INT. J. ELECTRONICS, 1992, VOL. 72, NO. 3, 331–355

A generalized maximum neural network for the module orientation problem

KUO CHUN LEE† and YOSHIYASU TAKEFUJI†

Several neuron models and artificial neural networks have been intensively studied since McCulloch and Pitts proposed the simplified neuron model in 1943. In this paper a generalized maximum neural network for parallel computing is introduced to solve the module orientation problem which belongs to the class of NP-complete problems. The goal of the module orientation problem in VLSI circuits or printed circuit boards is to minimize the total wire length by flipping each module with respect to its vertical and/or horizontal axes of symmetry. The circuit diagram of the generalized maximum neural network is shown and compared with the best known algorithm proposed by Libeskind-Hadas and Liu. The theoretical/empirical convergence analysis is discussed where a massive number of simulation runs were performed using more than one thousand instances. As far as we have observed the behavior of the proposed system, it converges within O(1) time regardless of the problem size and it performs better than the best known algorithm in terms of the solution quality and the computation time.

1. Introduction

Different methodologies have been studied for optimization problems which are very essential in computational research. Recent progress in software and VLSI technologies has motivated us to fully exploit parallelism of the artificial neural network for solving this hard problem. The study of the artificial neural network system began in the early 1940s. McCulloch and Pitts (1943) showed that a neuron can be modelled as a simple threshold device which performs a binary logic function. Later the neuron model was refined in Rosenblatt's Perceptron (Rosenblatt 1958) and Widrow and Hoff's Adaline (Widrow and Hoff 1960). During the 1970s, Carpenter and Grossberg (1987) proposed a neural network called Adaptive Resonance Theory (ART) based on the assumption that the brain spontaneously organizes itself into recognition codes. At the same time, Kohonen (1984) developed a self-organizing process by which an optimal topographical mapping of the signal is constructed. Recently, Rumelhart and McClelland (1986) proposed the back-propagation algorithm based on the feedforward multi-layered neural network. Hopfield and Tank (1985) used an artificial neural network to solve combinatorial optimization problems such as the travelling salesman problem (TSP).

The module orientation problem has been studied by Yamada and Liu (1988) and a similar problem has been investigated by Yao, Yamada, and Liu (1989). Libeskind-Hadas and Liu (1989) used the Hopfield neural network to solve the module orientation problem. In the Libeskind-Hadas and Liu's Hopfield neural network, the gradient descent method seeks the local minimum of the given Liapunov energy function E. Generally, the Liapunov energy function E is given by the constraints and the cost function. The synaptic weights between neurons are

Received 26 June 1991; accepted 1 August 1991.

†Department of Electrical Engineering and Applied Physics, Case Western Reserve University, Cleveland, OH 44106, U.S.A.

determined by the energy function E. In general, the state of the system in the Hopfield (1984) neural network is guaranteed to converge to one of the local minima instead of the global minimum. However, the local minima do not always represent feasible solutions. In other words, the proof of the local minimum convergence in the Hopfield neural network does not always guarantee the feasible solution although its fast convergence speed is very attractive. In this paper a generalized maximum neural network is proposed. The generalized maximum neural network not only enables the local minima to always be feasible solutions but also provides the fast local minimum convergence.

The NP-complete (Nondeterministic Polynomial) problems are classified as hard problems. It is believed that no polynomial time algorithm exists for NP-complete problems (Garey and Johnson 1979). Therefore, the near-optimum solution is practically acceptable as long as it can be obtained in a reasonable computation time. Most of the existing methods of solving NP-complete problem are based on heuristics or relaxation of the constraints. In this paper a near-optimum parallel algorithm based on the generalized maximum neural network is proposed for the module orientation problem which was proven to be NP-complete (Libeskind-Hadas and Liu 1989). A massive number of simulation runs were performed to investigate the capability of the proposed system.

2. Models of the processing element

An artificial neutral network consists of a massive number of simple processing elements that are highly interconnected. The processing element is called an artificial neuron, because it performs the simplified function of a biological neuron. The mathematical model of the artificial neural network is composed of two important components: neurons and synaptic links. The output signal generated by one neuron propagates to the others through the synaptic links. The new state of the neuron is determined by the linear sum of the weighted input signals. McCulloch and Pitts (1943) proposed a mathematical neuron model based on biological computation. The input/output function is given by $V_i = f(U_i) = 1$ if $U_i \geqslant 0$ and 0 otherwise where V_i and U_i are the output and the input of the ith neuron respectively. An artificial neural network for solving combinatorial optimization problems was first introduced by Hopfield and Tank (1985). The differentiable, continuous and non-decreasing neuron model, sigmoid function, was used to model the relation between the output V_i and the input U_i for the ith neuron given by

$$V_i = g(U_i) = \tfrac{1}{2}[1 + \tanh(\lambda U_i)] \qquad (1)$$

The constant λ is known as gain and determines the slope of the sigmoid function. The different slopes of the sigmoid function are shown in Fig. 1 In their paper (Hopfield and Tank 1985), the symmetric conductance matrix T with zero diagonal elements was used to guarantee the local minimum convergence. Hopfield and Tank mapped the TSP onto an $N \times N$ artificial neural network. The computational energy function E for an N-city problem is given by

$$E = \frac{A}{2}\sum_{x=1}^{N}\sum_{i=1}^{N}\sum_{j\neq i}^{N} V_{x,i}V_{x,j} + \frac{B}{2}\sum_{i=1}^{N}\sum_{x=1}^{N}\sum_{x\neq y}^{N} V_{x,i}V_{y,i} + \frac{C}{2}\left(\left(\sum_{i=1}^{N}\sum_{x=1}^{N} V_{x,i} - N\right)\right)^2$$

$$+ \frac{D}{2}\sum_{x=1}^{N}\sum_{y\neq x}^{N}\sum_{i}^{N} d_{x,y}V_{x,i}(V_{y,i+1} + V_{y,i-1}) \qquad (2)$$

where $d_{x,y}$ is the distance between city x and city y and A, B, C, and D are coefficient parameters.

The conductance matrix T in the Hopfield neural network is determined by the energy function E and is given by

$$T_{xi,yj} = -A\delta_{x,y}(1-\delta_{i,j}) - B\delta_{ij}(1-\delta_{x,y}) - C - Dd_{x,y}(\delta_{j,i+1} + \delta_{j,i-1}) \tag{3}$$

The external input bias current for each neuron is given by

$$I_{x,i} = CN \tag{4}$$

The equation of motion of the neuron in row x and column i is given by

$$\frac{\mathrm{d}U_{x,i}}{\mathrm{d}t} = -\frac{U_{x,i}}{\tau} - \frac{\partial E}{\partial V_{x,i}} \tag{5}$$

It has been shown that the decay term $-U_{xi}/\tau$ has an undesirable effect for the convergence of the system (Takefuji and Lee 1991), because it increases the energy function E in (2) under some conditions. In other words, E is not the Liapunov energy function of the system. Many researchers use a large value for τ in order to eliminate this undesirable effect. It is suggested that the decay term should be removed from the motion equation based on both theoretical and empirical reasons (Takefuji and Lee 1991). The McCulloch–Pitts neuron model and the modified McCulloch–Pitts neuron model have been successfully used (without the decay term in the motion equation) for graph planarization problems (Takefuji and Lee 1989), tiling problems Takefuji and Lee (1990*a*), RNA secondary structure prediction problems (Takefuji, Lin and Lee 1990, Takefuji, Chen, Lee and Huffman 1990), maximum independent set problems (Takefuji *et al.* 1990), sorting problems (Takefuji and Lee 1990*b*), *k*-colourability problems (Takefuji and Lee 1991), and spare allocation problems (Funabiki and Takefuji 1991).

