VLSI Circuit Placement with Rectilinear Modules using Three-Layer Force-Directed Self-Organizing Maps

Ray-I Chang and Pei-Yung Hsiao

***Abstract*— In this paper, a three-layer force-directed self-organizing map is designed to resolve the circuit placement problem with arbitrarily shaped rectilinear modules. The proposed neural model with an additional hidden layer can easily model a rectilinear module by a set of hidden neurons to correspond the partitioned rectangles. With the collective computing from hidden neurons, these rectilinear modules can correctly interact with each other and finally converge to a good placement result. In this paper, multiple contradictory criteria are accounted simultaneously during the placement process, in which, both the wire length and the module overlap are reduced. The proposed model has been successfully exploited to solve the time consuming rectilinear module placement problem. The placement results of real rectilinear test examples have been presented, which demonstrate that the proposed method is better than the simulated annealing approach in the total wire length. Furthermore, on the average, the central processing unit (CPU) time for the proposed method running on a sequential machine is 15 times faster than that required by the simulated annealing method. The appropriate parameter values which yield good solutions are also investigated.**

***Index Terms*— Force-directed placement method, molecule model, query-based learning, rectilinear circuit, three-layer self-organizing maps.**

I. Introduction

GIVEN a set of circuits and how they are connected to each other, the objective of circuit placement problem is to optimally locate these modules within a specified layout region. It determines the locations of modules such that all the constraints are satisfied and the estimated total wire length is minimized. In the physical circuit layout, the connection wire length between modules is minimized to reduce the delays associated with longer wire nets and speed up the operation of chip. Good circuit placement is a key aspect in the very large scale integration (VLSI) design, which has been proven to be NP-hard [1]. Although a number of heuristic algorithms have been proposed in the past with varying successes [2]–[8], [25]–[30], they are inherently sequential and unable to efficiently exploit massively parallel architecture. Artificial neural networks with massive parallelism have been demonstrated to resolve various optimization problems [9]–[11], [16]–[24], [43]. The development of a high-speed neural-based CAD system for the design of VLSI chips has become one of the interesting research topics. Moreover, current researches in neural-network implementations have shown the feasibility of building analog electronic neural networks with hundreds and thousands of neurons [12]–[13]. It also demonstrates that the massive parallelism neural network, e.g., Hopfield nets (HOP's) [14] or Kohonen's self-organizing maps (SOM's) [15], can be implemented as a hardware accelerator to accelerate the time consuming VLSI CAD processes. In this paper, a novel three-layer neural-network model based on SOM is proposed to solve the circuit placement problem with rectilinear modules. It is the first neural-based algorithm that is proposed to place the arbitrarily shaped rectilinear modules. Our experiments have shown that the proposed algorithm is better than Sechen's approach [25] in both the obtained total wire length and the simulated CPU time on a sequential machine. Moreover, the placement region is also smaller.

Over the years, a wide repertoire of neural-network-based placement methods have been suggested. These methods can be classified into two major categories, HOP-type networks and SOM-type networks. The HOP with a symmetrical interconnection matrix has been applied to minimize the connection wire length in one [16] and two-dimensional (2-D) circuit placement problems [17]–[21]. In 1989, Yu [17] has tried to obtain a placement result using HOP, but not with great success. The results of computer simulation show that this model gives much the same solutions as the min-cut algorithm. Kita *et al.* [18] have demonstrated that the quality of the placement solution obtained by means of this model is quite sensitive to some model parameters. Recently, Naft [19] have adapted this model for the traveling salesman problem (TSP) to multiobjective component placement based on wire length criteria and thermal reliability. Although HOP is more reasonable to model an optimization problem, the hardware complexity which represents the problem by a permutation matrix is very high. In 1990, neuron's self-organization property was presented to resolve the 2-D cell placement problem by Hemani and Postula [22]. In this research, a set of output neurons have been organized in a rectangular grid to correspond to the location of cells. An approach called SOAP (self-organization assistant placement) was presented [9] in 1992. In SOAP, SOM is applied as a preprocessor of other heuristic algorithms. An

Manuscript received March 4, 1995; revised May 17, 1995, June 8, 1995, and May 7, 1997. This work was supported in part by the National Science Council, R.O.C., under Contract NSC84–2213–E009–040.

R.-I Chang is with the Institute of Information Science, Academia Sinica, Taipei, Taiwan, R.O.C.

P.-Y. Hsiao is with the Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, R.O.C.

Publisher Item Identifier S 1045-9227(97)06049-9.

improvement of SOAP was presented by Zhang and Mlynski [23]. This approach is well suited for standard cell and gate array placement. However, it is not suitable to arbitrarily-sized macro cell placement. Recently, an unsupervised query-based neural network [37] based on the force-directed cell placement method [2] and Kohonen's SOM was proposed [24]. The *force-directed method* [41] explores the similarity between cell placement problem and classical mechanics problem of a system of bodies (circuit cells) attached to springs (wire nets). According to the Hook's law, the force exerted due to the stretching of the springs is proportional to the distance between the bodies connected to the spring. If the circuit cells were allowed to move freely, they would move by the force until the system achieved equilibrium. The same idea is used for SOM where the neurons connected to each other by networks are supposed to exert stimulus (forces) on each other. Following the query-based learning technique [42], the input samples can be queried by the force-directed method. This method has been successfully applied to the circuit placement problem with arbitrarily-sized cells in [24]. The experiments have shown that this approach is better than the previous neural-based placement methods in both the performance and the hardware requirement.

At present, VLSI circuit has become so dense and so complicated that a custom chip may contain lots of macro blocks. It is noted that the macro blocks used in VLSI design may be of arbitrary shapes. Compare to the physical custom chip layout, the major drawback of traditional neural-network-based placement algorithms is that they cannot handle the circuit blocks with arbitrary rectilinear shapes. In practice, the boundary of the modules are general rectilinear shape if the rectilinear hull for the layout of the predesigned modules is taken. The design process of a complex custom chip can cost millions of dollars and a year or more may elapse in moving from initial characterization of the chip to a working design. Thus, the industries need a placement technique which can well place the rectilinear modules with short turn-around time. Moreover, if the placement procedure can take the general rectilinear shape instead of taking the bounding rectangle or the limited L-shaped hull of the modules, the chip area can be optimally utilized as there will be less dead space. In this paper, a highly parallel method based on the force-directed SOM (FDSOM) [24] with additional hidden layer is proposed to resolve the more complex rectilinear module placement problem.

Key to this proposed approach is a novel representation method which simply models a rectilinear module by a set of decomposed rectangles, called the *molecule model*. Assume that the output layer and the input layer in the proposed neural network are used to represent the rectilinear modules and the input sample vector, respectively. We can model these decomposed rectangles by the neurons in an additional hidden layer. The connection weights from input layer to the output layer and the hidden layer are used to represent the positions of rectilinear modules and their decomposed rectangles, respectively. The adjustment of these weights can be interpreted as the movement of the corresponding blocks. The proposed network model is learned from a sequence of module movements and gradually organized toward a good placement solution. Different from the original two-layer model, there are connections between each output neuron and its corresponding hidden neurons to *synchronize* the motion of each rectilinear module and its decomposed rectangles. In the proposed three-layer model, the adaptation of an output neuron is decided by the adaptations of all the corresponding hidden neurons. In another word, the motion of a rectilinear module is decided by the motions of the corresponding rectangles. This study investigates the possibility of utilizing the collective computing property of neural network to obtain a better decision from the complex-description system, called *collective-decision*. Comparing with our previous [24] paper with limited rectangular blocks and reduced circle cells, this proposed three-layer SOM has lots of differences such as the molecule modeling, the ideal distance, and the dynamic objective function. Our experiments show that the proposed network model with collective computing property can obtain effective and efficient solutions for the rectilinear module placement problem. Experiments showed that the obtained total wire length was 16.5% better than the simulated annealing approach [25] on the average. The central processing unit (CPU) time running on a sequential machine was 15 times faster than that required by the simulated annealing method [15]. Besides, the applied layout area is also smaller. More comparisons with the heuristic constructive approach and the analytical method are also presented. The rest of this paper is organized as follows. We first review the previous works in the rectilinear module placement problem in Section II. Next, the proposed problem definition and useful notations are described in Section III. We also demonstrate that the rectilinear module placement problem is well suited to be solved by a collective computing neural model. In Section IV, the organization and behavior of the proposed three-layer neural network are described in comparison with the conventional models. Section V presents the proposed rectilinear module placement algorithm. Finally, experimental results and the conclusion are given in Section VI and Section VII, respectively.

II. Previous Rectilinear Module Placement Techniques

In macro/custom chip layout, one can easily encounter a circuit block that is either of rectilinear shape or that can be approximated by a rectilinear-shaped block. The placement of arbitrarily shaped rectilinear modules is more complicated than that of rectangular modules. Most of the earlier works in the circuit placement problem have implicitly assumed that the shapes of the placement modules are rectangular or limited type of rectilinear (like L-shape). Furthermore, they have conducted experiments on rectangular placement cases only. Unfortunately, the placement method that is good for rectangular modules or limited L-shaped modules is not necessarily applicable to the general rectilinear case. Without taking the actual rectilinear modules into consideration, their placement results always leave a lot of dead spaces and need a larger layout area. Since the placement of rectilinear modules

is so complex, only a few of algorithms are reported and tested by real rectilinear modules.

In 1988, Sechen [25] presented a simulated annealing algorithm to place rectilinear modules based on a theoretically derived statistical annealing schedule. In this algorithm, two cells can perturb with interchange or one cell can perturb with translation, reflection, or rotation in one iteration. They accept the movement if it reduces the objective function. On the other hand, if it increases the objective function, the movement is accepted or rejected based on the Boltzmann distribution function. This process is repeated until the temperature is too low or the system has no distinct change. Although it is proven that the simulated annealing technique can potentially achieve the global minimum, the computation cost is very expensive that requires enormous amounts of execution time. Furthermore, the obtained results are sensitive to process parameters and sometimes unacceptable in practice. Recently, Swartz and Sechen [28] have presented an algorithm based on this method to resolve the timing driven placement of rectilinear shaped macro cells. The timing constraint is simply added to the original objective function. Then, the simulated annealing technique is applied to minimize this new timing driven objective function. A heuristic constructive algorithm for the placement of rectilinear modules was presented by Wimer and Koren [26] in 1988. This algorithm applies a novel point list data structure to represent the partial layout result, and uses the cluster growth technique to construct the rest of the placement. The cluster growth algorithm first takes the module with maximum connections as the seed and then selects an unplaced module based on a selection function for the next placement. The defined selection function measures the module size and the number of connection wires to the already placed modules to select the best module placement. This algorithm decides the best placement position by a place function which computes the average weighted center of connection wires to the already placed modules without module overlap. Although the computations of the constructive method are simple and fast, as shown in our experiments, it is not possible to the change previous decisions and have the disadvantage to frequently trap into local minimum.

