A COMBINED DETERMINISTIC AND RANDOM OPTIMIZATION ALGORITHM FOR THE PLACEMENT OF MACRO-CELLS

by

S. Daijavad, E. Polak, and R-S Tsay

Memorandum No. UCB/ERL M87/86

20 November 1987

A COMBINED DETERMINISTIC AND RANDOM OPTIMIZATION ALGORITHM FOR THE PLACEMENT OF MACRO-CELLS

by

S. Daijavad, E. Polak, and R-S Tsay

Memorandum No. UCB/ERL M87/86

20 November 1987

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

A COMBINED DETERMINISTIC AND RANDOM OPTIMIZATION ALGORITHM FOR THE PLACEMENT OF MACRO-CELLS

S. Daijavad, E. Polak and R-S Tsay

Department of Electrical Engineering and Computer Sciences

University of California, Berkeley

ABSTRACT

This paper presents several new ideas for solving the macro-cell placement problem by optimization techniques. An original mathematical formulation using two different descriptions for constraints on nono-verlap of cells is presented. After describing a deterministic optimization method which results in good solutions and is insensitive to the starting point, we present a novel method for combining deterministic and random optimization techniques in search of the global minimum. Results on a test problem with 8 macro-cells and a more realistic problem with 33 cells are presented.

I. INTRODUCTION

The problem of placement of macro-cells, assuming that they are arbitrarily sized rectangles, has been addressed by many researchers in recent years. As is well known, placement algorithms decide the proper position of a set of components with interconnections among them. Among several approaches to solve this problem, heuristic algorithms have been used most frequently (see, e.g., [1],[2]). These algorithms form the basis of most existing software systems. The application of a powerful form of random optimization, namely, *simulated annealing* in solving the macro-cell placement problem has also been reported[3]. A new approach based on a rigorous formulation of the problem, which is suitable for the application of deterministic optimization algorithms, has been given by Sha and Dutton[4].

In this paper, we propose a part deterministic part random optimization algorithm for solving the macro-cell placement problem. Our motivation for developing a new algorithm is to take advantage of some nice features of the existing methods and to add other features along the way. The following five points summarize the features of our method.

1) Inspired by the work of Sha and Dutton[4], we develop a concise and original mathematical formulation for the problem and in particular for the constraints on nonoverlap of the cells. The formulation transforms an inherently combinatorial problem to a nonlinear programming problem which can be solved for a local solution using fast deterministic optimization techniques. Based on the framework of nonlinear programming, it is possible to extend the work in the future to include other constraints not discussed in the present paper, e.g., constraints arising from signal delay considerations.

2) In Sha's formulation, nonoverlap of cells at the solution is not guaranteed. This is due to the simplifications used in modeling of rectangles. By using an ellipse model for rectangles, we first develop continuously differentiable nonoverlap constraints which are highly desirable as far as optimization algorithms are concerned and are more robust than the constraints by Sha. However, these constraints also result in slightly overlapping solutions. We then use the so-called exclusion constraints which are more difficult to handle with optimization algorithms in order to achieve zero overlap of the blocks at the solution.

3) Similar to Sha's work, the best orientation (horizontal or vertical) for the cells is determined by the optimization process in our formulation. However, the freedom in orientation is achieved by using three variables per cell as opposed to four variables in Sha's work.

4) The shape constraints for L-shaped blocks which have been mentioned in Sha's work are rigorously formulated in this paper. To the best of our knowledge, this is an original contribution with immediate practical use. A simple method for simultaneous compaction or reduction of the chip area (the bounding box containing all cells), as the total wire length objective function decreases, is also described.

5) The important issue of selecting the starting point for the deterministic optimization algorithm is resolved in our method, so that a good local solution is reached from any arbitrary initial placement. However, due to the inherent combinatorial nature of the problem and the fact that any local optimization method is theoretically inadequate in finding the best possible solution of a problem with multiple local minima, we describe the consistent addition of a random optimization technique to the deterministic method. The major disadvantage of a random optimization technique such as simulated annealing is its intensive use of computer time in order to reach quality solutions. This is why some fast heuristic algorithms which do not solve complete optimization problems are preferred by some researchers. The use of a deterministic optimization algorithm is an effective procedure to obtain quality solutions in reasonable computational time.

This paper is organized as follows: the mathematical model for the problem including the objective function and constraints is first described. Next, we describe a penalty function method and the deterministic optimization part of the algorithm followed by a section on how to add a random optimization method to the overall algorithm. Finally, we present some results on an 8 block test problem and a more realistic 33 block problem.

II. MATHEMATICAL MODEL AND FORMULATION OF THE DETERMINISTIC OPTIMIZATION PROBLEM

A. Representation of Rectangles and Definition of Cost Function

The objectives and constraints of the macro-cell placement problem are formulated using a simple description for a rectangular cell. An L-shaped cell, which consists of two attached rectangles, will be discussed in more detail after the formulation using simple rectangles has been completed. The important issue of routability of the final placement is simplified by the assumption that the routing area for each cell is already included within the perimeter of the cell itself. Therefore, the problem to solve is to minimize the total wire-length and to achieve as small an overall chip area as possible, subject to constraints on the cell size and orientation and constraints which ensure nonoverlap of cells.

Consider m rectangular cells. For a rectangular cell B^i (i=1,...,m) with two length parameters $w^i > 0$ and $h^i > 0$ ($w^i > h^i$) which define its size, assuming that it can have only horizontal or vertical orientation, the coordinates of the bottom left vertex denoted by (x_1^i, y_1^i) and one coordinate of the top right vertex, determine both the position and the orientation of the cell (see Fig. 1). Denoting the coordinates of the top right vertex by (x_2^i, y_2^i), we have

$$(x_2^i - x_1^i) + (y_2^i - y_1^i) = w^i + h^i \,, \tag{1}$$

with either horizontal or vertical orientation. Therefore, given x_1^i, y_1^i and x_2^i, we can calculate y_2^i as

$$y_2^i = w^i + h^i + x_1^i - x_2^i + y_1^i \,. \tag{2}$$

The variables x_1^i, y_1^i and x_2^i (i=1,...,m) are used as optimization variables throughout this paper and the y_2^i's (i=1,...,m) are evaluated using (2), wherever they appear in the subsequent formulas. Formally, if z denotes the vector of optimization variables, we have

$$z = [\, x_1^1 \; y_1^1 \; x_2^1 \;\; \cdots \;\; x_1^m \; y_1^m \; x_2^m \,]^T \,. \tag{3}$$

It is clear that $z \in \mathbb{R}^{3m}$. An implicit assumption about the variables is that $x_2^i > x_1^i$ and $y_2^i > y_1^i$. It is redundant to ensure this assumption using explicit constraints, since the shape constraint, which will be introduced later in this section, ensures the validity of this assumption too.

To calculate the total wire length, we use the star model for each net with the gravity center [4] as the center of the star and with all the pins positioned at the cell centers. Without changing the remaining sections of this paper, alternative estimates such as the estimate using the half-perimeter of the bounding box which includes all the pins connected to a net, or a more realistic estimate using the actual pin positions to calculate the exact wire length could also be considered. The latter estimate makes use of the fact that the

relative position of pins corresponding to a cell with respect to its center is fixed.

Consider a signal net S_k which connects m_k cells and possibly an I/O pad on the periphery of the chip. The position of center points for cells are calculated as

$$x_c^i = \frac{x_1^i + x_2^i}{2} \quad ; \quad y_c^i = \frac{y_1^i + y_2^i}{2}, \tag{4}$$

where i=1,...,m_k. For signal net S_k, temporarily assuming

$(x_c^{m_k+1}, y_c^{m_k+1}) = (x_p^k, y_p^k)$ with (x_p^k, y_p^k) denoting the coordinates of the appropriate pad, the gravity center is calculated as

$$\bar{x}_k = \frac{1}{m_k^*} \sum_{i=1}^{m_k^*} x_c^i \quad ; \quad \bar{y}_k = \frac{1}{m_k^*} \sum_{i=1}^{m_k^*} y_c^i, \tag{5}$$

where $m_k^* = m_k$, if the I/O pad is not connected and $m_k^* = m_k + 1$, if the I/O pad is connected to the net. A measure for the wire length corresponding to S_k is calculated as

$$u_k(z) = \sum_{i=1}^{m_k^*} \left[(x_c^i - \bar{x}_k)^2 + (y_c^i - \bar{y}_k)^2 \right]. \tag{6}$$

It should be emphasized that the above formulation represents a measure for the wire length and strictly speaking, does not have the same dimensionality as the length. However, as is commonly referred to in the literature, the term wire length will also be used throughout this paper.

Denoting the total number of signal nets by m_s, the total wire length is computed as

$$U(z) = \sum_{k=1}^{m_s} u_k . \tag{7}$$

As suggested by other researchers [4], for larger nets, i.e., the ones that connect large numbers of cells, it is desirable to reduce the contribution of the corresponding u_k's, since each individual connection is less important. It is recommended that u_k be multiplied by a weighting coefficient, which is a function of m_k (e.g., inversely proportional to m_k)[4].

