dropped with the resulting submatrix remaining positive definite since $\mathbf{F}$ is positive definite. Consider the upper diagonal submatrix (ignoring the negative sign) $\mathbf{E}_2 + e_3 e_3^T / e_1$. Since this submatrix is positive definite, performing diagonal modifications will keep final incomplete factorization positive definite. Finally, any terms in the submatrix $\mathbf{A}_2 - a_1 e_3^T / e_1$ can be dropped without jeopardizing the incomplete factorization.

The result of the incomplete elimination is that the lower right submatrix remains quasi-definite. Thus, by induction, the elimination process can be continued for the first n columns without encountering zero pivots. The last m steps of the factorization require an incomplete factorization of a positive definite matrix (namely $\mathbf{F}$ updated by a series of rank 1 updates). This incomplete factorization can also proceed successfully using diagonal modifications.

4.4 Numerical Results

We now consider the effectiveness of the interior point method for solving the QP and LP relative placement formulations. The intention is to illustrate several differences in the two relative placement formulations, namely the resulting optimization problem sizes and the storage requirements. Computational times are also presented to demonstrate the reduced computational effort required when the QP formulation is used rather than the LP formulation. Finally, the benefits of using an iterative versus a direct method for solving the augmented equations are illustrated.

In producing the results, the following settings were used for the interior point method. In all cases, the interior point method was terminated when the primal and dual feasibility norms dropped below the threshold of 10^{-4} and when more than 2 significant digits of agreement were obtained between the primal and dual objective function values. Although these settings are not stringent (when compared to typical setting used on other types of optimization problems [45]), experimentation showed they were more than ac-

Circuit	QP Form.			LP Form.		
	Rows	Cols	$nz(\mathbf{K})$	Rows	Cols	$nz(\mathbf{K})$
Circuit1	2	833	11313	5422	2769	33203
Circuit2	2	3014	45468	18733	9096	114809
Biomed	2	6417	66687	28776	17609	187149
Industry1	2	2271	31357	13067	7157	81568
Industry2	2	12142	158720	69749	38124	435429
Industry3	2	15059	212225	124342	58723	740661

Table 4.1: Problem sizes and storage requirements for relative placement formulations.

ceptable for the relative placement problem (more stringent settings did not change the resulting cell positions significantly). For the iterative method, the following settings were used. The drop tolerance was initially set to 10^{-3} and the BiCGSTAB iterations were terminated when (i) the residual error dropped below 10^{-6} or (ii) when more than 100 iterations were required. In the cases where more than 100 iterations were required, the iterative solver was considered to have failed. However, rather than stopping the entire algorithm, the drop tolerance was reduced by a factor of 10, resulting in a more accurate preconditioner (fewer dropped terms), and the iterative solver was re-executed. This technique guaranteed convergence of the iterative method since reduction of the drop tolerance would eventually result in a direct factorization (if required). Failure of the iterative method was not considered acceptable for the relative placement problem, and therefore convergence was considered necessary regardless of the cost.

In Table 4.1, the optimization problem sizes are shown for the QP and LP relative placement formulations with the entire placement area taken as the only region. The number of variables, constraints and nonzeros in the augmented equations are shown in Table 4.1. As expected, the results presented in Table 4.1 indicate that the QP formulation results in optimization problems with significantly fewer variables and constraints than the equivalent LP formulation. In addition, the nonzeros in the matrix $\mathbf{K}$ (which measures the number of nonzeros in $\mathbf{Q}$ and $\mathbf{A}$, since $\mathbf{K}$ is composed from these two

Circuit	Dir. Meth.		Iter. Meth.		Imp. (Iter/Dir)	
	nz(L)	Time	*nz*(L)	Time	*nz*(L)	Time
Circuit1	18899	2.78	5671	3.20	0.30	1.15
Circuit2	142357	37.28	21055	22.70	0.15	0.61
Biomed	56691	57.24	28404	71.46	0.50	1.25
Industry1	42001	17.32	14217	22.04	0.34	1.27
Industry2	440143	296.74	71073	233.66	0.16	0.79
Industry3	1745903	1375.90	103692	334.80	0.06	0.24

Table 4.2: Relative placement results for the QP formulation.

matrices) indicates that the QP requires less storage space than the equivalent LP formulation. These observations comfirm the predictions made previously in Chapter 3 about the differences in the two relative placement formulations.

The performance of the interior point method for solving the QP relative placement formulation is illustrated in Table 4.2. In Table 4.2, we present the number of nonzeros in the matrix factorization L and the total computational time required to solve the QP relative placement formulation. These statistics are presented when both direct and iterative methods are used to solve the augmented equations. For the iterative method, the number of nonzeros in L presented in Table 4.2 represents the maximum number of nonzeros required over all interior point iterations. These results indicate that the iterative method typically requires significantly less storage (always less than 50 percent) for the matrix factorization when compared to the direct method. The computational times are not as good. In several cases, the direct method out performs the iterative method in terms of computational effort. However, for the larger circuits *industry2* and *industry3*, the presented results favour the iterative method.

In Table 4.3, the same statistics are presented to illustrate the performance of the interior point method for solving the LP formulation of the relative placement problem. Once again, the iterative method requires less storage space than the direct method, although the savings (expressed as the ratio) are not as significant as for the QP formu-

Circuit	Dir. Meth.		Iter. Meth.		Imp. (Iter/Dir)	
	$nz(\mathbf{L})$	Time	$nz(\mathbf{L})$	Time	$nz(\mathbf{L})$	Time
Circuit1	42396	23.89	27078	53.13	0.64	2.22
Circuit2	225524	256.11	120604	275.76	0.53	1.07
Biomed	175570	548.74	155847	507.08	0.89	0.92
Industry1	94831	109.69	74129	265.27	0.78	2.42
Industry2	690201	2706.33	398046	1603.32	0.58	0.59
Industry3	2377871	4243.13	776025	2037.17	0.33	0.48

Table 4.3: Relative placement results for the LP formulation.

lation. For the LP formulations, the direct method is preferable to the iterative method in several situations (namely the smaller problems *circuit1* and *industry1*) when computational effort is considered. However, for the larger problems (especially for the largest circuits *industry2* and *industry3*), where savings in computational effort are most crucial, the iterative method is preferable.

From both Tables 4.2 and 4.3, we make several observations to compare the QP and LP formulations. In all cases, the QP formulation requires less storage and exhibits lower solution times than the LP formulation. This is significant, since it is necessary to solve a sequence of relative placement problems during the constructive phase of the placement heuristic. However, no statements regarding the quality of the final placements can be made since the solution to the relative placement problem is not sufficient for creating a legal placement. Therefore, it is not necessarily feasible to ignore the LP formulation at this point.

4.5 Summary

In this chapter, a primal-dual interior point method suitable for solving the relative placement formulations previously described in Chapter 3 has been presented. The computational burden of the interior point method was shown to be the solution of the augmented equations which arise at each iteration of the interior point method. Both direct and iterative methods were described for solving the augmented equations. Iterative methods require the generation of a good preconditioning matrix. Preconditioning using drop tolerance techniques have been described, and a method for avoiding zero pivots based on modifying diagonal elements during the incomplete factorization was presented.

Numerical results on our set of test circuits indicate that the QP relative placement formulation results in smaller optimization problems requiring less storage space than the equivalent LP formulation. Moreover, regardless of the method used to solve the augmented equations, the QP formulation was solved with less computational effort than the equivalent LP formulation of the relative placement problem.

Numerical results also indicated that iterative methods require significantly less storage space than direct methods for solving the augmented equations. The performance of the iterative method was less definitive in terms of computational effort (when compared to the benefits in the storage requirements). In several cases the direct method was preferable to the iterative method. However, for the larger test circuits, which result in larger optimization problems and consume a significant amount of computational effort to solve, the iterative method was preferable. Therefore, we consider the application of the iterative method a promising approach.

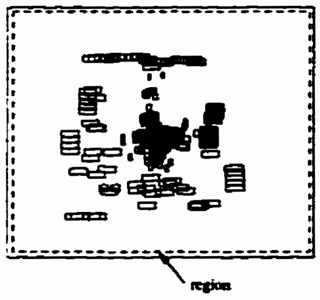
Although the numerical results presented in this chapter favour the QP formulation, it is not sufficient to ignore the LP formulation. No statements regarding the quality of the placement were made in this chapter since a single relative placement is not sufficient for the generation of a legal placement. Therefore, the LP formulation must still be considered to invesigate any potential tradeoffs in quality versus speed. In the following chapters, we continue with a description of the remaining components of the constructive placement method to further explore the QP and LP relative placement formulations, as well as the overall constructive placement method itself.

Chapter 5

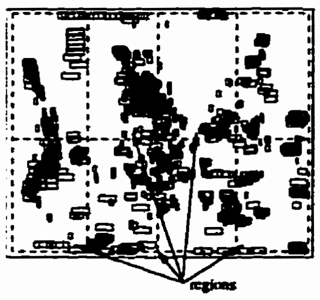
Partitioning the Placement Area

In Chapter 3, the relative placement problem was formulated under the assumption that the placement area had been divided into a number of disjoint regions and each free cell could only be positioned within one of these regions. These assumptions lead to a set of inequality constraints involving the cell positions and variables bounds on the acceptable cell positions.

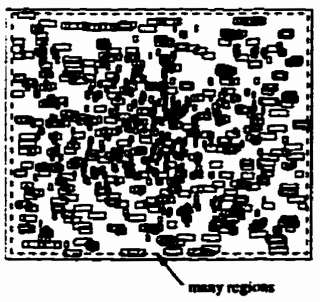
Several relative placements for *circuit1* are presented in Figures 5.1(a)-(c) to illustrate the necessity of dividing the placement area and the cells (the relative placements were obtained using the **QP** formulation in all cases. Similar results are obtained with the **LP** relative placement formulation). Figure 5.1(a) illustrates the relative placement obtained when cells were only required to be positioned within the placement area (that is, the placement area was "divided" into one region). Although cells are allowed to fall anywhere within the placement area, there is a tendency for the cells to group towards the middle of the region. The grouping of cells is a result of the minimization of wire length which naturally pulls connected cells together. The result is substantial cell overlap and poor utilization of the available placement area. In Figure 5.1(a), the movement of some cells away from the middle of the region is due to connections to the fixed I/O pads located



(a) Typical relative placement with one region.



(b) Typical relative placement with eight regions.



(c) Typical relative placement with > 100 regions.

Figure 5.1: Examples of relative placements.

around the perhiphery of the placement area (which pull connected cells outward) and the first moment constraints (which balance cells throughout the region). The cell positions provided by solution of a single relative placement problem are not sufficient for the creation of a legal placement since cells are not necessarily near to their final positions.

Figure 5.1(b) illustrates the relative placement obtained when the placement area is divided into eight regions. Clearly, area utilization is improved and the cells are distributed more evenly throughout the placement area. Overlap is prevented for cells assigned to different regions. Although cells still group together, there is more movement outward due to the I/O pads, the first moment constraints, and external connections to cells within other regions. Finally, the relative placement obtained when the placement area is divided into more than 100 regions is illustrated in Figure 5.1(c). Area utilization is very good and the cells are distributed evenly throughout the entire placement area. Cell overlap is also substantially lower. The cell positions obtained after division of the placement area into many regions are useful for determining good cell positions in an initial legal placement. Hence, it is necessary to have a progression of relative placements, starting from Figure 5.1(a), passing through the result shown in Figure 5.1(b), and terminating with the result shown in Figure 5.1(c). Obviously, the computational effort required to solve a sequence of relative placement problems (and obtain a relative placement similar to that illustrated in Figure 5.1(c)) is larger than that required to solve only a single relative placement problem. However, as we have illustrated, the additional computational effort is justified since it provides more useful (better) information for creating initial placements. The necessity of dividing the placement area in order to obtain "useful" relative cell positions has been observed by many researchers [23, 25, 37].

To divide the placement area and the cells in an intelligent manner, we introduce the concept of *circuit partitioning* [2, 3, 34]. In general terms, circuit partitioning involves dividing the cells of a circuit into a small number of disjoint blocks while minimizing a defined measure of connectivity between cells within different blocks. In other words, partitioning attempts to divide the circuit into relatively independent pieces. In the context of circuit placement, partitioning the cells within the placement area corresponds to dividing the cells into disjoint regions where each region has an underlying two-dimensional geometry. Therefore, circuit partitioning accomplishes several objectives, namely (i) overlap between cells within different regions is prevented and (ii) area utilization is improved by forcing cells into less utilized portions of the placement area. Furthermore, these objectives are accomplished in an intelligent manner since circuit partitioning naturally minimizes the connections between cells in different regions implying that wire lengths will be kept short (since nets will require wiring cells within only one or a few nearby adjacent regions).

This chapter is organized as follows. In Section 5.1, we describe circuit partitioning in greater detail. Its application in the context of cell placement is also described to illustrate how the two-dimensional properties of the placement area are taken into account. Our proposed approach for introducing circuit partitioning into the constructive method for circuit placement is described in Section 5.2. The relationship and interaction of the relative placement and circuit partitioning problems is also described. The intention is to illustrate how a progression of relative placements, such as those previously illustrated in Figures 5.1(a)-(c), are obtained. Termination criteria for the relative placement and partitioning problems is also described. Finally, a summary is provided in Section 5.3.

5.1 Circuit Partitioning

5.1.1 General Description

Circuit partitioning can be described as follows. Assume that we are given a set of c cells interconnected by n nets. It is assumed that each net connects to two or more cells. The

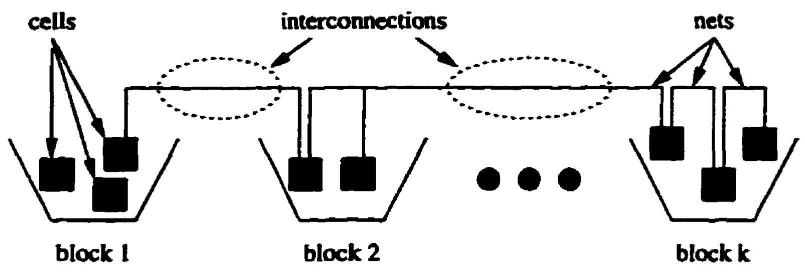


Figure 5.2: Illustration of circuit partitioning.

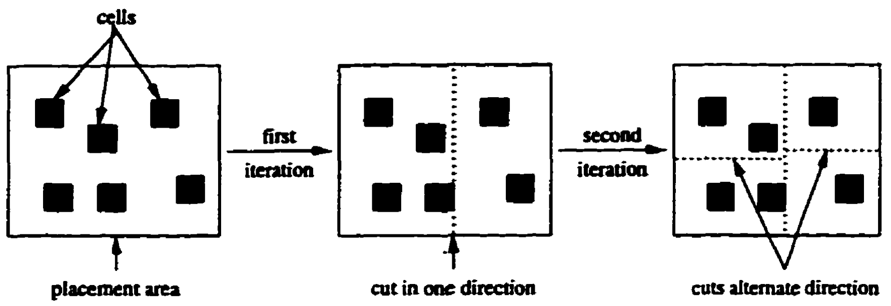
objective in circuit partitioning is to divide the cells into a small number of disjoint sets of cells denoted by B_1, B_2, $\cdots$, B_k while minimizing a measure of connectivity between cells located in different blocks. This is illustrated in Figure 5.2. Circuit partitioning arises in many applications other than VLSI design. For instance, partitioning naturally arises in any application where it is desireable to divide a set of interconnected items into a small number of relatively independent blocks. Examples of such applications are manufacturing, parallel algorithms, and so forth [34].

Circuit partitioning is known to be NP-hard [12], and it is therefore necessary to resort to partitioning heuristics to obtain near optimal partitions in reasonable time. Partitioning heuristics may be broadly classified as either constructive or iterative improvement methods. Constructive methods generate partitions directly from the circuit description. Typical methods are based on spectral approaches [3] or network algorithms [47]. These methods are generally computationally efficient and produce "globally" good partitions since the partitions are obtained directly from the circuit description and all interconnections are considered simultaneously. Therefore, the resulting partitions tend to have cells placed into the correct blocks. However, due to various approximations required to implement these heuristics, the partitions are not generally globally optimal.