In this paper a generalized maximum neural network is introduced for the module orientation problem. The generalized maximum neural network consists of M clusters where each cluster is composed of n processing elements. The total

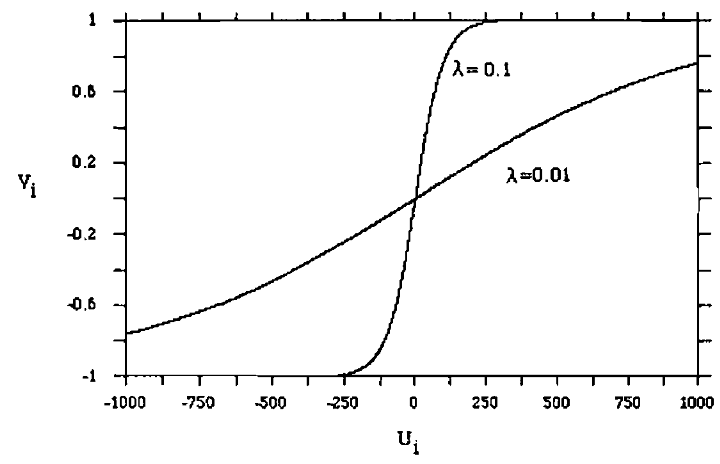


Figure 1. Sigmoid function for $\lambda=0{\cdot}1$ and $\lambda=0{\cdot}01$.

number of required processing elements is $M \times n$. One and only one processing element among n processing elements per cluster is always encouraged to fire in the generalized maximum neural network. The input/output function of the ith generalized maximum neutron in the mth cluster is given by: $V_{m,i}=1$ if $f(U_{m,i}) = \max\{f(U_{m,1}), \ldots, f(U_{m,n})\}$, 0 otherwise where f is a non-decreasing function. If the function $f(U_{m,i})$ is equal to $U_{m,i}$, the network is called a maximum neural network where it is a subset of the generalized maximum neural network. The maximum neural network model has been successfully used to solve the tiling problem (Takefuji and Lee 1990*a*), the max cut problem (Lee and Takefuji 1991), the bipartite subgraph problem (Lee *et al.* 1991), and the graph partition problem (Lee *et al.* 1991).

3. The module orientation problem

For the design of very large scale integrated circuits, it is important to place properly the individual modules while minimizing area and/or the total wire length. The module orientation problem has been studied by Yamada and Liu (1988) and a similar problem has been investigated by Yao *et al.* (1989). It is assumed that all the modules have already been placed according to some placement algorithm (Hanan *et al.* 1976, Preas and Karger 1988). The second assumption is that the pin positions on each module are fixed. The goal of the problem is to minimize the total wire length by flipping each module with respect to its vertical and/or horizontal axes of symmetry as shown in Fig. 2. This means that there are only four possible orientations per module. A similar problem, the module rotation problem, was discussed by Libeskind-Hadas and Liu (1988) who proved that the module orientation problem and the module rotation problem are both NP-complete.

For simplicity, the Euclidean metric is used to define the distance between two pins that are to be connected. Let p, q be two pins which belong to the same net N. Let (x_p, y_p), (x_q, y_q) represent the positions of p and q according to a certain orientation of the modules. The goal is to minimize the total wire length L where the summation is carried out over all pairs of pins which belong to the same net. The total wire length L in M modules is given by

$$L = \sum_{p,q \in N} [(x_p - x_q)^2 + (y_p - y_q)^2]^{1/2} = \frac{1}{2} \sum_{m}^{M} \sum_{m' \neq m}^{M} \sum_{i}^{4} \sum_{j}^{4} d_{mi,m'j} V_{m,i} V_{m'j} \tag{6}$$

where $d_{mi,m'j}$ denotes the total length of all wires between the mth module in the ith orientation and the m'th module in the jth orientation. Note that $V_{m,i}=1$ if the mth module is in the ith orientation, and 0 otherwise.

The problem is to determine the optimal orientation for each module in order to minimize L. Libeskind-Hadas and Liu mapped the problem onto the Hopfield

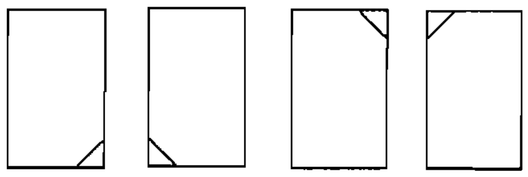


Figure 2. Four orientations for one module.

neural network using a $4 \times M$ neural array where M denotes the number of modules and each module has 4 possible orientations. The mapping procedure is similar to that of TSP proposed by Hopfield and Tank (1985) where an $N \times N$ neural array is employed for an N-city problem. For the module orientation problem, Libeskind-Hadas and Liu give the following energy function:

$$E=\frac{A}{2}\sum_{m=1}^{M}\sum_{i=1}^{4}\sum_{j\neq i}^{4}V_{m,i}V_{m,j}+\frac{B}{2}\left(\left(\sum_{m=1}^{M}\sum_{i=1}^{4}V_{m,i}-N\right)\right)^2$$

$$+\frac{C}{2}\sum_{m=1}^{M}\sum_{m'\neq m}^{M}\sum_{i=1}^{4}\sum_{j=1}^{4}d_{mi,m'j}V_{m,i}V_{m'j} \tag{7}$$

where $d_{mi,m'j}$ denotes the total length of all wires between the mth module in the ith orientation and the m'th module in the jth orientation. The feasible solution for the module orientation problem must have one and only one neuron fired per module.

The disadvantages of the computational energy E in (7) can be summarized as follows.

(1) Their proposed system requires more than one hundred iteration steps to converge for small size problems. It requires several thousand iteration steps for large size problems. The detection of the system convergence is ambiguous.

(2) The sigmoid neuron was used instead of the McCulloch–Pitts neuron. The output function of sigmoid is always smaller than 1 unless a very high gain is used. Therefore, it is very difficult to determine whether the neuron is fired or not. The first two terms in (7) will not be satisfied even if the system reaches one of the equilibrium states or the local minimum.

(3) If a very high gain λ is used for the sigmoid neuron, the energy function E and the sigmoid neuron model cannot guarantee that a valid configuration is reached although the system converges to the local minimum. In other words, the state of the local minimum is not always equivalent to a feasible solution.

(4) The second term in the (7) increases the complexity of connections in the network.

(5) There is no systematic way to tune the parameters in the computational energy function of (7) which has three coefficients A, B, and C. The scaling of the parameters will heavily influence the performance of the system and the solution quality. Analysis of the sensitivity of the parameters is questionable and the result is often inconclusive.

Remember that there are many candidates for the neural representation and the computation energy E. The first two terms in (7) can be replaced by the local constraint such that one and only one orientation is allowed for each module. The modified energy function E is given by

$$E=\frac{A}{2}\sum_{m=1}^{M}\left(\sum_{i=1}^{4}V_{m,i}-1\right)^2+\frac{C}{2}\sum_{m}^{M}\sum_{m'\neq m}^{M}\sum_{i=1}^{4}\sum_{j=1}^{4}d_{mi,m'j}V_{m,i}V_{m'j} \tag{8}$$

If the McCulloch-Pitts neuron is used, some of the disadvantages can be eliminated in (8). However, the serious problem for tuning parameters A and C still remains and the McCulloch-Pitts neuron model cannot always guarantee a valid configuration whenever the state of the system reaches a local minimum. In the next §, a generalized maximum neural network is introduced where it not only eliminates all of those disadvantages and difficulties but also provides better solutions than the best known algorithms.

4. Neural representaton using the generalized maximum neural network

The Hopfield neural network is based on the Liapunov energy function E. The energy function E usually contains several terms which describe the validity of the solution and the solution quality. For example, there are four terms in (2) for TSP and three terms in (7) for the module orientation problem. For TSP and the module orientation problem, the goal is to minimize the total distance or length. Remember that the last term only describes the total distance or length. For TSP, the percentage of the valid tour is very low even for the small size problem. Furthermore, the determination of the parameters in the computational energy function R is usually based on trial and error. Trial and error approaches are often criticized when the problem size increases.