Minimization of the overall chip area, in general, requires the introduction of a second objective function and therefore leads to a multi-criterion optimization problem. However, if we have I/O pads on the four sides of the chip periphery, minimization of area can be achieved via the minimization of wire length. To do this, chip dimensions X_T and Y_T are added to optimization variables. If (0,0) is chosen as

the bottom left vertex of the chip, X_T and Y_T will represent the coordinates of the top right vertex. As the chip dimensions vary, the I/O pad coordinates change. However, we avoid having the I/O pad coordinates as independent variables, by assuming that the relative position of I/O pads is fixed. Given the initial position of I/O pads and the initial dimensions of the chip, the coordinates of a pad (x_p, y_p) change in the following way as the chip dimensions X_T and Y_T vary :

$$x_p = \alpha X_T \quad ; \quad y_p = \beta Y_T \,, \tag{8}$$

where

$$\alpha = \frac{x_{p0}}{X_{T0}} \quad , \quad \beta = \frac{y_{p0}}{Y_{T0}} \,. \tag{9}$$

Subscript zero is used to denote initial values. Since α and β are constants, adding only X_T and Y_T to the optimization variables is sufficient to accommodate the varying position of all I/O pads. The vector of optimization variables z, given by (3), is augmented by X_T and Y_T and consequently, its dimensionality is increased to 3m+2. Using (8) and the fact that the pad coordinates affect the wire length objective function, it can be easily proved that the relationship between X_T, Y_T and U(z) is direct, i.e., as the chip dimensions and its area decrease, U(z) decreases.

B. Constraints for Rectangular Cells

There are three types of constraints in the macro-cell placement problem. The first type of constraint, referred to as the shape constraint, ensures that each cell has the prescribed dimensions. For rectangular cells with either horizontal or vertical orientation, we have

$$f_i(z) = (x_2^i - x_1^i - h^i)(x_2^i - x_1^i - w^i) = 0 \,, \; i=1,...,m \,. \tag{10}$$

Note that (10) also implies that $x_2^i > x_1^i$ and $y_2^i > y_1^i$ (using (1) for the latter inequality).

The second type of constraint ensures that each cell is located within the chip boundaries. For i=1,...,m, we have

$$g_{i1}(z) = x_1^i \geq 0 \,, \tag{11a}$$

$$g_{i2}(z) = y_1^i \geq 0 \,, \tag{11b}$$

$$g_{i3}(z) = X_T - x_2^i \geq 0 \,, \tag{11c}$$

$$g_{i4}(z) = Y_T - y_2^i \geq 0 \,. \tag{11d}$$

The third type of constraint ensures that the cells do not overlap. We have developed two new approaches to formulate the nonoverlap constraints. The first approach, which involves modeling of the rectangular cells by ellipses, results in differentiable constraints, however, it has the disadvantage that the error in modeling leads to solutions with slightly overlapping cells. For rectangles which approach squares, the error in modeling increases. The mathematical description of this method is as follows.

Consider the rectangular cell i modeled by an ellipse centered at (x_c^i, y_c^i). To avoid the overlap of cell j with this ellipse, the following inequality must hold

$$G_{ij}(z, \lambda_x, \lambda_y) = (y_1^j - y_c^i)^2 (x - x_c^i)^2 + (x_1^j - x_c^i)^2 (y - y_c^i)^2 - (x_1^j - x_c^i)^2 (y_1^j - y_c^i)^2 \geq 0 \quad (12)$$

for all x and y such that

$$x = x_1^j + \lambda_x (x_2^j - x_1^j), \quad 0 \leq \lambda_x \leq 1 \quad (13a)$$

and

$$y = y_1^j + \lambda_y (y_2^j - y_1^j), \quad 0 \leq \lambda_y \leq 1 . \quad (13b)$$

To satisfy (12) subject to (13), it is necessary and sufficient that the following inequality holds :

$$g_{ij}^e(z) = \min_{\lambda_x, \lambda_y} G_{ij}(z, \lambda_x, \lambda_y) \geq 0 . \quad (14)$$

Minimization of G_{ij} with respect to λ_x and λ_y gives

$$\lambda_x = \begin{cases} (x_c^i - x_1^j) / (x_2^j - x_1^j), & \text{if } x_1^j \leq x_c^i \leq x_2^j , \\ 0 & \text{, if } x_c^i < x_1^j , \\ 1 & \text{, if } x_c^i > x_2^j . \end{cases} \quad (15)$$

The expression for λ_y is obtained by replacing x with y in (15). Using (15) and the similar expression for λ_y, and substituting for λ_x and λ_y in (13) and then substituting (13) in (12), we can derive explicit formulas for $g_{ij}^e(z)$.

Assuming $A_1 = (y_1^j - y_c^i)^2$, $A_2 = (x_1^j - x_c^i)^2$, $A_3 = (x_1^j - x_c^i)^2$, $A_4 = (x_2^j - x_c^i)^2$, $A_5 = (y_1^j - y_c^i)^2$ and $A_6 = (y_2^j - y_c^i)^2$, we get

$$g_{ij}^{e}(z)=\begin{cases} -A_1A_2 & \text{, if } s_1 \text{ and } t_1, \\ A_1(A_3-A_2) & \text{, if } s_2 \text{ and } t_1, \\ A_1(A_4-A_2) & \text{, if } s_3 \text{ and } t_1, \\ A_2(A_5-A_1) & \text{, if } s_1 \text{ and } t_2, \\ A_1A_3+A_2(A_5-A_1) & \text{, if } s_2 \text{ and } t_2, \\ A_1A_4+A_2(A_5-A_1) & \text{, if } s_3 \text{ and } t_2, \\ A_2(A_6-A_1) & \text{, if } s_1 \text{ and } t_3, \\ A_1A_3+A_2(A_6-A_1) & \text{, if } s_2 \text{ and } t_3, \\ A_1A_4+A_2(A_6-A_1) & \text{, if } s_3 \text{ and } t_3, \end{cases} \tag{16}$$

where conditions s_1, s_2, s_3 and t_1, t_2, t_3 are as follows; $s_1: x_1^i \le x_c^i \le x_2^i$, $s_2: x_c^i < x_1^i$, $s_3: x_c^i > x_2^i$, $t_1: y_1^i \le y_c^i \le y_2^i$, $t_2: y_c^i < y_1^i$ and $t_3: y_c^i > y_2^i$. When considering all cells, we have i=1,...,m and j=i+1,...,m in (16).

The second approach in formulating the nonoverlap constraints uses the rectangles directly and ensures zero overlap of cells at the solution. The disadvantage of this approach is that it results in the so-called exclusion constraints [5], which are not only nondifferentiable but also of a combinatorial nature.

For two cells i and j, if we have

$$(x_1^i \ge x_2^i) \; or \; (x_1^i \ge x_2^i) \; or \; (y_1^i \ge y_2^i) \; or \; (y_1^i \ge y_2^i), \tag{17}$$

then there is no overlap between the cells. Equivalently, satisfying the constraint

$$\max\left\{(x_1^i - x_2^i), (x_1^i - x_2^i), (y_1^i - y_2^i), (y_1^i - y_2^i)\right\} \ge 0, \tag{18}$$

guarantees (17).

The combinatorial nature of this constraint is evident from the appearance of the "or" expression in (17) or alternatively from the fact that for the maximum of a number of expressions to be positive, it is sufficient that any one of them be positive.

A general and exhaustive algorithm for handling exclusion constraints of the form given in (18) requires that at each stage of the optimization, each one of the active constraints (in the present context, a constraint is active if it is equal to or less than a small positive number ε below the maximum value of all constraints) be tried separately in determining a search direction. Among all the search directions calculated, the best one is selected. (For each search direction, the step length is computed, the variables are updated and active constraints are evaluated. Comparison of the values of active constraints determines the

best search direction.) General algorithms of the type described have been developed in [5]. The major disadvantage of a general algorithm is its high computational expense. In this paper, instead of using a general algorithm which considers all possible search directions, we adopt a simple strategy which uses only one of the active constraints (and consequently, only one of several possible search directions) at each stage of the optimization. Our simplification is as follows. For cells i and j, assuming that there is overlap between the cells (for two nonoverlapping cells, the corresponding nonoverlap constraint is ignored in the penalty function formulation which will be used for solving the resulting optimization problem), the values of four terms $(x_1^i - x_2^j)$, $(x_1^j - x_2^i)$, $(y_1^i - y_2^j)$ and $(y_1^j - y_2^i)$ are compared (all four terms are negative when there is overlap between the cells) and the maximum is selected as the constraint $g_{ij}^r(z)$. If two, three or all four values equal the maximum, one is selected randomly. Therefore, for each combination of i and j (i=1,...,m , j=i+1,...,m) we work with only one of the four terms at a time. The possibility of finding the best solution is sacrificed by this simplification. However, the simplification is justified since we are trying to find a feasible solution using the deterministic part of our algorithm in as small a computational time as possible. We always have the option of searching for better or globally optimal solution with the random part of the algorithm.