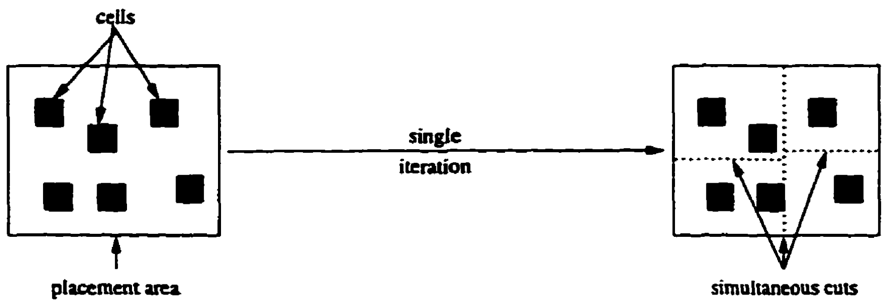
Iterative improvement methods, unlike their constructive counterparts, obtain a partition of cells by attempting to improve on an initial partition. Typically, iterative improvement methods for circuit partitioning are based on cell interchange heuristics [2, 24, 34, 35] which attempt to find an improved partition by moving cells from one block to another block. The advantage of cell interchange heuristics is their efficient implementations. However, cell interchange heuristics are local in nature and tend to get trapped into a locally optimal partition, often far from a globally optimal partition. Several different techniques are used to avoid locally minimal partitions. Often, cell interchange heuristics are performed from multiple random initial partitions with the best result selected as the final partition [17]. Meta-heuristics, such as Tabu Search [15], have also been proposed as a means of guiding cell interchange heuristics out of locally optimal partitions [2]. Another technique is to start with a single (or a few) good initial partitions produced by a constructive method [2]. Since the initial partitions are globally good, cell interchanges on a single (or a few) good initial partitions tend to avoid locally minimal partitions [2, 17]. Two-phased partitioning heuristics are also used. In these approaches, the circuit is initially *clustered* (condensed) into a smaller circuit. Partitioning is applied to the condensed circuit from a few random initial partitions. Subsequently, the initial results from the first phase of partitioning are flattened and used as the initial partitions for cell interchanges on the original circuit. Finally, it is possible to combine all of these techniques, where good initial partitions are created, but enhanced search heuristics are still applied to guide the cell interchange heuristic out of locally minimal partitions. Extensive investigations of these techniques have been done [2].

5.1.2 Partitioning and Placement

In the context of cell placement, where partitioning is used to divide the placement area, each block of a partition is associated with an underlying region of a specified geometry. Therefore, it is necessary to consider the underlying placement geometry when using



(a) Sequential bisection of the placement area.



(b) Quadsection of the placement area.

Figure 5.3: Methods for partitioning the placement area.

partitioning for the division of the placement area and the cells.

Previous approaches to partitioning a two-dimensional placement area have relied on two-way circuit partitioning (bisection) [8, 25, 37]. In this approach, each region of the placement area is divided into two halves of nearly equal size by first partitioning the cells into two blocks and subsequently applying a cut in either the x or the y direction. At subsequent iterations, the cuts alternate in the x and y directions. This approach is illustrated in Figure 5.3(a). In performing the bisectioning, the measure of connectivity is taken as the number of *cut* nets, where a net is considered to be cut if it connects cells in different blocks (crosses the cut line).

Bisection suffers from the disadvantage that it takes a one-dimensional view of the

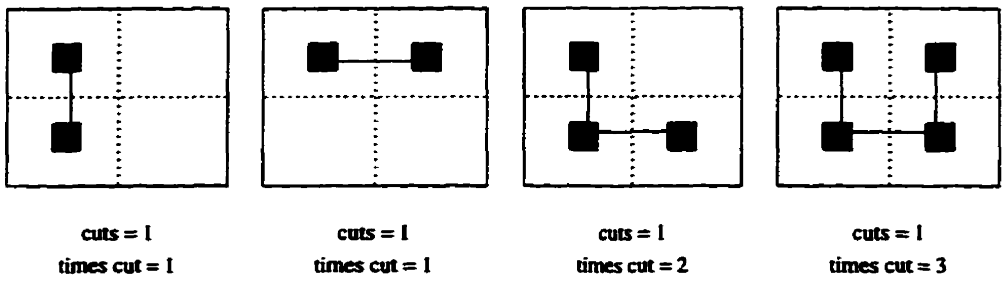


Figure 5.4: Accounting for different measures of connectivity.

problem since only one cut in either the x or the y direction is introduced at each stage. Since cell placement is inherently two-dimensional, four-way partitioning (quadsection) has been proposed [40] and is illustrated in Figure 5.3(b). Rather than applying the cuts in the x and y directions at separate iterations, all cuts are applied simultaneuously. The result is a division of any area into four regions simultaneously. Although quadsection can be accomplished by applying several stages of bisectioning, it has been demonstrated [2, 17, 34, 35, 40] that quadsection gives better results than multiple bisections when four blocks are required. This is true for the general circuit partitioning problem where there is no underlying geometry and in the context of division of a placement area. Moreover, in addition to improving the quality of the partitions, quadsection also reduces the number of partitioning steps since the division of cells into disjoint blocks in accomplished in a more timely manner. This is important from a computational aspect, since reducing the number of partitioning steps reduces the total computational effort.

A difficulty with four-way partitioning of a placement area is that it is no longer sufficient to minimize the number of cut nets. This is illustrated in Figure 5.4 where several examples of a single net connecting cells in two or more blocks are presented. Let b represent the number of blocks in which a net has a connected cell. Whenever $b \geq 2$, the net is cut and therefore contributes one to the measure of connectivity. However, if the wiring of the net is considered, nets connecting cells in a large number of blocks corresponds to an increase in wire length during the routing since the cells requiring connection are physically further apart. Therefore, it would be more appropriate to measure the *number of times a net is cut* which is equivalent to minimizing the number of blocks in which a net connects to a cell. This measure of connectivity is also illustrated in Figure 5.4 and has been incorporated into multi-way cell interchange heuristics [35, 40].

5.2 A Proposed Strategy

5.2.1 Incorporating Circuit Partitioning

We now consider how circuit partitioning is incorporated into the constructive placement method. The approach is illustrated in Figure 5.5. In Figure 5.5, it is assumed that the cell positions from the solution of the most recent relative placement problem are available. Each region currently within the placement area is selected one at a time and a list of cells within the region is composed. If the region is suitable for partitioning (we explain why a region may be unsuitable for partitioning in a subsequent section), the cells within the region are partitioned into four blocks, otherwise another region is selected.

Once a region is selected, the cells within the region are sorted according to their x and y positions. An initial partition for each region is created by partitioning the cells into two blocks of equal size based on the sorted y positions. Subsequently, these two blocks are further divided into two blocks each (resulting in a total of four blocks) based on the sorted x positions. Despite the grouping of cells towards the middle of the region, cells have moved towards their final positions. Hence, good initial partitions are possible (introducing cuts into the sorted lists). Furthermore, the initial partitions are generated quickly since the creation of the initial partitions is based on sorting cells which has a complexity of $O(c \log c)$, where c is the number of cells. Taken in the context of the

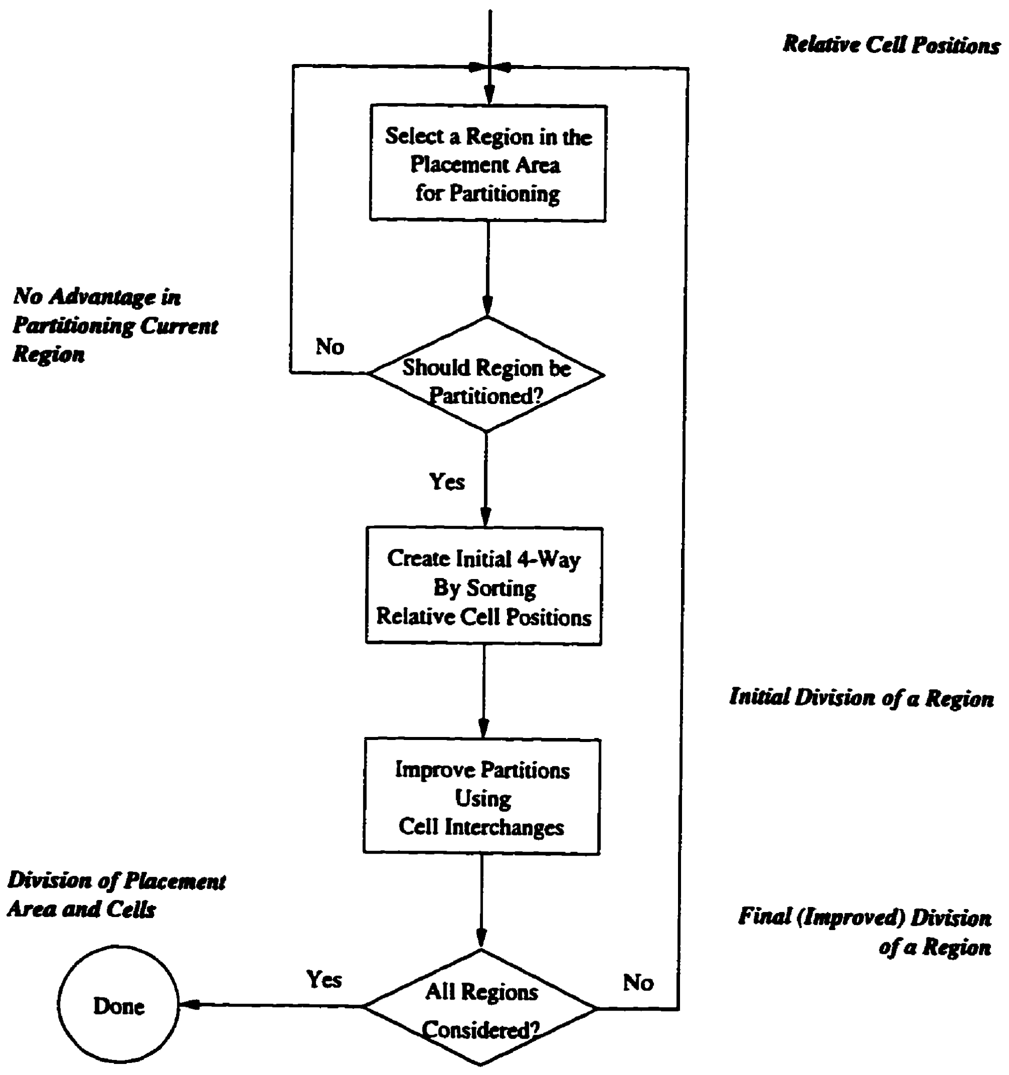

Relative Cell Positions
Select a Region in the Placement Area for Partitioning
No Advantage in Partitioning Current Region
No
Should Region be Partitioned?
Yes
Create Initial 4-Way By Sorting Relative Cell Positions
Initial Division of a Region
Improve Partitions Using Cell Interchanges
Division of Placement Area and Cells
Final (Improved) Division of a Region
Done
Yes
All Regions Considered?
No

Figure 5.5: Incorporating circuit partitioning into the placement heuristic.

previous description of partitioning methods, the creation of initial partitions based on sorting cell positions can be considered a constructive method for generating an initial partition.

Once the initial partition has been created, further improvement is still possible by considering the movement of cells from one block to another. This follows from the observation that partitioning the cells based on their relative cell positions corresponds to placing cut lines within the placement area near the middle of the region. Since this also corresponds to where cells tend to group during the relative placement, the division of cells close to the cut lines into their initial blocks may be arbitrary. To improve the initial partitions, a cell interchange heuristic [35] is used with the measure of connectivity counting the number of times each net is cut. We allow cells near the middle of the region (near the cut lines introduced in the x and y directions) to move between regions while (or locking) cells far from the cut lines into their initially selected partitions. Since cells within the current region under consideration may also have connections to cells outside of the region, terminal propagation [40] is also used to account for these external connections. The partitioning is accomplished efficiently, since the complexity of the cell interchange heuristic is $O(lpb(\log b + gl))$ per pass, where l is a level parameter (typically one or two) internal to the interchange heuristic, p is the number of pins in the circuit, b is the number of blocks and g is the maximum number of nets attached to any cell.

Finally, once each region within the current placement area has been considered and subdivided into a larger number of regions (with each new region containing fewer cells than the original). the circuit partitioning terminates. It follows from the description of the proposed approach that the incorporation of circuit partitioning into the constructive placement method follows the general partitioning technique of using a good initial placement which is subsequently improved using cell interchanges.

5.2.2 Relative Placement and Circuit Partitioning Interaction

The circuit partitioning required the solution of the relative placement problem as input for the creation of initial partitions. Similarly, the output of the circuit partitioning (a new division of the placement area and the cells) is used as input to the next relative placement problem (Recall the formulation of the relative placement problem in Chapter 3. The region information is used to revise or add constraints and variable bounds into the formulation such that a new relative placement of the cells can be found. We emphasize that the relative cell positions are found *simultaneously* once the relative placement formulation is revised using the partitioning information, and that it is *not* necessary to solve a separate relative placement problem for each region). Therefore, there is an interaction between the two problems which is illustrated in Figure 5.6. By performing several iterations of relative placement followed by circuit partitioning, a good distribution of cells throughout the placement area, such as that previously illustrated in Figure 5.1(a)-(c), is obtained.

It is necessary to develop an approach for terminating the relative placement and circuit partitioning iterations. As previously described, a region which is not appropriate for circuit partitioning may be selected. A region is not appropriate when the number of cells within the region drops below a small predefined threshold (for instance, 10 to 20 cells). In this situation, the region correponds to only a very small portion of the entire placement area and little improvement in area utilization and cell overlap is possible by partitioning the cells within region. Eventually, the circuit partitioning will reach a stage where the number of cells within every region has fallen below the threshold, and no region will be partitioned. Any further relative placements will produce identical results since the constraints and variable bounds will not change. Therefore, the relative placement and circuit partitioning iterations may be terminated.

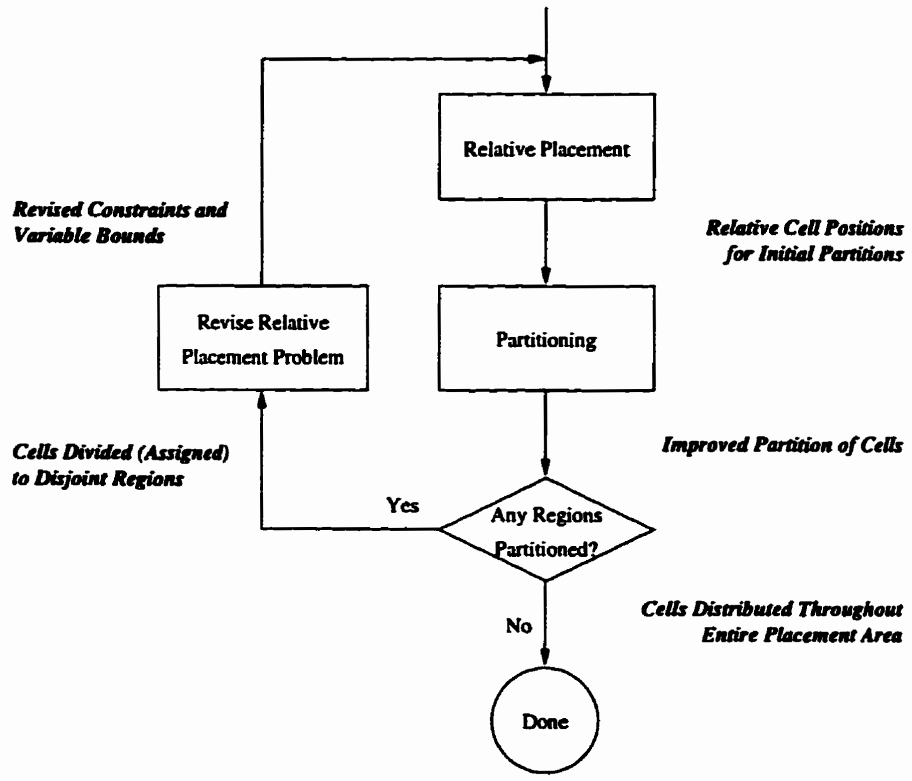

Relative Placement
Relative Cell Positions for Initial Partitions
Partitioning
Improved Partition of Cells
Any Regions Partitioned?
Yes
No
Cells Distributed Throughout Entire Placement Area
Done
Cells Divided (Assigned) to Disjoint Regions
Revise Relative Placement Problem
Revised Constraints and Variable Bounds

Figure 5.6: Relative placement and circuit partitioning interaction.

5.3 Summary

This chapter has illustrated that the solution of a single relative placement problem is not sufficient for providing useful information for the creation of an initial legal placement. This is a consequence of the grouping of cells towards the middle of the placement area after the solution of the initial relative placement problem. It is necessary to solve a sequence of relative placement problems where, at each iteration, the placement area is divided into more and more disjoint regions (with a subset of the cells restricted to positions within each region) using circuit partitioning. Several iterations of relative placement and circuit partitioning result in an even distribution of cells throughout the entire placement area, a reduction in cell overlap, and better utilization of the available placement area. In other words, cells are nearer to their desired positions, and therefore the final relative placement provides useful and reliable information for the creation of an initial legal placement.

