A Linear Programming-Based Algorithm for Floorplanning in VLSI Design

Jae-Gon Kim and Yeong-Dae Kim, *Member, IEEE*

***Abstract*—In this paper, we consider a floorplanning problem in the physical design of very large scale integration. We focus on the problem of placing a set of blocks (modules) on a chip with the objective of minimizing area of the chip as well as total wire length. The blocks have different areas and their shapes are either fixed (predetermined) or flexible (to be determined). We use the sequence-pair suggested by Murata *et al.* to represent the topology of nonslicing floorplans and present two methods to obtain a floorplan from a sequence-pair. One is a construction method, and the other is a method based on a linear programming model. The two methods are embedded in simulated annealing algorithms, which are used to find a near optimal floorplan. Results of computational experiments on the Microelectronics Center of North Carolina benchmark examples show that the proposed algorithms work better than existing algorithms.**

***Index Terms*—Floorplanning, linear programming (LP), sequence-pair, simulated annealing (SA).**

NOMENCLATURE

Parameters

n	Number of blocks.
m	Number of nets.
a_i	(Lower limit of) the area of block i.
S	Set of soft blocks.
H	Set of hard blocks.
E_k	Set of blocks included in net k.
p_i	Length of the shorter side of (hard) block i.
q_i	Length of the longer side of (hard) block i.
$\underline{\tau}_i, \bar{\tau}_i$	Lower and upper limits of the aspect ratio of block i.
$\underline{\tau}_c, \bar{\tau}_c$	Lower and upper limits of the aspect ratio of the chip.
α	Weight that specifies relative importance of chip size compared to total wire length in the objective function.
M	A very large positive number.

Decision Variables

w_i^b, h_i^b	Width and height of block i.
w_k^e, h_k^e	Width and height of the net-bounding box of net k.
x_i^{bl}, x_i^{br}	x coordinates of the left and right boundaries of block i.
y_i^{bb}, y_i^{bt}	y coordinates of the bottom and top boundaries of block i.
x_k^{el}, x_k^{er}	x coordinates of the left and right boundaries of the net-bounding box for net k.
y_k^{eb}, y_k^{et}	y coordinates of the bottom and top boundaries of the net-bounding box for net k.
w_c	Width of the chip.
h_c	Height of the chip.
r_{ij}	= 0 if block i is to be placed to the left of block j, i.e., $x_i^{br} \leq x_j^{bl}$, and 1 otherwise (if block i is free to be placed on any side of block j).
u_{ij}	= 0 if block i is to be placed below block j, i.e., $y_i^{bt} \leq y_j^{bb}$, and 1 otherwise.
v_i	= 0 if (hard) block i is placed horizontally, and 1 otherwise.

Manuscript received January 8, 2001; revised May 7, 2002. This paper was recommended by Associate Editor T. Yoshimura.

J.-G. Kim is with the School of Industrial Systems and Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: jgkim@isye.gatech.edu).

Y.-D. Kim is with the Department of Industrial Engineering, Korea Advanced Institute of Science and Technology, Yusong-gu, Daejon 305-701, Korea (e-mail: ydkim@mail.kaist.ac.kr).

Digital Object Identifier 10.1109/TCAD.2003.810748

I. INTRODUCTION

THIS PAPER focuses on the floorplan design problem of very large scale integrated (VLSI) circuits, which is the problem of placing a set of circuit blocks (modules) on a semiconductor chip to minimize chip size (area) and total wire length. Each block consists of several hundred or thousand cells performing logical or arithmetic operations such as AND, NAND, and flip-flops, and the area of each block is predetermined. Blocks are classified into two types according to their shape flexibility: hard blocks and soft blocks. Hard blocks are ones that are completely designed beforehand and have fixed shapes, while soft blocks are ones whose widths and heights are free to change as far as their aspect ratios are within given ranges. Here, the aspect ratio of a block is defined as the ratio of the height to the width of the block.

Because of its complexity, floorplanning is usually carried out in two steps, in which two decision problems are solved. In the first step, the relative positions of blocks (topology of the floorplan) are determined so that the total wire length is minimized, and in the second step, (exact) positions and dimensions (widths and heights) of the blocks are determined so that the chip size is minimized while the relative positions of the blocks being kept remain unchanged. In this study, the first problem is called the topology generation problem, and the second problem is called the floorplan area minimization problem (FAMP). Graph-based methods are often proposed for the first step [4], [10], [11], [24], while various optimal solution algorithms are suggested for the second step [2], [19], [22], [26], [27], [31]. Although the two-step approach has an advantage in that it can deal with problems with a large number of blocks, its solution quality may not be good since the two steps are done sequentially and decisions involved in the two steps are made relatively independently.

In most studies for the FAMP, it is assumed that each block has several possible alternatives for its dimension, which is defined by the width and height of the block. Recently, several attempts have been made to find optimal solutions for FAMPs in

which soft blocks have an infinite number of alternatives for dimensions within given ranges of aspect ratio. Moh *et al.* [14] formulate the problem as a geometric programming problem and find optimal solutions for slicing floorplans, while Young *et al.* [32] and Chen and Ku [1] use Lagrangian relaxation and linear programming (LP) approximation, respectively, for general nonslicing floorplans.

There have been efforts to minimize the total wire length and the chip size simultaneously in one step. Wong and Liu [28] propose a normalized Polish expression to represent slicing floorplans and use a simulated annealing (SA) algorithm to obtain a floorplan. Based on Wong and Liu's algorithm, Yamanouchi *et al.* [29] propose a hybrid floorplanning algorithm using partial clustering and module restructuring. Young *et al.* [33], [34] extend Wong and Liu's algorithm to handle cases in which some blocks should be adjacent to specific boundaries of a chip or should be placed within specific regions in the chip. To handle nonslicing floorplans, Murata *et al.* [16] and Murata and Kuh [17] suggest methods based on the sequence-pair, while Nakatake *et al.* [18] and Kang and Dai [7] propose methods based on the bound-sliceline-grid (BSG). In these methods, a floorplan topology is represented by a sequence-pair, which is a pair of sequences of block indexes (see the Appendix for more details), and a BSG, which is a plane dissected into several rectangles by horizontal or vertical line segments. Recently, Hong *et al.* [5] and Lin and Chang [12] suggested a corner block list (CBL) and a transitive closure graph (TCG), respectively, for topological representation of nonslicing floorplans.

This paper focuses on the floorplanning problem in which both hard blocks and soft blocks are to be placed within a chip for the objective of minimizing chip size and total wire length. In this study, we develop a mathematical model for the problem and use it to solve the problem, which was not done in most previous studies. We use the sequence-pair suggested by Murata *et al.* [16] to represent the topology of floorplans. We present two methods for generating a floorplan from a sequence-pair (a construction method and a method based on LP), in which positions of blocks, shapes of soft blocks, orientations of hard blocks, and width and length of a floor are determined. We also suggest an SA algorithm to find a sequence-pair that gives the best floorplan: one with the minimum weighted sum of total wire length and chip size.