The generalized maximum neural network can be used for solving the module orientation problem. The generalized maximum neuron function is given by: $V_{m,i}=1$ if $f(U_{m,i})$ is the maximum value among $f(U_{m,1})$,$f(U_{m,2})$,$f(U_{m,3})$ and $f(U_{m,4})$, 0 otherwise where f is a non-decreasing function. Due to the external constraint placed upon a collection of neurons, there is always one and only one neuron fired per module. The first term in (8) is always zero. Therefore, (8) is simply reduced to the following form:

$$E=\frac{C}{2}\sum_{m=1}^{M}\sum_{m'\neq m}^{M}\sum_{i=1}^{4}\sum_{j=1}^{4} d_{mi,m'j}V_{m,i}V_{m'j} \tag{9}$$

The equation of motion for the neuron of the mth module in the ith orientation without the decay term is given by

$$\frac{\mathrm{d}U_{m,i}}{\mathrm{d}t}=-C\sum_{m'\neq m}^{M}\sum_{j=1}^{4} d_{mi,m'j}V_{m'j} \tag{10}$$

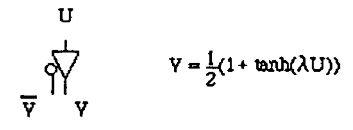
The conductance T is given by

$$T_{mi,m'j}=Cd_{mi,m'j} \tag{11}$$

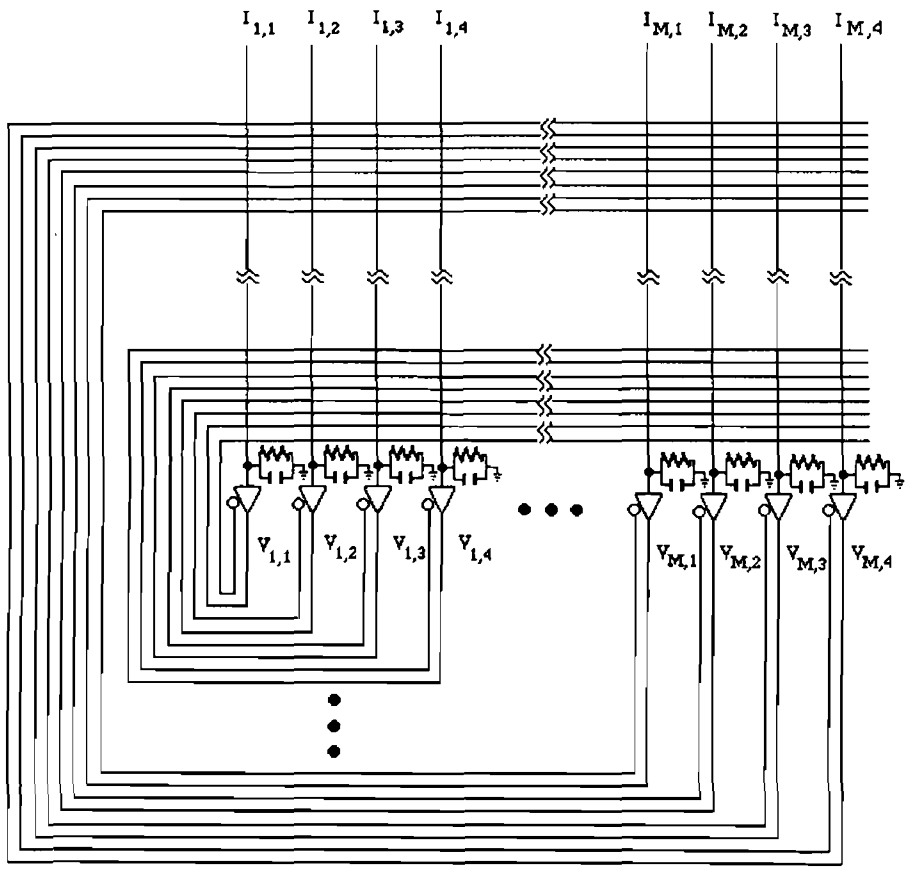
Note that there is no bias in the generalized neural network.

Note that minimization of the energy function E in (9) is exactly equal to minimization of L in (6) for $C=1$. The generalized maximum neural network always generates a valid configuration for the module orientation problem. It is not required to tune the coefficient parameters in the computational energy function E, while the energy function in (2), (7), and (8) must suffer from the presence of parameters that must be tuned. Whenever or wherever the system converges, the corresponding configuration is always forced to be a valid solution; none of the existing Hopfield neural networks can guarantee this property.

The circit diagrams for the Hopfield neural network, the generalized maximum neural network, and the maximum neural network are shown in Figs 3 and 4 respectively for the M-module orientation problem. In Fig. 3, the bias for each input is required in the Hopfield neural network. Note that no bias is needed in the generalized maximum neural network. If the bias of each input has to be adjusted individually, it will increase the circuit design cost. In the Hopfield neural network, there are two kinds of outputs (excitatory and inhibitory output) for each processing element. On the other hand, the generalized neural network requires only inhibitory outputs. This property not only decreases the number of required processing



(*a*)



(*b*)

Figure 3. The network architecture: (*a*) processing element with the input U and output V and $\bar{V}$; (*b*) Hopfield neural network for the module orientation problem.

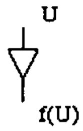
elements but also reduces the number of connections between processing elements from $4M^2$ to M^2 where M is the number of modules. The maximum neural network in Fig. 4(*e*) has the smallest chip size and the least number of interconnections among the three circuits.

5. Convergence of the generalized maximum neural network

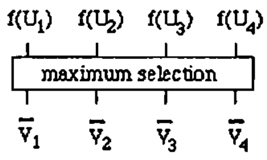
Lemma 1 and Lemma 2 are introduced to prove that the proposed system is always allowed to converge to the equilibrium state or the optimal (near-optimum) solution.

Lemma 1

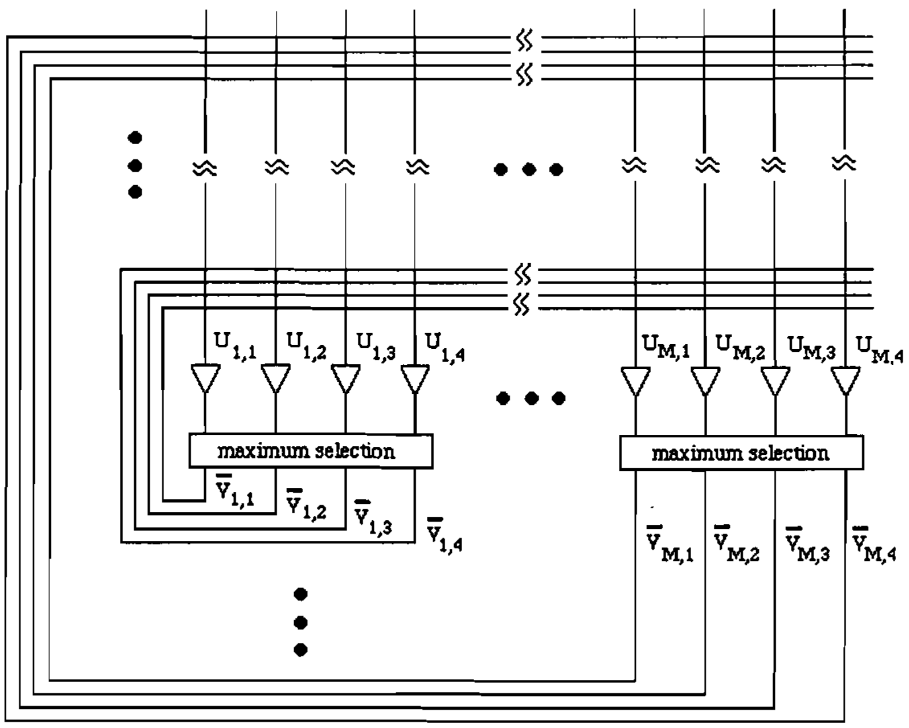
$\frac{dE}{dt} \leq 0$ is satisfied under two conditions such as $\frac{dU_i}{dt} = -\frac{\partial E}{\partial V_i}$ and $V_i = f(U_i)$ where $f(U_i)$ is a non-decreasing function.