In 1993, Lee [30] has presented a bounded 2-D contour searching algorithm to generate the floorplans with arbitrarily shaped rectilinear modules. The spirit of this algorithm is similar to some 2-D compaction techniques that tries to minimize the chip area along a selected compacting direction. During the compacting process, an additional novel incremental wire length calculation method is presented to minimize the connection cost. This algorithm iteratively improves the design from a given initial configuration and the system only accepts the best one as the next configuration. This greedy property may cause the algorithm to fall into a local minimum. Furthermore, the placement results are dependent on their initial configurations. Since the cell model used here is called *soft module*, which can change its shape within a specified aspect ratio range, it is hard to compare wit the conventional placement methods with fixed shape cells. In 1994, Ding *et al.* [33] presented a two-phase hybrid method with the min-cut approach [44] and simulated annealing [25]. They applied the placement result of the constructive method as the initial configuration for a medium-temperature simulated annealing method. Experiments show that this two-phase hybrid method [33] can obtain better results than that of the constructive method [26] and the simulated annealing method [25]. Moreover, it also takes fewer computation time and smaller layout area than that of the simulated annealing method.

III. Notations and Definitions

The representation of a rectilinear block is so complicated that most of the earlier works on circuit placement implicitly assumed a rectangular hull for all modules. In this section, we concentrate on the modeling of rectilinear modules and their placement problem. We begin our description with a novel molecule model to represent the arbitrary-shaped rectilinear modules, and then proceed to an effective local connection method for the wire nets among the rectilinear modules. The issue of problem definition based on the proposed molecule model and local connection method will also be discussed. Our representation enables the rectilinear module placement problem to be solved by a collective computing network model.

A. Arbitrarily Shaped Rectilinear Module

The arbitrarily shaped rectilinear module can be defined as a circuit block with the shape as a simple rectilinear polygon. A simple rectilinear polygon is a polygon whose sides are either vertical or horizontal and, for simplification, has no holes in it. Throughout this paper, the analysis of the rectilinear modules is based on a novel molecule modeling method that models a rectilinear system by the composed molecules, the elementary atoms and their structure. Our representation is analogous to the atomic structure of a element in nature. For example, the molecule of water (H_2O) is made up of two atoms of hydrogen (H) and one atom of oxygen (O). The structure of a water molecule is nearly L-shaped, and the unique oxygen is at the corner-point of this structure. The proposed molecule model is more reasonable to reflect the physical rectilinear circuit module which is arisen from the composition of some low-level rectangular circuit cells (rectangles, for short). Thus, a rectilinear module (molecule) can be made up by a set of rectangles (atoms) with some specified structures. In this paper, the problem input for the rectilinear modules $B = \{b_i | 1 \leq i \leq m\}$ and their decomposed rectangles (for example, $B_i = \{b_{ij} | 1 \leq j \leq m_i\}$ represents a set of rectangles to make up b_i) are given. m is the number of rectilinear modules, and m_i is the number of decomposed rectangles for the rectilinear module b_i. The decomposition of a rectilinear module can be classified into two types: the partition and the cover. If the resulting rectangles can overlap with each other, then the decomposition is a "cover." If the resulting rectangles cannot overlap with each other, then the decomposition is a "partition." Both approaches have been discussed in previous researches [31]. As the module overlap is more complicated for the cover case, in this paper, we use the partition case as the decomposition of a rectilinear module. For each partitioned rectangle b_{ij}, xb_{ij}, and yb_{ij} are the x and

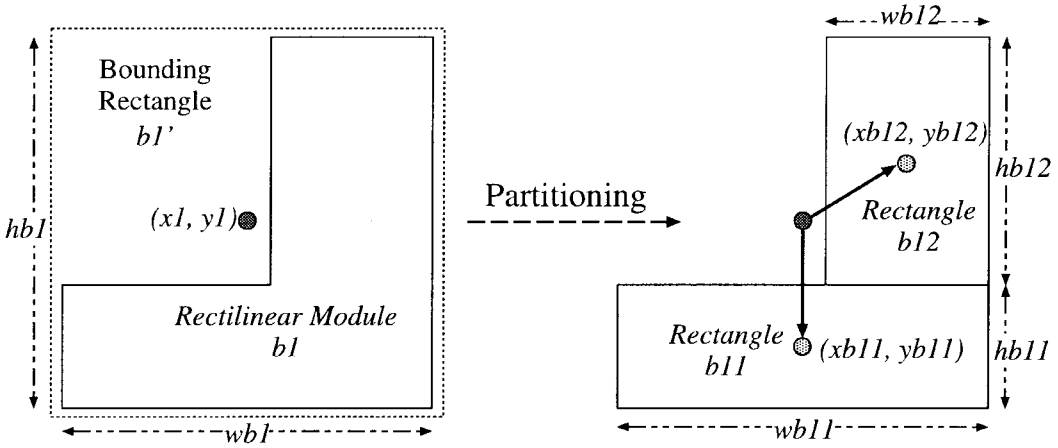


Fig. 1. A simple example to demonstrate the relationship between a rectilinear module b1 and its partitioned rectangles $b11, b12$. All the variables and constants used in this example are labeled to illustrate the relationship of these notations.

the y coordinate of the middle of rectangle b_{ij}. The width and the height of rectangle b_{ij} are wb_{ij} and hb_{ij}. Without considering the module rotation and reflection, in this paper, the structures of their decomposed rectangles would never be changed. We can define the decomposed rectangles in a rectilinear module as *solid-connected*, in which, the offsets among these rectangles are kept constant. Thus, during the placement process, we should keep the offsets among all the rectangles b_{ij} in a set B_i . For simplification, we give xo_{ij} and yo_{ij} as two constants to represent the x and the y offset from the middle of b_i to the middle of b_{ij}. For the definition of the middle of an arbitrarily shaped rectilinear module, we can model the rectilinear modules with their bounding rectangles. Assume that b'_i is the bounding rectangle of rectilinear module b_i. x_i and y_i are the x and the y coordinate of the middle of b'_i. The width and the height of b'_i are wb_i and hb_i. Then, we can define the middle of a rectilinear module b_i as the middle of its related bounding rectangle b'_i, and define the x-offset and the y-offset from b'_i to b_{ij} as $xo_{ij} = xb_{ij} - x_i$ and $yo_{ij} = yb_{ij} - y_i$, respectively. Fig. 1 presents a simple example to demonstrate the relationship between a rectilinear module and its partitioned rectangular cells. All the variables and constants used in this example are labeled to illustrate the relationship of these notations. The problem input for the limited layout region is given as w and h to represent the width and the height of given placement region. During the placement, the middle of each rectilinear module b_i should satisfy the bounding constraint: $wb_i/2 \le x_i \le w - wb_i/2$ and $hb_i/2 \le y_i \le h - hb_i/2$, to minimize the chip area. This means that the module should be at the inside of the placement layout region. A simple example to demonstrate the bounding constraints between a rectilinear module and the given layout region is shown in Fig. 2.

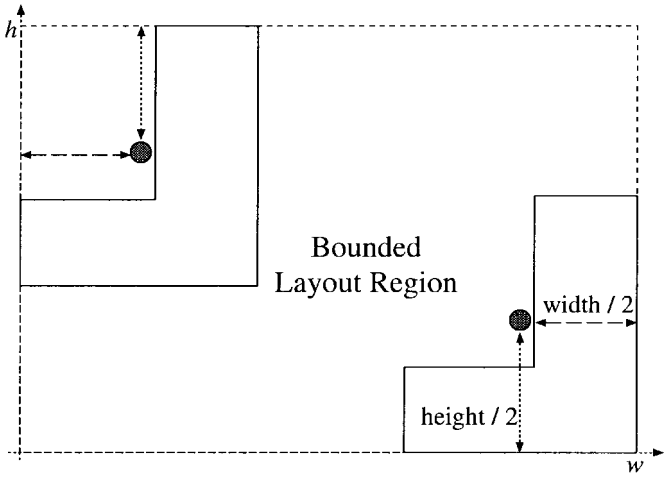


Fig. 2. A simple example to demonstrate the bounding constraint between the rectilinear module $b1$ and the given layout region with width w and height h.

B. Wire Connection and Problem Definition

In VLSI design, each wire net contains a set of pins around the rectilinear modules or the pads around the layout region. After circuit placement, the pins (or the pads) in the same wire net will be connected by the router. Circuit placement is to determine the locations of circuit modules such that the estimated total wire length is minimized. The problem input for the connection wire nets are given as $N = \{N_i | 1 \le i \le n\}$ where n is the number of connection wire nets. Define c_i as the strength of connection for the wire net N_i. The strength of connection, in the simplest case, could be the number of wires. In general, it could be defined as a weighted function of many constraints depending on the application. For example, in a timing driven placement problem, the connection strengths on the critical path would be higher than that of the others. For the simplification of computation, most of the previous approaches assumed that the wires are connected from, or to, the middles of rectilinear modules. This assumption which derived from the rectangular case is not accurate enough for the wire nets on the rectilinear case. In this paper, we assume that the wires are connected from, or to, the partitioned rectangles. The wire connection which connects decomposed rectangles is called *local connection*. Note that this local connection model allows user to estimate the total wire length with more precision without much effort. In this paper, we follow the idea and represent the strength

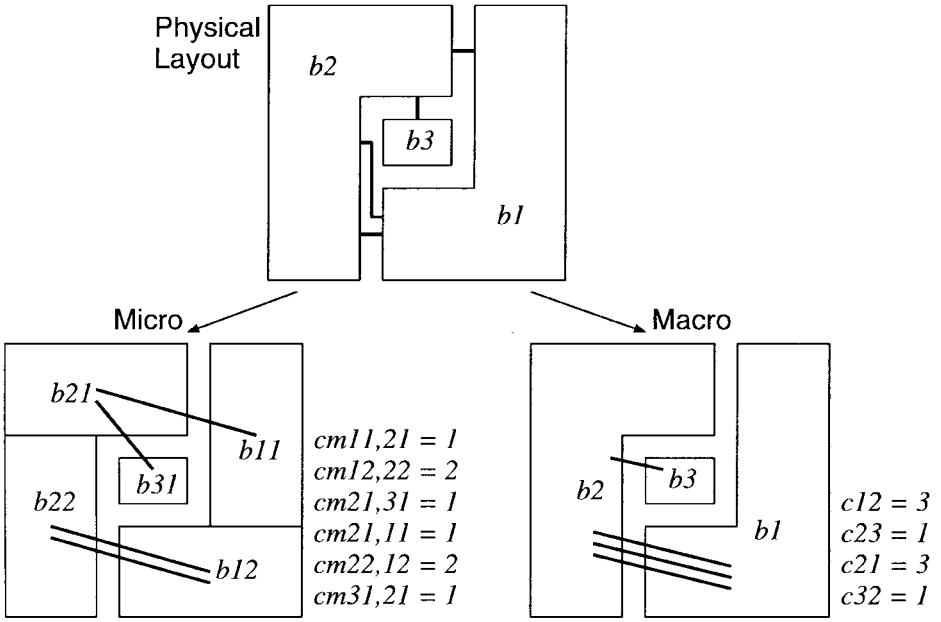


Fig. 3. A simple example to demonstrate the different representation of the wire net model.