The rest of this paper is organized as follows. In the next section, we describe the problem considered in this paper in detail and give a mixed integer nonlinear programming (MINLP) formulation. A construction method and an LP-based method are presented in Sections III and IV, respectively, and an SA algorithm is given in Section V. To test the performance of the methods, computational experiments are done on well-known benchmark problems, and results are reported in Section VI. Finally, Section VII concludes the paper with a short summary.

II. Problem Description

The floorplanning problem considered in this study is the problem of placing a set of circuit blocks on a semiconductor chip with the objective of minimizing chip size and total wire length. Here, the chip size is defined as the area of the smallest rectangle that encloses all blocks of the chip, while the total wire length is defined as the sum of wire lengths of nets. A net is composed of a set of pins through which blocks exchange electric signals among them and wires that connect those pins. Positions of pins within the shape of each hard block are predetermined and given (although pin positions in the chip should be determined), while positions of pins within soft blocks cannot be determined until exact shapes of the blocks are determined. Therefore, it is assumed that pins in soft blocks are located at the centers of the blocks as was done in other research [17], [33], [34]. To estimate the wire length of a net, we use the *half-perimeter estimation method* commonly used for the purpose. In this method, the wire length of a net is computed (estimated) as the half-perimeter of the smallest rectangle enclosing centers of all blocks in the net. In this paper, such a rectangle is called a *net-bounding box*.

In the problem, blocks are of rectangular shape, and their areas are given. Hard blocks have fixed shapes, but their spatial positions are to be determined. It is assumed that hard blocks can be placed either horizontally (the longer side is parallel to the x axis) or vertically (the longer side is parallel to the y axis). Soft blocks can have various shapes as far as their aspect ratios are within given ranges. The aspect ratio of the chip, i.e., the aspect ratio of the smallest rectangle that encloses all blocks of the chip, should also be within a predetermined range.

In the following, the floorplanning problem considered in this paper is presented formally as an MINLP. In the MINLP, it is assumed for the moment that pins of all blocks are located at the centers of the blocks to simplify the formulation (although exact positions of pins within hard blocks are considered when the total wire length is computed in the algorithm to be presented later in this paper).

$$[P1]\ \text{Minimize} \quad \sum_{k=1}^{m}(w_k^e + h_k^e) + \alpha \cdot w_c \cdot h_c$$

$$\text{subject to } x_k^{el} \le \frac{1}{2}(x_i^{bl} + x_i^{br}) \quad \forall i,k; i \in E_k \quad (1)$$

$$x_k^{er} \ge \frac{1}{2}(x_i^{bl} + x_i^{br}) \quad \forall i,k; i \in E_k \quad (2)$$

$$y_k^{eb} \le \frac{1}{2}(y_i^{bb} + y_i^{bt}) \quad \forall i,k; i \in E_k \quad (3)$$

$$y_k^{et} \ge \frac{1}{2}(y_i^{bb} + y_i^{bt}) \quad \forall i,k; i \in E_k \quad (4)$$

$$x_k^{er} - x_k^{el} = w_k^e \quad \forall k \quad (5)$$

$$y_k^{et} - y_k^{eb} = h_k^e \quad \forall k \quad (6)$$

$$x_i^{br} - x_i^{bl} = w_i^b \quad \forall i \quad (7)$$

$$y_i^{bt} - y_i^{bb} = h_i^b \quad \forall i \quad (8)$$

$$w_i^b = v_i \cdot p_i + (1 - v_i) \cdot q_i \quad \forall i \in H \quad (9)$$

$$h_i^b = (1 - v_i) \cdot p_i + v_i \cdot q_i \quad \forall i \in H \quad (10)$$

$$\underline{\tau}_i \le \frac{h_i^b}{w_i^b} \le \bar{\tau}_i \quad \forall i \in S \quad (11)$$

$$w_i^b \cdot h_i^b \ge a_i \quad \forall i \in S \quad (12)$$

$$r_{ij} + r_{ji} + u_{ij} + u_{ji} \le 3 \quad \forall i < j \quad (13)$$

$$x_i^{br} \le x_j^{bl} + M \cdot r_{ij} \quad \forall i,j; i \ne j \quad (14)$$

$$y_i^{bt} \le y_j^{bb} + M \cdot u_{ij} \quad \forall i,j; i \ne j \quad (15)$$

$$w_c \geq x_i^{br} - x_j^{bl} \quad \forall i,j \tag{16}$$

$$h_c \geq y_i^{bt} - y_j^{bb} \quad \forall i,j \tag{17}$$

$$\underline{\tau}_c \leq \frac{h_c}{w_c} \leq \bar{\tau}_c \tag{18}$$

$$v_i, r_{ij}, u_{ij} \in \{0,1\} \tag{19}$$

$$w_i^b, h_i^b, w_k^e, h_k^e, x_i^{bl}, x_i^{br}, y_i^{bb}, y_i^{bt}, x_k^{el}$$
$$x_k^{er}, y_k^{eb}, y_k^{et}, w_c, h_c \geq 0. \tag{20}$$

The objective is to minimize the weighted sum of total wire length and chip size. Constraints (1)–(4) are used to define the x and y coordinates of boundaries of net-bounding boxes, and constraints (5) and (6) are used to define widths and heights of net-bounding boxes, while constraints (7) and (8) are used to define widths and heights of blocks. Constraints (9) and (10) ensure that hard blocks have fixed shapes but can be placed either horizontally or vertically. In addition, constraint (11) ensures that aspect ratios of soft blocks are within given ranges, and constraint (12) ensures that soft blocks satisfy their area requirements. Constraints (13)–(15), which were used in the formulation of Montreuil [15] as well, prevent overlaps of blocks by letting each pair of blocks be separated in the x or y direction. Constraints (16) and (17) are used to define the width and the height of the chip, respectively, and constraint (18) ensures that the aspect ratio of the chip is within a given range.

In [P1], w_i^b, h_i^b, w_k^e, and h_k^e can be eliminated if they are replaced with $x_i^{br} - x_i^{bl}$, $y_i^{bt} - x_i^{bb}$, $x_k^{er} - x_k^{el}$, and $y_k^{et} - x_k^{eb}$, respectively, and constraints (5)–(8) are not needed by such replacements. However, in the above model (and throughout this paper), these variables are used instead of longer expressions to make the model easier to understand. (Note that those redundant variables are not used in computer codes for the algorithms developed in this study.)

III. Construction Method

A construction method is used to obtain a floorplan quickly from a given sequence-pair $\Gamma \equiv (\Gamma^+, \Gamma^-)$. In the construction method, dimensions (widths and heights) of blocks are determined one by one, and the blocks are placed to generate a floorplan with the minimum area (smallest chip size). A complete floorplan is obtained when dimensions and positions of all blocks are determined. Note that the dimensions specify aspect ratios for soft blocks and orientations for hard blocks. Blocks are selected in a nonincreasing order of the *block-effect value*, which is defined as $a_i(q_i/p_i)$ for hard blocks ($i \in H$) and $a_i \cdot \max(1/\underline{\tau}_i, \bar{\tau}_i)$ for soft blocks ($i \in S$). Large blocks and blocks with very large or very small aspect ratios have large block-effect values. Note that the chip size tends to be more affected by blocks with larger block-effect values.