Once the number of regions has become very large (and only a small number of cells are assigned to every region), the relative placement and circuit partitioning iterations terminate and a set of cell positions are available. Once the cell positions are "modified" such that cells are positioned within rows and any residual cell overlap is reduced, the constructive placement method is completed. In the next chapter, we consider how an initial legal placement is created using the relative cell positions.

Chapter 6

Initial Placement Construction

As previously described in Chapter 5, the relative placement and circuit partitioning iterations provide cell positions distributed throughout the entire placement area. However, these cell positions do not constitute a placement since the placement restrictions are not satisfied. There is still minor cell overlap and the cells are not positioned within the rows, as required. It is therefore necessary to "legalize" the placement by further adjusting the cell positions to satisfy the placement requirements. Once the repositioning of cells is accomplished, an initial legal placement is obtained and the constructive placement method is completed.

In this chapter, we consider the construction of an initial legal placement based on the cell positions provided by the relative placement and circuit partitioning iterations. In Section 6.1, we describe a simple heuristic for legalizing the placement by sorting the relative cell positions. Numerical results are presented in Section 6.2 to illustrate various aspects of the constructive placement method. Both the QP and LP relative placement formulations are considered. The computational effort required by the relative placement, circuit partitioning and legalization heuristics are presented. The quality of the initial placements are presented and judged by considering the estimates of wire length and the required placement areas. Additionally, we confirm through numerical results that the cell positions provided by a single relative placement are not sufficient for the creation of initial placements. Finally, a summary is provided in Section 6.3.

6.1 Legalization Using Relative Cell Positions

We propose the following simple, yet effective, approach for positioning cells within rows while removing cell overlap. Cells are sorted in ascending order according to their positions in both the x and y directions. Cells are assigned to rows based on the sorted y positions while keeping the row length nearly equal (implying a minimum width placement). Cells are allowed to move between rows in order to keep the row lengths approximately equal. Once a cell has been assigned to a row, its y position is updated to reflect its row assignment. Subsequently, the cells within each row are positioned adjacent to each other from left to right across the row based on the sorted x positions. As cells are shifted to remove overlap, the x positions of the cells are updated. Since the legalization heuristic is based on sorted cell positions, the complexity of the legalization heuristic is $O(c \log c)$, where c denotes the number of cells.

6.2 Numerical Results

The constructive placement method is completed once an initial legal placement is obtained. In this section, we consider the computational aspects of the constructive placement method and the quality of the initial legal placements. Additionally, we also justify the necessity of performing several relative placement and circuit partitioning iterations, rather than a single relative placement, in order to obtain cell positions which are "useful" for creating an initial placement. In presenting these results, we consider both the QP and LP formulations of the relative placement problem.

Circuit	Iter.	Time (CPU Secs)		
		Relative Place.	Circuit Part.	Legal.
circuit1	6	38.96	9.02	0.03
circuit2	6	232.93	92.57	0.12
biomed	7	716.64	171.85	0.26
industry1	6	135.23	53.36	0.08
industry2	7	2031.65	543.35	0.91
industry3	7	3293.77	710.30	0.89

Table 6.1: Computational effort with the QP relative placement formulation.

Circuit	Iter.	Time (CPU Secs)		
		Relative Place.	Circuit Part.	Legal.
circuit1	6	428.44	8.88	0.03
circuit2	6	2187.04	94.52	0.11
biomed	7	3970.78	158.91	0.27
industry1	7	1300.99	48.69	0.07
industry2	7	11854.62	505.36	0.85
industry3	7	27336.94	759.05	0.90

Table 6.2: Computational effort with the LP relative placement formulation.

6.2.1 Computational Effort

The computational effort required for the various components of the constructive method when the QP formulation of the relative placement problem is used are presented in Table 6.1. The number of relative placement and circuit partitioning iterations required to produce the relative cell positions used by the legalization heuristic are also presented. In Table 6.2, the same numerical results are presented when the LP formulation of the relative placement problem is used.

Several observations follow from the numerical results presented in Tables 6.1 and 6.2. In all cases, very few relative placement and circuit partitioning iterations (≤ 7 even on the largest test circuits) are required in order to spread the cells throughout the placement area. Regardless of the relative placement formulation, the circuit partitioning and legalization components of the constructive method consumes significantly less time than the solutions of the sequence of relative placement problems. Therefore, the "bottleneck"

Circuit	Solution Times (CPU Secs)		Improvement
	QP Form.	LP Form.	(QP/LP)
circuit1	48.01	437.35	0.1098
circuit2	325.62	2281.67	0.1427
biomed	888.75	4129.96	0.2152
industry1	188.67	1349.75	0.1398
industry2	2575.91	12360.83	0.2084
industry3	4004.96	28096.89	0.1425

Table 6.3: Total computational effort for initial placements.

of the constructive method is the solution of the relative placement problem.

Finally, it is possible to compare the computational effort required by the different relative placement formulations. The number of relative placement problems solved during the constructive method are comparable (equal on all test circuits with the exclusion of *industry1* where one additional iteration was required) for both the QP and LP formulations. As previously illustrated in Chapter 4, the QP formulation required substantially less time than the LP formulation when only one relative placement problem was solved. The results in Tables 6.1 and 6.2 again illustrate this observation over a sequence of relative placements. That is, the QP formulation results in substantially lower computational effort when compared the equivalent LP formulation.

6.2.2 Total Computational Effort and Quality of the Placements

The total computational times (taken as the summation of the times for the individual components presented in Tables 6.1 and 6.2) required to generate initial placements for each test circuit are presented in Table 6.3 for both relative placement formulations. Additionally, the ratio of the time required when using the QP formulation to that required when using the LP formulation is presented. As previously mentioned, the QP formulation requires significantly less computational effort than the equivalent LP formulation.

In Table 6.4, the length of the longest row in the initial placement for each test circuit

Circuit	Max. Row Length (μ m)		Improvement
	QP Form.	LP Form.	(QP/LP)
circuit1	6380	6270	1.0175
circuit2	12500	12540	0.9968
biomed	8368	8392	0.9971
industry1	4822	4810	1.0025
industry2	14344	14352	0.9994
industry3	28296	28280	1.0006

Table 6.4: Maximum row lengths for initial placements.

Circuit	HPWL (m)		Improvement
	QP Form.	LP Form.	(QP/LP)
circuit1	1.6154	1.5101	1.0697
circuit2	6.3227	5.8707	1.0770
biomed	4.0053	3.5557	1.1264
industry1	2.1200	1.9053	1.1127
industry2	29.3326	29.3229	1.0003
industry3	77.6508	73.1003	1.0622

Table 6.5: Wire length estimates for initial placements.

are presented as a measure of the required placement area. Row lengths are comparable regardless of the relative placement formulation used. This follows from the observation that the row lengths are a consequence of the legalization heuristic and have nothing to do with the relative placement formulation used.

Finally, estimates of wire lengths for the initial placements are presented in Table 6.5. As an estimate of the wire length, the half-perimeter wire length (HPWL) is used. Note that the LP formulation minimizes this estimate of wire length directly, whereas the QP formulation does not. Since rectilinear wiring is typically used in routing the circuit, the HPWL respresents an accurate estimate of the actual wire length. Additionally, other results presented in the literature (for example, [25, 41, 49]) typically used the HPWL for reporting placement results.

From Table 6.5, the LP formulation clearly produces better initial placements than the QP formulation when the HPWL is used as the estimate of wire length. This observation

Circuit	Time (CPU Secs)		Imp.	HPWL (m)		Imp.
	Sing.	Iter.	(Sing./Iter.)	Sing.	Iter.	(Sing./Iter.)
circuit1	6.48	48.01	0.13	1.8758	1.6154	1.16
circuit2	52.03	325.62	0.16	10.6007	6.3227	1.68
biomed	214.08	888.75	0.24	7.3746	4.0053	1.84
industry1	38.20	188.67	0.20	2.9814	2.1200	1.40
industry2	473.52	2575.91	0.18	47.8197	29.3326	1.63
industry3	774.87	4004.96	0.19	160.9700	77.6508	2.07

Table 6.6: Single versus Several Relative Placements.

agrees with those results obtained by other researchers [37]. However, these improved results come at the expense of significantly larger computational effort. Therefore, there is, as expected, a tradeoff in the quality of the initial placement versus the computational effort required to produce the initial placement.

6.2.3 Justification for Relative Placement and Circuit Partitioning

We now justify the necessity of performing several iterations of relative placement and circuit partitioning rather than using the cell positions provided by a single relative placement for the creation of an initial placement. In Table 6.6, we present numerical results from the creation of initial placements when the cell positions provided by the first relative placement problem were used as input to the legalization heuristic. In producing these results, the QP relative placement formulation was used (although similar results are obtained using the LP formulation). These results are compared with those obtained using the relative placement and circuit partitioning iterations (comparisons are made based on computational effort and wire length estimates since the row lengths were comparable in all cases).

The creation of initial placements using only one relative placement requires less computational effort than performing several iterations of relative placement and circuit partitioning. This is an obvious result, since less computational effort is necessary (only one relative placement problem is solved and no circuit partitioning is performed). However it is necessary to consider the qualities of the initial placements. Table 6.6 clearly illustrates that the resulting estimates of wire length are *substantially* worse than those obtained when several relative placement and circuit partitioning iterations are performed. As previously illustrated in Chapter 5, most cells group towards the middle of the placement area after the first relative placement problem (despite the tendency of some cells to move outward towards the periphery of the placement area due to connections to I/O pads). Using these initial cell positions as a "useful" indication of the desired cell positions in an initial placement is simply incorrect. The poorness of the wire length estimates presented in Table 6.6 confirm this observation. Hence, there is significant motivation for the additional computational effort required to perform several relative placement and circuit partitioning iterations in order to obtain a better placement.

6.3 Summary

In this chapter, a simple heuristic for creating an initial legal placement based on the relative cell positions provided by the relative placement and circuit partitioning iterations has been described. Since the relative cell positions provide a good indication of the general position for a cell, the initial legal placement was obtained by simply sorting the relative cell positions based on their x and y positions. The sorted y positions were used to assign cells to rows and the sorted x positions were used to order the cells (while removing overlap) within each row.

Numerical results were presented to illustrate the computational aspects of the constructive method. Regardless of the relative placement formulation, the circuit partitioning and legalization heuristics required only a fraction of the total computational effort. In other words, the relative placement problem was identified as the "bottleneck" step of the constructive method. Therefore, any additional enhancements to the relative placement problem, either in the formulation (for instance, more compact formulations) or in the solution methodology (improvements in the efficiency of the interior point method) will result in an overall improvment to the constructive method. Comparisons between the QP and LP formulations illustrated that the QP formulation required significantly less computational effort than the equivalent LP formulation on all test circuits considered.

In terms of quality, the row lengths (and therefore the width of the placement area) were comparable regardless of the relative placement formulation. The LP formulation was illustrated to provide placements with lower estimates of wire length than the equivalent QP formulation, but at the expense of increased computational effort. This illustrated a tradeoff in quality of the initial placement versus the computational effort required to produce the placement.

Finally, initial placements created using the relative cell positions available after the solution of a single relative placement problem were presented. Although less computational effort was required, the quality of the initial placements were quite poor. These results confirmed the necessity of the additional computational effort required in performing several relative placement and circuit partitioning iterations in order to obtain "useful" information for positioning cells in the initial legal placement.

Although the cell positions produced by the relative placement and circuit partitioning iterations provide a good indication of the final cell positions, accepting the placement immediately after the legalization may not be sufficient. The relative cell positions used by the legalizaton heuristic, although "globally" good, only provide an "indication" of where each cell belongs. Although each cell may only move a small amount using the legalization heuristics described in this chapter, the total amount of cell movement may be significant. Therefore, some effort should be expended at improving the initial placement produced by the constructive method. Improving the initial placements by the application of a simple search heuristic is the topic of the next chapter.

Chapter 7

A Simple Iterative Improvement Method

In this chapter, we consider a simple iterative iterative method to further improve the initial placements created by our constructive method. Although the constructive method provides a good initial placement, it is still only a heuristic method for globally positioning cells near to their final positions. Several shortcomings may be identified, namely (i) the relative placement and circuit partitioning iterations may have incorrectly forced some cells into non-optimal positions and (ii) the additional cell movement during the legalization heuristic may have deteriorated the quality of the initial placement. It is therefore reasonable and important (necessary) to consider a method for further improving the initial placements created by the constructive method.

As previously mentioned, iterative improvement methods take an initial placement and make small changes (that is, rearrangements of the cells) to generate a new and improved placement. The methods may be classified as either randomized or deterministic algorithms, depending on whether or not a newly generated placement is accepted.

Randomized algorithms accept worse placements with some probability and are capable of escaping locally optimal placements. Given enough computational time, these algorithms tend to achieve globally optimal placements. Conversely, deterministic algorithms accept only improved placements and are not capable of escaping from locally optimal placements. However, these algorithms require little computational effort.

In this chapter, we consider a highly localized method for improving our initial placements. Although there is some randomness in the way we select cells for rearrangement, we consider our method to be deterministic since any attempt at rearranging cells is only permitted if it leads to an improved placement (that is, it is a greedy algorithm). We consider such an algorithm for the following reason. For any circuit, the constructive method has provided an initial placement in which the cells are near their desired final positions. It is reasonable to assume that, although not globally optimal, the initial placement lies within the vincinity of either a near, or globally, optimal placement. Hence, local rearrangement of cells is reasonable.

This chapter is organized as follows. In Section 7.1, we describe our iterative improvement method. We illustrate that by using simple 1-opt (cell moves) and 2-opt (cell swaps) techniques for repositioning cells, that an improved placement is possible. In Section 7.2, we present numerical results to demonstrate the effectiveness of the method when the initial placement is provided by the constructive method. Numerical comparisions are presented to demonstrate that our method compares favourably with a well-known randomized iterative improvement method, namely Simulated Annealing [36]. Finally, a summary is provided in Section 7.3.

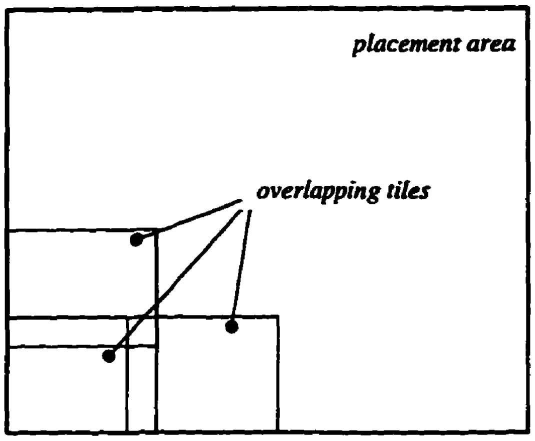


Figure 7.1: Tiles overlapping the placement area.

7.1 The General Strategy

7.1.1 Localizing the Improvement

Iterative improvement methods rely on rearrangements of cells to achieve an improved placement. In our placement heuristic, the initial placement is assumed good, and therefore any attempted improvements in the placement should reflect the assumption about the quality of the initial placement. To localize the rearrangement of cells, we introduce a novel technique of overlapping *tiles*, or *windows*, throughout the placement area as illustrated in Figure 7.1. The introduction of tiles throughout the placement area is useful for restricting the rearrangement of cells and is common in all types of iterative improvement methods to some extent [7, 26, 36]. Each tile contains a small subset of the cells. Furthermore, cells may belong to more than one tile due to the overlap which exists between tiles.

The iterative improvement method works as follows. A tile is selected, and a list of cells within the tile is generated. Subsequently, cells within the selected tile are rearranged in some fashion. In rearranging the cells, the cells are restricted to positions within the tile boundaries which keeps cells close to their original positions. The computational effort required to find improved cell positions is reduced, since the search for improved cell positions is restricted to positions within the tile boundaries. Since tiles overlap, cells near the boundaries of a tile may be permitted to move between tiles.

The question arises as to how tiles are selected. We use the idea of *passes* or *generations* [7]. During one pass, tiles are selected randomly, and only once during each pass. After each pass, the quality of the placement is evaluated and compared to placements generated by previous passes. The algorithm terminates when either a maximum number of passes is exceeded, or when the improvement in the placements over a number of consecutive passes is neglible.

7.1.2 Cell Rearrangement Within a Tile

Once a tile is selected, it is necessary to rearrange the cells within the tile. Several approaches have been proposed for this rearrangement.

One approach requires removing all cells from the tile and rearranging all cells simultaneously by solving a minimum cost maximum flow network problem [7]. Since cells are of differing width, this approach required cells to be divided into a number of subcells of equal width prior to creating and solving the network flow problem. Potentially, cells could split into disconnected pieces and a heuristic is therefore required to rejoin the cells. Additionally, since the cells within a tile were repositioned simultaneously, wire length approximations were required to account for cell interconnections within the tile.