C. Constraints for L-Shaped Cells

A simple extension of the formulation described for rectangular cells enables us to handle L-shaped cells. An L-shaped cell consists of two rectangular cells with their longer sides being either perpendicular (Fig. 2a) or parallel (Fig. 2b) to each other. As Fig. 2 illustrates, we have opted to take two overlapping rectangles for an L-shaped block.

Having an L-shaped block among the macro-cells is similar to having two separate rectangles in formulating the objective function, nonoverlap and chip boundary constraints. Assume that rectangles i and j form an L-shaped cell. We use previously formulated nonoverlap constraints to ensure that the remaining rectangles do not overlap either i or j. A minor difference with having two separate rectangles is that i and j may overlap (and indeed i and j should overlap according to the shape constraints which will be formulated shortly). The constraints which ensure that the cells are within the chip boundaries, operate on an L-shaped block exactly the same way as on two separate rectangles.

The major difference between having an L-shaped block and two separate rectangles is in the shape constraints. For two attached rectangles i and j, only one is free to take either horizontal or vertical orientation and the orientation of the second one is dictated by the orientation of the first one. Assuming that rectangle i is free, we have equality constraint (10) for rectangle i and one of the following two constraints for rectangle j :

$$[(x_2^i - x_1^i - h^i)^2 + (x_2^j - x_1^j - w^j)^2][(x_2^i - x_1^i - w^i)^2 + (x_2^j - x_1^j - h^j)^2] = 0 , \quad (19)$$

or

$$[(x_2^i - x_1^i - h^i)^2 + (x_2^j - x_1^j - h^j)^2][(x_2^i - x_1^i - w^i)^2 + (x_2^j - x_1^j - w^j)^2] = 0 . \quad (20)$$

Equation (19) is used for a case similar to the one shown in Fig. 2a and (20) is used for the case in Fig. 2b. Note that only one of (19) or (20) can be true for a given L-shaped block. A simple interpretation of (19), considering that (10) should also hold, is that if rectangle i has a vertical orientation, rectangle j must have a horizontal orientation and vice versa.

A more important shape constraint for two rectangles i and j which form an L-shape is a constraint which ensures that the two rectangles are attached at the solution, while allowing rotations and mirror operations that achieve minimum wire length. Figs. 3a-3d illustrate four possible orientations for an L-shaped block, with each one obtained by a 90 degree rotation of another one in the set. Figs. 3e-3h show four other orientations with each one obtained by mirror-imaging one of the orientations in Figs. 3a-3d. For instance, Fig. 3e is the mirror image of Fig. 3d. The following equality constraint keeps rectangles i and j attached in an L-shape, while allowing all 8 possible orientations :

$$[(x_1^i - x_1^j)(x_2^i - x_2^j)]^2 + [(y_1^i - y_1^j)(y_2^i - y_2^j)]^2 = 0 . \quad (21)$$

This constraint simply states that we should have either $x_1^i = x_1^j$ and $y_1^i = y_1^j$ (as in Figs. 3a and 3e), or $x_1^i = x_1^j$ and $y_2^i = y_2^j$ (as in Figs. 3b and 3f), or $x_2^i = x_2^j$ and $y_2^i = y_2^j$ (as in Figs. 3c and 3g), or $x_2^i = x_2^j$ and $y_1^i = y_1^j$ (as in Figs. 3d and 3h).

D. Penalty Function Formulation

Among all techniques for solving the resulting nonlinear programming (NLP) problem, which is characterized by the cost function U(z) given by (7), the equality constraints $f(z)$ (given by (10) and one of (19) or (20) plus (21) for L-shaped cells) and the inequality constraints $g(z)$ (given by (11) and either

one of $g_{ij}^{e}(z)$ or $g_{ij}^{r}(z)$), we select the penalty function method, as has also been suggested by Sha and Dutton[4]. This method does not guarantee a feasible solution for a finite penalty coefficient, but it has proved to be quite effective in our computational experience. The constraints are always "adequately" satisfied for large penalty coefficients and, compared to the more sophisticated nondifferentiable "exact" penalty method (exact in the sense that the solution of the penalty problem yields the exact solution to the original NLP with a finite penalty coefficient), the computational speed is remarkable. For problems with a large number of cells, exact penalty methods, which are generally more robust, become extremely slow since they usually use linear or quadratic programming techniques in determination of a search direction at each iteration.

A penalty function P(z,c_l) is constructed as

$$P(z,c_l) = U(z) + c_l \left\{ \sum_{i=1}^{m} \alpha_i f_i^2(z) + \sum_{i=1}^{m_g} \alpha_i [min(0,g_i(z))]^2 \right\}. \tag{22}$$

The total number of inequality constraints denoted by m_g is given by

$$m_g = 4m + [1+2+...+(m-1)] = \frac{m(m+7)}{2}. \tag{23}$$

(Notice the i and j variation for (16) to justify the term in square brackets.) For every L-shaped block, the number of equality constraints is increased by one and the number of inequality constraints is decreased by one. In equation (22), α_i is a nonnegative weighting coefficient and c_l is the important penalty coefficient. $\{c_l\}$, l=1,2,... is a sequence tending to infinity such that, $c_1 \geq 0$ and $c_{l+1} > c_l$. Practically, c_l is gradually increased until the constraints are satisfied within the required accuracy.

Selecting c_1=0, i.e., minimization of U(z) without taking the constraints into account gives a solution which, although not feasible, is of great significance. The reason is that it resolves the key issue of choosing starting values for optimization variables in order to achieve a good local minimum. The minimum of the quadratic function U(z) given by (7), is its global minimum. Therefore, with c_1=0, we reach identical positions for cell centers, regardless of the starting point. This unique solution which is visually meaningless due to the extreme violation of constraints, provides a good starting point for the constrained problem. The unconditional decrease in the wire length objective results in an excellent relative position for the cells. As the penalty coefficient increases, there is increasing emphasis on satisfying the

constraints, while the relative position of the cells remains almost unchanged. Therefore, minimization with c_1=0 before taking the constraints into account, strengthens our algorithm so as to avoid bad local minima. An important note is that the chip dimensions, i.e., variables X_T and Y_T, should be fixed with c_1=0. This is obvious because otherwise, U(z) will be reduced to zero, with all the cell centers taking the same position.

Minimization of $P(z,c_l)$ is performed using a conjugate gradient method. We utilized a standard available routine in the IMSL library[6], which uses the restarting algorithm suggested by Powell[7]. We also implemented the more recent three-term algorithm of Nazareth[8] with the guidelines provided by Dixon[9]. The two algorithms performed equally well on most examples. In the second algorithm, the frequency of restarting the procedure using the steepest descent search direction is controlled by the user.

III. RANDOM OPTIMIZATION STAGE OF THE ALGORITHM

Having addressed the part of our proposed algorithm which uses a deterministic optimization approach, namely, minimization using the conjugate gradient method, we consider the consistent addition of random optimization techniques to the existing method. In this section, we assume that the cells are rectangles, although generalization to L-shaped blocks is possible by considering an L-shaped block as two attached rectangles.

In the previous section we argued that the optimization without constraints provides a good starting point for the constrained optimization in the sense that we are almost guaranteed to reach one of the best local minima. The use of a random optimization technique as an optional tool to search for better local minima or ultimately the global minimum is recommended. In the last few years, Simulated Annealing (SA) [10], which belongs to the more general class of Probabilistic Hill Climbing (PHC) algorithms [11], has become a powerful tool for solving general combinatorial optimization problems. The application of SA algorithm in the placement of macro-cells has been reported [3]. While robust SA techniques have provided excellent quality solutions to placement problems, the CPU time required is very high. To speed up the process, the use of multiprocessor architectures in applying a parallel SA algorithm has been reported [12].

In this section, we discuss the use of a random optimization technique inspired by the PHC algorithms in conjunction with a deterministic algorithm. Random optimization is used only when it is felt that a better local minimum, than the one currently reached by the deterministic algorithm, exists. The randomizing process for the macro-cell placement problem corresponds to the random movement of cells. The key property of all PHC algorithms, namely, allowing an increase in the objective function value, is preserved. A new acceptance rule, which uses the rate of convergence of the deterministic algorithm, is devised.

Consider a solution vector z^* obtained by minimizing the penalty function of (22) after the gradual increase of c_l, such that the constraints are satisfied within the required accuracy. We calculate a measure of the convergence rate for the deterministic algorithm (e.g., the conjugate gradient algorithm) in the following way.