To determine dimensions (widths and heights) of blocks, we consider several candidates for each block. For hard blocks, two candidates are considered since hard blocks can be placed in two orientations: horizontally and vertically. On the other hand, five candidates are considered for each soft block, though there are an infinite number of possible dimensions. Let $\hat{w}_{ij}$ and $\hat{h}_{ij}$ denote the width and height of block i in its jth candidate dimension, respectively. In this study, the value of $\hat{w}_{ij}$ is determined as follows. Note that $\hat{h}_{ij}$ can be determined as $\hat{h}_{ij} = a_i/\hat{w}_{ij}$, once $\hat{w}_{ij}$ is given.

For hard blocks (if $i \in H$)

let $\hat{w}_{i1} = p_i$, and $\hat{w}_{i2} = q_i$

For soft blocks (if $i \in S$)

let
$$\hat{w}_{i1} = \sqrt{\frac{a_i}{\underline{\tau}_i}}$$
$$\hat{w}_{i2} = 0.75\sqrt{\frac{a_i}{\underline{\tau}_i}} + 0.25\sqrt{\frac{a_i}{\bar{\tau}_i}}$$
$$\hat{w}_{i3} = 0.5\sqrt{\frac{a_i}{\underline{\tau}_i}} + 0.5\sqrt{\frac{a_i}{\bar{\tau}_i}}$$
$$\hat{w}_{i4} = 0.25\sqrt{\frac{a_i}{\underline{\tau}_i}} + 0.75\sqrt{\frac{a_i}{\bar{\tau}_i}}, \text{ and}$$
$$\hat{w}_{i5} = \sqrt{\frac{a_i}{\bar{\tau}_i}}.$$

In the above five candidates for soft block i, the first candidate has the smallest aspect ratio ($\underline{\tau}_i$), and the fifth candidate has the largest aspect ratio ($\bar{\tau}_i$).

We evaluate each candidate for the dimension of a selected block by generating floorplans using the sequence-pair evaluation algorithm of Tang *et al.* [23], which runs in $O(n \log n)$ time. The following summarizes the procedure used to obtain a floorplan for a given sequence-pair in the construction method.

Procedure 1.

step 0) Let $w_i^b = h_i^b = p_i$ for all $i \in H$, and $w_i^b = h_i^b = \min(\hat{w}_{i5}, \hat{h}_{i1})$ for all $i \in S$.

step 1) Select a block, say block $\hat{i}$, with the maximum block effect value, i.e., $\hat{i} = \arg\max[\max_{i \in H}\{a_i q_i/p_i\}, \max_{i \in S}\{a_i/\underline{\tau}_i, a_i\bar{\tau}_i\}]$.

step 2) For each candidate (j) for dimensions of block $\hat{i}$, let $w_{\hat{i}}^b = \hat{w}_{\hat{i}j}$ and $h_{\hat{i}}^b = \hat{h}_{\hat{i}j}$, and obtain a floorplan using the algorithm of Tang *et al.* **[23]**.

step 3) Select a candidate, say candidate $\hat{j}$, that gives the best floorplan. Let $w_{\hat{i}}^b = \hat{w}_{\hat{i}\hat{j}}$ and $h_{\hat{i}}^b = \hat{h}_{\hat{i}\hat{j}}$. If dimensions of all blocks are determined, obtain a floorplan using the single-pass algorithm and stop. Otherwise, go to step 1.

If more candidates are considered for the dimension of each soft block in the construction method, a longer computation time is required, although a better floorplan can be obtained. In this research, the number of candidates to be considered was determined after a series of preliminary tests considering both solution quality and computation time.

IV. LP-Based Method

Although the construction method can quickly generate a floorplan from a sequence-pair, the floorplan may not be good since dimensions of blocks are determined one by one, not as a whole, and only a small number of candidates are considered for dimensions of soft blocks in the construction method. In this section, we present another method, which is based on a mathematical model. In this method, [P1], the MINLP model presented in Section II, is used. It is very hard to solve [P1]

optimally since [P1] contains a large number of binary variables and nonlinear terms in the objective function and constraints. In the method suggested here, we fix binary variables, linearize nonlinear terms and remove redundant constraints in [P1] to obtain an LP model, which is used to obtain a floorplan from a sequence-pair.

A. Fixing Binary Variables

Once a sequence-pair is given, binary variables r_{ij} and u_{ij} in [P1] can be fixed at zero or one according to the given sequence-pair. For example, when $\Gamma = (1,3,2;2,1,3)$, r_{13}, u_{21}, and u_{23} can be fixed at zero, and $r_{12}, r_{21}, r_{23}, r_{31}, r_{32}, u_{12}, u_{13}$, u_{31}, and u_{32} can be fixed at one. Therefore, constraint (13) is no longer needed and can be deleted. On the other hand, binary variables $v_i s$, which are used to represent orientations of hard blocks, are fixed using a floorplan obtained with the construction method from the given sequence-pair. If block i ($i \in H$) is placed horizontally in the floorplan, v_i is set to zero, and v_i is set to one otherwise.

B. Linearizing Nonlinear Terms

The nonlinear term in the objective function is linearized by approximating the area of the chip with the perimeter of the chip, as was done in Yamazaki *et al.* [30]. Note that the area of a rectangle tends to increase (or decrease) as the perimeter of the rectangle increases (or decreases). In this study, $w_c \cdot h_c$ is replaced with $(w_c + h_c)$ in the objective function, and hence, the objective function is changed to $\sum_{k=1}^{m} (w_k^e + h_k^e) + \beta \cdot (w_c + h_c)$, where β is a constant that specifies relative importance between the total wire length and the perimeter of the chip. Note that β should be determined in such a way that $\beta \cdot (w_c + h_c)$ has a value close to $\alpha w_c h_c$. We set $\beta = \alpha\sqrt{A}/2$, assuming $w_c = h_c = \sqrt{A}$, where A is the sum of areas of the blocks.

In addition, nonlinear constraints (12) are linearized by using their surrogate constraints $2(w_i^b + h_i^b) \geq 3\sqrt{a_i} + \lambda_i \cdot s_i$ for $i \in S$, where $s_i = \max(w_i^b, h_i^b)$, and λ_i is a parameter of which the value must be determined for each soft block considering a given range of the aspect ratio of the block. In the surrogate constraint, $2(w_i^b + h_i^b)$ represents the perimeter of (soft) block i, while $3\sqrt{a_i} + \lambda_i \cdot s_i$ represents a lower bound on the perimeter that satisfies the constraint associated with the area of the block. These surrogate constraints are used based on the presumption that the perimeter of a soft block should be increased to maintain the area of the block if s_i (or the aspect ratio) increases.

Feasibility of floorplans cannot be guaranteed if the above surrogate constraints are used since constraints associated with the areas of soft blocks may be violated. Although we can obtain a feasible floorplan by letting $\lambda_i s$ have large values, the objective function value will be increased since the areas of the blocks are overestimated. Therefore, $\lambda_i s$ should be set in such a way that the constraints associated with areas of soft blocks are satisfied as tightly as possible. How values for $\lambda_i s$ are selected will be presented later in this section. Those surrogate constraints were first used by Meller *et al.* [13], in which the same value (λ) was used for the parameters ($\lambda_i s$) of all blocks. Note that we can approximate the areas of soft blocks more exactly by using different values for $\lambda_i s$ for different blocks.