of connections between each pair of rectangles with the local connectivity matrix $CM_1 = \{cm_{ij,kl} | 0 \le i,k \le m, 1 \le j \le m_j, 1 \le l \le m_l\}$. $cm_{ij,kl}$ is defined as the summation of wire connectivities between rectangle b_{ij} and rectangle b_{kl}. For the self-organization process of output neurons, the strength of connections between each pair of rectilinear modules (global connections) are also given as the connectivity matrix $CM_2 = c_{ik} | 0^1 \le i,k \le m$ where

$$c_{ik} = \sum_{j=1}^{m_i} \sum_{l=1}^{m_k} cm_{ij,kl}. \tag{1}$$

An example to illustrate the relationship of these notations is shown in Fig. 3. Given the connectivity $cm_{ij,kl}$, the connection wire length between the rectangles b_{kl} and b_{ij} can be simply defined as $cm_{ij,kl} \times d_{ij,kl}$. The Euclidean distance $d_{ij,kl}$ between b_{kl} and b_{ij} is defined as $d_{ij,kl} = \sqrt{dx_{ij,kl}^2 + dy_{ij,kl}^2}$, where $dx_{ij,kl} = xb_{kl}$ - xb_{ij} and $dy_{ij,kl} = yb_{kl} - yb_{ij}$ are defined as the x and y coordinated displacement from the middle of b_{kl} to the middle of b_{ij}, respectively.

We can define the circuit placement problem as a constrained optimization problem which shorten the connection wire length among circuit blocks without module overlap. In other words, circuit placement is to minimize the total wire length C_{WL} under the criterion that the module overlap C_{MO} is zero. As the module partitioning described above, we can define C_{WL} and C_{MO} as follows. It is noted that these definitions have satisfied the collective computing property and can be easily modeled by a three-layer neural network

$$C_{WL} = \sum_{i=1}^{m} \sum_{j=1}^{m_i} \sum_{k=i+1}^{m} \sum_{l=1}^{m_k} cm_{ij,kl} \times d_{ij,kl}$$
$$C_{MO} = \sum_{i=1}^{m} \sum_{j=1}^{m_i} \sum_{k=i+1}^{m} \sum_{l=1}^{m_k} od_{ij,kl}. \tag{2}$$

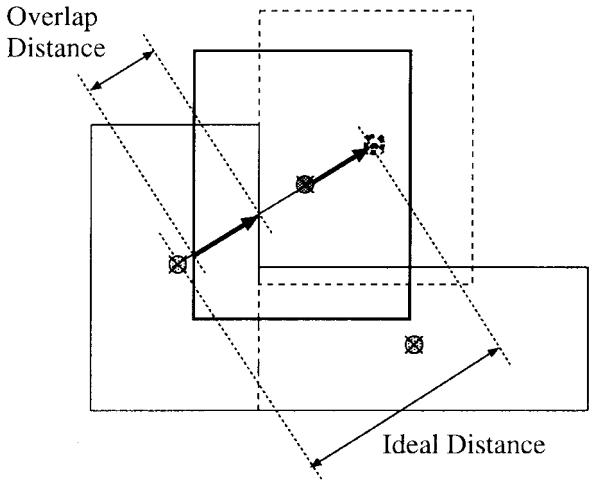


Fig. 4. A example with an L-shaped module and a rectangular module is presented to demonstrate the ideal distance and the overlap distance.

In (2), the overlap distance, $od_{ij,kl}$, is our symbolic representation of the module overlap between b_{kl} and b_{ij}. It can be defined as follows:

$$od_{ij,kl} = (id_{ij,kl} - d_{ij,kl}) \times f_h(id_{ij,kl} - d_{ij,kl})$$
$$id_{ij,kl} = \min\left(\frac{hb_{ij} + hb_{kl}}{2 \times |dy_{ij,kl}|}, \frac{wb_{ij} + wb_{kl}}{2 \times |dx_{ij,kl}|} \right) \times d_{ij,kl} \tag{3}$$

where $id_{ij,kl}$ is the ideal minimum distance between b_{kl} and b_{ij} that minimizes the connection wire length and module overlap. Function $f_h(.)$ used in (3) is the 0-1-hardlimit threshold function where $f_h(x) = 1$ if $x > 0$, otherwise $f_h(x) = 0$. Fig. 4 presents an example with a simple L-shaped module and a rectangular module to demonstrate this novel representation method. Note that, in this paper, the definitions of wire length and module overlap are both dependent on their distance $d_{ij,kl}$.

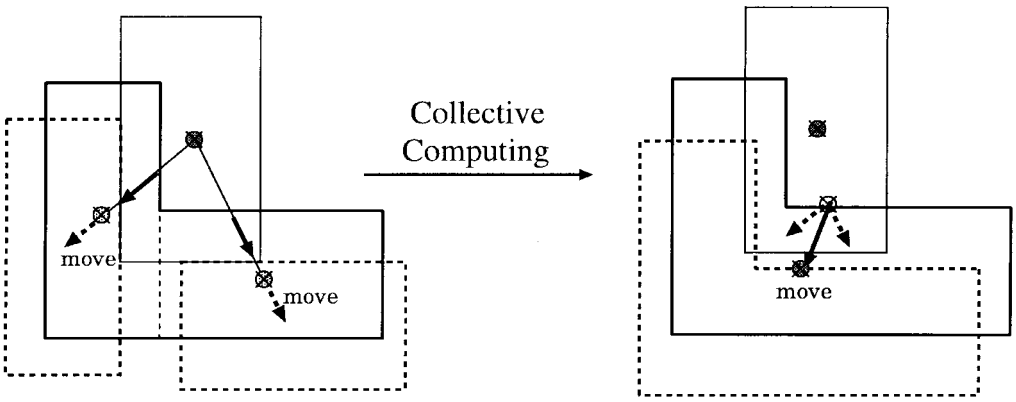


Fig. 5. An example of collective computation is shown in which the motion of a rectilinear module can simply be decided by the summation of the motions of the corresponding rectangles.

This definition is well suitable to model the placement problem with SOM that represents the stimulus between neurons by their distance.

IV. Network Organization and Behavior

The problem definition described in (2) and (3) naturally leads the rectilinear module placement problem to be implemented on FDSOM with three collective computing layers: input layer, hidden layer, and output layer. The traditional FDSOM uses only two layers that correspond to the input and the output layers. In this paper, the output layer which describes the pattern classes in the real world is called macro-description-layer. The additional hidden layer which represents the micro features of the patterns is called micro-feature-layer. With this micro-feature-layer, the proposed neural network can represent a complex pattern by a set of simple features. Computation between two sets of simple features is usually simpler than that between two complex objects. In this paper, we apply this idea to model the complex rectilinear modules in circuit placement problem. The partitioned rectangles are treated as a kind of features to represent their corresponding rectilinear modules. Fig. 5 shows an example of the collective computation where the motion of a rectilinear module can be simply decided by the summation of the motions of the corresponding rectangles.

For simplification, the neurons in the input layer, hidden layer, and output layer are called input-neurons, micro-neurons, and macro-neurons, respectively. We can represent the rectilinear module b_i by a macro-neuron hn_i in this proposed model. The micro-neuron mn_{ij} corresponds to the partitioned rectangle b_{ij}. Connection weight between micro-neurons mn_{ij} and mn_{kl} specifies the strength of connection between rectangles b_{ij} and b_{kl}. It is defined as the local connectivity $cm_{ij,kl}$. The connectivity c_{ij} which represent the connection weight between macro-neurons hn_i and hn_j is used as their degree of neighboring. As in the SOM, each input-neuron is corresponding to one coordinate of the placement space. It uses only k input-neurons to represent a k-dimensional placement space. In the case of placement on a 2-D space, it is $w_{i1} = x_i$ and $w_{i2} = y_i$. The input-neuron in_j is used to represent the jth coordinate value of the input sample vector. Connection weight between input-neuron in_j and macro-neuron hn_i, say w_{ij}, specifies the jth coordinate value of the rectilinear module b_i. It is called the jth position weight of the macro-neuron hn_i. This method can be easily extended to the placement with various constraints on their connection and dimension. For example, w_{i3} is an additional weight needed for the placement on three-dimensional space to represent the z coordinate value. The same to the weights of macro-neurons, connection weights between micro-neurons and input-neurons are used to specify the location (xb_{ij}, yb_{ij}) of each rectangle b_{ij} in a given 2-D region. The set of micro-neurons which represents the partitioned rectangles of a rectilinear module is called a micro-group. Neurons in a micro-group are connected to an unique output-neuron which represents the corresponding rectilinear module. The relationship between the proposed neuron model and the rectilinear placement model is illustrated in Fig. 6. Note that the spatial relation of the partitioned rectangles are also a kind of feature to represent a rectilinear module. It should be taken into consideration for the design of our proposed algorithm.