For the minimization with the final value of c_l, we ignore the first few and the last few iterations and assume that the remaining iterations are numbered from 0 to n_i. With P_i denoting the penalty function at iteration i, and P^* representing P(z^*), $P_i - P^*$ is approximated by $(P_0 - P^*)\theta^i$ (θ to the power i). The best value of θ is obtained by solving a simple data-fitting problem in the least-squares sense as

$$\ln \theta = \frac{\sum i \ln(P_i - P^*) - \ln(P_0 - P^*) \sum i}{\sum i^2}, \tag{24}$$

where summation is from 1 to n_i.

The computed θ is used to define a rule for accepting or rejecting the new variable values after a random change is performed at z^*. Suppose that a better local minimum z^{**} exists such that

$$P(z^{**}) < \mu P(z^*), \mu < 1. \tag{25}$$

For sufficiently large c_l in (22), satisfying (25) almost guarantees that at z^{**} the constraints are satisfied at least as well as at z^*. Now, the procedure is to make random changes in variables until a point is found from which, based on the value of θ, it is projected that $\mu P(z^*)$ can be reached in a user-selected number of iterations. An important side issue here is the strategy for random change in variables such that generation of many useless points is avoided. The strategy used in TimberWolf[9], a package based on the SA algorithm, is also appropriate here. It is as follows: (1) A random number between 1 and m (the total

number of cells) is generated. (2) A second random number between 1 and 10m is generated. (3) If the second number is between 1 and m and not equal to the first number, the positions of the corresponding cells are exchanged, otherwise the first cell is displaced to a random position (without violating chip boundaries). Clearly, pairwise exchanges are infrequent compared to cell displacements.

Continuing with the main issue, we accept a random point z_r if $P(z_r) < \mu P(z^*)$ (which rarely happens), or if

$$(P(z_r) - P(z^{**}))\,\theta^{n_{max}} \leq \mu P(z^*) - P(z^{**}), \tag{26}$$

where n_{max} is the user-selected maximum number of iterations allowed. μ is also a user-selected parameter controlling the acceptance criterion. $P(z^{**})$ is guessed as a value below $\mu P(z^*)$, once μ has been selected. If inequality (26) is not satisfied after a certain maximum number of random changes in variables (e.g., 100m random moves), the best set of variables among the sets tried (the one with the lowest value of P), is selected. The deterministic optimization is then performed starting from this set with the hope that a different and better local minimum can be reached. Another alternative is to adjust some of the user-selected parameters and repeat the process.

What has been suggested in this section is a general methodology for combining deterministic and random optimization techniques. The choice of parameters μ, n_{max} or the rules for adaptive change in these parameters is problem dependent. This is analogous to the selection of the rule for reducing temperature parameter or inner loop criterion in simulated annealing.

IV. RESULTS

Two examples are given to show the performance of the deterministic part of the algorithm discussed. In both examples, an arbitrary starting point was used, minimization without taking the constraints into account (c_1=0) was performed and then c_l was gradually increased. As discussed in Section III, the random part of our algorithm, which may take considerable amount of CPU time, is reserved for the cases where it is clear that a better solution, than the one reached by the deterministic algorithm, exists. Since the solutions reached for the two examples were reasonably good, no attempt was made to continue the process with the random optimization. However, a third example is given to illustrate the effect of random optimi-

zation.

Example 1 is adopted from [13], and has 8 cells, 24 pads and 38 signal nets. Starting from an identical position for all cells (x_1=1.6, x_2=1.7 and y_1=1.0), the solution reached with c_1=0 is illustrated in Fig. 4a. As discussed in Section II, the chip dimensions (X_T=3.3 and Y_T=2.3, in this example) are fixed during the optimization with c_1=0. The computational time for this stage was 0.97 seconds on a VAX Station II machine. The objective function has the unique value of 16.5, reachable from any starting point. Next, we increased c_l to 100 and then to 500 and finally at c_l=500, we changed the weighting coefficient for the shape constraints from 1 to 10. The final result, which was reached in a total of 145 seconds, is illustrated in Fig. 4b. The wire length objective function is 8.7 at this solution and the penalty function is 8.8, indicating how well the constraints are satisfied. The chip dimensions, which were added to the optimization variables after the first stage, are X_T=2.07 and Y_T=1.30. In this experiment, nonoverlap of the cells has been ensured using the exclusion constraints. Finally, the building block routing package BBL [13],[14] was used to obtain a complete layout for the chip. The chip dimensions for the result, which is illustrated in Fig. 4c, are X_T=2.38 and Y_T=1.53.

To test the shape constraints for L-shaped blocks, we added the requirement that blocks 3 and 7 in the above problem should be attached in an L-shaped form. Starting from the position of Fig. 4b, the solution illustrated in Fig. 5 was obtained after a gradual increase in the weighting coefficient for the new shape constraint from 10 to 5000.

Example 2 is a reasonably complex problem with 33 blocks, 38 pads and 121 nets. Using an arbitrary starting point, minimization with c_1=0 resulted in the configuration of Fig. 6. The computational time on a VAX Station II was 16 seconds with the objective function at the solution having the unique value of 10.9 (global minimum reachable from any starting point). Starting from this solution, c_l was increased to 100, 500, 1000 and 2000, allowing only 100 simulations (i.e., evaluation of functions and gradients associated with the objective function and constraints) for each value of c_l. As an example for an intermediate situation, in Fig. 7 we have illustrated the solution reached with c_l=100, after 100 simulations. With c_l=2000, the weight on the shape constraints was increased to 10 and after 400 simulations, the final result shown in Fig. 8 was obtained. The total CPU time (including the first stage) was 30 minutes. The final wire length

objective function is equal to 12.9 and the penalty function (with c_l=2000) has a value of 13.5. The final chip dimensions are X_T=2.32 and Y_T=1.62. Finally, the layout of this chip was completed by BBL ([13],[14]). The result is illustrated in Fig. 9.

In the third example, we consider the 8-block problem of example 1 and demonstrate how the random optimization stage of the algorithm is used to move from one local minimum to a better one. Starting from the initial position of cells as illustrated in Fig. 10 (which does not seem to be worse than the starting position used in example 1), but without performing the unconstrained optimization, we used a penalty coefficient of 100 and reached a solution with the wire length equal to 10.6 and the penalty function equal to 12.0 (28 seconds of CPU time). In an attempt to improve on the violation of constraints, the penalty coefficient was increased to 500 and the solution of Fig. 11 was obtained (wire length equal to 16.4 and the penalty function equal to 17.0). This solution which is clearly worse than the solution reached in example 1, demonstrates the sensitivity of the deterministic local optimization algorithm to the initial placement and justifies our previous emphasis on an initial unconstrained optimization stage.

To show that the solution reached by the deterministic optimizer can be improved with the help of a random optimizer, we performed random changes in variables according to the strategy discussed in Section III, following the stage in which a solution with the penalty coefficient equal to 100 was reached (an estimate for θ was established). A solution with the wire length equal to 11.2 and the penalty function equal to 10.0 was obtained after only 20 seconds of CPU time. This solution was used as a starting point for the deterministic optimizer with the penalty coefficient increased to 500. The result is illustrated in Fig. 12 (wire length equal to 14.1 and the penalty function equal to 14.4). This solution, which is still worse than the one in example 1, can be further improved by repeating the combination of deterministic and random optimization methods. Slightly overlapping areas in both Figs. 11 and 12 are not significant as far as the above methodology is concerned. They only indicate that the penalty coefficient of 500 is not large enough.

Due to different objective function and constraints, we could not directly compare our results with the results of simulated annealing algorithm. However, to appreciate the CPU times reported above, we refer to an example with 30 blocks[12] which takes an average of 210 minutes on a similar machine with

SA (using a parallel algorithm with 8 physical processors, time has been reduced to 33.6 minutes).

V. CONCLUSIONS

We have described a new methodology based on the combination of deterministic and random optimization methods to solve the macro-cell placement problem. Two new mathematical descriptions for nonoverlap constraints were developed and solution of the resulting nonlinear programming problem using penalty function method was discussed. The problem of selecting the starting point for the deterministic optimization was addressed, so that a bad local minimum is avoided regardless of the initial placement.

When compared to other optimization-based algorithms such as simulated annealing, our computational results are promising in terms of the CPU time required to reach a good solution

Future work will investigate the theoretical properties of combining deterministic and random optimization methods. From a more practical point of view, we will use the current framework to deal with other aspects of the placement problem, such as formulation of more realistic objective functions and inclusion of constraints related to signal delay considerations.

ACKNOWLEDGEMENTS

We wish to thank Prof. E. Kuh for his advice during this project. This work was supported by a Postdoctoral Fellowship of the Natural Sciences and Engineering Research Council of Canada, the National Science Foundation grants ECS-8517362 and MIP-8506901, the Air Force Office Scientific Research grant 86-0116, the Office of Naval Research contract N00014-86-K-0295, and the California State MICRO program.

REFERENCES

[1] B.T. Preas and W.M. vanCleemput, "Placement algorithms for arbitrarily shaped blocks", *Proc. 16th Design Automation Conf.*, pp. 474-480, 1979.