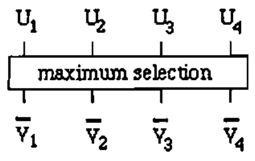
4(*a*)



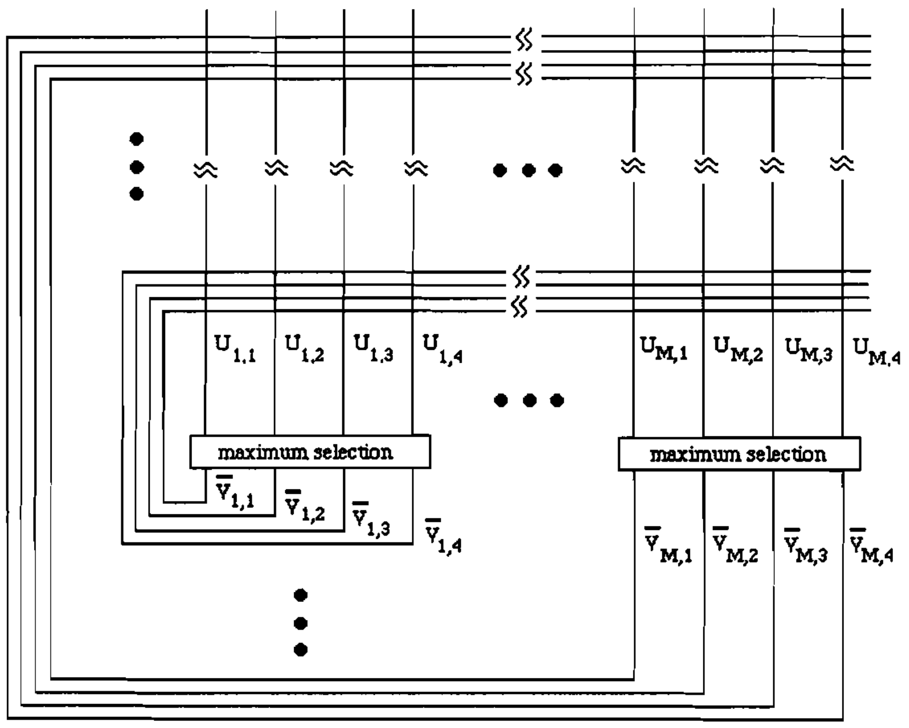
4(*b*)



4(*c*)



(*d*)



$U_{1,1}$ $U_{1,2}$ $U_{1,3}$ $U_{1,4}$
$U_{M,1}$ $U_{M,2}$ $U_{M,3}$ $U_{M,4}$
maximum selection
maximum selection
$\bar{V}_{1,1}$ $\bar{V}_{1,2}$ $\bar{V}_{1,3}$ $\bar{V}_{1,4}$
$\bar{V}_{M,1}$ $\bar{V}_{M,2}$ $\bar{V}_{M,3}$ $\bar{V}_{M,4}$

(*e*)

Figure 4. The generalized maximum neural network architecture: (*a*) processing element; (*b*) maximum selection function block; (*c*) the hardware structure for the module orientation problem; (*d*) maximum selection function block; (*e*) the maximum neural network for the module orientation problem.

Proof

$$\frac{dE}{dt}=\sum_i \frac{dU_i}{dt}\frac{dV_i}{dU_i}\frac{dE}{dV_i}$$

$$=-\sum_i \left(\frac{dU_i}{dt}\right)^2 \frac{dV_i}{dU_i} \quad \text{where } \frac{dE}{dV_i} \text{ is replaced by } \frac{dU_i}{dt} \text{ (condition 1)}$$

$$\leqslant 0 \quad \text{where } \frac{dV_i}{dU_i} > 0 \text{ (condition 2)}$$

Lemma 1 states that the state of the system finally reaches an equilibrium state. The input/output function must follow a nondecreasing function. In Lemma 2, the local minimum convergence of the generalized maximum neural network for the module orientation problem is given.

Lemma 2

$\frac{dE}{dt} \leqslant 0$ is satisfied under two conditions such as

(1) $\frac{dU_{m,i}}{dt} = -\frac{\partial E}{\partial V_{m,i}}$ and

(2) $V_{m,i} = 1$ if $f(U_{m,i}) = \max\{f(U_{m,1}), f(U_{m,2}), f(U_{m,3}), f(U_{m,4})\}$

0 otherwise

where f is a non-decreasing function.

Proof

Consider the derivatives of the computational energy E with respect to time t.

$$\frac{dE}{dt} = \sum_m \sum_i \frac{dU_{m,i}}{dt} \frac{dV_{m,i}}{df(U_{m,i})} \frac{df(U_{m,i})}{dU_{m,i}} \frac{\partial E}{\partial V_{mi}}$$

$$= -\sum_m \sum_i \left(\frac{dU_{m,i}}{dt}\right)^2 \frac{dV_{m,i}}{df(U_{m,i})} \frac{df(U_{m,i})}{dU_{m,i}} \quad \text{where } \frac{\partial E}{\partial V_{m,i}} \text{ is replaced by}$$

$$-\frac{dU_{m,i}}{dt} \text{ (condition 1)}$$

Let $\frac{dV_{m,i}}{df(U_{m,i})}$ be $\frac{V_{m,i}(t+dt) - V_{m,i}(t)}{f[U_{m,i}(t+dt)] - f[U_{m,i}(t)]}$. Let $\frac{df(U_{m,i})}{dU_{m,i}}$ be $\frac{f(U_{m,i}(t+dt)] - f[U_{m,i}(t)]}{U_{m,i}(t+dt) - U_{m,i}(t)}$. Let $\frac{dU_{m,i}}{dt}$ be $\frac{U_{m,i}(t+dt) - U_{m,i}(t)}{dt}$. Let us consider the term $\sum_i \left(\frac{dU_{m,i}}{dt}\right)^2 \frac{dV_{m,i}}{df(U_{m,i})} \frac{df(U_{m,i})}{dU_{m,i}}$ for each module separately. Let $U_{m,a}(t+dt)$ be the maximum at time $t+dt$ and $U_{m,b}(t)$ be the maximum at time t for the input of the module m. Because the function f is a non-decreasing function, $f(U_{m,a}(t+dt)]$ and $f[U_{m,b}(t)]$ are the maximum at $t+dt$ and at t respectively.

$$U_{m,a}(t+dt) = \max\{U_{m,1}(t+dt), U_{m,2}(t+dt), U_{m,3}(t+dt), U_{m,4}(t+dt)\},$$

$$U_{m,b}(t) = \max\{(U_{m,1}(t), U_{m,2}(t), U_{m,3}(t), U_{m,4}(t)\},$$

$$f[U_{m,a}(t+\mathrm{d}t)]=\max\{f[U_{m,1}(t+\mathrm{d}t)], f[U_{m,2}(t+\mathrm{d}t)], f[U_{m,3}(t+\mathrm{d}t)],$$
$$\times f[U_{m,4}(t+\mathrm{d}t)]\},$$
$$f[U_{m,b}(t)]=\max\{f[U_{m,1}(t)], f[U_{m,2}(t)], f[U_{m,3}(t)], f[U_{m,4}(t)]\}$$

It is necessary and sufficient to consider the following two cases:

(1) $a=b$

(2) $a\neq b$

If the condition (1) is satisfied, then there is no state change for the module m. Consequently,

$$\sum_i\left(\frac{\mathrm{d}U_{m,i}}{\mathrm{d}t}\right)^2\frac{\mathrm{d}V_{m,i}}{\mathrm{d}U_{m,i}} \text{ must be zero.}$$