A similar approach was proposed in [26]. Rather than repositioning all of the cells, a measure of goodness is calculated for each cell to estimate its appropriateness in its current location. Cells with high goodness are correctly positioned, whereas those with a low goodness are incorrectly positioned. A subset of cells with low goodness are removed from the placement and repositioned into the empty space created by their removal. This assignment is performed using a weighted matching algorithm, where the weights were selected based on an estimate of the appropriateness of repositioning each cell in each of the empty positions.

Both of the aforementioned methods are capable of avoiding locally minimal placement since cells are repositioned simultaneously. That is, given the subproblem these methods attempt to solve, the application of network methods results in optimal repositioning of the cells. However, in order to apply these methods, estimates in the objectives are required during the formulation of the network problems. This implies that, although the subproblem being solved is optimal, it may not be optimal in terms of the actual placement since wire lengths approximations are required.

We consider a third alternative method for rearranging cells based on simple 1-opt (cell moves) and 2-opt (cell swaps) movement of cells. This approach is more local than the network methods since it does not necessarily provide an optimal rearrangement of cells. However, repositioning using cell moves and swaps is a valuable method since it (i) is simple and efficient to implement, (ii) requires no estimates to the design objectives, and (iii) provides good results since the initial placement is good anyway.

The rearrangement of cells within a tile using moves and swaps proceeds as follows. To rearrange the tile, we randomly select a cell i and consider moving it to another position within the tile. If the new position for cell i does not violate any row length constraints, the move is considered. If the move violates a row length constraint, a second cell j closest to the new position is found and the cells i and j are swapped (provided that the swap does not violate any row length constraints). The motivation for the random selection of cells is computational effort. Selecting cells at random is constant, whereas attempting to select the "best" cell (or pair of cells) to move (swap) would require some additional computational effort to decide which cell is "best" to consider for rearrangement. Once the move or swap is performed, additional cells within the tile may require shifting to remove overlap and accomodate the repositioned cell. Shifting is required for cell moves and for cell swaps when the swapped cells are unequal in width. Cell moves and swaps, combine with potential shifting, are illustrated in Figure 7.2.

Figure 7.2 illustrates an additional aspect of the algorithm that must be mentioned. When cells are shifted within a tile, overlap may be introduced with cells external to tile under consideration. We ignore this overlap for several reasons. First, the overlap is temporary since it can be removed once the rearrangement of the tile is completed. Shifting cells external to the current tile after every move or swap (and estimating the effects of these shifts) requires a large amount of computational effort. However, the amount of temporary overlap is typically very small (since the tiles are selected to contain a small number of cells), and its effect on the total estimate of wire length is negligible. Therefore, ignoring the temporary overlap has little impact on quality, but substantially improves the computational efficiency of the tile rearrangement. Ignoring overlap external to the current tile being rearranged represents the one approximation used in our iterative improvment method.

Once a move or cell swap is made, and any cell shifting within the tile is performed, the change in the HPWL is estimated. Only nets connected to cells begin moved, swapped, or shifted will have a change in their HPWL (again, if cells external to the tiles were shifted, a significant number of nets would require updates to their HPWLs, which is computational prohibitive). If the change in HPWL leads to a reduction in the total HPWL, the move or swap is permitted and the placement is updated, otherwise the placement is left unchanged. The number of attempted moves and/or swaps for each tile is selected to be linear in the number of cells within the selected tile. The algorithm rearranging cells within a tile is outlined in Figure 7.3.

We consider the complexity of rearranging each tile using cell moves and swaps. As-

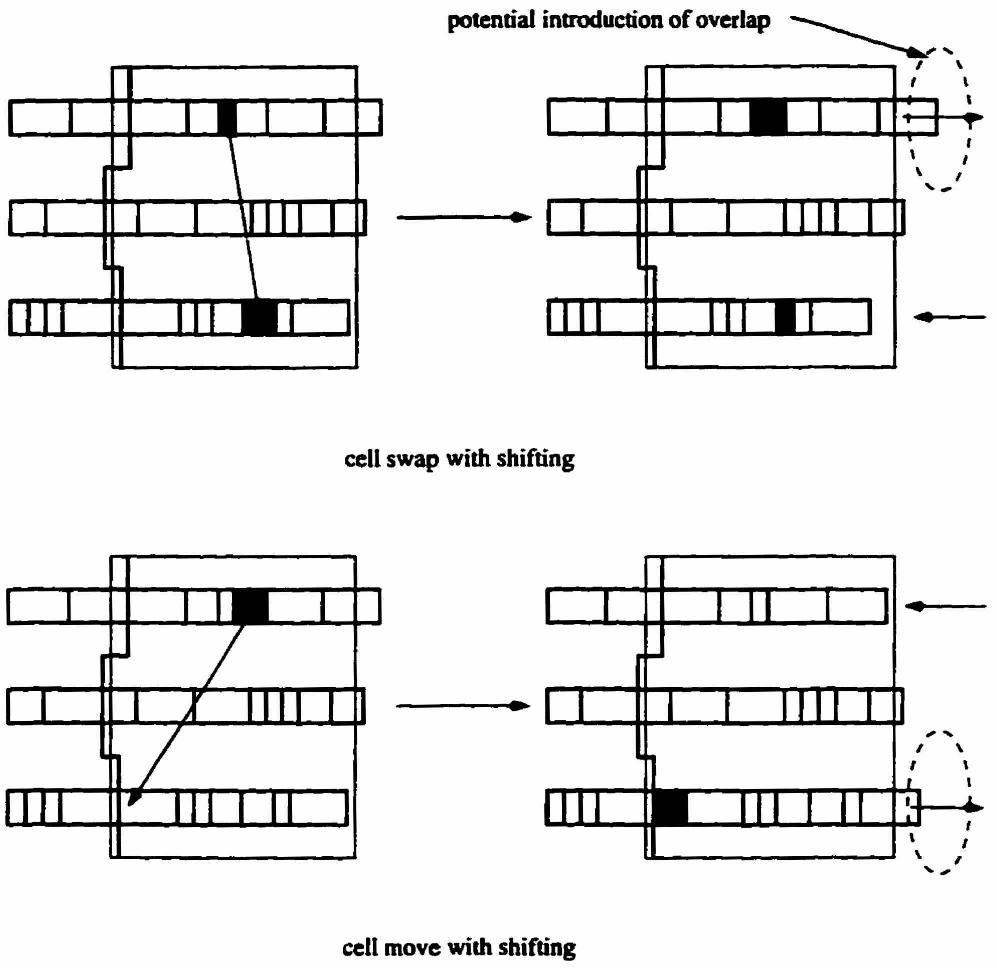


Figure 7.2: Example of cell moves and swaps.

```
    **procedure** *tile_rearrangment()*

1   *estimate_hpwl(initial_placement)*
    **repeat**
2       *select cells i and j*
3       *determine_cell_shifts()*
        *estimate_hpwl(new_placement)*
4       **if** $\Delta HPWL \leq 0$ **do**
            *update_placement()*
    **until** *attempted_moves* $\geq$ *max_attempts*
```

Figure 7.3: Algorithm for tile rearrangement.

sume that a tile contains c cells. Furthermore, assume that the number of nets connected to each cell is bounded by a small constant, and that the number of cells on each net is similarily bounded (a reasonable assumption for practical circuits since the circuit statistics previously presented in Chapter 2 indicated that most cells (nets) connect to very few nets (cells)).

Step 1 of the algorithm requires estimating the wire length for each net connected to a cell within the tile. This operation is linear in the number of connection points. However, the number of nets on each cell is bounded, and the estimate of wire length is performed in $O(c)$ time. Step 2 requires the random selection of two cells, and a check for row length violations. This operation is done in $O(1)$ time. The shifting operation in Step 3 requires a linear search of the current cell positions to remove any overlap, which is accomplished in $O(c)$ time. If in Step 4, the new placement is better than the previous placement, new cell positions must be saved. This can be accomplished by a copy of the new cell positions which is done in $O(c)$ time. Therefore, each attempted cell move or swap requires $O(c)$ time. Since the number of attempted tile rearrangements is selected linear in the number of cells, the complexity of rearranging one tile is $O(c^2)$.

Circuit	QP Form. (CPU Secs)		Improv.	LP Form. (CPU Secs)		Improv.
	Construct.	Iter. Improv.	Passes	Construct.	Iter. Improv.	Passes
circuit1	48.01	153.54	10	437.35	160.02	10
circuit2	325.62	711.34	10	2281.67	727.56	11
biomed	888.75	997.82	12	4129.96	1006.15	12
industry1	188.67	381.28	11	1349.75	352.39	10
industry2	2575.91	3583.69	12	12360.83	3467.36	12
industry3	4004.96	2421.74	11	28096.89	2862.02	12

Table 7.1: Constructive and iterative improvement times

7.2 Numerical Results

We now consider the effects of the iterative improvement method on the initial placements provided by the constructive method. The iterative improvement method was terminated when the reduction in wire length over the previous placement was less than 1/4 % for more than 3 passes, or when a maximum of 12 passes were performed.

We consider several aspects of the algorithm, namely (i) computational aspects of the iterative improvement method and the entire placement heuristic, and (ii) time and quality of the overall placement heuristic compared to a well-known Simulated Annealing (SA) placement package (TimberWolfSC V4.2) [36]. Results are presented using the initial placements created using both the QP and LP relative placement formulations.

7.2.1 Computational Aspects

Timing results for the overall placement heuristic are illustrated in Table 7.1. The times shown include that taken by the constructive method (extracted from those results previously presented in Chapter 6) and that taken for the iterative improvement method. The results are presented using both the QP and LP relative placement formulations (different initial placements imply a different sequence of events during the iterative improvement, and therefore potentially different times). Finally, the number of iterative improvement passes required are also illustrated in Table 7.1.

Table 7.1 illustrates that the initial placement (whether created from the QP or the LP relative placement formulation) has little impact on the amount of computational effort required by the iterative improvement method. This observation follows since the number of improvement passes (and therefore the computational effort) is nearly identical regardless of the initial placement used. If the computational effort of the iterative improvement method is compared to the constructive method, the following observations can be made. The computational time required by the iterative improvement method is either comparable or greater than that required by the constructive method when the QP relative placement formulation is used. Conversely, when the LP relative placement formulation is used during the constructive method, the time required by the iterative improvement method represents only a portion of the overall computational effort. This is a direct consequence of the differences in computational effort required by the different relative placement formulations which were previously illustrated in Chapter 7.

We illustrate the progression of the iterative improvement method in Figure 7.4. In Figure 7.4, the estimate of wire length and cumulative computational effort is plotted versus the pass number of the iterative improvement method for *circuit1* (similar plots are obtained regardless of the test circuit considered). The time required by the iterative improvement method increases linearly with the number of passes. However, the estimated wire length decreases rapidly near the beginning of the iterative improvement method and tapers off towards the last few passes. This is an important aspect (observation) for the local improvement heuristic since, if the computational effort required by the iterative improvement method is considered too large for any given circuit, decreasing the number of permitted passes will still result in a reasonable amount of improvement in the placement with less computational effort.

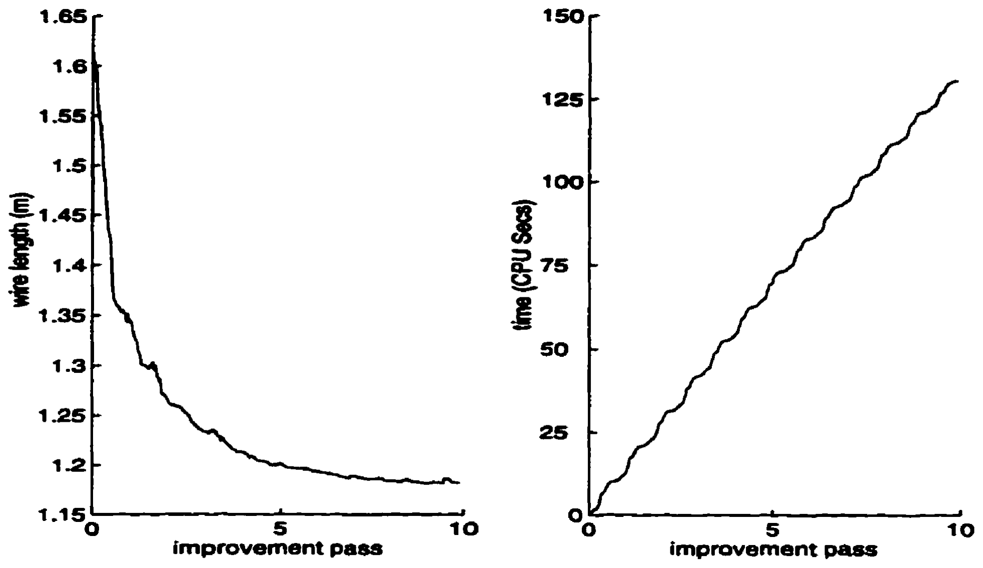


Figure 7.4: Progression of the iterative improvement method

7.2.2 Total Computational Effort and Quality of the Placements

The total computational effort for the entire placement heuristic (constructive and iterative improvement methods) is presented in Table 7.2. Results using both QP and LP relative placement formulations are presented. Additionally, the computational effort required by the Simulated Annealing placement algorithm is also presented. These results indicate that our placement heuristic requires less computational time than that required

Circuit	Solution Times (CPU Secs)			Improvement	
	QP Form.	LP Form.	Sim. Anneal.	(QP/SA)	(LP/SA)
circuit1	201	575	656	0.31	0.88
circuit2	1037	3009	3961	0.26	0.76
biomed	1887	5136	10309	0.18	0.50
industry1	570	1702	2629	0.22	0.65
industry2	6160	15828	25405	0.24	0.62
industry3	6427	30959	36267	0.18	0.85

Table 7.2: Final solution times

Circuit	Max. Row Length (μm)			Improvement	
	QP Form.	LP Form.	Sim. Anneal.	(QP/SA)	(LP/SA)
circuit1	6280	6280	6260	1.003	1.003
circuit2	12500	12500	13010	0.961	0.961
biomed	8424	8424	8464	0.995	0.995
industry1	4842	4840	5366	0.902	0.902
industry2	14432	14432	15240	0.947	0.947
industry3	28480	28480	31768	0.896	0.896

Table 7.3: Final placement areas

Circuit	Est. HPWL (m)			Improvement	
	QP Form.	LP Form.	Sim. Anneal.	(QP/SA)	(LP/SA)
circuit1	1.180	1.181	1.186	0.995	0.996
circuit2	4.676	4.697	4.822	0.968	0.974
biomed	2.843	2.661	2.572	1.105	1.034
industry1	1.743	1.656	1.933	0.902	0.856
industry2	19.304	19.686	21.716	0.889	0.906
industry3	56.798	64.694	68.589	0.828	0.943

Table 7.4: Final estimates of wire length

by Simulated Annealing. This observation is valid regardless of the relative placement formulation used. Moreover, the savings in computational effort is substantial when the QP formulation of the relative placement problem is used during the constructive phase of the placement heuristic.

Table 7.3 illustrates the longest row lengths (taken as an indication of the final placement widths) in the final placements. In all cases, our placement heuristic generates placements with row lengths better than or comparable to those obtained using Simulated Annealing. The different relative placement formulations result in placements with exactly identical row lengths. The observation that the row lengths for the different relative formulations are identical is a coincidence (they were not required to be identical).

Finally, comparisions of the estimated wire lengths are presented in Table 7.4. Our placement heuristic generates placements with lower wire lengths for all test circuits considered, with the exception of *biomed*. Analysis of circuit *biomed* indicated that several extremely long nets were present which were apparently "handled" more effectively by Simulated Annealing than by our placement heuristic. We also note that the LP relative placement formulation was more effective at handling these long nets than the equivalent QP formulation. To summarize, our placement heuristic (which uses a combination of relative placements, circuit partitioning and iterative improvement) yields placements that are up to 10 – 17% better than Simulated Annealing in terms of estimated wire lengths while requiring 5 times less computational effort (for the QP formulation).

One interesting observation can be made from the results presented in Table 7.4. Although it provides better initial placements, the LP formulation *does not* necessarily result in better placements than those obtained using the equivalent QP formulation *after* the application of the iterative improvement method (with the exception of circuit *biomed*, where the additional improvement using the LP formulation is about 7 percent). This leads to an important observation, namely that by performing iterative improvement, the QP formulation is *sufficient* for the creation of good initial placements which lead to very good final placements. Given the substantial reduction is computational effort necessary to obtain an initial placement, the QP formulation appears preferable to the LP formulation of the relative placement problem.