C. Removing Redundant Constraints

Some of the overlap avoidance constraints in [P1], i.e., (14) and (15), are redundant and can be removed. For example, when $\Gamma^+ = (1,3,2,4)$ and $\Gamma^- = (2,1,3,4)$, the overlap avoidance constraint that defines relative positions in the x axis between blocks 1 and 2 is not needed since block 1 is to be placed above block 2 according to the sequence-pair. Also, the overlap avoidance constraints between 1 and 4 can be removed since overlap of blocks 1 and 4 can be avoided by overlap avoidance constraints between blocks 1 and 3 and between blocks 3 and 4. In other words, overlap avoidance constraints between blocks 1 and 4 can be removed if there is a block, say block 3, that should be placed between the two blocks. Similarly, one can delete overlap avoidance constraints that define relative positions in the y axis among the blocks. In the following, we present a systematic procedure for removing redundant constraints.

Procedure 2.

Step 1) Find pairs of blocks that are separated in the x direction in the given sequence-pair, and delete overlap avoidance constraints in the y direction between them. Similarly, find pairs of blocks that are separated in the y direction and delete overlap avoidance constraints in the x direction between them.

Step 2) For all pairs of blocks i and j, check if there is a block whose index is between i and j in both Γ^+ and Γ^-. If there is such a block, eliminate overlap avoidance constraints between blocks i and j.

D. Linear Program

Using the above procedures, [P1] can be reduced to the following linear program [P2], once a sequence-pair is given. Let $\hat{v}_i$ denote the value of v_i that is fixed by the method given above, and let R_x and R_y be the sets of pairs of blocks whose overlap avoidance constraints in the x and y directions are found to be redundant, respectively.

$$[P2]\ \text{Minimize} \quad \sum_{k=1}^{m} (w_k^e + h_k^e) + \beta \cdot (w_c + h_c)$$

$$\text{subject to} \quad (1) - (9),\ (16) - (18),\ (20),\ \text{and}$$

$$w_i^b = \hat{v}_i \cdot p_i + (1 - \hat{v}_i) \cdot q_i \quad \forall i \in H \tag{21}$$

$$h_i^b = (1 - \hat{v}_i) \cdot p_i + \hat{v}_i \cdot q_i \quad \forall i \in H \tag{22}$$

$$2(w_i^b + l_i^b) \geq 3\sqrt{a_i} + \lambda_i \cdot s_i \quad \forall i \in S \tag{23}$$

$$s_i \geq w_i^b \quad \forall i \in S \tag{24}$$

$$s_i \geq l_i^b \quad \forall i \in S \tag{25}$$

$$x_i^{br} \leq x_j^{bl} \quad \forall i,j; (i,j) \notin R_x \tag{26}$$

$$y_i^{bt} \leq y_j^{bb} \quad \forall i,j; (i,j) \notin R_y. \tag{27}$$

E. Obtaining a Feasible Floorplan

[P2] can be solved optimally even for large-sized problems using a commercial software package for linear programs. As stated earlier, a floorplan generated from an optimal solution of [P2] may not be feasible since constraints (23)–(25) are not exact constraints but surrogate constraints for (12), which is the

constraint associated with the areas of blocks. In this study, the following procedure is used to obtain a feasible floorplan. In the procedure, $I_{\max}$, δ, and ρ are parameters whose values should be determined, and I and $\hat{a}_i$ denote the iteration count and the area of block i in the solution of [P2], respectively.

Procedure 3.

Step 1) Let $I = 0$ and $\lambda_i = 1$ for all $i \in S$, and solve [P2]. Let $g_{\min} = \sum_i (a_i - \hat{a}_i)$ and $\lambda_i^* = \lambda_i$ for all i.

Step 2) If $I > I_{\max}$ or $|\hat{a}_i - a_i| / a_i < \delta$ for all $i \in S$, go to step 4. Otherwise, let $I = I + 1$, and go to step 3.

Step 3) Let $\lambda_i = (a_i/\hat{a}_i)^\rho \cdot \lambda_i$ for all $i \in S$, and solve [P2]. If $\sum_i (a_i - \hat{a}_i) < g_{\min}$, let $g_{\min} = \sum_i (a_i - \hat{a}_i)$ and $\lambda_i^* = \lambda_i$ for all $i \in S$. Go to step 2.

Step 4) Let $\lambda_i = \lambda_i^*$ for all $i \in S$, and solve [P2]. If $\hat{a}_i \geq a_i$ for all $i \in S$, stop. Otherwise, go to step 5.

Step 5) For all $i \in S$, let $\lambda_i^* = (a_i/\hat{a}_i)^\rho \cdot \lambda_i^*$ if $\hat{a}_i < a_i$. Go to step 4.

In the procedure, if the area of soft block i is smaller (larger, respectively) than its lower limit in the solution of [P2], λ_i is increased (decreased). The amount of the increase or decrease is controlled by a parameter ρ. After a series of preliminary tests, $I_{\max}$, δ, and ρ are set to 20, 0.001, and 0.35, respectively, considering solution quality and computation time.

V. Simulated Annealing (SA) Algorithm

In this study, we use an SA algorithm to find a sequence-pair which gives the best floorplan. (The construction method or the LP-based method is used to obtain a floorplan from a sequence-pair.) In SA algorithms, an initial solution is repeatedly improved by making small alterations until further improvements cannot be made by such alterations. Unlike greedy-type local search algorithms, SA algorithms can avoid entrapment in a local minimum by allowing occasional uphill moves which deteriorate the objective function value. The uphill move is allowed with the probability given by $\exp(-\Delta/T)$, where T is a control parameter called the *temperature*, and Δ is the difference between objective function values of the current and neighborhood solutions. The temperature is initially set with a certain method and gradually lowered in a predetermined method, called the *cooling schedule*.

The following shows how the SA algorithm is implemented for the floorplanning problem.

A. Objective Function

Once a floorplan is generated (from a sequence-pair) by the construction method or LP-based method, the total wire length is newly calculated while taking account of exact positions of pins within the hard blocks and chip input–output (I/O) pads. To obtain exact pin positions for the hard blocks, we compare four possible pin positions (current, 180° rotation, flip, flip and 180° rotation) for each hard block without changing the width and height of the block and select the best pin position. In addition, to consider I/O pads on the boundaries of the chip, net-bounding boxes for nets which contains I/O pads are extended so that their areas include the I/O pads.

A floorplan generated by the construction method always satisfies all constraints of the problem except for the shape constraint of the chip, i.e., the constraint associated with the aspect ratio of the chip. (Note that the LP-based method always generates a feasible layout for a given sequence-pair.) In the suggested SA algorithm, the shape constraints are converted into a penalty so that feasible solutions can be obtained in the search procedure. That is, the objective function value of a solution in the SA algorithm is computed as $\left\{\sum_{k=1}^{m}(\tilde{w}_k^e + \tilde{h}_k^e) + \alpha \cdot w_c \cdot h_c\right\} \cdot (1 + 0.1\zeta)$, where $\tilde{w}_k^e$ and $\tilde{h}_k^e$ are, respectively, the width and height of a net-bounding box of net k that are (newly) calculated by considering the I/O pads, and $\zeta = \max(0, h_c/w_c - \bar{\tau}_c, \underline{\tau}_i - h_c/w_c)$ is a value indicating how much the aspect ratio of the chip deviates from a given range.