If (2) is satisfied, then

$$\sum_i\left(\frac{\mathrm{d}U_{m,i}}{\mathrm{d}t}\right)^2\frac{\mathrm{d}V_{m,i}}{\mathrm{d}f(U_{m,i})}\frac{\mathrm{d}f(U_{m,i})}{\mathrm{d}U_{m,i}}$$

$$=\left[\frac{U_{m,a}(t+\mathrm{d}t)-U_{m,a}(t)}{\mathrm{d}t}\right]^2\frac{V_{m,a}(t+\mathrm{d}t)-V_{m,a}(t)}{U_{m,a}(t+\mathrm{d}t)-U_{m,a}(t)}+\left[\frac{U_{m,b}(t+\mathrm{d}t)-U_{m,b}(t)}{\mathrm{d}t}\right]^2$$
$$\times\frac{V_{m,b}(t+\mathrm{d}t)-V_{m,b}(t)}{U_{m,b}(t+\mathrm{d}t)-U_{m,b}(t)}$$

$$=\left[\frac{U_{m,a}(t+\mathrm{d}t)-U_{m,a}(t)}{\mathrm{d}t}\right]^2\frac{1}{U_{m,a}(t+\mathrm{d}t)-U_{m,a}(t)}+\left[\frac{U_{m,b}(t+\mathrm{d}t)-U_{m,b}(t)}{\mathrm{d}t}\right]^2$$
$$\times\frac{-1}{U_{m,b}(t+\mathrm{d}t)-U_{m,b}(t)}$$

$$=\frac{U_{m,a}(t+\mathrm{d}t)-U_{m,a}(t)}{(\mathrm{d}t)^2}-\frac{U_{m,b}(t+\mathrm{d}t)-U_{m,b}(t)}{(\mathrm{d}t)^2}$$

$$=\frac{1}{(\mathrm{d}t)^2}[U_{m,a}(t+\mathrm{d}t)-U_{m,a}(t)-U_{m,b}(t+\mathrm{d}t)+U_{m,b}(t)]$$

$$=\frac{1}{(\mathrm{d}t)^2}[U_{m,a}(t+\mathrm{d}t)-U_{m,b}(t+\mathrm{d}t)+U_{m,b}(t)-U_{m,a}(t)]$$

$$>0$$

because $U_{m,b}(t+\mathrm{d}t)$ is the maximum at time $t+\mathrm{d}t$ and $U_{m,b}(t)$ is the maximum at time t for the module m.

The contribution from each term is either 0 or positive, therefore

$$\sum_i \left(\frac{sU_{m,i}}{\mathrm{d}t}\right)^2 \frac{\mathrm{d}V_{m,i}}{\mathrm{d}U_{m,i}} \geqslant 0 \quad \text{and} \quad -\sum_m \sum_i \left(\frac{\mathrm{d}U_{m,i}}{\mathrm{d}t}\right)^2 \frac{\mathrm{d}V_{m,i}}{\mathrm{d}U_{m,i}} \leqslant 0 \Rightarrow \frac{\mathrm{d}E}{\mathrm{d}t} \leqslant 0$$

The generalized maximum neural network performs a parallel improvement algorithm which can be implemented either on a sequential machine or on a parallel machine. The solution quality is always improved as time elapses until no further improvement can be achieved. The termination condition is given by the convergence state of the system. When the system reaches a stable point or an equilibrium state, the execution is terminated. The equilibrium state is reached when all firing neurons have the smallest change rate of the input per cluster. The condition of the equilibrium state is given by:

$$V_{m,i}(t)=1 \text{ and } \frac{\mathrm{d}U_{m,i}(t)}{\mathrm{d}t}=\min\left\{\frac{\mathrm{d}U_{m,1}(t)}{\mathrm{d}t}, \frac{\mathrm{d}U_{m,2}(t)}{\mathrm{d}t}, \frac{\mathrm{d}_{m,3}(t)}{\mathrm{d}t}, \frac{\mathrm{d}U_{m,4}(t)}{\mathrm{d}t}\right\} \text{ for } m =1,\ldots,M$$

Note that none of the existing Hopfield neural networks clearly define the condition of the system convergence.

6. Parallel algorithm and simulation results

The parallel algorithm based on the generalized maximum neural network is implemented in C on Dec3100 and in Turbo Pascal on Macintosh SE/30. The following procedure describes the proposed parallel algorithm based on the first order Euler method.

```
Step 0. Set $t=0$
Step 1. for $m=1$ to $M$ do
            for $i=1$ to 4 do in parallel
                $U_{m,i}(t) \leftarrow$ Random number $\in (-1,1)$
Step 2. for $m=1$ to $M$ do
            for $i=1$ to 4 do in parallel
                $V_{m,i}(t)=1$ if $f[U_{m,i}(t)]=\max\{f[U_{m,1}(t)], f[U_{m,2}(t)], f[U_{m,3}(t)], f[U_{m,4}(t)]\}$
                    0 otherwise
Step 3. for $m=1$ to $M$ do
            for $i=1$ to 4 do in parallel
                $\Delta U_{m,i}(t)=-\sum_{m'\neq m}^{M}\sum_{j=1}^{4} d_{m,i,m'j}V_{m',j}$
Step 4. for $m=1$ to $M$ do
            for $i=1$ to 4 do in parallel
                $U_{m,i}(t+1)=U_{m,i}(t)+\Delta U_{m,i}(t)$
```

Step 5. Increment t by 1.

Step 6. If the state of the system reaches an equilibrium state then stop else go to Step 2.

In the proposed parallel alogorithm, the initial state is randomly generated for the input of each processing element at step 1. At step 2, the output of each processing element is evaluated according to the generalized maximum function. The equation of motion in (10) determines the change for each input of the processing elemets at step 3. The new state of each input is updated by the first-order Euler method at step 4. We have examined our algorithm using more than one thousand examples including up to 300-module problems. The simulation results for the module orientation problem based on the Libeskind-Hadas and Liku's method, the maximum neural network, and the generalized neural network are shown in Table 1 where λ denotes the gain of the sigmoid function. Each element except the last column in Table 1 is represented by the average value and standard deviation of 100 simulation runs. In this paper the sigmoid function is used in the generalized maximum function. The last column represents the best solutions found by the Libeskind-Hadas and Liu's method. Originally, Yamada and Liu tested six optimization methods for the module orientation problem and found that the simulated annealing method provides the best performance among the six methods. Libeskind-Hadas and Liu have tested the module orientation problem based on the neural network using problems up to 96 modules. Their solutions are generally better than those found by Yamada and Liu's simulated annealing algorithm. However, the performance of the maximum neural network and the generalized maximum neural network is far better than that of Libeskind-Hadas and Liu in terms of the solution quality and the computation time. Note that the number of convergence steps in the Libeskind-Hadas and Liu system is in the order of several thousand. However, the maximum neural network and the generalized maximum neural network converge within 100 iteration steps. The other distinguished characteristics lies in that the percentage of the obtained valid solutions for Libeskind-Hadas and Liu's system is averagely below 15%.

Figure 5(*a*) shows the initial configuration of the 4-module problem, and the final result based on the maximum neural network is shown in Fig. 5(*b*). For four-module to 14-module problems, we were able to verify the optimality of our algorithm by exhaustive search. The searching complexity for 14 modules is $4^{14}(\approx 10^8)$ where it took 15 h to find the global minimum on a Macintosh SE/30 machine. One hundred simulation runs were performed for each instance where the initial state of the system is randomly generated for every simulation run. The simulation results are shown in Table 2.