[2] U. Lauther, "A min-cut placement algorithm for general cell assemblies based on. a graph representation", *Proc. 16th Design Automation Conf.*, pp. 1-10, 1979.

[3] C. Sechen and A. Sangiovanni-Vincentelli, "The TimberWolf placement and routing package", *Proc. 1984 Custom Integrated Circuit Conf.*, Rochester, NY, 1984.

[4] L. Sha and R.W. Dutton, "An analytical algorithm for placement of arbitrarily sized rectangular blocks", *Proc. 22nd Design Automation Conf.*, pp. 602-608, 1985.

[5] D.Q. Mayne and E. Polak, "Algorithms for optimization problems with exclusion constraints", *Journal of Optimization Theory and Applications*, vol. 51, pp. 453-473, 1986.

[6] ZXCGR subroutine specification, *IMSL Library*, Houston, TX, 1982.

[7] M.J.D. Powell, "Restart procedures for the conjugate gradient method", *Mathematical Programming*, vol. 12, pp. 241-254, 1977.

[8] L. Nazareth, "A conjugate gradient algorithm without line searches", *Journal of Optimization Theory and Applications*, vol. 23, pp. 373-387, 1977.

[9] L.C.W. Dixon, D.G. Ducksbury and P. Singh, "A new three term conjugate gradient method", Numerical Optimization Center, Hatfield Polytechnic, England, Technical Report No. 130, 1983.

[10] S. Kirkpatrick, C.D. Gellatt, M.P. Vecchi, "Optimization by simulated annealing", *Science*, vol. 20, pp. 671-680, 1983.

[11] F. Romeo and A. Sangiovanni-Vincentelli, "Probabilistic hill climbing algorithms: properties and applications", U.C. Berkeley Electronic Research Laboratory, Memorandum No. UCB/ERL M84/34, 1984.

[12] A. Casotto, F. Romeo and A. Sangiovanni-Vincentelli, "A parallel simulated annealing algorithm for the placement of macro-cells", *Proc. Int. Conf. on Computer-Aided Design*, Santa Clara, CA, pp. 30-33, 1986.

[13] N-P. Chen *et al.*, "BBL.2 user's manual", U.C. Berkeley Electronic Research Laboratory, Memorandum No. UCB/ERL M85/2, 1985.

[14] N-P. Chen, C.P. Hsu and E.S. Kuh, "The Berkeley Building-Block Layout system for VLSI design", *Proc. International Conference on VLSI*, Norway, pp. 37-44, 1983.

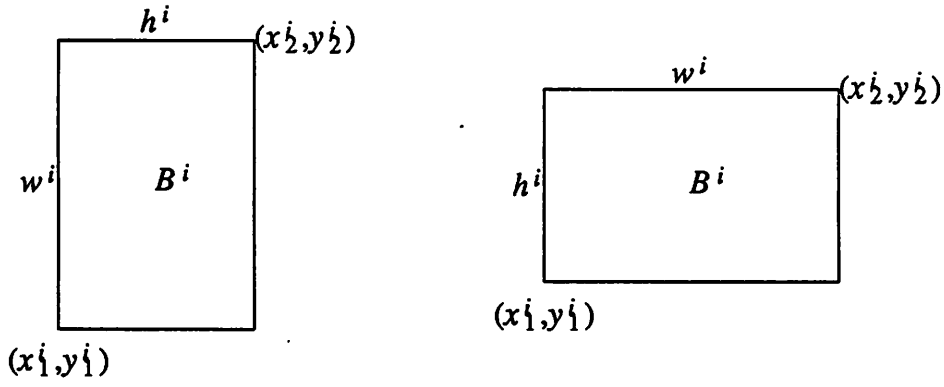

Fig. 1 Vertical and horizontal orientations for a rectangular cell

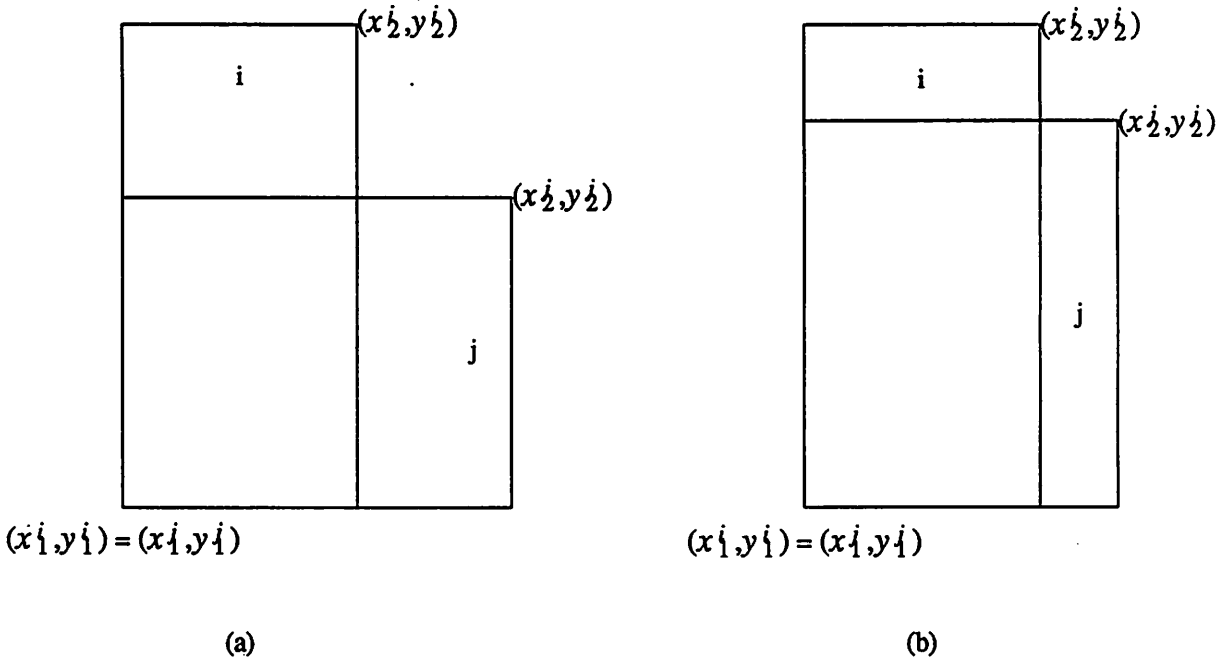

Fig. 2 L-shaped rectangular block with a) the long sides perpendicular b) the long sides parallel.