7.3 Summary

In this chapter, a greedy and localized iterative improvement method based on simple cell moves and cell swaps has been presented. By "tiling" the placement area and restricting cells to positions within their assigned tile, the computational efficiency of the iterative improvement method was greatly enhanced. The tiling was facilitated by the construction of a good initial placement, where cells were already near to their desired final positions.

Numerical results highlighted several important aspects of the overall placement heuristic. The computational effort of the iterative improvement method was illustrated to be linear in the number of passes performed (where one pass consisted of considering each tile within the placement area once, and rearranging the cells within each selected tile). However, the reduction in estimated wire length was illustrated to be greatest at the beginning of the iterative improvement method. Therefore, for any given circuit, if the computational effort required by the iterative improvement method is considered too large, a reduction in the overall computational effort is possible by reducing the number of passes performed while still maintaining a reasonable improvement in the quality of the placement.

Comparisions with a Simulated Annealing placement heuristic illustrated the effectiveness of our combination of constructive and iterative improvement methods. In all cases, our heuristic produced placements of comparable or superior quality, both in terms of the widths of the required placement areas and the total estimates of wire length. Additionally, our heuristic required less computational effort (substantially less when the QP relative placement formulation is used during the creation of the initial placements) to produce these higher quality placements.

Comparisions of the final placements also illustrated several important aspects regarding the relative placement formulations. Although, as previously illustrated in Chapter 6, the LP formulation provides better initial placements than the equivalent QP formulation, after the application of iterative improvement the QP formulation provides (in most cases) either comparable or better results. The savings in computational effort for the creation of the initial placement using the QP formulation versus the LP formulation was reflected in the final computational times of the overall placement heuristic. Therefore, we concluded that our placement heuristic, with the QP formulation of the relative placement problem, is the preferable placement heuristic.

Although we have completely described the combination of constructive and iterative improvement methods for cell placement (and illustrated they produce favourable results compared to a well known Simulated Annealing placement heuristic), we consider an additional enhancement to our constructive method in the next chapter. Specifically, we consider *circuit clustering* which may be used to improve the efficiency of the constructive placement method by reducing the sizes of the resulting relative placement formulations (regardless of the relative placement formulation used).

Chapter 8

Circuit Clustering

We now consider a method for reducing the computational effort of the constructive method, while maintaining or improving the quality of the resulting initial placements, by introducing the concept of circuit clustering. Circuit clustering involves identifying strongly connected components of a circuit and merging these components into a large number of small groups called clusters. For highly complex circuits containing many thousands of cells, efficient placement procedures require a reduction in problem sizes. Circuit clustering achieves this objective since grouping cells into clusters is equivalent to condensing the circuit, where clusters of cells may be considered as single cells and nets which connect cells entirely within the same cluster are effectively removed from the problem.

When applied to a condensed circuit, our constructive method results in a sequence of smaller optimizations problems for determining relative cell positions. This implies reduced storage requirements and reduced computational efforts. Circuit clustering can also lead to improved placements since cells that should be close to each other are merged into the same cluster. Therefore, the possibility of making "bad" decisions early in the constructive method (that is, forcing highly connected cells apart) is reduced. Circuit clustering is applicable not only to cell placement, but to other VLSI problems as well. For instance, it has been incorporated into a two-phase circuit partitioning heuristic [2], where it resulted in better final partitions with less computational effort (when compared to those final partitions obtained without clustering). Hence, we expect similar benefits when applied to the placement problem.

This chapter is organized as follows. In Section 8.1 we describe previously proposed circuit clustering heuristics. We identify several shortcomings of these previous approaches which make them unattractive when applied to the cell placement problem. In Section 8.2, we describe a new greedy clustering heuristic based on graph connectivity. The heuristic is illustrated to be capable of producing a large number of clusters of nearly equal size due to limits imposed on the size of the clusters. We illustrate that the storage requirements are reasonable in practical situations, and that the complexity of the heuristic is polynomial in the number of cells in the circuit. The incorporation of the clustering heuristic into the constructive placement method is also described. Numerical results are presented in Section 8.3. These results illustrate that the proposed clustering heuristic is effective at reducing the netlist sizes while requiring reasonable computational effort. The reduction in computational effort and the quality of the initial legal placements when clustering is included into the constructive placement method is also illustrated. Final placements obtained after the application of the local improvement heuristic are also presented and compared to Simulated Annealing. Finally, a summary is provided in Section 8.4.

8.1 Previous Approaches

We consider a circuit clustering heuristic effective if (i) it exhibits reasonable storage requirements (ii) it is efficient to implement and requires little computational effort to run to completion, and (iii) it is capable of producing a large number of clusters of nearly equal size while still providing large reductions in circuit sizes. Since clustering is most beneficial when applied to large circuits, its implementation requirements (the storage requirements) should not prohibit its application. Since circuit clustering is typically used as a preprocessing step for a subsequent problem, any computational benefits achieved at later stages should not be lost during the clustering. Finally, by generating a large number of clusters of nearly equal size, no "skewing" (that is, no preference for a cluster based on size) occurs during the solution of subsequent problems.

Many clustering heuristics have been proposed which achieve some, but not all, of the aforementioned characteristics. Recently, an effective clustering heuristic capable of producing a large number of clusters of nearly equal size has been proposed [2] based on a combination of a multi-way cell interchange heuristic [34] with the GRASP heuristic [9]. The heuristic was illustrated to be effective at reducing netlist sizes while requiring little computational effort. One shortcoming of this heuristic, however, is the storage requirements necessary for its implementation. Since it is based on a cell interchange heuristic, it is necessary to store the so called "gain entries" [34] associated with cell interchange heuristic which is prohibitive for extremely large circuits with a large number of clusters.

A clustering heuristic based on a modification of the uniform multicommodity flow problem has been proposed [51]. This heuristic produces natural clusters where the final number of clusters is unknown until the heuristic runs to completion. Unfortunately, this heuristic typically produces very few clusters which vary greatly in size. For instance, results presented in [51] on an example test circuit produced the following: 1 cluster of 3184 cells, 6 clusters of 8 cells, 2 clusters of 4 cells, and 6 clusters of single cells. Therefore, it is not useful as preprocessing step for a subsequent problem as a result of the skewing in the cluster sizes. Another approach for generating natural clusters based on circuit connectivity was proposed in [11]. The heuristic requires the selection of two parameters whose optimal values cannot be determined prior to running the algorithm. Therefore, the resulting clusters are not controllable.

A clustering heuristic based on random walks has been proposed [18]. Although this approach was illustrated to produce good clusters, the length of each random walk required is $O(c^2)$ where c is the number of cells in the circuit. The total complexity of the heuristic is $O(c^3)$ which makes it undesireable when used as a preprocessing step for a subsequent problem.

8.2 Proposed Clustering Strategy

In this section, we propose a new simple and greedy clustering heuristic based on circuit connectivity. The heuristic produces a large number of clusters (with each cluster containing a small number of cells) of nearly equal size. We also describe the incorporation of circuit clustering into our constructive placement method.

8.2.1 Description

To implement the heuristic, it is first necessary to convert the circuit netlist into a weighted adjacency graph, where each node in the adjacency graph represents a cell in the circuit and the edge weight between any pair of nodes provides a measure of how "tightly connected" the corresponding cells are in the circuit netlist. The adjacency graph is created by defining, for each net i in the original circuit, the net weight

$$w_i = \frac{1}{|N_i| - 1}, \tag{8.1}$$

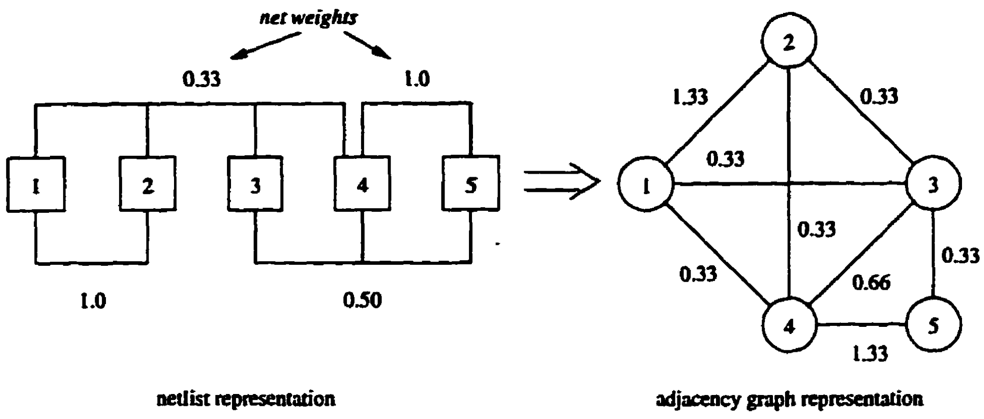


Figure 8.1: Conversion from circuit netlist to adjacency graph.

where N_i denotes the cells on net i. Clearly, short nets which connect only a few cells are assigned larger weights than long nets which connect many cells. The edge weight, denoted by w_{ab}, between any two nodes a and b in the adjacency graph is given by

$$w_{ab} = \sum_{i \in C_a \cap C_b} w_i, \tag{8.2}$$

where C_a and C_b represent the set of nets incident on cells a and b, respectively.

Defining the edge weights in this way accompishes several objectives. Pairs of cells with no common nets have zero edge weights, whereas cells sharing many nets have large edge weights. Therefore, the weights reflect the amount of connectivity between pairs of cells in the netlist. Preferential treatment is given to short nets since they make larger contributions than long nets when computing the edge weights. This is desireable for circuit clustering since it is precisely short nets which tend to get absorbed when clustering cells, whereas long nets cannot be eliminated. An example of the conversion of a netlist into a weighted adjacency graph is illustrated in Figure 8.1.

Our proposed clustering heuristic is based on merging pairs of nodes in the adjacency graph according to the edge weights. More precisely, the heuristic continually selects a maximally weighted edge in the adjacency graph of maximal and attempts to merge the two incident nodes together. After each merge, the adjacency graph is updated to reflect the consequence of the merge as follows. Let the two nodes being merged be nodes a and b, and let node a be the representative after the merge. Node b and all its incident edges are removed from the adjacency graph. The adjacencies to node a are replaced by those nodes in $Adj(a) \cup Adj(b)$. Finally, the edge weights incident to node a are updated according to the rule $w_{ac} = w_{ac} + w_{bc}$ for all $c \in Adj(a) \cup Adj(b)$. Since nodes in the adjacency graph correspond to cells in the circuit netlist, the circuit netlist also changes along with the adjacency graph (cells in the circuit netlist are merged together and form the desired clusters). Nodes are merged together until no further merges are possible. The total number of merges is limited as follows. Let the desired cluster size be denoted by T, the maximum cluster size be denoted by M, and select a cluster size penalty parameter α. After selecting nodes a and b for merging, the size penalty $p_{ab} = T - \alpha(S_a + S_b)$ is computed, where S_a and S_b represent the current number of nodes represented by nodes a and b, respectively. That is, the size penalty represents the difference between the resulting cluster (weighted by some factor) and the target cluster size. If the selected merge would result in a cluster with size greater than M, or the selected edge weight plus the size penalty is negative, the merge is discarded from future consideration. When all potential merges are rejected, the clustering heuristic terminates. An example illustrating the merging of nodes, along with the associated changes in the circuit netlist, is provided in Figure 8.2.

A pseudo-code description of the clustering heuristic is provided in Figure 8.3 where it is assumed that the edge weights in the adjacency graph are provided as input. The complexity of the heuristic is computed as follows. In the worst-case, the adjacency graph created from the circuit netlist will be a complete graph consisting of $c(c-1)/2$ edges

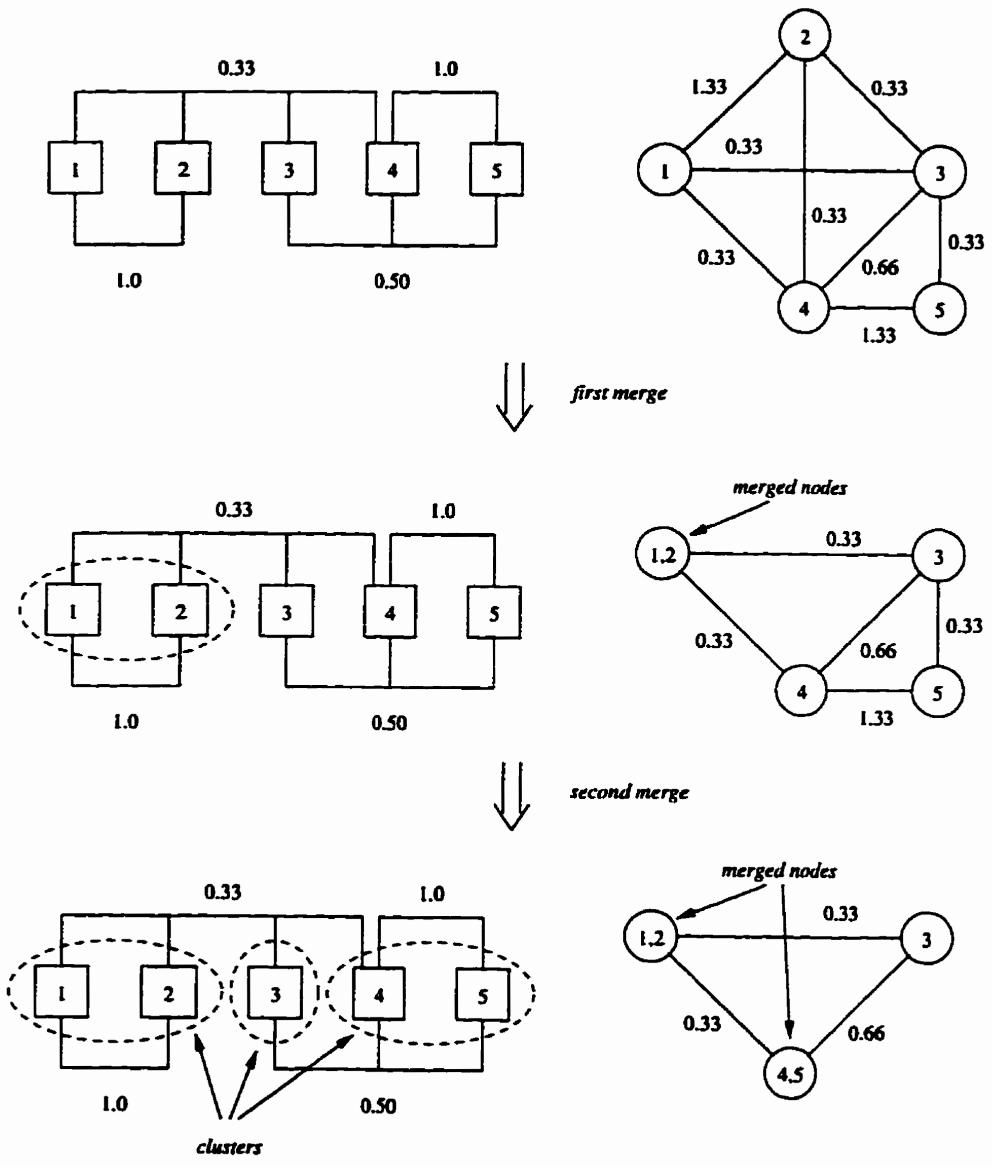


Figure 8.2: Progression of the clustering heuristic.

```
    procedure cluster();
1   insert edge weights into maximum heap
    $k = 1$
    repeat
2       Select maximally weighted edge $w_{ab}$
        { merge node $b$ into node $a$ }
        { update adjacencies, edge weights and heap }
3       for $c \in Adj(a) \cup Adj(b)$ do
            $w_{ac} = w_{ac} + w_{bc}$
        $k = k + 1$
    endfor
    until no possible merges;
```

Figure 8.3: Pseudo-code for the clustering heuristic.

and c nodes. Step 1 of the heuristic requires the insertion of the edge weights into an appropriate data structure to facilitate the retrieval of the maximally weighted edges at each step of the heuristic. To faciliate the selection of such edges, the edge weights are inserted into a maximum heap [50]. The creation of a heap in linear in the number of entries in the heap. Therefore, Step 1 of the heuristic is $O(c^2)$.

Once initialized, the heuristic continues until no further merges are possible. Step 2 requires the selection of the maximally weighted edge which is $O(1)$, since the edge weights are stored in a maximum heap. In the worst-case, where the merging proceeds until all nodes are collapsed into a single node, the outer loop of the heuristic is executed $c - 1$ times. Therefore, over the course of the entire heuristic, the selection of maximally weighted edges is $O(c)$. After the kth merge, the number of remaining nodes in the adjacency graph is $c - k$ and the maximum number of edges is $(c - k)(c - k - 1)/2$. Step 3 of the heuristic requires updating the adjacencies of node a to reflect the consequence of merging node b into node a. This requires removing any adjacencies to both nodes, updating the edge weights and reinserting the updated edge weights back into the heap.