The proposed algorithm always finds the global minimum or near-optimum with several ten iteration steps. Figure 6(*a*) to Fig. 6(*g*) shows the initial state and the converged state of the system for four-module through ten-module problems respectively. Figure 7(*a*), Fig. 8 (*a*), and Fig. 9(*a*) show the initial state and the converged state of the system for the 100-module, 200-module, and 300-module problem respectively. The relation between the energy and iteration steps for 100-module, 200-module, and 300-module problem are also shown in Fig. 7(*b*), Fig. 8(*b*), and Fig. 9(*b*) respectively. The energy represents the total wire length.

		Maximum neural network		Generalized maximum neural network				Libeskind-Hadas and Liu's method, best solution
				$\lambda = 1e-8$		$\lambda = 1e-7$		
Size	Initial length	Solution	Iteration steps	Solution	Iteration steps	Solution	Iteration steps	
10	2101·73 ± 116·44	1835·41 ± 1·47	21·76 ± 6·70	1836·10 ± 1·13	23·24 ± 8·57	1835·62 ± 1·45	21·73 ± 6·42	1862·14
20	3881·83 ± 121·65	3441·36 ± 8·33	20·00 ± 0·00	3445·60 ± 6·63	20·04 ± 0·20	3442·90 ± 8·38	20·00 ± 0·00	3615·03
30	8077·71 ± 123·42	7572·56 ± 38·89	26·99 ± 8·71	7578·47 ± 37·14	21·09 ± 3·00	7566·29 ± 40·32	26·03 ± 2·64	7787·16
40	11653·56 ± 237·04	10915·48 ± 19·70	20·15 ± 0·36	10916·29 ± 15·76	20·10 ± 0·39	10914·51 ± 20·03	20·14 ± 0·35	11098·94
50	14465·39 ± 166·13	13613·96 ± 18·88	20·00 ± 0·00	13613·35 ± 18·79	20·00 ± 1·50	13615·01 ± 18·89	20·00 ± 0·00	13997·05
60	17355·84 ± 213·49	16252·41 ± 5·68	28·13 ± 2·03	16247·34 ± 5·00	39·78 ± 3·18	16250·75 ± 5·50	35·13 ± 7·70	17025·57
70	20732·71 ± 251·92	19536·33 ± 6·42	35·06 ± 6·23	19534·47 ± 4·99	32·23 ± 2·70	19536·12 ± 6·36	29·79 ± 2·40	20317·16
80	23343·07 ± 262·78	21987·59 ± 10·45	20·88 ± 2·32	21987·05 ± 10·31	20·23 ± 0·66	21987·76 ± 10·16	20·85 ± 2·28	22863·13
90	28500·73 ± 147·09	27286·13 ± 8·74	20·12 ± 0·55	27283·37 ± 7·13	21·22 ± 4·03	27284·49 ± 7·83	20·37 ± 1·90	28220·61
100	29202·87 ± 166·83	27786·89 ± 22·61	21·68 ± 4·85	27785·75 ± 20·20	20·39 ± 2·19	27786·94 ± 22·45	21·27 ± 3·91	28872·39
110	32428·37 ± 332·73	30500·13 ± 5·33	21·84 ± 3·28	30501·38 ± 3·53	24·02 ± 4·27	30500·12 ± 5·25	22·30 ± 3·92	31820·94
120	36310·35 ± 268·47	34313·00 ± 12·62	26·60 ± 3·68	34310·02 ± 10·64	30·52 ± 3·00	34312·11 ± 12·91	48·09 ± 6·82	35710·07
130	38712·22 ± 211·43	36739·19 ± 5·10	22·92 ± 9·08	36739·16 ± 4·52	24·03 ± 5·11	36739·38 ± 4·82	23·97 ± 5·57	38458·68
140	39833·82 ± 169·51	38017·12 ± 12·04	27·40 ± 6·43	38017·23 ± 10·74	23·45 ± 6·78	38017·54 ± 12·17	25·86 ± 15·06	39490·21
150	41936·09 ± 247·46	39797·70 ± 10·68	45·31 ± 8·53	39787·02 ± 9·48	31·04 ± 2·57	39808·17 ± 41·87	39·67 ± 3·65	41432·39
160	47461·52 ± 322·05	45004·91 ± 9·51	21·33 ± 5·63	45004·43 ± 9·25	25·44 ± 2·91	45005·50 ± 9·83	26·22 ± 4·88	46753·53
170	48908·64 ± 339·95	46306·76 ± 19·05	42·49 ± 7·71	46305·36 ± 14·16	35·07 ± 4·99	46313·57 ± 10·10	38·01 ± 4·28	48583·02
180	53067·66 ± 404·72	50215·30 ± 20·92	20·24 ± 0·51	50227·34 ± 18·44	20·47 ± 0·77	50220·14 ± 19·73	20·33 ± 0·57	52262·82
190	57300·09 ± 654·85	53939·43 ± 18·81	21·14 ± 2·06	53944·75 ± 15·77	22·39 ± 3·04	53940·88 ± 18·94	21·11 ± 1·79	56009·94
200	61660·71 ± 624·13	58011·41 ± 15·50	20·66 ± 1·63	57994·55 ± 1·84	52·13 ± 4·50	58010·25 ± 15·39	20·72 ± 1·52	60533·42
210	63037·32 ± 550·81	59674·39 ± 18·90	31·38 ± 15·12	59696·45 ± 6·66	38·77 ± 3·70	59673·82 ± 19·10	33·08 ± 16·19	62340·82
220	66242·21 ± 438·35	62810·16 ± 8·84	20·25 ± 0·48	62808·45 ± 0·74	56·96 ± 7·02	62811·14 ± 8·93	20·27 ± 0·55	65517·85
230	70237·66 ± 404·48	66649·87 ± 18·96	24·97 ± 7·83	66632·19 ± 0·72	52·46 ± 5·47	66645·47 ± 15·47	25·01 ± 7·56	69416·02
240	76063·41 ± 403·92	72461·73 ± 12·85	44·70 ± 6·16	72440·91 ± 1·13	48·34 ± 6·87	72490·26 ± 72·42	37·92 ± 3·82	75178·57
250	77558·16 ± 336·19	73982·45 ± 12·66	32·22 ± 3·86	73973·49 ± 1·12	43·07 ± 5·55	73982·72 ± 12·22	52·95 ± 35·73	76856·05
260	81739·41 ± 485·51	77883·85 ± 18·53	39·15 ± 9·08	77887·16 ± 10·61	57·54 ± 7·35	77885·36 ± 18·83	53·64 ± 48·53	80752·46
270	82243·45 ± 325·69	78258·86 ± 9·85	25·00 ± 12·84	78251·20 ± 6·27	49·89 ± 10·07	78258·55 ± 11·16	23·96 ± 9·46	81635·57
280	86999·08 ± 423·88	82805·05 ± 13·91	31·25 ± 9·86	82797·45 ± 2·17	45·12 ± 3·35	82868·00 ± 86·72	35·98 ± 4·84	86288·80
290	91282·96 ± 310·18	87366·05 ± 17·80	36·14 ± 6·08	87389·06 ± 0·51	46·30 ± 5·54	87368·08 ± 17·61	34·35 ± 24·50	89890·12
300	91871·88 ± 351·93	87672·34 ± 19·70	20·46 ± 1·48	87657·81 ± 0·58	52·37 ± 6·23	87674·15 ± 19·96	20·51 ± 1·60	90107·05

Table 1. Comparisons of the solution quality and the number of iteration steps among the maximum neural network, the generalized maximum neural network, and Liu's method.