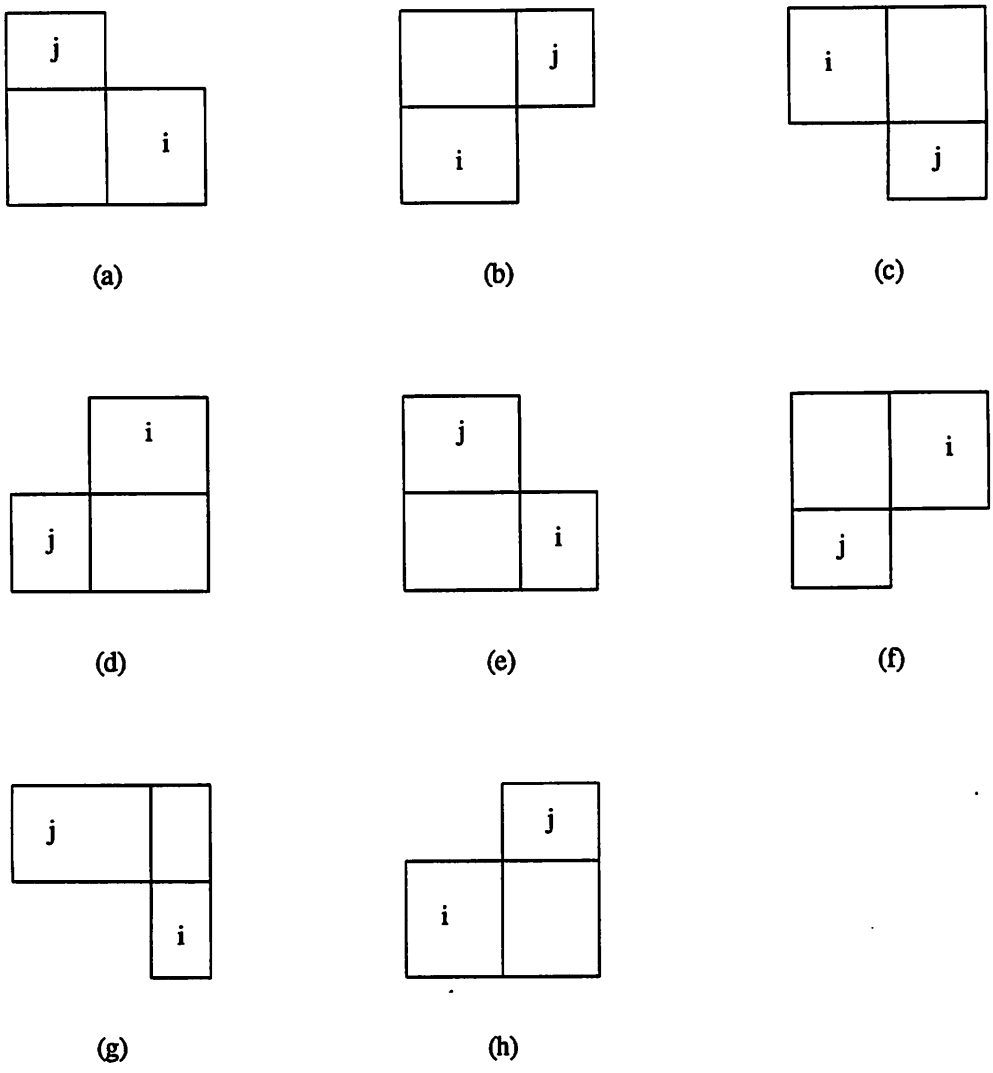

Fig. 3 Eight possible orientations for an L-shaped block

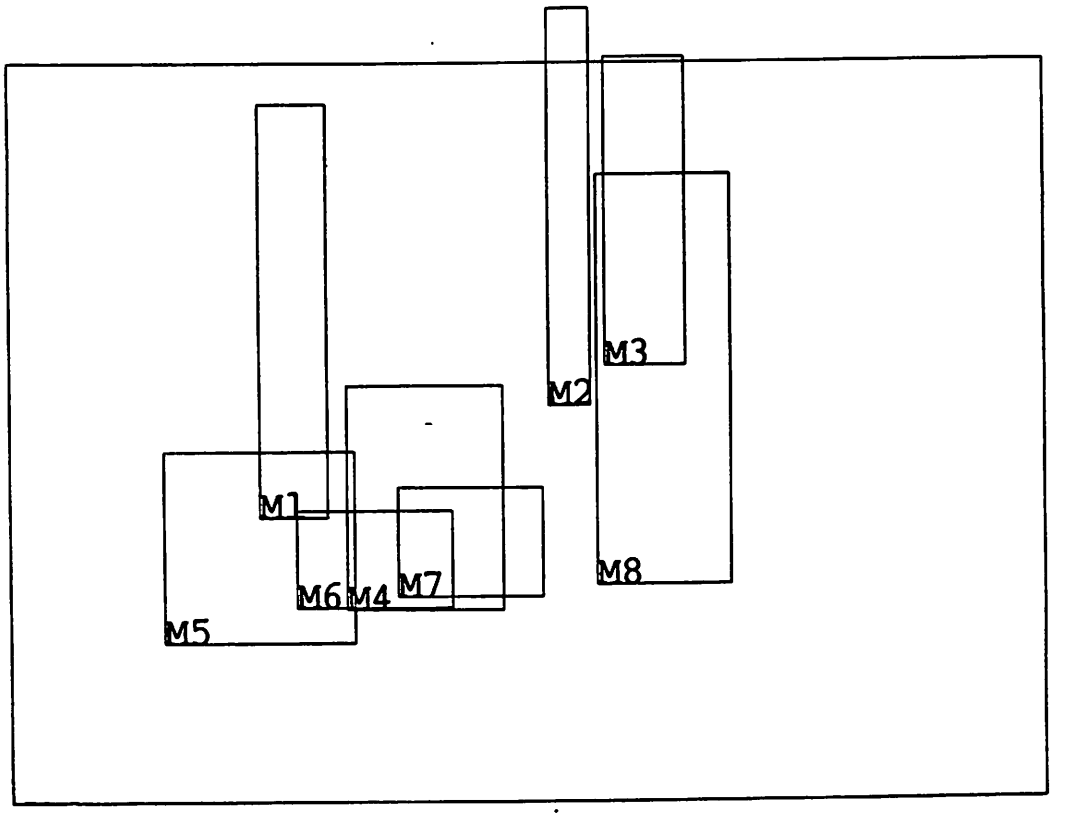

Fig. 4a The solution of the unconstrained optimization problem for the 8-block example.

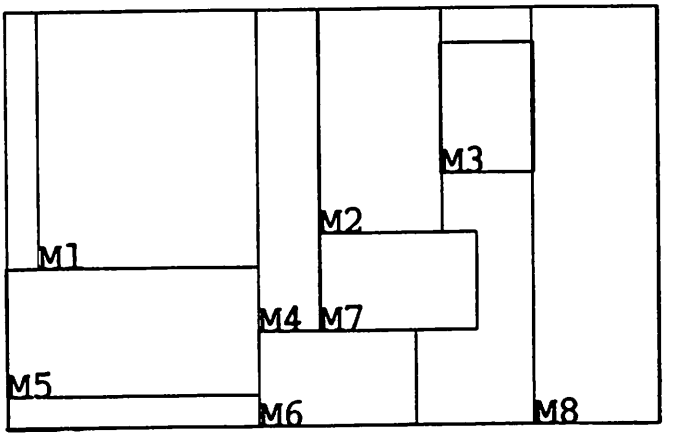

Fig. 4b Final solution for the 8-block problem.

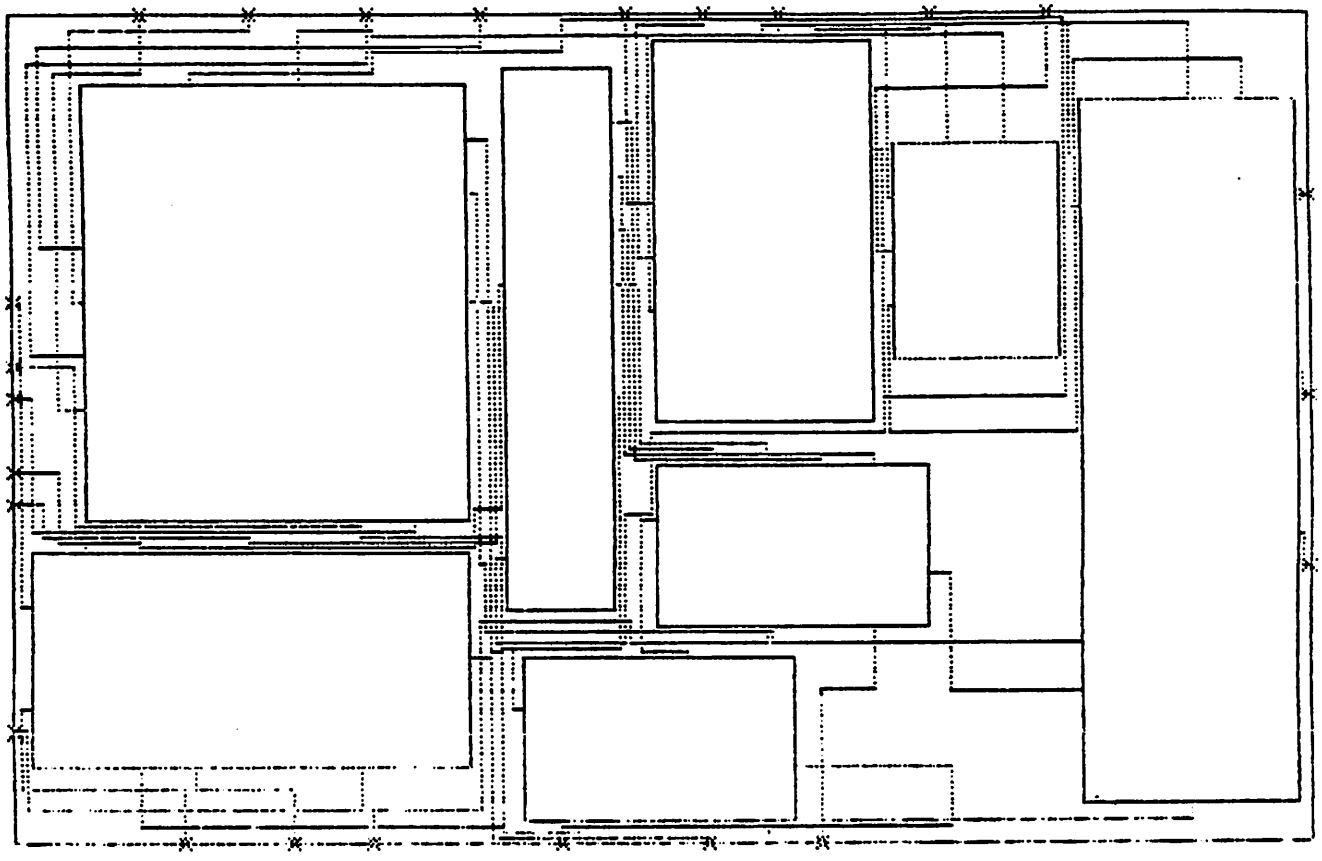

Fig. 4c Final layout of the 8-block example after detailed routing.