The objective of this algorithm is to iteratively adapt the weights of neurons, so as that the total wire length is minimized without module overlap. To minimize the connection wire length, modules with more connections should be placed closer together. Thus, we can use the self-organization rule to adjust weights of neurons, so that neurons connected closely in topology are placed closer. Kohonen's rule for self-organization is known to create an optimal 2-D feature map of higher dimensional sample vectors. We input the sample vectors from the network stimulus to adjusts weights of the synapses. In this paper, the excitation stimulus and the inhibition stimulus between neurons are used to represent the criteria of wire length and module overlap between cells, respectively. Based on this proposed model, a self-organization algorithm is proposed to find the positions of modules such that the closely related modules are placed near one another without overlapping, or, loosely speaking, the whole interconnection wire length is minimized without module overlap. Different from the rectangular case, during the rectilinear module placement process, the adaptation of the output neuron is feedback to its connected micro-neurons

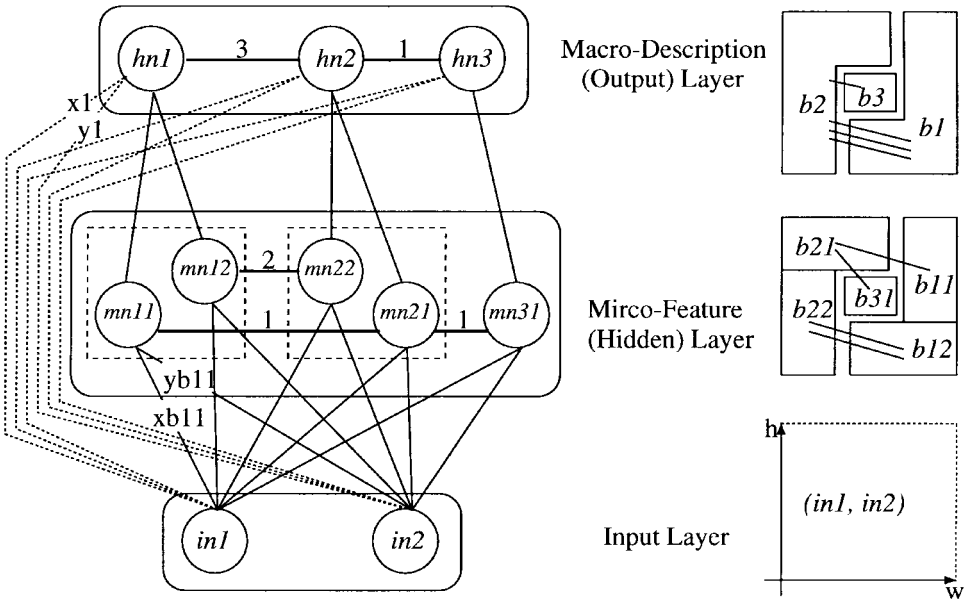


Fig. 6. The relationship between the proposed three-layered neural network and the rectilinear module placement problem.

to update the spatial relation of the partitioned rectangles. In our proposed algorithm, the initial placement evolves in a continuous manner toward the final solution. This continuous representation is quite different from the one found in the traditional algorithms (like HOP), where all intermediate solutions being examined are various permutations of cells. By associating iterative position rearrangements of modules with the adaptations of neuron weights, a significant gain in speed can also be achieved over the conventional software algorithms running on sequential computers.

The criteria of less module overlap and shorter wire length are contradictory. Odawara *et al.* [3] have analyzed the circuit placing of expert designers and found two phases in their placement technique: relative placement phase and spacing phase. In the first phase, wire length criterion is more important. The placement with some module overlap would be permitted in this phase. In the second phase, the module overlap criterion is more important. Expert designers remove the module overlap such that the total wire length is kept as low as possible. We can apply expert knowledge to minimize these contradictory criteria simultaneously [39]. Our approach is to define a dynamic objective function C_{CP} incorporating wire length and module overlap with a gain function as follows:

$$C_{CP} = (1 - g(t)) \times C_{WL} + g(t) \times C_{MO} \tag{4}$$

where the gain function $g(t)$ is used to simulate the degree of importance of the module overlap criterion from expert knowledge. In this paper, we simply define the gain function $g(t)$ as a sigmoid function. The value of $g(t)$ increases gradually over time t to preserve the current configuration and to reduce the overlap and wire length between cells.

V. Circuit Placement with Rectilinear Modules

Although there are many algorithms presented to place rectangular or limited L-shaped modules in a given region [33], they are hard to extend to more complicated arbitrarily shaped rectilinear case. Consider the circuit placement with rectilinear modules, these algorithms always model the modules with their bounding rectangles, and leave lots of dead space in their placement results. In this paper, a rectilinear module placement algorithm is proposed which applies the self-organizing optimization technique and can be implemented with a hardware accelerator.

A. Query-Based Self-Organizing Optimization Technique

Self-organizing neural networks offer a paradigm to understand the internal representation of information in the brain, in particular, the structures and internal organizations. Over the years, a vast amount of effort has already been dedicated to the study of the self-organizing optimization technique [9], [22]–[24], [43]. In this paper, a query-based self-organizing optimization algorithm using a three-layer FDSOM is studied. We use an adaptation of Kohonen's SOM as it is well suited for implementation on massively parallel architecture. In traditional self-organization algorithm, the topological relation between neurons is represented as the lateral interaction. The lateral interaction, stimulus of excitation and inhibition, is used to preserve the degree of similarities and differences between neurons. Using the problem criteria to query (or produce) the input stimulus and adapting the representation weights, the problem solution is achieved if the neurons are self-organized. In this paper, the simultaneously considered criteria of module overlap and wire length are represented as the inhibition and excitation stimulus, respectively [24]. The total stimulus function from micro-neuron mn_{kl} to micro-neuron mn_{ij} is defined as

$$F(d_{ij,kl}) = (1 - g(t)) \times F_w(d_{ij,kl}) + g(t) \times F_o(d_{ij,kl}) \tag{5}$$

where function $F_w(.)$ is a penalty function to minimize the wire length between rectangle b_i and rectangle b_j [41], and it is also defined as the lateral excitation from micro-neuron

mn_{kl} to micro-neuron mn_{ij}. We can simply define $F_w(.)$ as follows [24]:

$$F_w(d_{ij,kl}) = cm_{ij,kl} \times \frac{d_{ij,kl}}{id_{ij,kl}}. \quad (6)$$

The overlap penalty function is calculated between micro-neurons to reduce the module overlap. It is noted that the traditional SOM usually uses the distance measurement to represent the stimulus between neurons. We redefine the overlap penalty function to preserve the ideal distance between rectangles. The overlap penalty function $F_o(.)$ from micro-neuron mn_{kl} to mn_{ij} is defined as the inhibition from micro-neuron mn_{kl} to mn_{ij}. As shown in [24], the overlap penalty function $F_o(.)$ can be simply defined as follows:

$$F_o(d_{ij,kl}) = -(\frac{id_{ij,kl}}{d_{ij,kl}})2 \times f_h(id_{ij,kl} - d_{ij,kl}) \quad (7)$$

where $f_h(.)$ is a 0-1-hardlimit threshold function used as (3). This model is reasonable to reflect the connection and overlap between rectangles directly. Therefore, the placement problem can be solved if the related self-organization problem is solvable. The intermodule and module-to-chip overlaps are removed in such a way that the total wire length is kept as low as possible. Using the collective computing of the three-layer neural network described in Section IV, this query-based self-organizing optimization technique can be easily applied to the rectilinear module placement problem.

B. Rectilinear Module Placement Algorithm

In the following, an intuitive overview of the algorithm will be described and then a more detailed description will be given. The system flow of the algorithm is as shown in Fig. 7. Initially, we set the positions of each macro-neuron hn_i as random values and setup the positions of micro-neurons mn_{ij} by the corresponding macro-neuron hn_i and prespecified offset (xo_{ij}, yo_{ij}). For each iteration, we randomly select a macro-neuron, say hn_i, as the winner from a uniform distribution. All the micro-neurons which connect to the winner macro-neuron are set to be active. For each active micro-neuron, the total stimulus (excitation and inhibition) work on it would be computed. Then, using collective computation property, we generate the input sample vector V_i by the summation of stimulus. Finally, the weights associated with the winner macro-neuron hn_i and its neighbors are adapted to make these neurons more responsive to the current input V_i. The macro-neuron moves toward the sample vector and induces its micro-neurons to do so. Constraint of the bounded placement region is modeled as the threshold function of macro-neurons. The bounding threshold functions for macro-neuron z are easily defined as follows:

$$\begin{aligned} fx_z(a) &= w_z/2 && \text{if } (a < w_z/2) \\ &= w - w_z/2 && \text{if } (a > w - w_z/2) \\ &= a && \text{others.} \\ fy_z(a) &= h_z/2 && \text{if } (a < h_z/2) \\ &= h - h_z/2 && \text{if } (a > h - h_z/2) \\ &= a && \text{others.} \end{aligned} \quad (8)$$

This process is repeated until the system is convergent. A more detailed description of the proposed algorithm is described as follows.

Step 1) Initialization: $t = 0$. Set the weight (x_i, y_i) as random values, and all neurons be nonactive. The position weight (x_{ij}, y_{ij}) can be initialized as $(x_i + xo_{ij}, y_i + yo_{ij})$.

Step 2) Randomly (or with some priorities) select a nonactive macro-neuron, say hn_i. If there is no nonactive macro-neuron, go to Step 7).

Step 3) Set the macro-neuron hn_i and the related micro-neurons mn_{ij} to be active. Compute the sample vector

$$V_i = (vx_i, vy_i).vx_i = x_i(t) + \frac{Fx_i}{\sqrt{Fx_i{}^2 + Fy_i{}^2}}$$

and

$$vy_i = y_i(t) + \frac{Fy_i}{\sqrt{Fx_i{}^2 + Fy_i{}^2}} \quad (9)$$

where

$$Fx_i = \sum_{j=1}^{m_i} \sum_{k=1}^{m} \sum_{l=1}^{m_k} F(d_{ij,kl}) \times \frac{dx_{ij,kl}}{d_{ij,kl}}$$

$$Fy_i = \sum_{j=1}^{m_i} \sum_{k=1}^{m} \sum_{l=1}^{m_k} F(d_{ij,kl}) \times \frac{dy_{ij,kl}}{d_{ij,kl.}}$$

The total force (stimulus) function F(.) is defined in (5).

Step 4) Update the position of macro-neuron b_i and its neighbors by Kohonen's learning rule. For all such macro-neurons b_z, perform

$$\begin{aligned} x_z(t+1) &= x_z(t) + \eta(u,t) \times (vx_i - x_z(t)) \\ y_z(t+1) &= y_z(t) + \eta(u,t) \times (vy_i - y_z(t)) \end{aligned} \quad (10)$$

where $\eta(u,t)$ is the neighbor function and u is the topological distance between b_z and b_i. As the bounded constraint in (8), we have

$$\begin{aligned} x_z(t+1) &= fx_z(x_z(t+1)) \text{ and } y_z(t+1) \\ &= fy_z(y_z(t+1)). \end{aligned}$$

Note that, the moving distances of the neurons are decreased as the number of iterations is increased or as u is increased.

Step 5) For all b_z, update the position of corresponding micro-neurons, say b_{zj}

$$\begin{aligned} xb_{zj}(t+1) &= xb_{zj}(t) + (x_z(t+1) - x_z(t)) \\ yb_{zj}(t+1) &= yb_{zj}(t) + (y_z(t+1) - y_z(t)). \end{aligned} \quad (11)$$

Step 6) Go to Step 2).

Step 7) Increase t. Set all neurons be nonactive. If (system doesn't converge) then go to Step 2).

Assume that the system is convergent if $t > t_{max}$. A large number of experiments are conducted to evaluate the performance of the proposed algorithm.