Step 3 is executed at most $c-k$ times during the kth iteration (this occurs since in the worst-case, node a is adjacent to every other node in the graph) and the insertion and removal of entries from a heap is logarithmic in the number of entries. Therefore, the complexity of the updating step over the course of the entire heuristic is given by

$$\sum_{k=1}^{c-1} O((c-k)\log[(c-k)(c-k-1)/2]) = \sum_{k=1}^{c-1} O((c-k)\log(c-k) = O(c^2 \log c)$$

Therefore, the clustering heuristic is $O(c^2 \log c)$. Finally, the storage requirements of the heuristic are $O(c^2)$, namely the space required to store the edge weights in the heap.

We now make some practical observations. A circuit netlist is typically sparse and the initial adjacency graph will not be a complete graph. The number of edges will be substantially less than $c(c-1)/2$ which implies reasonable storage requirements. This also implies the computational effort will be reduced since fewer entries will be present in the heap. Its creation, as well as the continual insertions and deletions during the updating step will proceed in a more timely manner. The sparsity of the netlist also implies that only a few edges will have to be considered during the updating step of the heuristic, since it is unlikely that a single node will be adjacent to every other node in the graph. Finally, since the clustering heuristic is never allowed to merge all nodes together (due to the size penalty and maximum cluster size imposed on the heuristic), the outer loop of the heuristic will not be executed $c-1$ times, implying fewer updating operations and so forth.

We make several final observations to explain an "objective function" associated with the proposed clustering heuristic. Let the set of cells in cluster k be denoted by B_k. Furthermore, define the objective function

$$\sum_{k=1}^{B} \sum_{\{\forall i | N_i \cap B_k \neq \emptyset\}} (\mid N_i \cap B_k \mid -1) w_i. \tag{8.3}$$

This objective function "counts" the total weight of the nets absorbed into a cluster when each net in the circuit netlist is represented as a tree. Note that when all cells for a given net i are merged into the same cluster, the total contribution to this objective function is 1. Conversely, when only one cell on any net i is placed into a cluster, the contribution to the objective function is 0. Therefore, since the proposed clustering heuristic is based on merging highly connected cells, it may be considered a greedy method for maximizing the total number of edges abosrbed into the clusters.

8.2.2 Incorporating Clustering into the Constructive Method

Incorporating circuit clustering into the constructive placement method is illustrated in Figure 8.4. When clustering is included, the circuit netlist is clustered *once*, *prior* to beginning the relative placement and circuit partitioning iterations. When the iterations begin, the relative placement problem is formulated using the *condensed netlist* produced by the clustering heuristic. Since the clustered netlist is smaller, the resulting optimization problems will require fewer variables and less storage, implying an overall reduction in the computational effort. When the solution of the relative placement problem is found, the relative positions of the *clusters* are known rather than the positions of the individual cells. Cells are assigned to the position of their assigned cluster.

Cell positions are provided to the circuit partitioning portion of the constructive method as usual, and the circuit partitioning proceeds as previously described in Chapter 5. In other words, the circuit partitioning portion of the method is still applied to the *cells*, and is not directly affected by the circuit clustering. Once the partitioning is completed, it is necessary to revise the clustered netlist prior to revising the constraints and variable bounds of the relative placement problem. During the circuit partitioning, cells within the same cluster may be assigned to different regions. Therefore, it is necessary to "check" each cluster in the condensed netlist. Any clusters containing cells assigned to

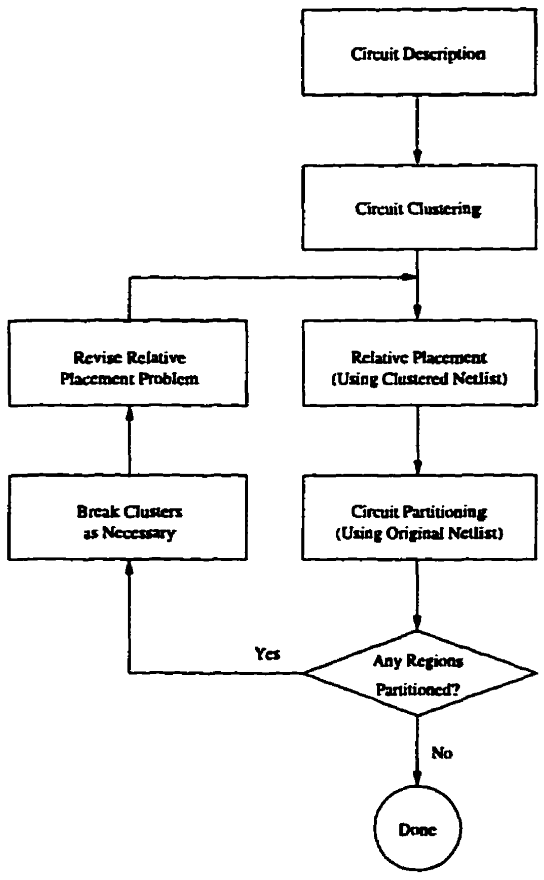

Circuit Description
Circuit Clustering
Revise Relative Placement Problem
Relative Placement (Using Clustered Netlist)
Break Clusters as Necessary
Circuit Partitioning (Using Original Netlist)
Any Regions Partitioned?
Yes
No
Done

Figure 8.4: Incorporating circuit clustering into the placement method.

different regions of the placement area are "broken" into smaller pieces to account for the separation of the cluster during the partitioning. Once completed, the relative placement problem is revised using the circuit partitioning information and reformulated using the revised condensed netlist. The termination of the algorithm (that is, when each region in the placement area contained fewer than a preselected number of cells) is identical to that previously described in Chapter 5.

8.3 Numerical Results

In this section, we consider numerical aspects of the clustering heuristic proposed in this chapter. We consider the reductions in the circuit sizes when clustering is applied and the amount of computational effort required by the clustering heuristic. The effects of circuit clustering on the performance of the proposed placement heuristic are also considered. Specifically, when clustering is implemented within the constructive phase of the placement heuristic, we consider the quality of the initial and final legal placements, as well as the computational effort required to produce the placements. Both QP and LP formulations of the relative placement problem are considered (although numerical results presented in previous chapters indicate that the QP formulation of the relative placement problem is the preferred choice).

8.3.1 Computational Effort and Impact on Netlist Sizes

The effects of the clustering heuristic when applied to the set of test circuits are shown in Table 8.1. In performing the clustering, the target cluster size was set to $5A$ and the maximum cluster size to $10A$, where A is the average area of the cells. The size penalty parameter was set to 10. The number of cells and nets in the original and clustered circuits are presented in Table 8.1. The improvement is the number of cells and nets

Circuit	Original Netlist		Clustered Netlist		Improv.		Time
	Cells	Nets	Cells	Nets	Cells	Nets	(CPU Secs)
circuit1	833	983	214	555	0.26	0.56	1.43
circuit2	3014	3136	770	1616	0.26	0.51	13.03
biomed	6417	5742	1864	1746	0.29	0.30	14.76
industry1	2271	2478	743	1427	0.33	0.57	11.93
industry2	12142	13419	3096	7214	0.25	0.53	177.01
industry3	15059	21938	3754	14202	0.25	0.64	181.43

Table 8.1: Effects of clustering on netlist sizes.

(measures and the number of cells (nets) in the clustered netlist divided by the number of nets (cells) in the original netlist) is also provided. Finally, the computational effort required by the clustering heuristic is also presented.

Table 8.1 clearly illustrates that the clustering heuristic requires little computational effort to cluster each test circuit (compared to those times previously presented in Chapter 6 for the creation of the initial legal placements). Therefore, the clustering heuristic will not degrade the constructive placement method from the perspective of the required computational effort. The numerical results in Table 8.1 also illustrates that the clustering heuristic is very effective at reducing circuits. Although the number of cells permitted within each cluster is relatively small (always less than $10A$), the number of nets effectively removed from the clustered netlists is typically greater than 50 percent for all test circuits, which is clearly a significant amount.

8.3.2 Computational Effort and Quality of Initial Placements

We now consider the effects of circuit clustering on the initial legal placements created by the constructive placement method. The computational effort required to create initial legal placements, both with and without clustering, is illustrated in Table 8.2. Both the QP and LP relative placement formulations are considered. We note that the running times presented in Table 8.2 without clustering are taken from the results previously

Circuit	QP Form. (CPU Secs)		Improv.	LP Form. (CPU Secs)		Improv.
	Without	With		Without	With	
circuit1	48.01	23.45	0.49	437.35	174.89	0.40
circuit2	325.62	167.61	0.51	2281.67	958.16	0.42
biomed	888.75	402.38	0.45	4129.96	1308.81	0.32
industry1	188.67	116.84	0.62	1349.75	598.08	0.44
industry2	2575.91	1216.03	0.47	12360.83	6904.00	0.56
industry3	4004.96	1883.53	0.47	28096.89	15910.45	0.56

Table 8.2: Computational effort with and without clustering.

Circuit	QP Form. (μm)		Improv.	LP Form. (μm)		Improv.
	Without	With		Without	With	
circuit1	6380	6230	0.98	6270	6370	1.02
circuit2	12500	12460	1.00	12540	12450	1.00
biomed	8368	8376	1.00	8392	8390	1.00
industry1	4822	4806	1.00	4810	4808	1.00
industry2	14344	14336	1.00	14352	14352	1.00
industry3	28296	28288	1.00	28280	28248	1.00

Table 8.3: Row lengths with and without circuit clustering.

presented in Chapter 6. Additionally, when circuit clustering is used, the times reported in Table 8.2 include the computational effort required to cluster the netlist. Taking the ratio of the time required with clustering to that required without clustering, the results presented in Table 8.2 clearly indicate that circuit clustering results in substantial savings in the required computational effort, yielding a typical savings around 50 percent for all test circuits.

The longest row lengths in the initial legal placements are presented in Table 8.3. The results presented in Table 8.3 indicate that the longest row lengths, and therefore the placement area, are not effected by circuit clustering.

Estimates of the total wire lengths for all test circuits are presented in Table 8.4. Again, both relative placement formulations are considered. The results presented in Table 8.4 illustrate that the incorporation of circuit clustering results in initial legal placements of either comparable or better estimates of wire length when compared to

Circuit	QP Form. (m)		Improv.	LP Form. (m)		Improv.
	Without	With		Without	With	
circuit1	1.6154	1.5743	0.97	1.5101	1.6030	1.06
circuit2	6.3227	6.5673	1.04	5.8707	5.9814	1.02
biomed	4.0053	3.2970	0.82	3.5557	3.1739	0.89
industry1	2.1200	2.0162	0.95	1.9053	1.8722	0.98
industry2	29.3326	27.4624	0.94	29.3229	27.4941	0.94
industry3	77.6508	75.2041	0.97	73.1003	72.3994	0.99

Table 8.4: Estimates of wire lengths with and without circuit clustering.

those initial placements obtained without clustering (especially on the *larger* test circuits). In other words, circuit clustering has a *positive* impact on the *quality* of the initial placements while requiring *less* computational effort.

8.3.3 Computational Effort and Quality of Final Placements

Although clustering has no direct impact on the local improvement heuristic (of course, different initial placements are created by the constructive method when clustering is used which in turn effects the *final results* of the local improvement heuristic), we still consider the final placements obtained after the application of local improvement when clustering is used during the creation of the initial placements. Essentially, it is these numerical results which represent the final results (placements) of the proposed placement heuristic since all components (that is, the constructive method, the iterative improvement method *and* the clustering heuristic) proposed in this thesis are used in combination.

Table 8.5 presents the total computational effort required by the constructive method with clustering (taken from Table 8.2) and the iterative improvement method. Previously, it was shown in Chapter 7, that the constructive method typically required either comparable or more time than the iterative improvement method. However, Table 8.5 clearly demonstrates that the effect of clustering is to "shift" the computational burden away from the constructive method. In other words, the computational effort required by the

Circuit	QP Form. (CPU Secs)		Improv.	LP Form. (CPU Secs)		Improv.
	Construct.	Iter. Improv.	Passes	Construct.	Iter. Improv.	Passes
circuit1	23.45	218.43	12	174.89	234.52	12
circuit2	167.61	844.45	12	958.16	930.94	12
biomed	402.38	1061.72	12	1308.81	1216.65	12
industry1	116.84	476.95	12	598.08	477.16	12
industry2	1216.03	4110.14	12	6904.00	4273.66	12
industry3	1883.53	3118.23	12	15910.45	3321.28	12

Table 8.5: Constructive and iterative improvement times.

Circuit	CPU Time (CPU Secs)			Imp. (Clustering)		Imp. (No Clustering)	
	QP	LP	SA	(QP/SA)	(LP/SA)	(QP/SA)	(LP/SA)
circuit1	241	405	656	0.37	0.62	0.31	0.88
circuit2	1010	1877	3961	0.25	0.47	0.26	0.76
biomed	1412	2482	10309	0.14	0.24	0.18	0.50
industry1	590	1067	2629	0.22	0.41	0.22	0.65
industry2	5317	11056	25405	0.21	0.44	0.24	0.62
industry3	4985	19070	36267	0.14	0.53	0.18	0.85

Table 8.6: Final solution times.

iterative improvement method becomes comparable, or larger, than that required by the constructive method. Of course, using the LP relative placement formulation for the larger circuits (*industry2* and *industry3*) still requires more computational effort during the constructive method (this difference may be further offset by increasing the desired cluster size, resulting in smaller LP problems during the initial stages of the constructive method).

The final solution times comparing our placement heuristic using the QP formulation and the LP formulation to the Simulated Annealing placement heuristic are presented in Table 8.6. As previously demonstrated in Chapter 7, our placement heuristic requires less computational effort than the Simulated Annealing heuristic, regardless of the relative placement formulation. Circuit clustering serves to further improve this result as illustrated in Table 8.6. A comparison of the results presented in Table 8.6 to those previously presented in Table 7.2 (the results presented in the last two columns of Table 8.6 are reproduced directly from Table 7.2 to faciliate the comparison of results with and without

Circuit	Max. Row Length (μm)			Imp. (Clustering)		Imp. (No Clustering)	
	QP	LP	SA	(QP/SA)	(LP/SA)	(QP/SA)	(LP/SA)
circuit1	6280	6280	6260	1.003	1.003	1.003	1.003
circuit2	12500	12500	13010	0.961	0.961	0.961	0.961
biomed	8424	8424	8464	0.995	0.995	0.995	0.995
industry1	4842	4842	5366	0.902	0.902	0.902	0.902
industry2	14432	14432	15240	0.947	0.947	0.947	0.947
industry3	28480	28472	31768	0.896	0.896	0.896	0.896

Table 8.7: Final placement areas.

clustering) illustrates that circuit clustering reduces the overall computational effort by a significant amount (this observation is especially true for the LP relative placement formulation, since it is the LP formulation which benfits the most from the reduction in the optimization problem sizes during the constructive method).

Table 8.7 illustrates that the final placement areas, as measured by the length of the longest row in the final placement, are better than those obtained by the Simulated Annealing heuristic. A comparision of the row lengths presented in Table 8.7 to those obtained without circuit clustering previously presented in Table 7.3 (results previously reported in Table 7.3 are reproduced in the last two columns of Table 8.7 for comparision purposes) illustrates that clustering has no effect on the final row lengths resulting from our placement heuristic.

The final estimates of wire length obtained using our placement heuristic (with both the QP and LP relative placement formulations) and Simulated Annealing are presented in Table 8.8 (of course, the Simulated Annealing results are reproduced from Chapter 7). Our placement heuristic with clustering results in placements with final estimates of wire lengths which are either comparable to or better than those obtained with Simulated Annealing. Several additional observations must be made concering the results presented in Table 8.8 to fully analyse the effects of clustering. A comparison of the final estimated wire lengths obtained with and without clustering (results obtained without clustering are

Circuit	Est. HPWL (m)			Imp. (Clustering)		Imp. (No Clustering)	
	QP	LP	SA	(QP/SA)	(LP/SA)	(QP/SA)	(LP/SA)
circuit1	1.191	1.193	1.186	1.004	1.006	0.995	0.996
circuit2	5.161	4.946	4.822	1.070	1.026	0.968	0.974
biomed	2.752	2.708	2.572	1.070	1.053	1.105	1.034
industry1	1.739	1.628	1.933	0.899	0.842	0.902	0.856
industry2	20.628	20.145	21.716	0.950	0.928	0.889	0.906
industry3	58.611	66.737	68.589	0.855	0.973	0.828	0.943

Table 8.8: Final estimates of wire length.

taken from Table 7.4 and are reproduced in the last two columns of Table 8.8 to facilitate the comparison), illustrates that the placements obtained with clustering experience a slight degradation in quality. This result occurs despite the fact that the initial placements created with clustering are *better* than those created without clustering.