B. Neighborhood Generation

A neighborhood solution of the current solution is generated by the following three methods.

1) Two blocks are randomly selected and they are interchanged in both Γ^+, and Γ^-.
2) Two blocks are randomly selected, and they are interchanged either in Γ^+ or Γ^- (with equal probability).
3) Two blocks i and j are randomly selected, and block i is moved just before block j either in Γ^+ or Γ^- (with equal probability).

In each iteration of the suggested SA algorithm, the three methods are randomly selected with probabilities 0.3, 0.4, and 0.3.

C. Cooling Schedule

In general, cooling schedules can be defined by the initial temperature, the epoch length, and the method to decrease the temperature. In the suggested algorithm, the initial temperature T_0 is chosen in such a way that the fraction of accepted uphill moves in a trial run of the annealing process becomes approximately F_0. In the SA algorithm, $3n$ moves are made and the average increase in the objective function value $\overline{\Delta}$ is calculated with uphill moves only, and then T_0 is obtained from an equation $\exp(-\overline{\Delta}/T_0) = F_0$. The epoch length specifies the number of moves made with the same temperature. In the suggested algorithm, the epoch length is set to $\varepsilon \cdot n$, where ε is a parameter to be determined. The temperature is decreased in such a way that the temperature at the kth epoch is given by $T_k = \gamma T_{k-1}$, where γ is a parameter, called the cooling ratio, with a value less than one.

D. Stopping Condition

Among various methods to determine when to terminate the search procedure, we adopt a method given by Johnson *et al.* [6]. In this method, a counter is increased by one when an epoch is completed with the fraction (or percentage) of accepted moves less than a predetermined limit (ζ) and the counter is reset to 0

when a new incumbent solution is found. The SA algorithm is terminated when the counter reaches a given limit (ξ).

E. Parameters

To find appropriate values for the five parameters, F_0, ε, γ, ζ, and ξ in the SA algorithm, we use a procedure developed by Park and Kim [20]. The procedure employs the simplex method for nonlinear programming to find good parameter values relatively quickly without much human intervention. See Park and Kim [20] for more details of the procedure. In this study, 0.7, 3, 0.95, 0.001, and 1 were selected for the values of F_0, ε, γ, ζ, and ξ, respectively.

VI. Computational Experiments

There are two SA algorithms developed in this study, since two methods (the construction method and the LP-based method), are used for generating a floorplan from a sequence-pair. SA algorithms embedding the construction method and the LP-based method are denoted as SA-CT and SA-LP, respectively. Since it is expected that the LP-based method generates a better floorplan than the construction method from the same sequence-pair, we may obtain a better final floorplan if we apply the LP-based method to the best sequence-pair found by SA-CT. Therefore, the LP-based method is used to improve the final floorplan obtained by SA-CT in a new algorithm, SA-CT+LP.

In this study, we first compare two methods for generating a floorplan from a sequence-pair. Then, we compare SA-CT+LP and SA-LP with the algorithms of Yamanouchi *et al.* [29], Hong *et al.* [5], Lin and Chang [12], and Murata and Kuh [17] on the Microelectronics Center of North Carolina (MCNC) benchmark examples. SA-CT was not included in the tests since SA-CT+LP always gives better (or at least equal) floorplans than SA-CT but requires negligibly short additional computation time. For the algorithms suggested in this study, α and β, coefficients in the objective function were set to $m/2\sqrt{A}$ and $m/4$, respectively. The tests were done on a personal computer with a 500-MHz Pentium III processor and CPLEX 6.0 was used to solve linear programs in the algorithms.

A. Comparison of the Construction Method and the LP-Based Method

Performance of the construction method and the LP-based method was compared on randomly generated sequence-pairs. For this comparison, we randomly generated 35 problems, in which there are six levels for the number of soft blocks (0, 10, 20, 30, 40, and 50) and six levels for the number of hard blocks (0, 10, 20, 30, 40, and 50). Areas of the soft blocks were randomly generated from DU(1, 10), which is the discrete uniform distribution with range [1, 10], and the range of aspect ratio was set to (0.25, 4.0). For each hard block, the length of the shorter side was randomly generated from DU(1, 4), and then the length of the longer side was generated by adding a random number generated from DU(0, 3) to the length of the shorter side. The number of nets was generated by multiplying the number of blocks by a random number generated from DU(3, 8). The number of blocks included in each net was randomly generated from DU(1, $\lceil n/2 \rceil$), where $\lceil n/2 \rceil$ is the smallest integer that is not less than $n/2$. Blocks were randomly selected and included in each net but in such a way that the number of blocks included in each net became equal to the predetermined value. The lower and upper limits on the aspect ratio of the chip were set to 0.2 and 5.0, respectively.

TABLE I
Average Ratio of the Solution Values of the LP-Based Method to Those of the Construction Method

Number of soft blocks	Number of hard blocks					
	0	10	20	30	40	50
0	-	0.75	0.79	0.83	0.70	0.77
10	0.62	0.67	0.69	0.62	0.70	0.81
20	0.89	0.57	0.78	0.76	0.75	0.82
30	0.59	0.76	0.59	0.61	0.74	0.80
40	0.70	0.56	0.78	0.73	0.61	0.73
50	0.67	0.55	0.65	0.72	0.75	0.73

TABLE II
Comparisons of CPU Times (in Seconds) of the Construction and LP-Based Methods

	Number of soft blocks	Number of hard blocks					
		0	10	20	30	40	50
Construction method	0	-	0.001	0.001	0.002	0.003	0.005
	10	0.001	0.002	0.003	0.004	0.006	0.008
	20	0.002	0.003	0.005	0.007	0.009	0.012
	30	0.004	0.005	0.007	0.010	0.014	0.018
	40	0.006	0.008	0.012	0.015	0.018	0.026
	50	0.011	0.014	0.020	0.023	0.029	0.034
LP-based method	0	-	0.09	0.41	2.27	1.43	6.10
	10	0.57	1.27	2.16	3.03	7.47	14.8
	20	3.89	4.09	15.7	19.8	24.4	36.1
	30	5.68	23.7	8.76	6.78	18.4	45.3
	40	33.2	6.12	39.0	35.5	18.3	34.6
	50	42.4	11.4	28.4	37.2	68.1	54.0

We generated ten sequence-pairs for each problem and obtained floorplans from them using the construction method and the LP-based method. Tables I and II show the average ratio of the solution value of the LP-based method to that of the construction method and the average CPU times required to obtain a floorplan with the two methods.

As expected, the LP-based method gave much better (over 30% better on average) floorplans than the construction method. The LP-based method gave better floorplans than the construction method for all 350 tested sequence-pairs. The construction method required very short CPU time. The computation times of the two methods are more affected by the number of soft blocks than by that of hard blocks. This is because we consider more candidates for the dimension of a soft block than those of a hard block (five versus two) to generate a floorplan in the construction method. Also, in the LP-based method, as the number of soft blocks increases, the number of alternative floorplans increases very quickly and more LPs should be solved to satisfy the constraints associated with areas of soft blocks.