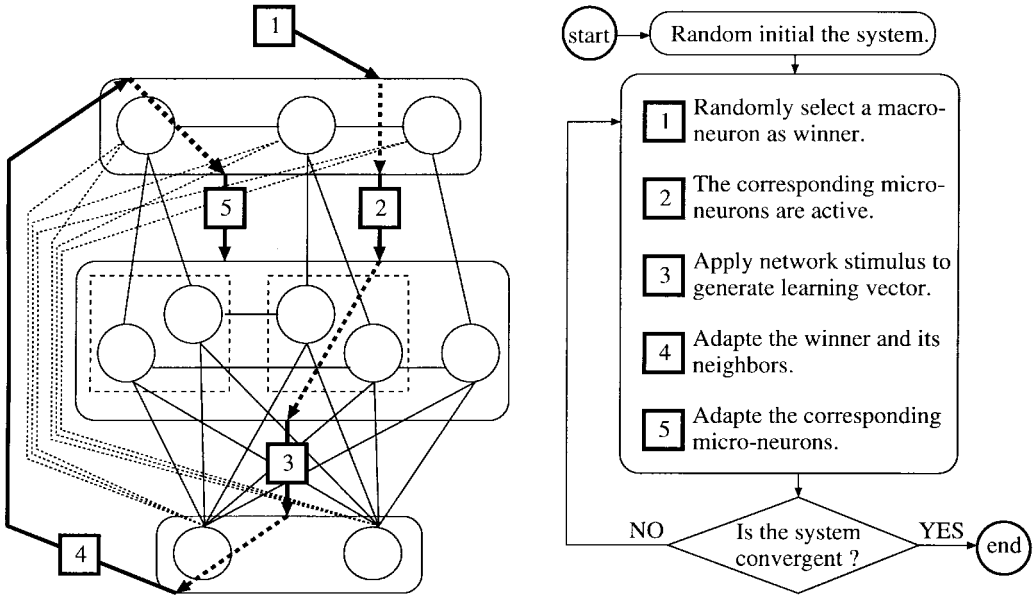


Fig. 7. The system flow of the proposed rectilinear module placement algorithm on a three-layer neural network.

VI. Experimental Results

The proposed algorithm has been implemented in C language on a Sun SPARC-IPC workstation. The current version of the system is running under Sun OS 4.1.1. Results on many test examples are given to illustrate the performance and feasibility of the proposed algorithm. In this paper, test data were grouped into two sets. The first two test examples, TEST1 and TEST2, are taken from Ding *et al.* [33]. There are 42 modules and 178 wire nets in test example TEST1, and 50 modules and 211 wire nets in test example TEST2, respectively. We compared the performance of the proposed method with the constructive approach [26], the simulated annealing method [25] and the two-phase hybrid approach [33]. The obtained results show that the proposed algorithm is better than the previous in both the area of layout region and the total wire length. The next two test examples, TEST3 and TEST4 [27], are arbitrarily-sized rectangular module placement cases with 12 cells/140 wires and 28 cells/236 wires, respectively. Their running results are given and compared with BITOPT [27]. The next two test examples are TEST5 and TEST6 with 70 rectilinear cells/335 wires and 100 rectilinear cells/450 wires, respectively. The large test examples TEST7 with 225 rectilinear cells/50 676 wires and TEST8 with 1024 rectilinear cells/1 569 294 wires are also tested. The last two test examples are called TEST9 and TEST10 with 20 more complex rectilinear cells. Table I summarizes the statistics of the test examples; the notations cell, net and type are the number of modules, wire nets, and the shape of modules respectively. In this section, we first investigate the efficiency of the proposed molecule modeling method by testing with the TEST1 example. The relations between the obtained layout area/wire length and the applied gain term g_r [as shown in (5)] are also presented. Then, the solution quality of the proposed algorithm is studied by testing with the TEST2 example. The test results of large size test examples TEST5, TEST6, TEST7, and TEST8 are proposed. Two more complex test examples TEST9 and TEST10 are also tested. This placement algorithm is also examined taking some other rectangular examples of different characteristics, e.g., TEST4 and TEST3. Experimental results show that the total wire lengths obtained are better than the previous.

TABLE I
The Statistics of the Presented Test Examples; the Notations Cell, Net and Type Are the Respective Number of Modules, Wire Nets, and the Shape of Modules

Test Example	*# of circuit cells*	*# of wire nets*	*circuit shape type*
TEST1	42	178	Rectilinear
TEST2	50	211	Rectilinear
TEST3	12	140	Rectangular
TEST4	28	236	Rectangular
TEST5	70	335	Rectilinear
TEST6	100	450	Rectilinear
TEST7	225	50676	Rectangular
TEST8	1024	1569294	Rectangular
TEST9	20	50	Rectilinear
TEST10	20	50	Rectilinear

A. Solution Efficiency for the Molecule Modeling Method

In order to demonstrate the solution efficiency for the proposed molecule modeling method, Fig. 8 shows four snapshots from a running trace of the placement of the test example TEST1. The initial positions, as shown in Fig. 8(a), are random numbers within the given placement region. Apply the competitive rule of the self-organizing neural network, modules are finally spread out over the whole region which minimizes the total wire length without module overlap. Furthermore, due to the collective computing of hidden-neurons, we can see that most of the dead-spaces among rectilinear modules are removed as shown in Fig. 8(d). The

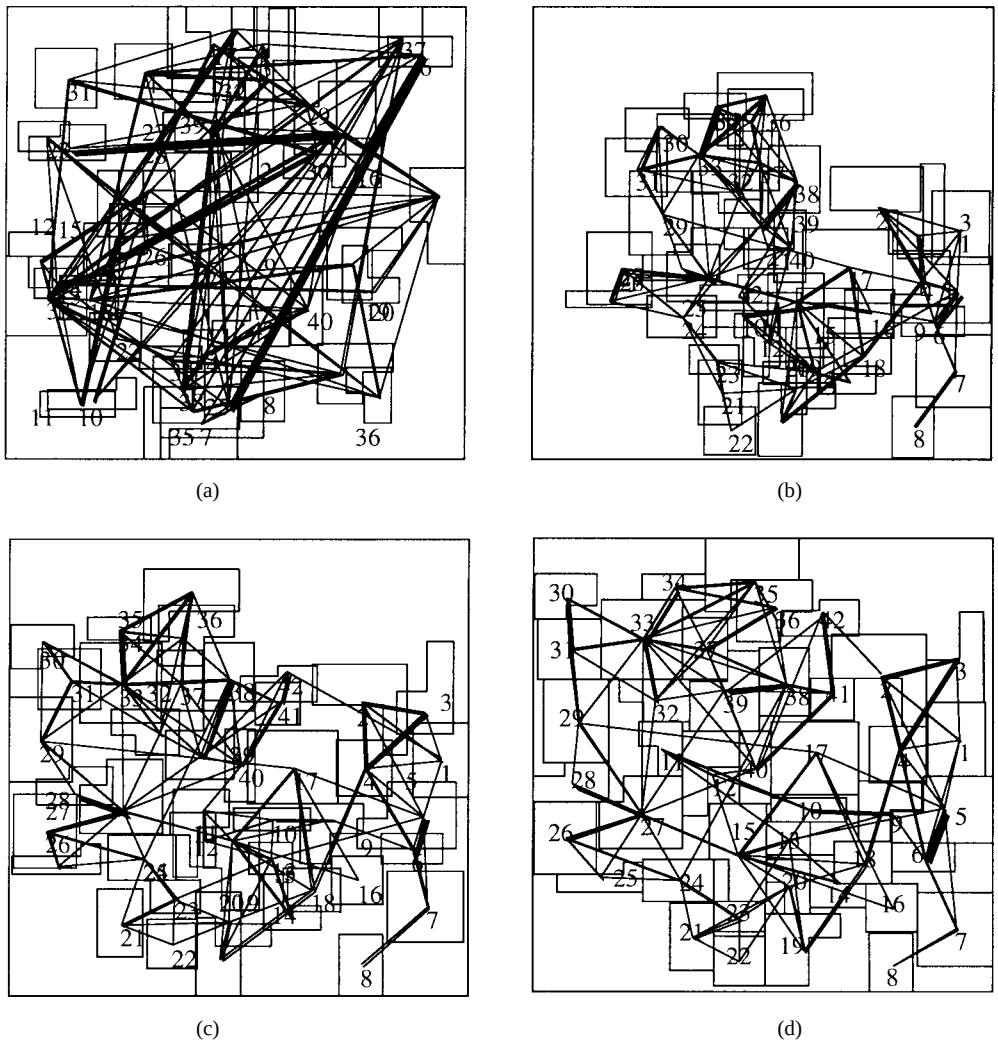


Fig. 8. Four snapshots from a running trace of the test example TEST1 are shown. (a) The initial positions are raandom numbers within the placement region. (b) (c) With the competitive rule, modules are finally spread out over the whole region. (d) Due to the collective computing, most of the dead spaces among the rectilnear modules are removed.

best total wire length obtained by the proposed algorithm from 100 placement trials is 9998. We do get a better result than that achieved by the simulated annealing approach [25] and the heuristic constructive approach [26], where they used 14 800 and 16129, respectively. During the placement of TEST1, the change of total wire length and module overlap versus the first 200 iteration number are shown in Fig. 9. Note that, throughout this paper, the measurement of estimated module overlap are taken from (2). As the definition of the dynamic objective function, the placement with some module overlap is permitted at the early stage to minimize the total wire length. It is seen that the variation of total wire length is initially high and is gradually degraded. As the process continues, we cautiously remove the module overlap such that the total wire length is kept as low as possible. Finally, the total wire length converges to some constant value and the module overlap reduces to zero. Our experiments also demonstrate that the number of module overlap decreases rapidly while the total wire length increases slowly to toward a good placement result. Thus, the placement process is related to the changes of $g(t)$. In the next section, we have observed the obtained layout area and wire length for different dynamic objective functions to obtain a better definition of function $g(t)$.