Despite this observation regarding the slight degradation in the quality of the placements as a result of clustering, the application of clustering is still important. Clustering has a positive impact on the quality of the initial placements and results in a *significant* reduction in the computational effort required to construct the initial placements (which also implies a reduction in the total computational effort required by the entire placement heuristic). Good quality final placements are still obtained (especially for the larger circuits such as *industry2* and *industry3*) and are comparable to or better than those provided by Simulated Annealing. Therefore, an important future path is to focus on the enhancement of the iterative improvement method.

8.4 Summary

In this chapter, we have proposed a new clustering heuristic based on circuit connectivity. The heuristic was illustrated to be greedy in nature, and capable of producing a large number of clusters of nearly equal size. The complexity of the heuristic was illustrated to be polynomial in the number of cells in the circuit. The incorporation of the circuit clustering heuristic into the constructive placement method was also described.

Numerical results indicated that the clustering heuristic was quite effective at reducing the sizes of the netlists while requiring little computational effort to do so. When applied to the constructive method, an overall reduction in the computational effort required to create an initial placement was achieved, regardless of the relative placement formulation used. Furthermore, clustering resulted in better initial placements when compared to those initial placements created without circuit clustering. Therefore, circuit clustering represents an important extension to the constructive method since: (i) it aids in the creation of better initial placements and (ii) significantly reduces the required computational effort.

Numerical results were also presented to investigate the final placements obtained once the initial placements (created using clustering) were subjected to the iterative improvement method previously proposed in Chapter 7. These final placements were compared to those obtained using Simulated Annealing. As expected, circuit clustering reduced the overall computational effort of our placement heuristic (regardless of the formulation of the relative placement problem), making our heuristic even more attractive than Simulated Annealing from a computational point of view. Additionally, the estimated wire lengths of the final placements were comparable to or better than those obtained using Simulated Annealing. Unfortunately, final placements with clustering were slightly inferior to those placements obtained without clustering (as previously presented in Chapter 7). This observation was despite that fact that clustering aided the constructive method in creating better initial placements. This observation, do not detract from the benefits of circuit clustering (*significant* reduction in computational effort and good quality placements), but rather demonstrates the potential benefit of further enhancements to the iterative improvement method previously described in Chapter 7.

Chapter 9

Conclusions and Future Directions

The division of the integrated circuit design procedure into a sequence of interacting design steps is an appropriate decision, given the complexity of modern integrated circuits. Unfortunately, the division of the procedure does not represent a complete solution to all the difficulties associated with integrated circuit design. Each individual design step, even when considerd as a single problem, still remains difficult. Continual research is therefore required to develop more effective, efficient and robust techniques for each problem which arises during the design of integrated circuits. Only through the *continual* development of such techniques will circuit designers be capable of designing more complex circuits which take advantage of both existing and future technologies.

When solving any problem associated with integrated circuit design, it is common that only an *approximate solution* of the exact problem may be possible. Techniques capable of providing near optimal solutions while requiring reasonable computational effort must be implemented. Furthermore, *different* solution methodologies, such as *mathematical programming*, *combinatorial optimization*, *search heuristics*, and so forth, may all prove useful for tackling a given problem. Finally, it is necessary to consider the *combination* of techniques in situations where one particular solution methodology is not sufficent for the solution of a specific problem. These issues have been illustrated in this thesis through the investigation of the cell placement problem, a *difficult* subproblem associated with a *single* design step in the circuit design procedure.

In this closing chapter, we summarize the contents of this thesis. The various contributions made towards the development of a better heuristic for cell placement are described. Contributions made on related subproblems encountered during the development of the placement heuristic are also described. We describe potential avenues for future research which we consider to be important for the enhancement of placement heuristics. Finally, closing comments are provided.

9.1 Summary and Contributions

In Chapter 2, cell placement for semi-custom design was described. Placement heuristics were classified as either constructive or iterative improvement methods, where the classification was shown to depend on how cell placements were obtained. Through a description of the advantages and disadvantages of the both methods, it was proposed that a *combination* of methods would prove beneficial as a means of exploiting the advantages of both constructive and iterative improvement methods, while avoiding the disadvantages associated with each method.

9.1.1 Constructive Placement

Chapters 3 through 6 were dedicated to the description of a constructive method for creating an initial legal placement. It was illustrated that several iterations of relative placement and circuit partitioning provided a good distribution of cell positions throughout the placement area. This iterative combination of different approaches provided *useful* information for positioning cells in an initial legal cell placement.

In Chapter 3, the relative placement problem was described as a means of determining the general positions of the cells while ignoring several placement restrictions. Two formulations of the relative placement problem were described. Depending on the estimate of wire length, the relative placement problem was formulated as both a quadratic program and a linear program. An analysis of the two formulations revealed that the quadratic program formulation resulted in smaller optimization problems (in terms of the number of unknowns, constraints and required storage) than the equivalent linear program formulation.

A primal-dual interior point method was proposed for solving both relative placement formulations in Chapter 4. Interior point methods have typically been ignored for solving the relative placement problem, despite their efficient implementations and polynomial-time complexities. However, when compared to previously proposed solution methodologies, the proposed interior point method was illustrated to represent a more efficient and flexible solution methodology since it is capable of handling a reasonably wide variety of constraints. Numerical results presented in Chapter 4 also confirmed the advantages of the quadratic program relative placement formulation in terms of computational efficiency and storage requirements when compared to the linear program formulation.

Several issues regarding the efficient implementation of the interior point method were also illustrated in Chapter 4. The computational "bottleneck" of the interior point method was identified as the solution of the augmented equations at each iteration of the method. Typically, direct methods have been proposed [46] for solving the augmented equations. However, we introduced the concept of *iterative methods* which were illustrated to offer several potential benefits in terms of computational efficiency and storage requirements when compared to their direct method counterparts. The importance of generating a good preconditioning matrix when using an iterative method was described and drop tolerance preconditioning was considered as an effective approach. Furthermore, it was illustrated that it was possible to *guarantee* the existence of the preconditioning matrix by applying diagonal modifications during the generation of the preconditioning matrix. Numerical results indicated the iterative method performed favourably in comparision with direct methods when applied to large, sparse optimization problems arising from the formulations of the relative placement problem. Iterative methods typically resulted in a reduction in the computational effort and storage requirements of the interior point method. Therefore, we concluded that for large and sparse problems such as the relative placement problem, which do not require a high degree of accuracy in the solution, iterative methods are beneficial.

In Chapter 5, it was illustrated that the solution of a single relative placement problem was not sufficient for providing *useful* information regarding final cell positions in an initial legal placement. Therefore, circuit partitioning was introduced as a means of improving the distribution of cells by forcing cells, in an intelligent manner, into under utilized portions of the placement area. The combination of relative placement and circuit partitioning was illustrated to result in an iterative procedure for determining "good" relative cell positions. Furthermore, the relative placement and circuit partitioning steps interacted with each other, where the information from one problem was used as input for improving the results of the other problem.

The creation of initial placements was illustrated in Chapter 6. Since the relative placement and circuit partitioning iterations provided a good indication of the initial cell positions, a simple and effective heuristic based on sorted cell positions was proposed for eliminating placement violations and creating an initial legal placement. Numerical results were presented to illustrate many aspects of the constructive placement method proposed in this thesis. Regardless of the relative placement formulation used, the solution of the optimization problems resulting from the formulation of the relative placement problem was identified as the computational "bottleneck" of the constructive placement method. Therefore, it was identified that any improvements in the interior point method or the relative placement formulations would be benficial for the computational efficiency of the overall placement heuristic.

Differences in the relative placement formulations were also illustrated. It was confirmed empirically that the quadratic program formulation of the relative placement problem required significantly less computational effort than the equivalent linear program formulation. However, the numerical results indicated that the linear program formulation resulted in initial placements with overall lower estimates of wire length. This represented an important observation regarding the constructive method, namely the potential trade-off in quality of the placement versus the required computational effort. Finally, it was also confirmed that a single relative placement was not sufficient for the creation of an initial legal placement. Numerical results indicated that the quality of the initial placements generated from a single relative placement problem were *substantially* worse than those obtained when several iterations of relative placement and circuit partitioning were performed.

9.1.2 Improved Placement

In Chapter 7, a simple iterative improvement method based on localized cell moves and cell swaps was proposed for repositioning cells to achieve a further improvment in the cell placement. The application of iterative improvement was illustrated to be necessary for the detailed placement of cells, since the constructive method only provided globally good information. Conversely, the iterative improvement method was illustrated to benefit from the exploitation of the a priori knowledge of the global goodness of the initial placement.

Numerical results indicated the computational effort required by the iterative improvement method was linear in its duration, whereas the improvement in the quality of the placement increased most rapidly during the initial portion of the iterative improvement. This was illustrated to be beneficial since a reduction in the computational effort of the iterative improvement method was possible by simply shorting its duration while still maintaining a reasonable improvement in the quality of the placement.

Comparisions with an established Simulated Annealing placement heuristic were presented to illustrated that the combination of constructive and iterative improvement methods was effective when compared to established methods. In all cases, our heuristic produced placements of comparable or superior quality, both in terms of the widths of the required placement areas and the total estimates of wire length. Additionally, our heuristic required less computational effort to produce these higher quality placements.

Comparisions of the final placements also illustrated several important aspects regarding the relative placement formulations. Although the initial placements provided by the linear program formulation of the relative placement problem were illustrated to be better than the equivalent quadratic program formulation, the application of iterative improvement eliminated the differences in quality. That is, the application of iterative improvement resulted in placements of equal quality regardless of the relative placement formulation. The savings in computational effort for the creation of the initial placement using the quadratic versus the linear program was reflected in the final computational times of the overall placement heuristic. Therefore,we concluded that our placement heuristic, with the relative placement problem formulated as a quadratic program, was the preferable placement heuristic.

9.1.3 Placement Enhancements

In Chapter 8, we considered circuit clustering as a means for reducing the computational effort required by the constructive placement method. By initially clustering the circuit, the solution to a sequence of smaller relative placement problems was required, implying an overall reduction in the computational effort and storage requirements. A greedy clustering heuristic based on circuit connectivity was proposed. The heuristic was illustrated to be capable of generating a large number of clusters, each containing a few cells, of nearly equal size. Furthermore, the complexity of the heurstic was illustrated to be polynomial in the size of the circuit. When applied to *practical* circuits, the heurstic was illustrated to be efficiently implemented with reasonable storage requirements.

Numerical results indicated that the proposed clustering heuristic was effective at reducing circuit sizes. Furthermore, when incorporated into the constructive placement method, circuit clustering was found to reduce computational efforts substantially, while *simultaneously* improving the quality of the resulting initial placements.

9.2 Future Directions

Many possible extensions may be applied to the basic techniques proposed in this thesis and we consider several possibilities in this section for enhancing and extending the placement heuristic.

9.2.1 Improved Cell Distribution

As illustrated in this thesis, the relative placement and circuit partitioning iterations are required to obtain a good distribution of cells throughout the placement area. It may prove useful to include additional constraints *directly* into the relative placement formulation (in addition to the first moment constraints) to prevent the grouping of cells towards the middle of the placement area. That is, additional "spreading constraints" incorporated into the relative placement problem may prove useful for eliminating overlap by effectively *forcing* cells apart. One example constraint (although it is likely that additional consideration may reveal "better" constraints) would be a second moment (variance) constraint applied to the cells within each region [48].

The consequence of including such constraints is twofold. By forcing cells further apart during the relative placement, the circuit partitioning may proceed in a more timely fashion since forcing cells apart results in less indecision during the creation of the initial partitions (since cells would have less tendency to collapse onto the cut lines). Therefore, it may be sufficient to create partitions of the cells and the placement area based strictly on sorted cell positions while avoiding the necessity of performing cell interchanges. Furthermore, additional spreading of cells into less utilized portions of the placement area would result in better initial partitions as well as a better indication of where cells belong in an initial placement earlier during the relative placement and partitioning iterations.

Several disadvantages may arise, however, when such constraints are included into the problem. Appropriate constraints for improving cell separation may be nonlinear implying modifications to the proposed interior point method. Furthermore, the constraints may be nonconvex implying that it may only be possible to guarantee a locally optimal solution to the relative placement problem. Despite these difficulties, the inclusion of constraints directly within the relative placement formulation to improve cell separation earlier on in the relative placement and circuit partitioning iterations would prove beneficial.

9.2.2 Interior Point Methods

Since the computational bottleneck of the constructive method is the solution of the relative placement problems, any additional enhancements to the solution methodology would be beneficial. For the interior point method, reduction in computational effort would require a reduction in the number of interior point iterations necessary to reach optimality or a reduction in the effort spent during each iteration.

Iterative versus direct methods for solving the augmented equations deserve more consideration. In general, iterative methods may not perform as well as their direct method counterparts on optimization problem requiring high degrees of accuracy (that is, many digits of agreement between the primal and dual objective values, feasibility norms approaching zero, and so forth). However, for optimization problems which are large, sparse and require only approximate solutions (such as the relative placement problem), iterative methods are useful.

Another interesting possibility is the extension of the interior point method to handle nonlinear (and possibly nonconvex) constraints. The development of such an interior point method would significantly enhance the flexiblity of the solution methodology and allow a greater variety of constraints to be included into the relative placement formulation.

9.2.3 Iterative Improvement

Iterative improvement methods are typically based on the application of *search heuristics* to repositions cells, and this is the approach which was followed in this thesis. Although it was illustrated that a simple iterative improvement method was capable of providing high quality final cell positions (assuming a globally good initial placement), enhancements may still be possible.

Additional computationally efficient search heuristics, such as GRASP [9] or genetic algorithms [2], for rearranging cells within localized portions of the placement area may prove useful. Additionally, incorporation of *meta-heuristics* such as Tabu Search [15] may prove useful guiding the search heuristic to better quality placements. The effects of improvements to the search heuristic used within the iterative improvement method would result in several benefits. First, by making more "intelligent" perturbations to an existing placement, better final placements would be possible without an increase in the required computational effort. Additionally, a reduction in the computational effort for a single perturbation would imply that more perturbations could be attempted over the entire duration of the iterative improvement method.

9.2.4 Performance Driven Placement

Finally, an important direction for the enhancement of the cell placement heuristic is in the direction of performance-driven placement [23, 38, 39, 42]. For combinational circuits, it is essential that a signal propagating along a net from a driver to a set of sinks arrives quickly enough in order to guarantee the proper functionality of the circuit. For instance, each functional element in a circuit has a *required arrival time* which represents the latest time that all signals must be present at the element's inputs in order to guarantee the element provides the correct outputs. Timing information is typically obtained from a timing analysis or a circuit simulation. This is especially important in today's high speed circuitry where timing violations are common occurances during the design of an integrated circuit, and can result in extended design cycles costing substantial time and money.

Previous research for performance driven placement has been done in the context of constructive placement methods [23, 38, 39, 42]. However, these previous approaches have used rather simplistic delay models to estimate the timing effects. Therefore, the inclusion of more accurate delay models (whether nonlinear, nonconvex, and so forth) into the relative placement formulation (and therefore the constructive placement method), along with a suitable solution methodology, would represent an extremely useful extension to the overall placement heuristic.

9.3 Epilogue

The investigation of the cell placement problem has proven to be an extremely interesting area of research. Application of the ideas and techniques presented within this thesis may also prove to be very effective when applied to other VLSI problems, and may extend to many other problems to which I have not yet been exposed.

The continuing evolution of mathematical programming techniques and advances in different types of search heuristics will continue to facilitate the creation of better, more powerful, algorithms for solving many different problems. As research never ceases to provide an avenue for both creativity and learning, I expect to be very busy in the years to come.