B. Comparison With the Algorithm of Yamanouchi et al. [29]

Next, SA-CT+LP and SA-LP are compared with the algorithm of Yamanouchi *et al.* [29], which is denoted by YTK in this paper. Note that YTK was developed to solve problems in which there were hard blocks only, and it showed better performance than the algorithm of Murata *et al.* [16]. We used the largest MCNC floorplan benchmark example (ami49) for the

TABLE III
COMPARISON OF THE SUGGESTED ALGORITHMS WITH YTK

	Width (μ)	Height (μ)	Area (mm^2)	Wire length (mm)	CPU time (seconds)
YTK	5824	6440	37.6	723.9	56.3
SA-CT+LP	6524	5754	37.5	752.7	158.7
SA-LP	5628	6734	37.9	701.7	16683.5

The test on YTK was done on a Sun Sparc Station 20 while the tests on the other two algorithms were done on a personal computer with a 500MHz Pentium processor.

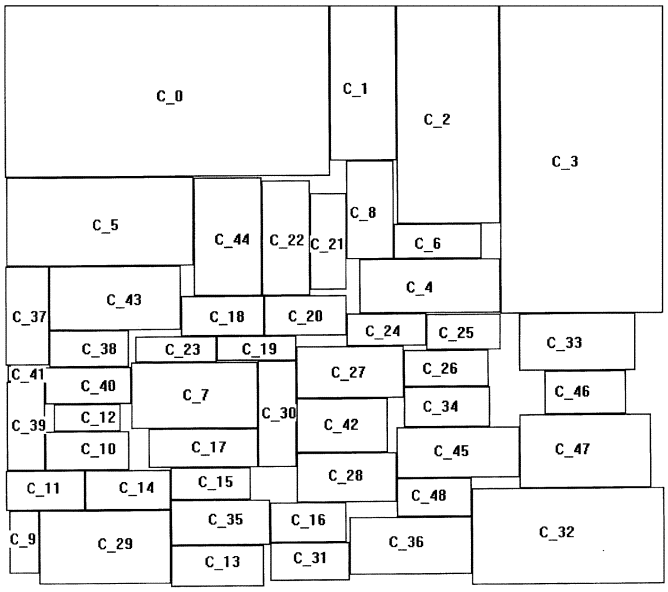

Fig. 1. Final floorplan obtained by SA-CT+LP for ami49.

comparison since it was used for testing performance of YTK in Yamanouchi *et al.* [29]. In ami49, the numbers of (hard) blocks and nets are 49 and 408, respectively. In the test problem, the lower and upper limits on the aspect ratio of the chip were set to 0.8 and 1.25, respectively, as was done in Yamanouchi *et al.* [29].

Table III shows results of the test. Note that the test result of YTK was obtained from a Sun Sparc Station 20. SA-CT+LP gave a floorplan with a slightly less chip area than that from YTK although the total wire length was about 4% longer. On the other hand, SA-LP gave a floorplan with a slightly larger chip area but at about 3% shorter total wire length compared with the floorplan obtained from YTK. It seems that there was not much difference in the solution quality between the three algorithms on the test problem in which there are only hard blocks. Fig. 1 shows the final floorplan obtained by SA-CT+LP for ami49.

C. Comparison With the Algorithms of Hong et al. [5] and Lin and Chang [12]

The suggested algorithms are compared with the algorithms of Hong *et al.* [5] and Lin and Chang [12], denoted here by CBL and TCG, respectively, on five MCNC benchmark examples. CBL and TCG use a corner block list and a transitive closure graph, respectively, for topological representation of nonslicing floorplans, and they use SA algorithms to find the best solution. In the test problem, the lower and upper limits on the aspect ratio of the chip were set to 0 and ∞, respectively, since they were not considered in [5] and [12].

TABLE IV
COMPARISON OF THE SUGGESTED ALGORITHMS WITH CBL AND TCG

		apte (9, 97)†	xerox (10, 203)	hp (11, 83)	ami33 (33, 123)	ami49 (49, 408)
CBL	Area (mm^2)	47.43	20.23	NA	1.226	38.38
	Wire length (mm)	194.95	403.47	NA	51.67	732.84
	CPU seconds	NA	NA	NA	NA	NA
TCG	Area	48.48	20.42	9.49	1.237	38.20
	Wire length	378.0	385.0	151.8	50.29	663.1
	CPU seconds	49	114	59	939	3613
SA-CT+LP	Area	47.30	20.42	9.26	1.197	37.75
	Wire length	294.33	430.97	141.12	41.08	625.88
	CPU seconds	1.13	2.31	2.16	65.21	180.2
SA-LP	Area	47.44	20.42	9.25	1.227	37.54
	Wire length	275.57	443.73	138.3	46.14	752.4
	CPU seconds	298.2	589.23	501.1	2902.1	16198.4

The test on TCG was done on a 400MHz Sun Sparc Ultra-60 workstation while the tests on the suggested algorithms were done on a personal computer with a 500MHz Pentium processor.

† a and b in (a, b) denote the numbers of (hard) blocks and nets, respectively.

Table IV shows results of the test. Solution values of problem hp and CPU times for CBL are not given here since they were not reported in [5], and test results of TCG were obtained from a 433-MHz SUN Sparc Ultra-60 workstation [12]. In general, SA-CT+LP and SA-LP gave better solutions (in terms of both the total wire length and the chip area) than CBL and TCG for the large sized problems including hp, ami33, and ami49 but similar or worse solutions for the rest of the problems. There was no significant difference in the solution qualities of SA-LP and SA-CT+LP for the problems because the LP-based method is based on the result of the construction method for the hard-block problems in which dimensions of the blocks are predetermined.

D. Comparison With the Algorithms of Murata and Kuh [17]

The suggested algorithms are compared with two algorithms of Murata and Kuh [17], known as EXACT and DIS2+POST on five (modified) MCNC benchmark examples (apte-s, xerox-s, hp-s, ami33-s, and ami49-s). In these problems, all blocks are soft blocks whose aspect ratios should be within the range (0.1, 10). (Note that Murata and Kuh modified the MCNC benchmark examples for their test. In the original MCNC benchmark examples, all blocks are hard blocks.) EXACT and DIS2+POST are similar to the algorithms suggested in this research in that they use the sequence-pair to represent the topology of a floorplan and use an SA algorithm to find the best sequence-pair. However, they use different methods for obtaining a floorplan from a sequence-pair and for generating neighborhood solutions in the SA algorithm. See Murata and Kuh [17] for more details on EXACT and DIS2+POST. In the test problems, both the lower and upper limits on the aspect ratio of the chip were set to 1.0, that is, the chip should be of a square shape.