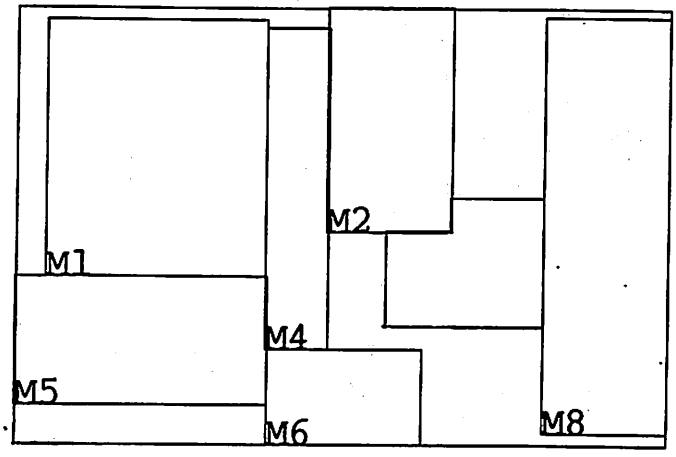

Fig. 5 The solution of the 8-block problem with one L-shaped cell.

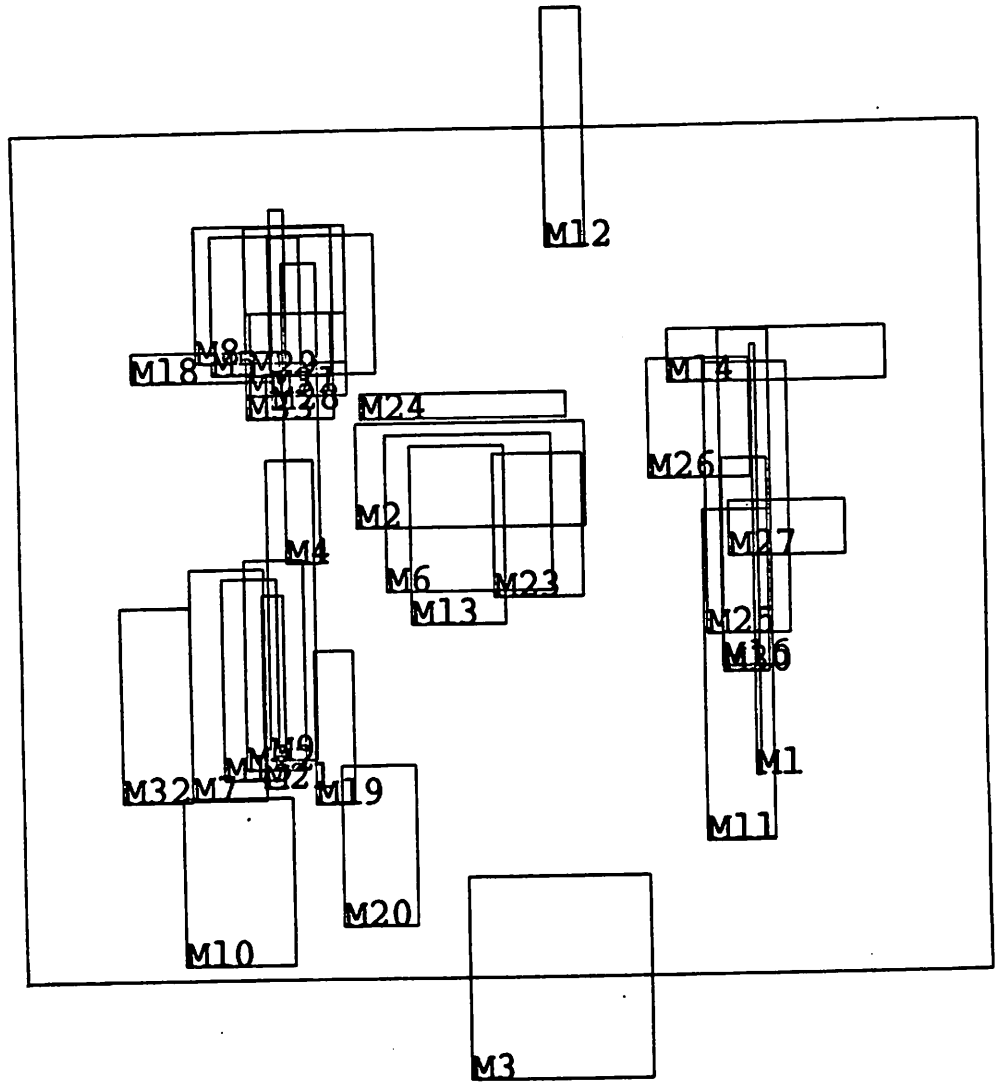

Fig. 6 The solution of the unconstrained optimization problem for the 33-block example.

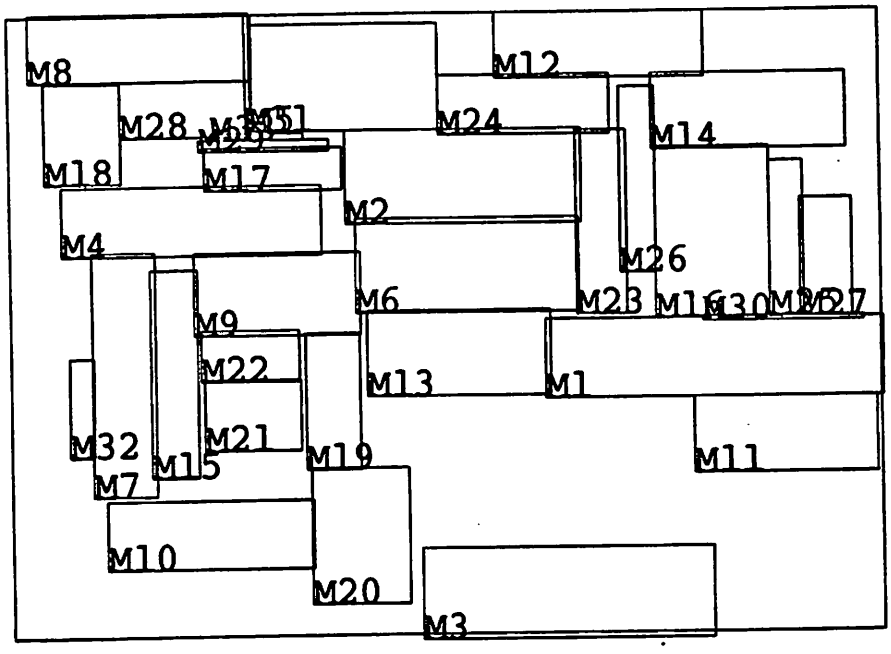

Fig. 7 An intermediate solution for the 33-block example.

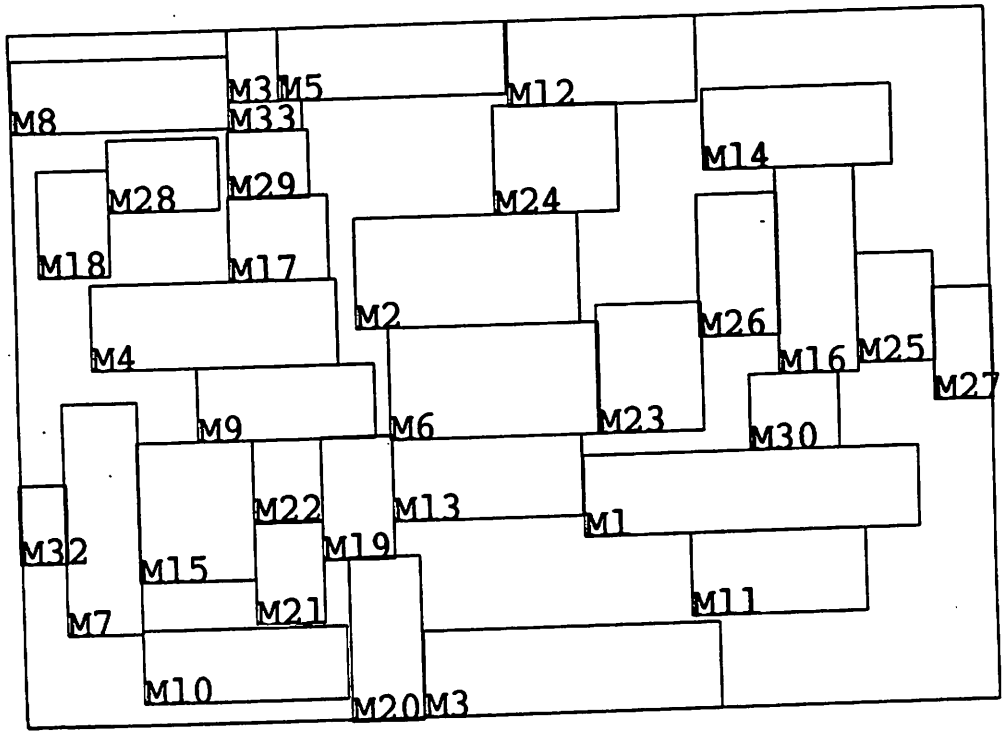

Fig. 8 Final placement for the 33-block example.