Appendix A

Derivation of the Augmented Equations

We are interested in deriving the augmented equations which yield the search direction at each iteration of the interior point method described in Chapter 4. The first order optimality conditions for the logarithmic barrier problems were shown to be given by

$$\begin{aligned}
\mathbf{Ax} + \mathbf{p} &= \mathbf{b} \\
\mathbf{x} + \mathbf{s} &= \mathbf{u} \\
\mathbf{A}^T\mathbf{y} + \mathbf{w} + \mathbf{r}_1 - \mathbf{Qx} &= \mathbf{c} \\
\mathbf{w} + \mathbf{r}_2 &= \mathbf{0} \\
\mathbf{y} + \mathbf{r}_3 &= \mathbf{0} \\
\mathbf{XR}_1 &= \mu\mathbf{e} \\
\mathbf{SR}_2 &= \mu\mathbf{e} \\
\mathbf{PR}_3 &= \mu\mathbf{e}
\end{aligned} \tag{A.1}$$

Applying Newton's method to these first order conditions results in the system of equations given by

$$\begin{aligned}
\mathbf{A}\Delta\mathbf{x}+\Delta\mathbf{p} &= \mathbf{b}-\mathbf{A}\mathbf{x}-\mathbf{p} &&:= \rho \\
\Delta\mathbf{x}+\Delta\mathbf{s} &= \mathbf{u}-\mathbf{x}-\mathbf{s} &&:= \tau \\
\mathbf{A}^T\Delta\mathbf{y}+\Delta\mathbf{w}+\Delta\mathbf{r}_1-\mathbf{Q}\Delta\mathbf{x} &= \mathbf{c}-\mathbf{A}^T\mathbf{y}-\mathbf{w}-\mathbf{r}_1+\mathbf{Q}\mathbf{x} &&:= \sigma \\
\Delta\mathbf{w}+\Delta\mathbf{r}_2 &= -\mathbf{w}-\mathbf{r}_2 &&:= \beta \\
\Delta\mathbf{y}+\Delta\mathbf{r}_3 &= -\mathbf{y}-\mathbf{r}_3 &&:= \gamma \\
\Delta\mathbf{x}+\mathbf{R}_1^{-1}\mathbf{X}\Delta\mathbf{r}_1 &= \mu\mathbf{R}_1^{-1}\mathbf{e}-\mathbf{X}\mathbf{e} &&:= \phi_1 \\
\Delta\mathbf{s}+\mathbf{R}_2^{-1}\mathbf{S}\Delta\mathbf{r}_2 &= \mu\mathbf{R}_2^{-1}\mathbf{e}-\mathbf{S}\mathbf{e} &&:= \phi_2 \\
\Delta\mathbf{p}+\mathbf{R}_3^{-1}\mathbf{P}\Delta\mathbf{r}_3 &= \mu\mathbf{R}_3^{-1}\mathbf{e}-\mathbf{P}\mathbf{e} &&:= \phi_3
\end{aligned} \tag{A.2}$$

This system of equations can be written in the following matrix form:

$$\begin{bmatrix}
\mathbf{R}_1^{-1}\mathbf{X} & & & \mathbf{I} & & & & \\
& \mathbf{R}_2^{-1}\mathbf{S} & & & \mathbf{I} & & & \\
& & \mathbf{R}_3^{-1}\mathbf{P} & & & \mathbf{I} & & \\
\mathbf{I} & & & -\mathbf{Q} & & & \mathbf{A}^T & \mathbf{I} \\
& \mathbf{I} & & & & & & \mathbf{I} \\
& & \mathbf{I} & & & & \mathbf{I} & \\
& & & \mathbf{A} & & \mathbf{I} & & \\
& & & \mathbf{I} & \mathbf{I} & & &
\end{bmatrix}
\begin{bmatrix} \Delta\mathbf{r}_1 \\ \Delta\mathbf{r}_2 \\ \Delta\mathbf{r}_3 \\ \Delta\mathbf{x} \\ \Delta\mathbf{s} \\ \Delta\mathbf{p} \\ \Delta\mathbf{y} \\ \Delta\mathbf{w} \end{bmatrix}
=
\begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \sigma \\ \beta \\ \gamma \\ \rho \\ \tau \end{bmatrix} \tag{A.3}$$

The pivot blocks $\mathbf{R}_1^{-1}\mathbf{X}$, $\mathbf{R}_2^{-1}\mathbf{S}$ and $\mathbf{R}_3^{-1}\mathbf{P}$ can be used to eliminate the variables $\Delta\mathbf{r}_1$, $\Delta\mathbf{r}_2$ and $\Delta\mathbf{r}_3$, respectively as follows:

$$\begin{aligned} \Delta\mathbf{r}_1 &= \mathbf{R}_1\mathbf{X}^{-1}(\phi_1 - \Delta\mathbf{x}) \\ \Delta\mathbf{r}_2 &= \mathbf{R}_2\mathbf{S}^{-1}(\phi_2 - \Delta\mathbf{s}) \\ \Delta\mathbf{r}_3 &= \mathbf{R}_3\mathbf{P}^{-1}(\phi_3 - \Delta\mathbf{p}) \end{aligned} \tag{A.4}$$

With these substitutions, the system of equations is reduced to the following:

$$\begin{bmatrix} -\mathbf{Q} - \mathbf{R}_1\mathbf{X}^{-1} & & & \mathbf{A}^T & \mathbf{I} \\ & -\mathbf{R}_2\mathbf{S}^{-1} & & & \mathbf{I} \\ & & -\mathbf{R}_3^{-1}\mathbf{P} & \mathbf{I} & \\ \mathbf{A} & & \mathbf{I} & & \\ \mathbf{I} & \mathbf{I} & & & \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{s} \\ \Delta\mathbf{p} \\ \Delta\mathbf{y} \\ \Delta\mathbf{w} \end{bmatrix} = \begin{bmatrix} \hat{\sigma} \\ \hat{\beta} \\ \hat{\gamma} \\ \rho \\ \tau \end{bmatrix}, \tag{A.5}$$

where

$$\begin{aligned} \hat{\sigma} &:= \sigma - \mathbf{R}_1\mathbf{X}^{-1}\phi_1, \\ \hat{\beta} &:= \beta - \mathbf{R}_2\mathbf{S}^{-1}\phi_2, \\ \hat{\gamma} &:= \gamma - \mathbf{R}_3\mathbf{P}^{-1}\phi_3. \end{aligned} \tag{A.6}$$

Next use the pivot blocks $-\mathbf{R}_2\mathbf{S}^{-1}$ and $-\mathbf{R}_3\mathbf{P}^{-1}$ to eliminate the variables $\Delta\mathbf{s}$ and $\Delta\mathbf{p}$, respectively as follows:

$$\begin{aligned} \Delta\mathbf{s} &= -\mathbf{R}_2^{-1}\mathbf{S}(\hat{\beta} - \Delta\mathbf{w}) \\ \Delta\mathbf{p} &= -\mathbf{R}_3^{-1}\mathbf{P}(\hat{\gamma} - \Delta\mathbf{y}) \end{aligned} \tag{A.7}$$

With these substitutions, the system of equations is reduced to the following:

$$\begin{bmatrix} -\mathbf{Q} - \mathbf{R}_1\mathbf{X}^{-1} & \mathbf{A}^T & \mathbf{I} \\ \mathbf{A} & \mathbf{R}_3^{-1}\mathbf{P} & \\ \mathbf{I} & & \mathbf{R}_2^{-1}\mathbf{S} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \\ \Delta\mathbf{w} \end{bmatrix} = \begin{bmatrix} \hat{\sigma} \\ \rho + \mathbf{R}_3^{-1}\mathbf{P}\hat{\gamma} \\ \tau + \mathbf{R}_2^{-1}\mathbf{S}\hat{\beta} \end{bmatrix} := \begin{bmatrix} \hat{\sigma} \\ \hat{\rho} \\ \hat{\tau} \end{bmatrix} \tag{A.8}$$

Finally, the pivot block $\mathbf{R}_2^{-1}\mathbf{S}$ can be used to eliminate the variables $\Delta\mathbf{w}$ as follows:

$$\Delta\mathbf{w} = \mathbf{R}_2\mathbf{S}^{-1}(\hat{\tau} - \Delta\mathbf{x}) \tag{A.9}$$

Making this final substitution, we arrive at the following system of equations:

$$\begin{bmatrix} -\mathbf{Q} - \mathbf{R}_1\mathbf{X}^{-1} - \mathbf{R}_2\mathbf{S}^{-1} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{R}_3^{-1}\mathbf{P} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \end{bmatrix} = \begin{bmatrix} \hat{\sigma} - \mathbf{R}_2\mathbf{S}^{-1}\hat{\tau} \\ \hat{\rho} \end{bmatrix} \tag{A.10}$$

This final system of equations is symmetric quasi-definite, which is the desired result.

Appendix B

Publications

The following publications are a consequence of the work done in this thesis.

- A. Vannelli, A. Kennings and P. Chin, Interior point approaches for the VLSI placement problem, In *Interior Point Methods of Mathematical Programming*, Ed. T. Terlaky, Kluwer Academic Publishers, pp. 501-528, 1996.

- A. Kennings and A. Vannelli, An efficient interior point approach for QP and LP models of the relative placement problem, In *Midwest Symposium on Circuits and Systems*, Ames, Iowa, August 18-21, 1996.

- A. Kennings and M. Frazer, Circuit clustering and its effects on a multiway circuit partitioning heuristic, In *Canadian Conference on Electrical and Computer Engineering*, St. John's, Newfoundland, May 25-28, 1997.

- A. Kennings and A. Vannelli, VLSI placement using quadratic programming and network partitioning techniques, Submitted *International Transactions on Operations Research*, 1996.

Bibliography

[1] M. A. Ajiz and A. Jennings. A robust incomplete choleski-conjugate gradient algorithm. *International Journal for Numerical Methods in Engineering*, 20:949–966, 1984.

[2] S. Areibi. *Towards optimal circuit layout using advanced search techniques.* PhD thesis, University of Waterloo, Waterloo, Ontario, 1995.

[3] E. R. Barnes. An algorithm for partitioning the nodes of a graph. *SIAM Journal of Algebraic and Discrete Methods*, 3(4):541–550, December 1982.

[4] J. R. Bunch and B. N. Parlett. Direct methods for solving symmetric indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 8:639–655, 1971.

[5] P. Chin and A. Vannelli. Computational methods for an LP model of the placement problem. Technical report, University of Waterloo, Waterloo, Ontario, 1994. UW E & C-94-02.

[6] J. K. Dickinson and P. A. Forsyth. Preconditioned conjugate gradient methods for three-dimensional linear elasticity. *International Journal for Numerical Methods in Engineering*, 37:2211–2234, 1994.

[7] K. Doll, F. M. Johannes, and K. J. Antreich. Iterative placement improvement by network flow methods. *IEEE Transactions on Computer-Aided Design of Intergrated Circuits and Systems*, 13(10):1189–1200, 1994.

[8] A. E. Dunlop and B. W. Kernighan. A procedure for placement of standard-cell vlsi circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 5:92–98, 1985.

[9] T. Feo, M. Resende, and S. Smith. A greedy randomized adaptive search procedure for the maximum independent set. *Operations Research*, 1994.

[10] R. Fourer and S. Mehrotra. Performance of an augmented system approach for solving least-squares problems in an interior point method for linear programming. *Mathematical Programming Society COAL Newsletter*, 19:26–31, 1991.

[11] J. Garbers, H. J. Promel, and A. Steger. Finding clusters in vlsi circuits. In *IEEE International Conference on Computer-Aided Design*, pages 520–523, 190.

[12] M. R. Garey and D. S. Johnson. *Computers and Intractability*. Freeman, San Francisco, CA, 1979.

[13] A. George, J. Liu, and E. Ng. The evolution of the minimum degree ordering algorithm. *SIAM Review*, 31:1–19, 1989.

[14] P. E. Gill, W. Murray, D. B. Ponceleon, and M. A. Saunders. Preconditioners for indefinite systems arising in optimization. *SIAM Journal on Matrix Analysis and Applications*, 13(1):292–311, 1992.

[15] F. Glover. Tabu search part I. *ORSA Journal on Computing*, 1(3):190–206, 1990.

[16] J. Gondzio. Presolve analysis of linear programs prior to applying an interior point method. Technical report, University of Geneva, Department of Management Studies, 1994. Technical Report 1994.3.

[17] S. W. Hadley, B. L. Mark, and A. Vannelli. An efficient eigenvector approach for finding netlist partitions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(7):885–892, July 1992.

[18] L. Hagen and A. B. Kahng. A new approach to effective circuit clustering. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 422–427, 1992.

[19] W. W. Hager. *Applied Numerical Linear Algebra*. Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[20] K. M. Hall. An r-dimensional quadratic placement algorithm. *Management Science*, 17(3):219–229, November 1991.

[21] T. Hamada, C. K. Cheng, and P. M. Chau. Prime: A timing-driven placement tools using a piecewise linear reisitive network approach. In *30th ACM/IEEE Design Automation Conference*, pages 531–536, 1991.

[22] T. C. Hu and E. S. Kuh. Theory and concepts of circuit layout. In T. C. Hu and E. S. Kuh, editors, *VLSI Circuit Layout: Theory and Design*, pages 3–18. IEEE Press, New York, 1985.

[23] M. A. B. Jackson and E. S. Kuh. Performance-driven placement of cell based ic's. In *26th ACM/IEEE Design Automation Conference*, volume 4, pages 370–375, 1989.

[24] B. W. Kerninghan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, pages 291–307, 1970.

[25] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich. GORDIAN: VLSI placement by quadratic programming and slicing optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(3):356–365, 1991.

[26] R. Kling and P. Banerjee. ESP: Placement by simulated evolution. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 8:245–256, 1989.

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

[28] K. Kozminski. Benchmarks for layout synthesis – evolution and current status. In *28th ACM/IEEE Design Automation Conference*, pages 265–270, 1991.

[29] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout.* J. Wiley, 1990.

[30] D. G. Luenberger. *Introduction to Linear and Nonlinear Programming.* Addison-Wesley Pub. Co., Reading, Mass., 1973.

[31] R. D. C. Monteiro, I. Adler, and M. G. C. Resende. A polynomial-time primal-dual affine scaling algorithm for linear and convex quadratic programming and its power series extensions. *Mathematics of Operations Research*, 15(2):191–214, 1990.

[32] R. H. J. M. Otten. Eigensolutions in top-down layout design. In *IEEE International Symposium on Circuits and Systems*, pages 602–608, 1982.

[33] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12:617–629, 1975.

[34] L. A. Sanchis. Multiple-way network partitioning. *IEEE Transactions on Computers*, 38(1):62–81, January 1989.

[35] L. A. Sanchis. Multiple-way network partitioning with different cost functions. *IEEE Transactions on Computers*, 42(12):1500–1504, December 1993.

[36] C. Sechen, K. W. Lee, B. Swartz, D. Chen, and M. Lee. *The TimberWolfSC Standard Cell Placement and Global Routing Package – User's Guide for Version 4.2c.* Yale University, 1987.

[37] G. Sigl, K. Doll, and F. M. Johannes. Analytical placement: A linear or quadratic objective function? In *23rd ACM/IEEE Design Automation Conference*, pages 57–62, 1991.

[38] A. Srinivasan. An algorithm for performance-driven initial placement of small-cell ics. In *28th ACM/IEEE Design Automation Conference*, pages 636–639, 1991.

[39] A. Srinivasan, K. Chaudhary, and E. S. Kuh. RITUAL: A performance driven placement for small-cell ics. In *International Conference on Computer-Aided Design*, pages 48–51, 1991.

[40] P. R. Suaris and G. Kedem. An algorithm for quadrisection and its application to standard cell placement. *IEEE Transactions on Circuits and Systems*, 35:294–303, 1988.

[41] W. J. Sun and C. Sechen. Efficient and effective placements for very large circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(3):349–359, 1995.

[42] R. S. Tsay and J. Koehl. An analytical net weighting approach for performance optimization in circuit placement. In *28th ACM/IEEE Design Automation Conference*, pages 620–625, 1991.

[43] H. A. van der Vorst. BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric lineat systems. *SIAM Journal on Scientific and Statistical Computing*, 13(2):631–644, 1992.

[44] R. J. Vanderbei. Symmetric quasi-definite matrices. Technical report, Princeton University, Princeton, NJ, 1991. SOR-91-10.

[45] R. J. Vanderbei. LOQO: An interior point code for quadratic programming. Technical report, Princeton University, Princeton, N.J., 1994.

[46] R. J. Vanderbei and T. J. Carpenter. Symmetric indefinite systems for interior point methods. *Mathematical Programming*, 58:1–32, 1993.

[47] A. Vannelli and S. W. Hadley. A Gomory-Hu cut tree representation of a netlist partitioning problem. *IEEE Transactions on Circuits and Systems*, 37(9):1133–1139, September 1990.

[48] A. Vannelli, A. Kennings, and P. Chin. Interior point approaches for the vlsi placement problem. In T. Terlaky, editor, *Interior Point Methods of Mathematical Programming*, chapter 13, pages 501–528. Kluwer Academic Publishers, 1996.

[49] B. X. Weis and D. A. Mlynski. A new relative placement procedure based on MSST and linear programming. In *IEEE International Symposium on Circuits and Systems*, volume 2, pages 564–567, 1987.

[50] G. Whale. *Data Structures and Abstraction using C.* PWS Publishing Company, Boston, Massachusetts, 1996.

[51] C. W. Yeh, C. K. Chen, and T. T. Y. Lin. A probabilistic multicommodity flow solution to the circuit clustering problem. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 428–431, 1992.