Table V shows results of the comparison. Test results of EXACT and DIS2+POST were obtained from a 250-MHz DEC Alpha workstation [17]. Floorplans obtained from the algorithms suggested in this study are much better in terms of the total wire length and slightly better in terms of the chip size than those from the existing algorithms. EXACT and SA-LP required very long computation time (especially for large sized problems), while DIS2+POST and SA-CT+LP found floorplans within a reasonably short computation time. SA-LP showed the best performance in terms of the total wire length and the chip size for all problems except for apte-s. Compared

TABLE V
COMPARISON OF THE SUGGESTED ALGORITHMS WITH EXACT AND DIS2+POST

		apte-s (9, 97)†	xerox-s (10, 203)	hp-s (11, 83)	ami33-s (33, 123)	ami49-s (49, 408)
EXACT	Area (μ^2)	46553329	19509889	8826841	1159929	35581225
	Wire length (μ)	344358	401254	118819	53393	775104
	CPU seconds	789	1198	1346	75684	612103
DIS2+POST	Area	46635241	19474569	8868484	1162084	36048016
	Wire length	421174	533990	157166	51346	850305
	CPU seconds	1.32	2.06	2.05	28.09	50.40
SA-CT+LP	Area	46726924	19352812	8835280	1158363	35613678
	Wire length	243412	452686	121519	51383	746924
	CPU seconds	2.74	3.92	5.01	115.1	402.4
SA-LP	Area	46561640	19350308	8830586	1157829	35520644
	Wire length	278178	376092	101011	45759	663689
	CPU seconds	496.5	4872.0	1659.9	16260.0	68995.1

The tests on EXACT and DIS2+POST were done on a 250 MHz Dec Alpha workstation while the tests on the other two algorithms were done on a personal computer with a 500MHz Pentium processor.
† a and b in (a, b) denote the numbers of (soft) blocks and nets, respectively.

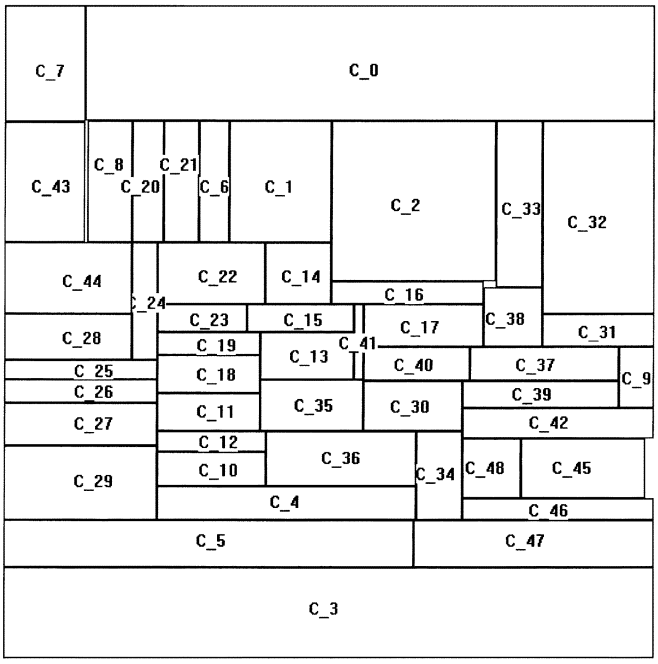

Fig. 2. Final floorplan obtained by SA-LP for ami49-s.

to the cases in which chips consist of hard blocks (see Tables III and IV), SA-LP showed much better performance (in terms of the solution quality) than SA-CT+LP. This may be because dimensions of soft blocks are optimized using LP solutions in SA-LP, while only five candidates for the dimension of each block are considered in SA-CT+LP. However, although SA-CT+LP was not as good as SA-LP, it showed better performance than EXACT and DIS2+POST and required much shorter computation time than SA-CT+LP. Fig. 2 shows the final floorplan obtained from SA-LP for ami49-s.

VII. CONCLUSION

In this paper, we considered the floorplanning problem with the objective of minimizing the total wire length and the chip size. A mathematical programming formulation is given for the problem, and the sequence-pair was used to represent the topology of nonslicing floorplans. We developed two methods to obtain a floorplan from a sequence pair: a construction method and a method based on LP. These two methods were embedded in SA algorithms, which are used to find a near-optimal floorplan. Results of computational tests on well-known test problems showed that the suggested algorithms outperformed existing algorithms and gave good solutions in a reasonable amount of time.

This research can be extended in several ways. For instance, we may consider floorplanning problems in which locations of pins in soft blocks are to be determined simultaneously with positions and dimensions of the blocks. To solve these problems, the LP model given in this paper should be modified and the LP-based method should be modified accordingly. Also, we may consider a problem in which blocks have nonrectangular shapes. For this problem, one may divide nonrectangular blocks into several smaller rectangular blocks. In this case, we must consider constraints for representing positional relationships among the divided blocks. Finally, since the floorplanning problem is closely related with the facility layout problem, which is the problem of determining a physical layout of a (manufacturing or service) system, algorithms and models of recent studies for facility layout problems [3], [8], [9], [21], [25] may be used for the floorplanning problem after proper modifications or extensions.

APPENDIX

A sequence-pair $\Gamma = (\Gamma^+; \Gamma^-)$ is a pair of sequences (Γ^+ and Γ^-) of indexes of blocks. For example, (1, 3, 2; 2, 1, 3) is a sequence-pair in cases where there are three blocks. According to Γ, relative positions of the blocks are represented as follows.

1) If i appears before j in both Γ^+ and Γ^-, (the right boundary of) block i is to the left of (the left boundary of) block j in the floorplan (corresponding to Γ), that is, $x_i^{br} \leq x_j^{bl}$. (See Section II for notation.)
2) If i appears after j in both Γ^+ and Γ^-, block i is to the right of block j in the floorplan, that is, $x_i^{bl} \geq x_j^{br}$.
3) If i appears after j in Γ^+ and before j in Γ^-, (the top boundary of) block i is below (the bottom boundary of) block j in the floorplan, that is, $y_i^{bt} \leq y_j^{bb}$.
4) If i appears before j in Γ^+ and after j in Γ^-, block i is above block j in the floorplan, that is, $y_i^{bb} \geq y_j^{bt}$.

Murata *et al.* [16] prove that for an arbitrary Γ, there always exist floorplans of which the topology is represented by Γ. Note that there are an infinite number of floorplans of which the topology is represented by an arbitrary sequence-pair, since there are an infinite number of ways to place blocks on a continual plane while keeping relative positions among them the same. On the contrary, for every floorplan, we can obtain a unique sequence-pair that represents the topology of the floorplan, if we assume without a loss of generality that when block i is above (below) block j and at the same time, to the left (or right) of block j, block i is considered to be above (below) block j. See Murata *et al.* [16] for more details on the sequence-pair.

REFERENCES

[1] P. Chen and E. S. Kuh, "Floorplan sizing by linear programming approximation," in *Proc. 37th Design Automation Conf.*, 2000, pp. 468–471.