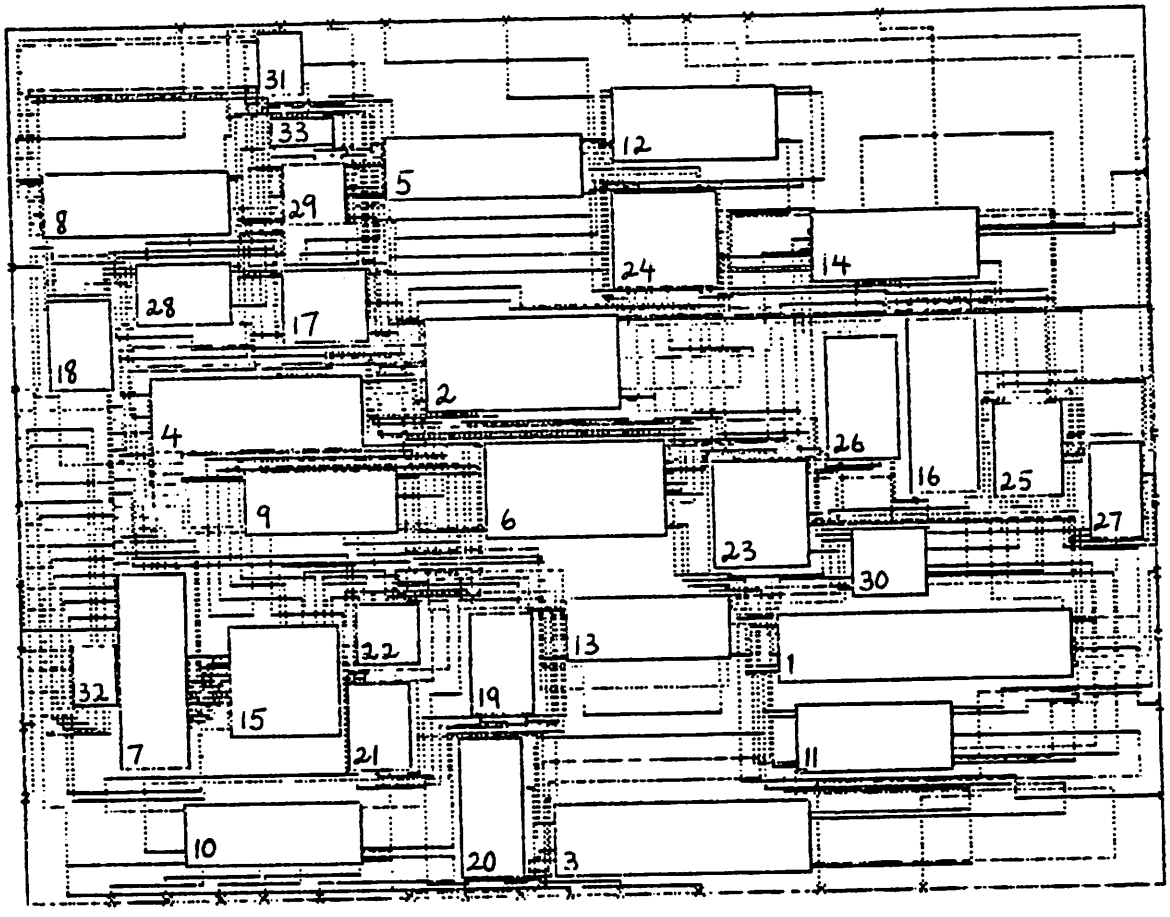

Fig. 9 Final layout of the 33-block example after detailed routing.

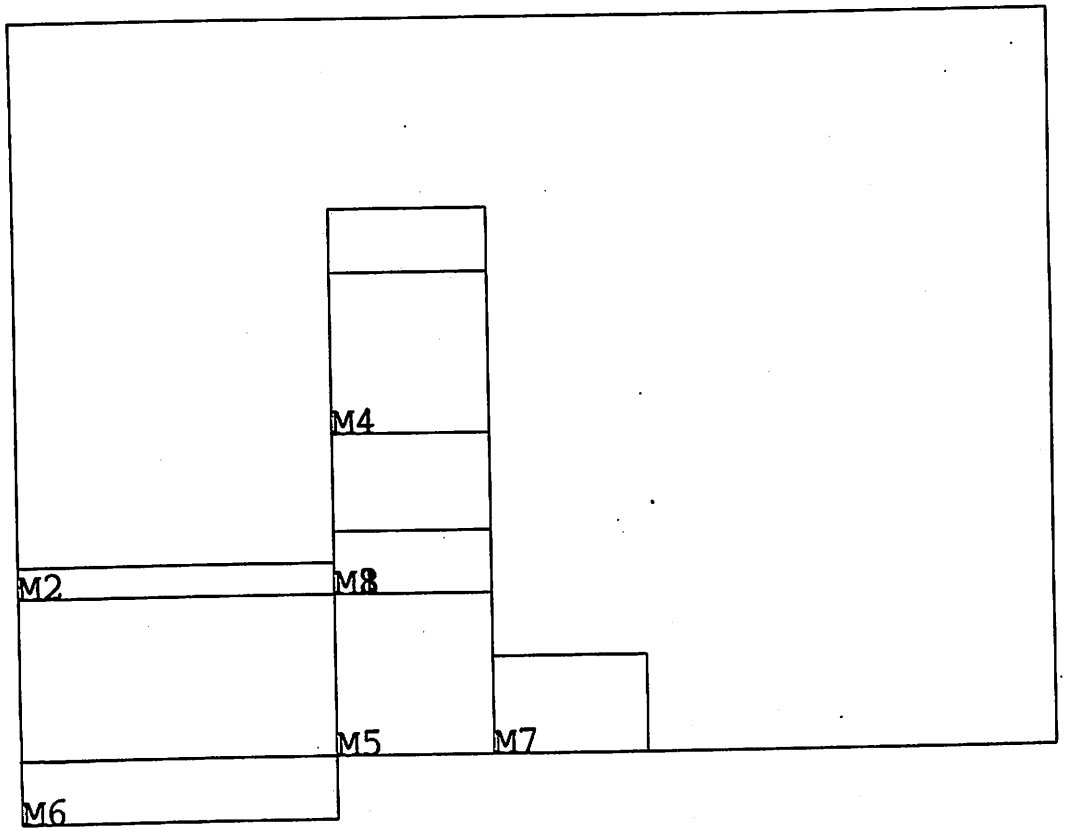

Fig. 10 The initial placement for example 3.

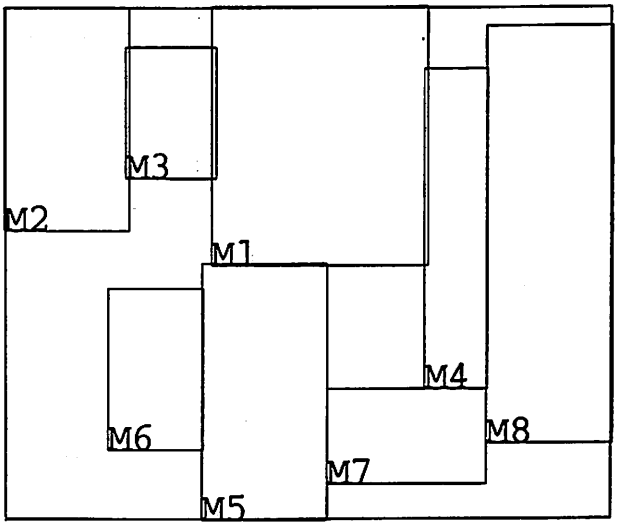

Fig. 11 Solution of example 3 without the random optimization stage.

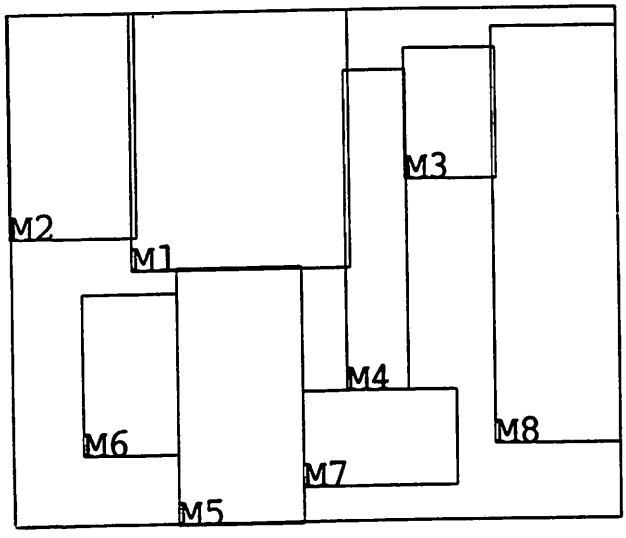

Fig. 12 Solution of example 3 after using an intermediate random optimization stage.