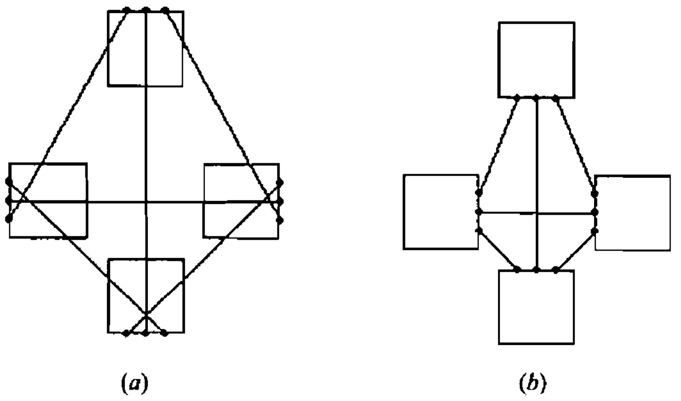
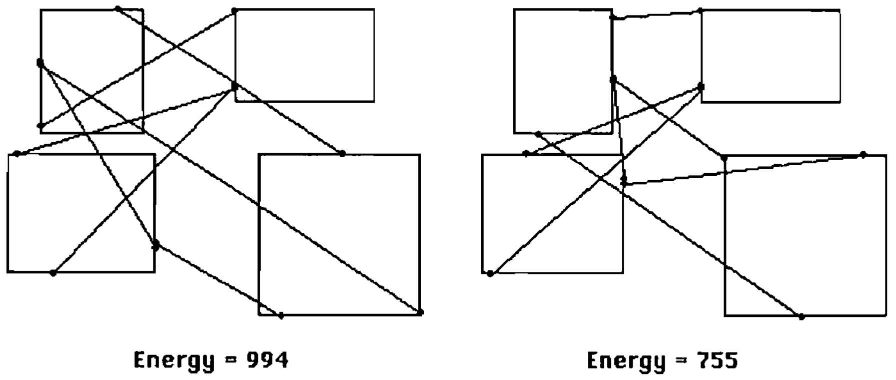
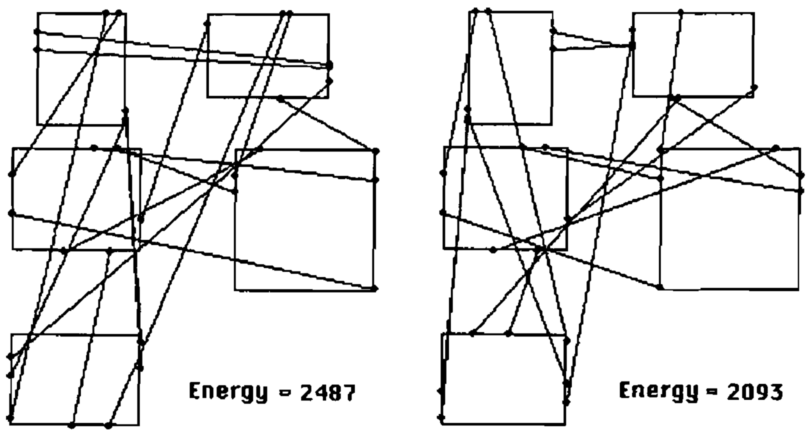


Figure 5. 4-module problem: (*a*) initial configuration of the testing example; (*b*) final solution.

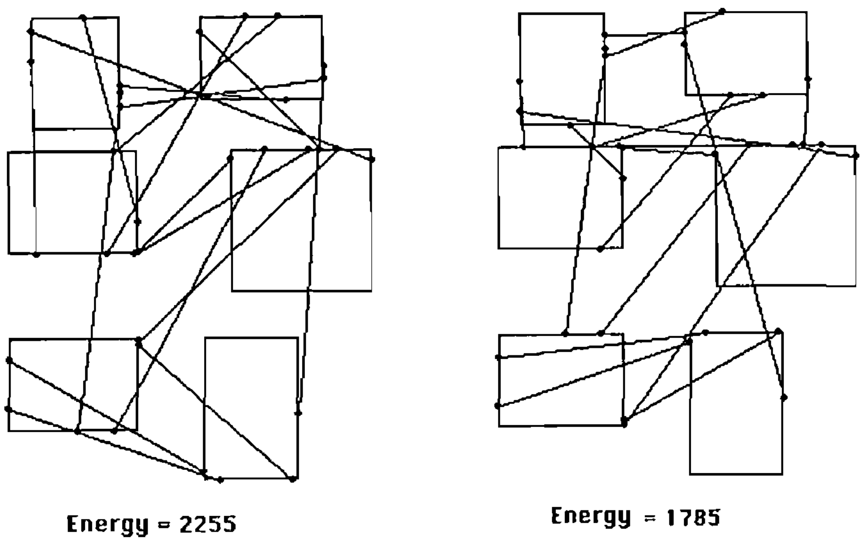


(*a*) Module size 4.

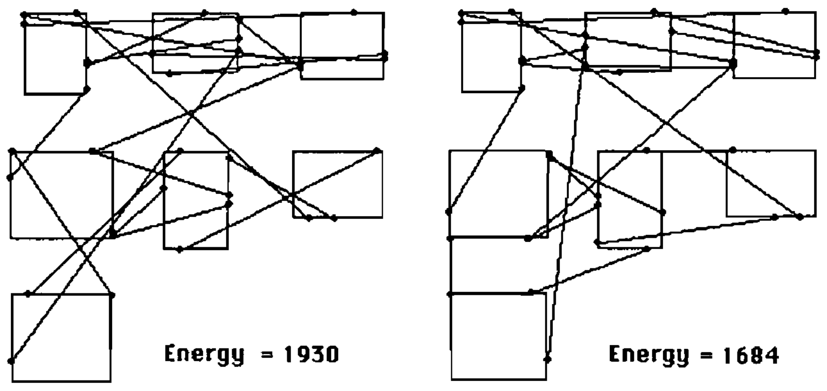


(*b*) Module size 5.

Figures 6(*a*) & 6(*b*). Module orientation problems with module size 4 and 5. The left picture and the right picture describe the initial configuration and the final converged solution, respectively.

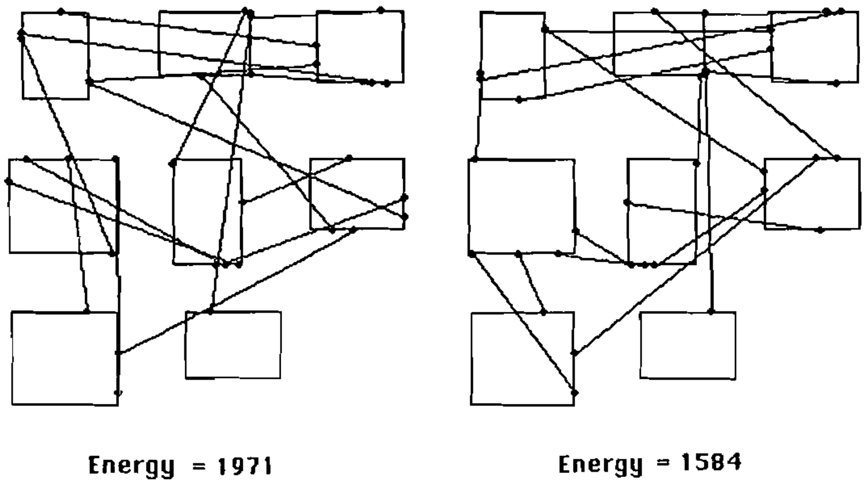


(*c*) Module size 6.

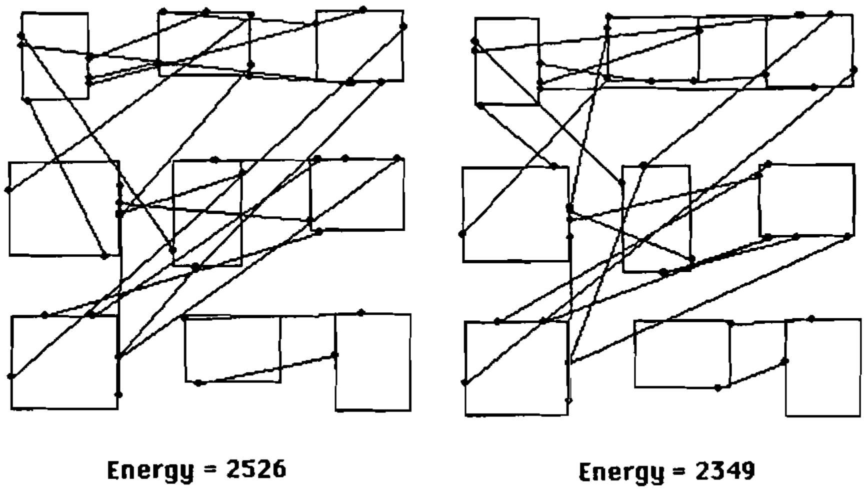


(*d*) Module size 7.