[2] K. Chong and S. Sahni, "Optimal realizations of floorplans," *IEEE Trans. Computer-Aided Design*, vol. 12, pp. 793–801, June 1993.
[3] S. K. Das, "A facility layout method for flexible manufacturing systems," *Int. J. Prod. Res.*, vol. 31, no. 2, pp. 279–297, 1993.
[4] P. S. Dasgupta, S. Sur-Kolay, and B. B. Bhattacharya, "A unified approach to topology generation and optimal sizing of floorplans," *IEEE Trans. Circuits Syst. I*, vol. 17, pp. 126–135, Feb. 1998.
[5] X. Hong, G. Huang, Y. Cai, J. Gu, S. Dong, C.-K. Cheng, and J. Gu, "Corner block list: An effective and efficient topological representation of nonslicing floorplan," in *Proc. Int. Conf. Computer-Aided Design*, 2000, pp. 8–12.
[6] D. Johnson, C. Aragon, L. McGoech, and C. Schevon, "Optimization by simulated annealing: An experimental evaluation; Part I, Graph partitioning," *Oper. Res.*, vol. 37, no. 6, pp. 865–892, 1989.
[7] M. Kang and W. Dai, "General floorplanning with L-shaped, T-shaped and soft blocks based on bounded slicing grid structure," in *Proc. Asia South Pacific Design Automation Conf.*, 1997, pp. 265–270.
[8] J.-G. Kim and Y.-D. Kim, "A space partitioning method for facility layout problems with shape constraints," *IIE Trans.*, vol. 30, no. 10, pp. 947–957, 1998.
[9] ——, "Layout planning for facilities with fixed shapes and input and output points," *Int. J. Prod. Res.*, vol. 38, no. 18, pp. 4635–4653, 2000.
[10] S. S. Kolay and B. B. Bhattacharya, "Canonical embedding of rectangular duals with applications to VLSI floorplanning," in *Proc. 29th ACM/IEEE Design Automation Conf.*, 1992, pp. 69–74.
[11] U. Lauther, "A min-cut placement algorithm for general cell assemblies based on a graph representation," *J. Digital Syst.*, vol. IV, no. 1, pp. 21–34, 1980.
[12] J.-M. Lin and Y.-W. Chang, "TCG: A transitive closure graph-based representation for nonslicing floorplans," in *Proc. 38th Design Automation Conf.*, 2001, pp. 764–769.
[13] R. D. Meller, V. Narayanan, and P. H. Vance, "Optimal facility layout design," *Oper. Res. Lett.*, vol. 23, no. 3-5, pp. 117–127, 1999.
[14] T.-S. Moh, T.-S. Chang, and S. L. Hakimi, "Globally optimal floorplanning for a layout problem," *IEEE Trans. Circuits Syst. I*, vol. 43, pp. 713–720, Sept. 1996.
[15] B. Montreuil, "A modeling framework for integrating layout design and flow network design," in *Proc. Mater. Handling Res. Colloq.*, Hebron, KY, 1990, pp. 43–58.
[16] H. Murata, K. Fujiyoshi, and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair," *IEEE Trans. Computer-Aided Design*, vol. 15, pp. 1518–1524, Dec. 1996.
[17] H. Murata and E. S. Kuh, "Sequence-pair based placement method for hard/soft/preplaced modules," in *Proc. Int. Symp. Physical Design*, 1998, pp. 167–172.
[18] S. Nakatake, K. Fujiyoshi, H. Murata, and Y. Kajitani, "Module placement on BSG-structure and IC layout applications," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1996, pp. 484–491.
[19] P. Pan and C. L. Liu, "Area minimization for floorplans," *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 123–132, Jan. 1995.
[20] M.-W. Park and Y.-D. Kim, "A systematic procedure for setting parameters in simulated annealing algorithms," *Comput. Oper. Res.*, vol. 25, no. 3, pp. 207–217, 1998.
[21] M. Rajasekharan, B. A. Peters, and T. Yang, "A genetic algorithm for facility layout design in flexible manufacturing systems," *Int. J. Prod. Res.*, vol. 36, no. 1, pp. 95–110, 1998.
[22] W. Shi, "A fast algorithm for area minimization of slicing floorplans," *IEEE Trans. Computer-Aided Design*, vol. 15, pp. 1525–1532, Dec. 1996.
[23] X. Tang, R. Tian, and D. F. Wong, "Fast evaluation of sequence pair in block placement by longest common subsequence computation," in *Proc. Design Automation Test Eur.*, 2000, pp. 106–111.
[24] K. Tani, S. Tsukiyama, I. Shirakawa, and H. Ariyoshi, "Area-efficient drawings of rectangular duals for VLSI floorplan," in *Proc. Int. Symp. Circuits Syst.*, 1988, pp. 1545–1548.
[25] D. M. Tate and E. A. Smith, "Unequal-area facility layout by genetic search," *IIE Trans.*, vol. 27, no. 4, pp. 465–472, 1995.
[26] T.-C. Wang and D. F. Wong, "Optimal floorplan area optimization," *IEEE Trans. Computer-Aided Design*, vol. 11, pp. 992–1002, Aug. 1992.
[27] S. Wimer and I. Koren, "Optimal aspect ratios of bulding blocks in VLSI," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 139–145, Feb. 1989.
[28] D. F. Wong and C. L. Liu, "A new algorithm for floorplan design," in *Proc. 23rd ACM/IEEE Design Automation Conf.*, 1986, pp. 101–107.
[29] T. Yamanouchi, K. Tamakashi, and T. Kambe, "Hybrid floorplanning based on partial clustering and module restructuring," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1996, pp. 478–483.
[30] H. Yamazaki, K. Sakanushi, and Y. Kajitani, "Optimum packing of convex-polygons by a new data structure sequence-table," in *Proc. IEEE Asia Pacific Conf. Circuits Syst.*, 2000, pp. 821–824.
[31] K. H. Yeap and M. Sarrafzadeh, "A unified approach to floorplan sizing and enumeration," *IEEE Trans. Computer-Aided Design*, vol. 12, pp. 1858–1867, Dec. 1993.
[32] F. Y. Young, C. C. N. Chu, W. S. Luk, and Y. C. Wong, "Handling soft modules in general nonslicing floorplan using lagrangian relaxation," *IEEE Trans. Computer-Aided Design*, vol. 20, pp. 687–692, May 2001.
[33] F. Y. Young, D. F. Wong, and H. H. Yang, "Slicing floorplans with boundary constraints," *IEEE Trans. Computer-Aided Design*, vol. 18v, pp. 1385–1389, Sept. 1999.
[34] ——, "Slicing floorplans with range constraint," *IEEE Trans. Computer-Aided Design*, vol. 19, pp. 272–278, Feb. 2000.

Jae-Gon Kim received the B.S., M.S., and Ph.D. degrees in industrial engineering from the Korea Advanced Institute of Science and Technology, Daejon, in 1994, 1996, and 2001, respectively.

He is currently a Postdoctoral Research Fellow at the School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta. His research interests include manufacturing system design, VLSI layout design, logistics, and production planning and scheduling.

Yeong-Dae Kim (M'98) received the B.S. degree in industrial engineering from Seoul National University, Seoul, Korea, in 1979, the M.S. degree in industrial engineering from the Korea Advanced Institute of Science and Technology, Daejon, and the Ph.D. degree from the University of Michigan, Ann Arbor, in industrial and operations engineering, in 1981 and 1988, respectively.

He is a Professor in the Department of Industrial Engineering, Korea Advanced Institute of Science and Technology. His research areas include design and operation of manufacturing systems, operations scheduling, and production planning and inventory management.

Prof. Kim is a Senior Member of IIE, a member of the Intsitute for Operations Research Management Sciences, and the Operational Research Society.