B. Layout Area/Wire Length with Different Decreasing Ratios

As described above, it can be found that all these processes of dynamic objective are controlled by the decreasing gain function$g'(t) = 1 - g(t)$. The large variation of total wire length at the early stage is due to the large gain term which illustrates the importance of total wire length minimization. Then, the changes are reduced because the magnitude of this gain term is decreased. For simplification, assume that $g'(t+1) = g'(t) \times g_r$, and $0 \leq g_r \leq 1$. The gain term g_r is called a decreasing ratio of function g′(.). Fig. 10 shows the impact of the decreasing ratio g_r on the obtained total wire length and module overlap. Although the total wire length decreases along with the gain term g_r, the module overlap increases very fast if the value of g_r is over 0.99. The results also show that a decreasing ratio g_r of 0.98 to 0.99 gives a good compromise. We adopted a decreasing ratio of 0.98 throughout the following experiments. Note that the decreasing

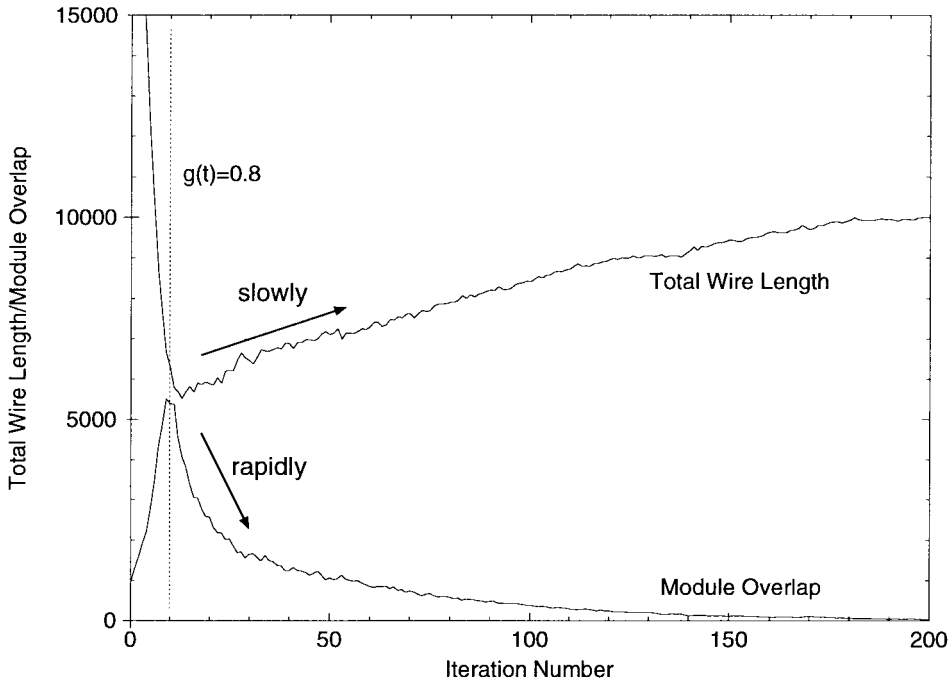


Fig. 9. The changes of total wire length and module overlap versus the first 200 iteration number are shown. At the early stage, the placement with some module overlap is permitted to minimize the total wire length. As the process continues, we cautiously remove the module overlap such that the total wire length is kept as low as possible.

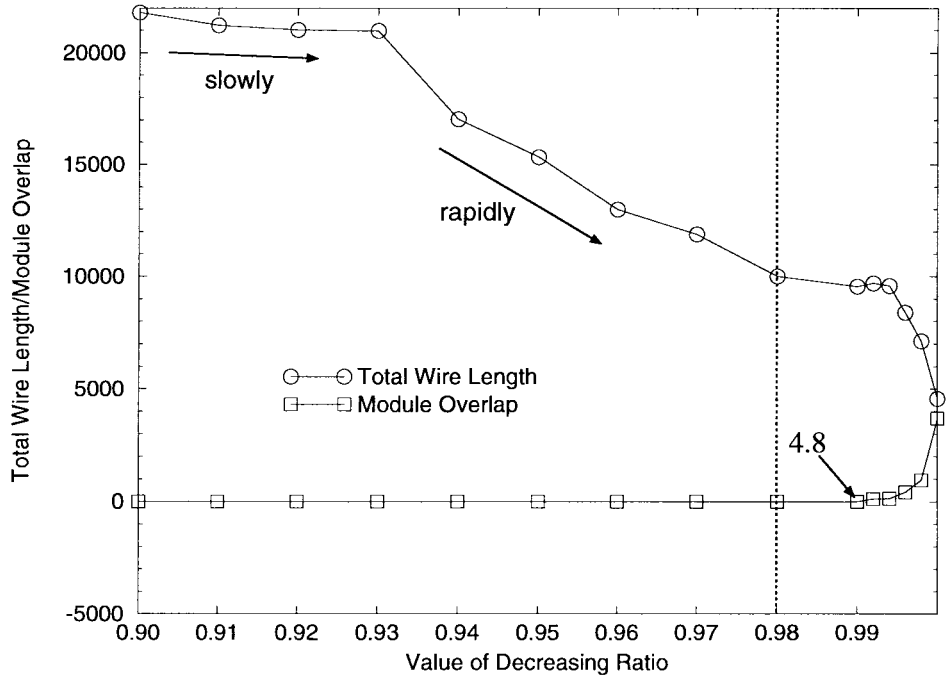


Fig. 10. The impact of the decreasing ration on the obtained total wire length and the number of module overlap is presented. Although the total wire length decreases along with the decreasing ratio, the module overlap is not free if the decreasing ration is over 0.99. Note that the decreasing of the total wire length is slow if the decreasing ration is small. We select the decreasing ration as 0.98 throughout this paper.

of total wire length is slow if g_r is smaller than 0.93 as shown in the graph. The reason is that $g'(t)$ is usually small if g_r is small. Then, the major objective of the whole process is to minimize the module overlap. In this case, the obtained total wire length depends on its initial configuration and have no distinct reduction. Fig. 11 shows the impact of the layout area on the obtained total wire length and module overlap. Account the total module size of test example TEST1 is 55 070. If the size of the layout region is smaller than the total module size, the module overlap is large and the obtained total wire length is hard to estimate. If the size of layout region is larger than the total module size, the module overlap is nearly zero

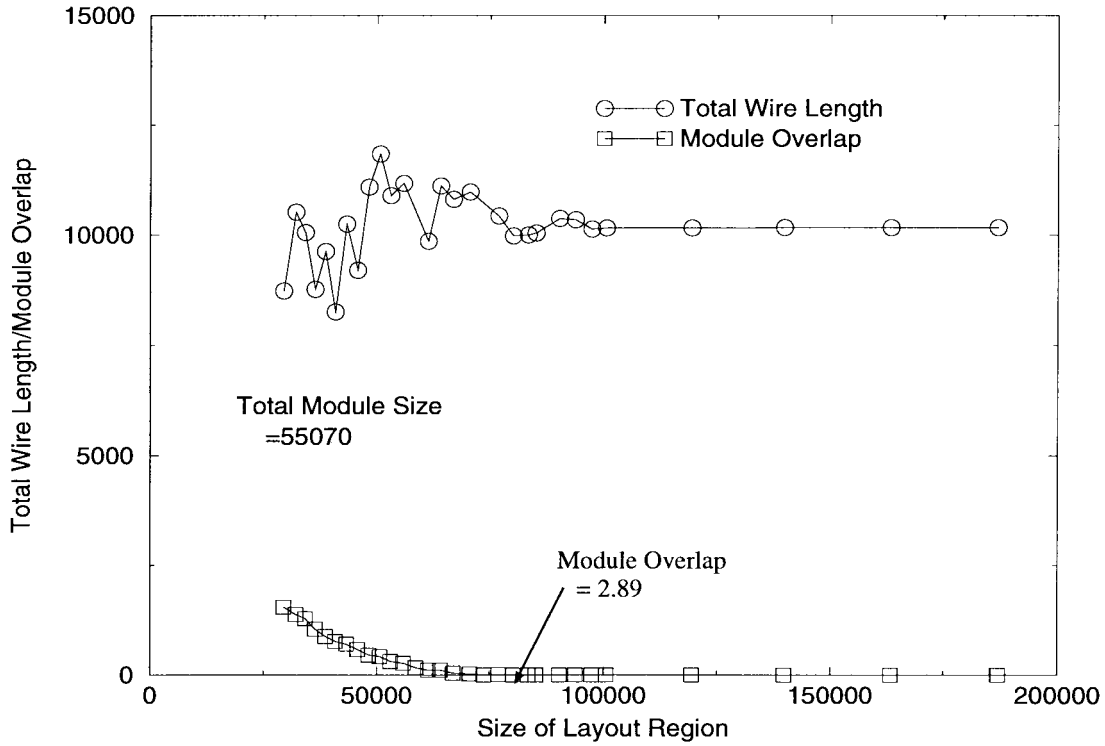


Fig. 11. The impact of the layout area on the obtained total wire length and module overlap is presented. If the size of the layout region is smaller than the total module size, the module overlap is large. If the size of the layout region is larger, the obtained total wire length is nearly a constant. If shown that the proposed method can minimize the area of the layout region.

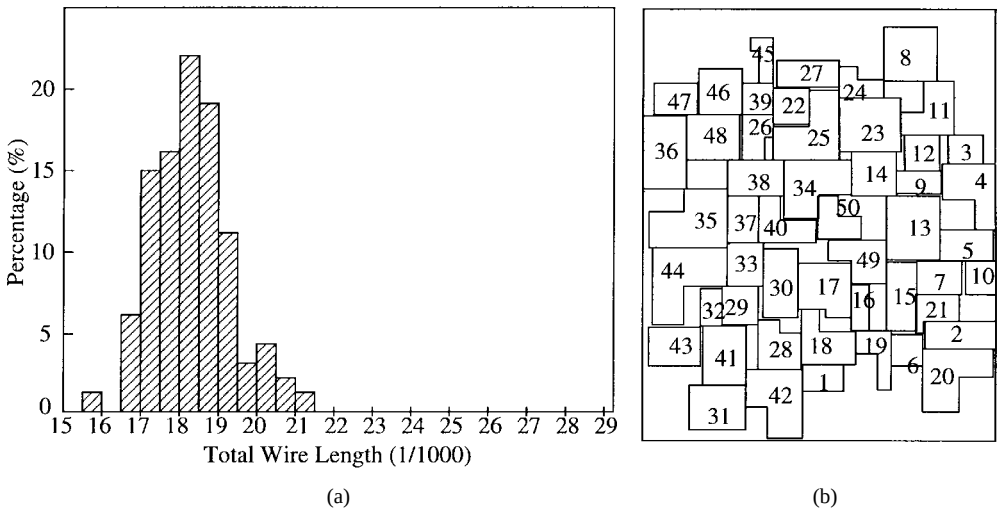
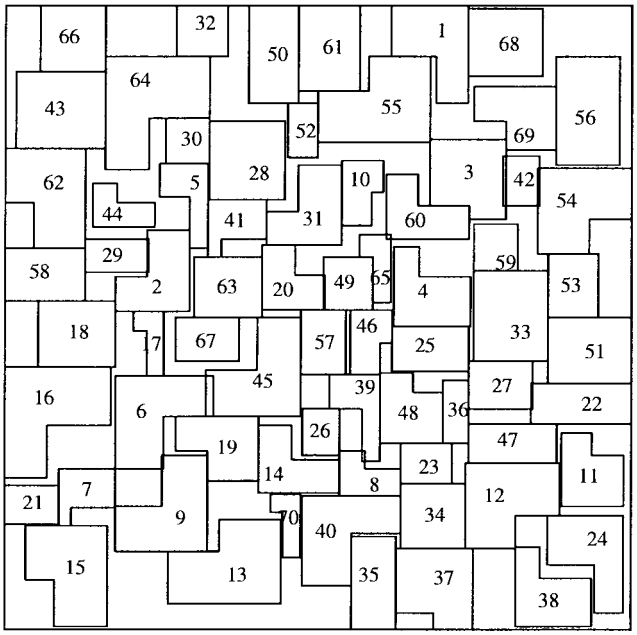


Fig. 12. (a) The graph of percentage of trials versus the total wire length obtained during the 100 placement trials of the test example TEST2. All the trials started from different randomly generated initial configurations. (b) The best solutoin obtained is presented.

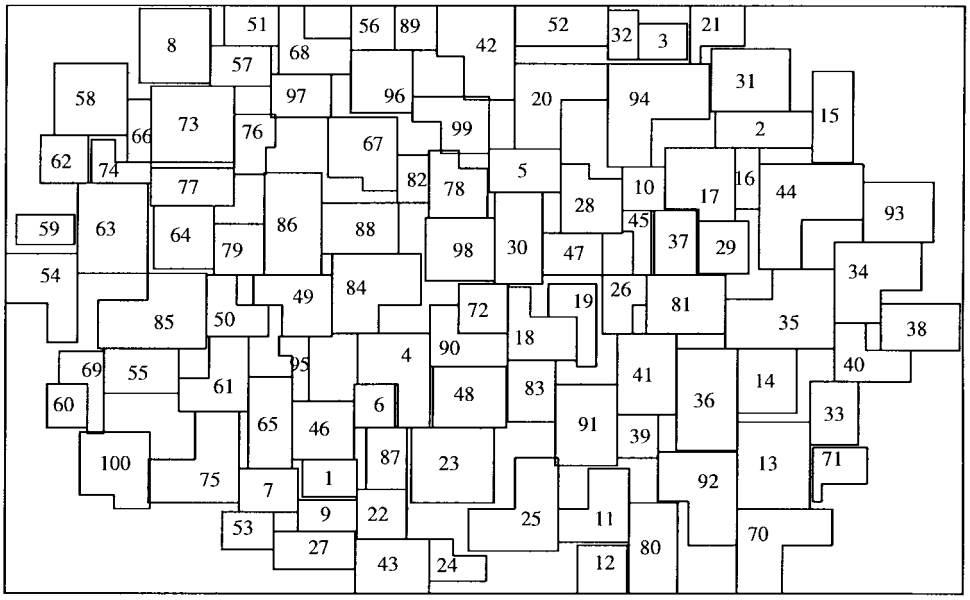
and the total wire length is nearly a constant value in the proposed algorithm. In other words, if the area of layout region is large enough, the total wire length and the module overlap are insensitive to the difference in the layout area. The results show that the layout area within the range 60 000 to 70 000 is acceptable for TEST1. In the paper, we assume the area of layout region is 83 496 (284 $\times$ 294). Our chip area utilization is better than that of the simulated annealing approach and the heuristic constructive approach, where they used 108 999 and 99840, respectively, including lots of dead spaces.

C. Solution Quality of the Proposed Method

The solution quality of the proposed solution model is discussed through the comparison with three rectilinear circuit placement algorithms described in [26] (a constructive approach), [25] (a simulated annealing approach), and [33] (a two-phase hybrid approach). Fig. 12(a) demonstrates the graph of percentage versus the total wire length obtained during the 100 placement trials of test example TEST2. All these trials are started from different randomly generated initial



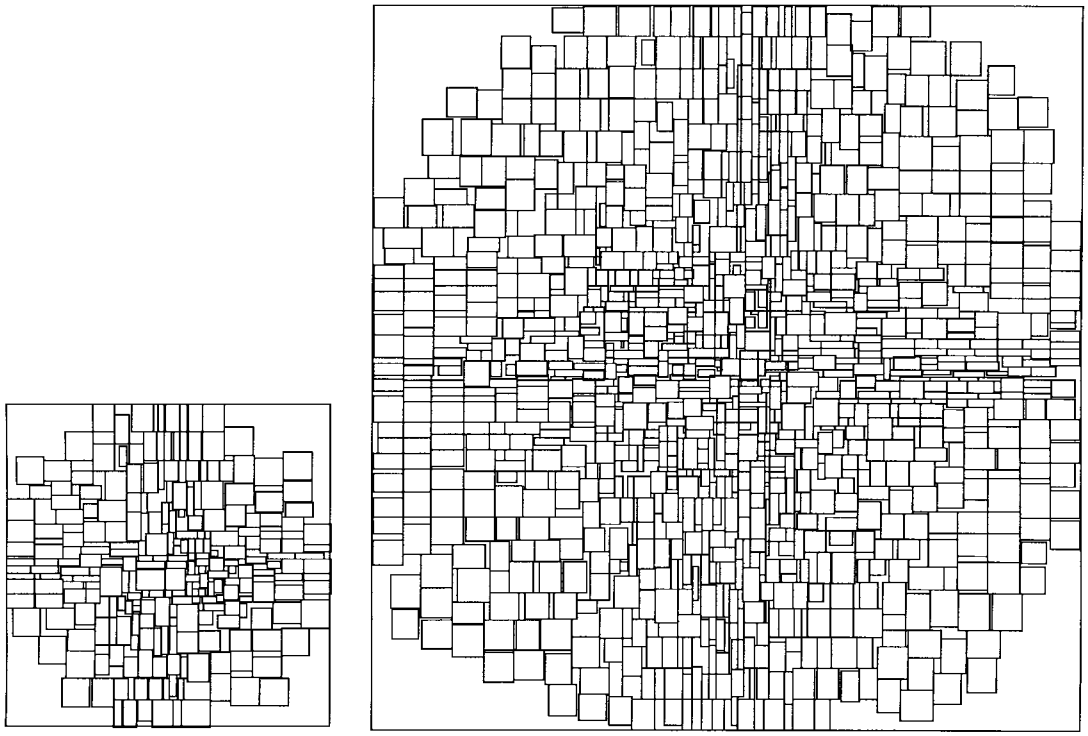
(a)



(b)

Fig. 13. (a) The obtained placement result for TEST4 with total wire length 35239. (b) The obtained placment result for TEST5 with total wire length 53954.

configuration. The average total wire length obtained by the proposed model was 18 324.35. We did get a better result than that achieved by the simulated annealing method [25] and the constructive approach [26], where they used 20 426 and 29875, respectively. It is also better than that obtained by the two-phase hybrid method [33] with 20 053 total wire length. Our obtained solution is shown in Fig. 12(b). These results show that the obtained total wire length is not sensitive to the difference in the initial configuration. In other words, the proposed algorithm successes in yielding a proper placement under different configurations. The time behaviors as shown in Fig. 8 give a possible answer, in which, the effect of start configuration is reduced with the large $g(t)$ in initial time. Table II presents the detail comparison results of the obtained total wire length for test examples TEST1 and TEST2 with three previous rectilinear circuit placement algorithms. As described in [33], in this paper, the size of modules are assumed to be the size of actual modules with the spaces left around the modules for the wire routing. Our experiments show that the proposed method can obtain smaller layout area and fewer total routing length. In these comparisons, the applied simulated annealing method [25] is started from high-temperature and computed with $400m$ iterations (m is the number of circuit cells) to low-temperature. However, as shown in Fig. 9, the number of iterations used for the proposed method is only $200m$. Compared to the simulated



(a) (b)

Fig. 14. (a) The placement result for the large-size problem with 225 cells. (b) The obtained placment result for the large-size problem with 1024 cells.

TABLE II
THE DETAIL COMPARISON RESULTS OF THE OBTAINED TOTAL WIRE LENGTH FOR TEST EXAMPLES TEST1 AND TEST2 WITH THREE PREVIOUS RECTILINEAR CIRCUIT PLACEMENT ALGORITHMS

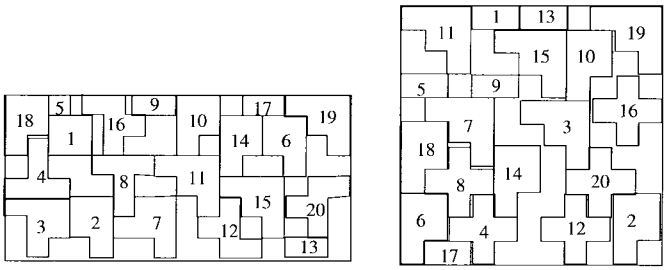
Test Examples		[26]	[25]	[33]	Ours
TEST1	layout area	99840	108999	83496	83496
	wire length	16129	14800	12368	11446*
	CPU time	49.49'	2016.86'	677.97'	153.42'
TEST2	layout area	102898	128426	108224	108224
	wire length	29875	20426	20053	18324*
	CPU time	85.07'	3453.26'	1306.83'	194.70'

* The average total wire length of 100 placement trials.

annealing method, our placement result shows an average 16.5% improvement in total wire length and 93% improvement in CPU time. In Ding *et al.*'s approach [33], the number of iterations for this medium-temperature simulated annealing method is given as $350m$. Experiments show that the proposed method is also better than which uses $350m$ iterations. Its CPU time is smaller than that of [25], however, still larger than that of the proposed method. Although simulated annealing could give a better result if it run for a very long time. However, its computation cost would be very expensive.

D. Other Test Examples

An algorithm that is good for the rectilinear model should be also good for the rectangular model, as the latter is a special case of the former. In this paper, two test circuits, TEST3 and TEST4, with rectangular modules have been tried. The obtained total wire length for test examples TEST3 and TEST4 are 9099 and 7025, respectively. Comparing the placement result, we find that the proposed algorithm achieved shorter total wire length than BITOPT [27]. It obtains 9626 and 9470 for the total wire length of test examples TEST3 and TEST4, respectively. Average wire length reduced by the proposed algorithm over BITOPT is 15%. An algorithm that is good for the rectilinear model should be also good for the rectangular model, as the latter is a special case of the former. An algorithm that is good for the rectilinear model should be also good for the rectangular model, as the latter is a special case of the former. Our experience indicates that our algorithm is competitive with the other approaches mentioned above. The obtained total wire length for large size rectilinear test examples TEST5 and TEST6 are 35 239 and 53954, respectively. Their obtained placement results are shown in Fig. 13(a) and (b). Besides, another two large size problems, TEST7 and TEST8, with more than 1000 circuit cells are also presented to demonstrate the effectiveness of the proposed algorithm. Fig. 14(a) presents the obtained solution for placing TEST7 within a 20×20 layout region. The total wire length obtained was 466 441. The obtained solution for placing TEST8 within an 100×100 layout region is shown in Fig. 14(b) with the total wire length 50 198 793. Our experiments demonstrate that the proposed method can handle large size cell placement problems. These obtained results are quite reasonable, in which, small circuit cells are placed in the center of layout region. The narrow blocks are placed on the x and the y coordinates surrounding with large cell blocks. At last, two complex test examples TEST9 and TEST10 have also been tested to show that the proposed method can handle



(a) (b)

Fig. 15. The placement results for the complex test examples (a) TEST9 and (b) TEST10.

complex rectilinear cell placement problems. We have placed TEST9 within a 16 × 8 layout region. The total wire length obtained was 237.70. The obtained solution result for placing TEST10 within an 11 × 11 layout region has the total wire length 285.21. Their obtained placement results are shown in Fig. 15(a) and (b).

VII. Conclusion

A three-layer neural-network model based on the force-directed self-organizing maps is presented in the context to solve the circuit placement problem with arbitrarily shaped rectilinear modules. The pattern/feature representation method is applied to model the complex rectilinear modules. Furthermore, a new overlap penalty function with collective computation is proposed. In this algorithm, both the relative placement and the overlap-removing process are represented as a single model and solved at the same time. The experimental results are found to better than the previous methods. Note that, in this paper, the placement problem is simplified by assuming that modules can not be rotated during the process. We plan to investigate these possible extensions to our approach.

Acknowledgment

The authors would like to thank the anonymous referees and Y.-L. J. Chiu for their valuable suggestions about this paper.

References

[1] W. E. Donath, "Complexity theory and design automation," in *Proc. 17th ACM/IEEE Design Automa. Conf.*, 1980, pp. 412–419.
[2] K. Shahookar and P. Mazumder, "VLSI cell placement techniques," *ACM Computer Surveys*, vol. 23, pp. 143–220, 1991.
[3] G. Odawara, T. Hamuro, K. Iijima, T. Yoshino, and Y. Dai, "A rule-based placement system for printed wiring boards," in *Proc. 24th Design Automat. Conf.*, 1987, pp. 777–785.
[4] A. Alon and U. Ascher, "Model and solution strategy for placement of rectangular blocks in the Euclidean plane," *IEEE Trans. Computer-Aided Design*, vol. CAD-7, pp. 378–386, 1988.
[5] S. Goto, "An efficient algorithm for the two-dimensional placement problem in electrical circuit layout," *IEEE Trans. Circuit Syst.*, vol. SAC-28, pp. 12–18, Jan. 1981.
[6] R. B. Lin and E. Shragowitz, "Fuzzy logic approach to placement problem," in *Proc. 29th ACM/IEEE Design Automat. Conf.*, 1991, pp. 153–158.
[7] E. H. L. Aarts, F. M. J. de Bont, J. H. M. Korst, and J. M. J. Rongen, "An efficient macro-cell placement algorithm," *INTEGRATION, VLSI J.*, vol. 9, no. 10, pp. 299–317, 1991.
[8] C. M. Kyung, J. Widder, and D. A. Mlynski, "Adaptive cluster growth: A new algorithm for circuit placement in rectilinear regions," *Computer-Aided Design*, vol. 24, no. 1, pp. 27–35, Jan. 1992.
[9] S. S. Kim and C. M. Kyung, "Circuit placement in arbitrarily shaped regions using the self-organization principle," *IEEE Trans. Computer-Aided Design*, vol. 11, pp. 844–854, 1992.
[10] N. Funabiki and Y. Takefuji, "A parallel algorithm for channel routing problems," *IEEE Trans. Comput.-Aided Design*, vol. 11, pp. 464–474, 1992.
[11] J. S. Yih and P. Mazumder, "A neural-network design for circuit partitioning," *IEEE Trans. Comput.-Aided Design*, vol. 9, pp. 1265–1271, Dec. 1990.
[12] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE Acoust., Speech, Signal Processing Mag.*, Apr. 1987.
[13] T. Kohonen, "The 'neural' phonetic typewriter," *IEEE Comput.*, vol. 10, no. 8, pp. 11–22, 1988.
[14] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biol. Cybern.*, vol. 52, pp. 141–152, 1985.
[15] T. Kohonen, "An introduction to neural computing," *Neural Networks*, vol. 1, no. 1, pp. 11–22, 1988.
[16] G. Persky, "Experiments in cell placement with a simulated neural network," in *Proc. IEEE Int. Wkshp. Placement and Routing*, 1987, pp. 10–13.
[17] M. L. Yu, "A study of the applicability of Hopfield decision neural nets to VLSI CAD," in *Proc. 26th Design Automat. Conf.*, 1989, pp. 412–417.
[18] H. Kita, H. Odani, and Y. Nishikawa, "Solving a placement problem by means of an analog neural network," *IEEE Trans. Ind. Electron.*, vol. 39, pp. 543–551, Dec. 1992.
[19] J. Naft, "Neuropt: Neurocomputing for multiobjective design optimization for printed circuit board component placement," in *Proc. Int. J. Conf. Neural Networks*, 1989, pp. I/321–325.
[20] M. Sriram and S. M. Kang, "A modified Hopfield network for two-dimensional module placement," in *Proc. IEEE Int. Symp. Circuits Syst.*, 1990, pp. 1664–1667.
[21] H. Date, M. Seki, and T. Hayashi, "LSI module placement methods using neural computational network," in *Proc. IEEE Int. Conf. Neural Networks*, 1990, pp. 831–836.
[22] A. Hemani and A. Postula, "Cell placement by self-organization," *Neural Networks*, vol. 3, no. 3, pp. 377–383, 1990.
[23] C. X. Zhang and D. A. Mlynski, "Neural somatotopical mapping for VLSI placement optimization," in *Proc. Int. Joint Conf. Neural Networks*, 1991, pp. 863–868.
[24] R.-I Chang and P. Y. Hsiao, "Arbitrarily sized cell placement by self-organizing neural networks," in *Proc. IEEE Int. Symp. Circuits Syst.*, 1993, pp. 2043–2046.
[25] C. Sechen, "Chip-planning, placement, and global routing of macro/custom cell integrated circuits using simulated annealing," in *Proc. 25th Design Automat. Conf.*, 1988, pp. 73–80.
[26] S. Wimer and I. Koren, "Analysis of strategies for constructive general block placement," *IEEE Trans. Computer-Aided Design*, vol. 7, pp. 371–377, 1988.
[27] M. Mir and M. H. Iman, "Topology optimization of arbitrarily-sized blocks using a bivariate formulation," *Computer-Aided Design*, vol. 24, no. 10, pp. 556–564, 1992.
[28] W. Swartz and C. Sechen, "New algorithms for the placement and routing of macro cells," in *Proc. IEEE Int. Conf. Comput.-Aided Design*, 1990, pp. 336–339.
[29] A. Herrigel and W. Fichtner, "An analytic optimization technique for placement of macro-cells," in *Proc. 26th Design Automat. Conf.*, 1989, pp. 376–381.
[30] T. C. Lee, "A bounded 2-D contour searching algorithm for floorplan design with arbitrarily shaped rectilinear and soft modules," in *Proc. 30th ACM/IEEE Design Automat. Conf.*, 1993, pp. 525–530.
[31] W. T. Liou, J. J. M. Tan, and R. C. T. Lee, "Minimum rectangular partition problem for simple rectilinear polygons," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 720–733, 1990.
[32] E. Rich and K. Knight, *Artificial Intelligence*, 2nd ed. New York: McGraw-Hill, 1991.
[33] R. H. Ding, W. S. Feng and S. J. Chen, "Design and analysis of rectangular and L-shaped cell placement," *J. Chin. Eng. Inst.*, 1994.
[34] B. Preas and K. Roberts, in *Physical Design Wkshp.*, Hilton Head, SC, 1987.
[35] T. Kohonen, *Self-Organization and Associative Memory*, 3rd ed. New York: Springer-Verlag, 1989.
[36] M. Mir and M. H. Imam, "A gradent-based method for module placement," *Comput. Electron. Eng.*, vol. 16, pp. 109–113, 1990.
[37] R.-I Chang and P.-Y. Hsiao, "Unsupervised query-based learning of neural networks using selective-attention and self-regulation," *IEEE Trans. Neural Networks*, vol. 8, pp. 205–217, Mar. 1997.
[38] T. Kohonen, "The self-organizing map," *Proc. IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[39] R.-I Chang and P.-Y. Hsiao, "Self-organizing fuzzy techniques for macro-cell placement," in *Proc. 4th VLSI Design/CAD Wkshp.*, 1993, pp. 44–47.
[40] C. C. Tsai, S. J. Chen, P.-Y. Hsiao, and W. S. Feng, "New iterative construction approach to routing with compacted area," *Proc. Inst. Elec. Eng. pt. E*, vol. 138, no. 1, pp. 57–72, 1991.
[41] N. R. Quinn and M. A. Breuer, "A force-directed component placement procedure for printed circuit boards," *IEEE Trans. Circuits Syst.*, vol. CAS-26, pp. 377–388, June 1979.
[42] J. N. Hwang, J. J. Choi, S. Oh, and R. J. Marks, II, "Query-based learning applied to partially trained multilayer perceptrons," *IEEE Trans. Neural Networks*, vol. 2, pp. 131–136, Jan. 1991.
[43] R.-I Chang, K. Y. Huang, and H. T. Yen, "Self-organizing neural network for picking of seismic horizons," in *Proc. Symp. Telecommun.*, 1990, pp. 431–434.
[44] U. Lauther, "A min-cut placement algorithm for general cell assemblies based on a graph representation," in *Proc. 16th Design Automat. Conf.*, 1979, pp. 1–10.

Ray-I Chang was born on February 7, 1967 in Taichung, Taiwan, R.O.C. He received the B.S. degree and the Ph.D. degree in computer and information science from National Chiao Tung University, Hsinchu, Taiwan, R.O.C., in 1989 and 1996, respectively.

Since 1996, he had been a Postdoctoral Research Fellow in the Institute of Information Science, Academa Sinica, Taiwan, R.O.C. He has published more than 20 scientific papers in neural networks, fuzzy logic, genetic algorithms, image processing, and VLSI/CAD. His interests include real-time scheduling and distributed multimedia systems.

Pei-Yung Hsiao received the B.S. degree in chemical engineering from Tunghai University, Taichung, Taiwan, R.O.C., and the M.S. and Ph.D. degrees, both in electrical engineering, from National Taiwan University in 1987 and 1990, respectively.

Since 1990 he has been an Associate Professor in the Department of Computer and Information Science at National Chiao Tung University, Hsinchu, Taiwan, R.O.C. Since 1991, he has also been a Technique Consultant in the Piping Engineering Department, CTCI Corp., Taipei, Taiwan, R.O.C. His interests include VLSI/CAD, piping engineering design automation, expert system applications, and database systems.