Figures 6(*c*) & 6(*d*). Module orientation problems with module size 6 and 7. The left picture and the right picture describe the initial configuration and the final converged solution, respectively.

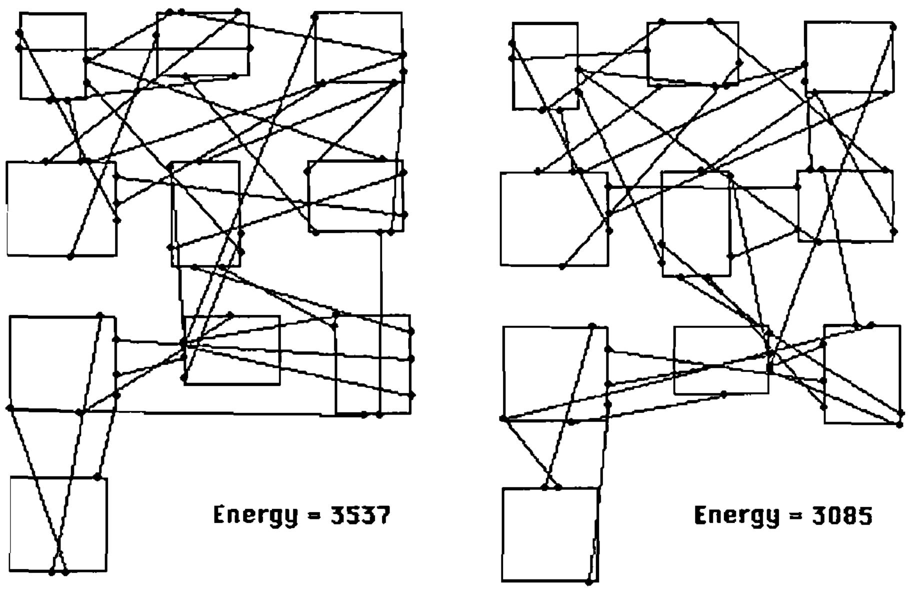


(*e*) Module size 8.



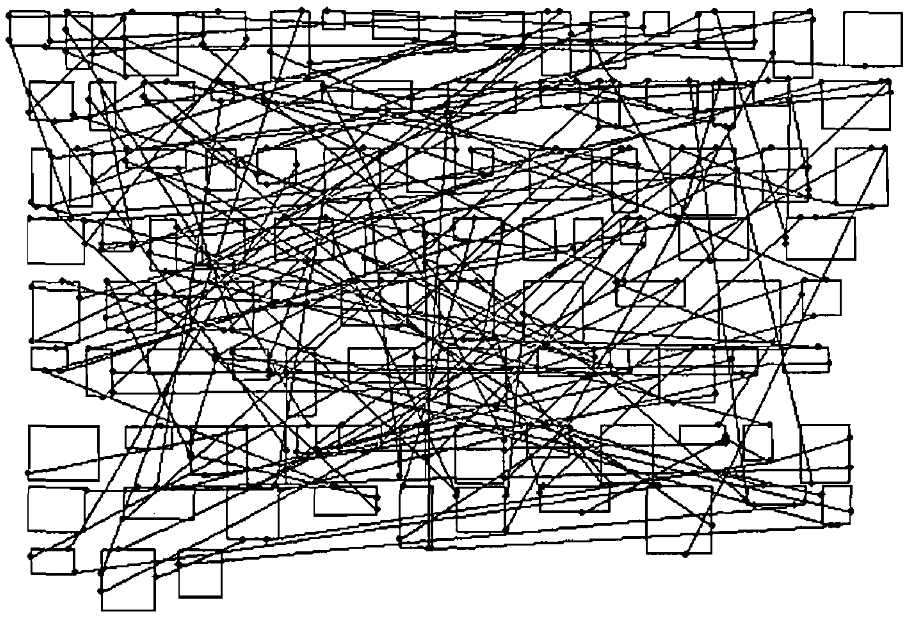
(*f*) Module size 9.

Figures 6(*e*) & 6(*f*). Module orientation problems with module size 8 and 9. The left picture and the right picture describe the initial configuration and the final converged solution respectively.

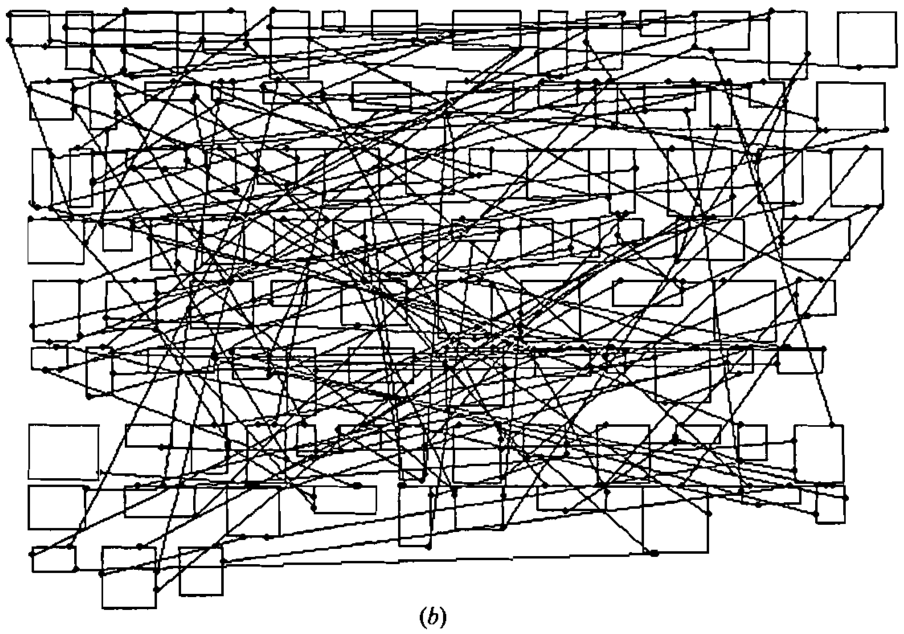


(*g*)

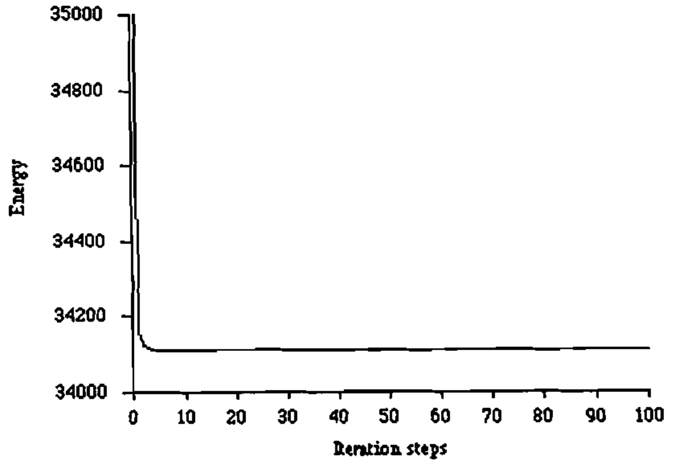
Figure 6(*g*). Module orientation problems with module size 10. The left picture and the right picture describe the initial configuration and the final converged solution respectively.



7(*a*)

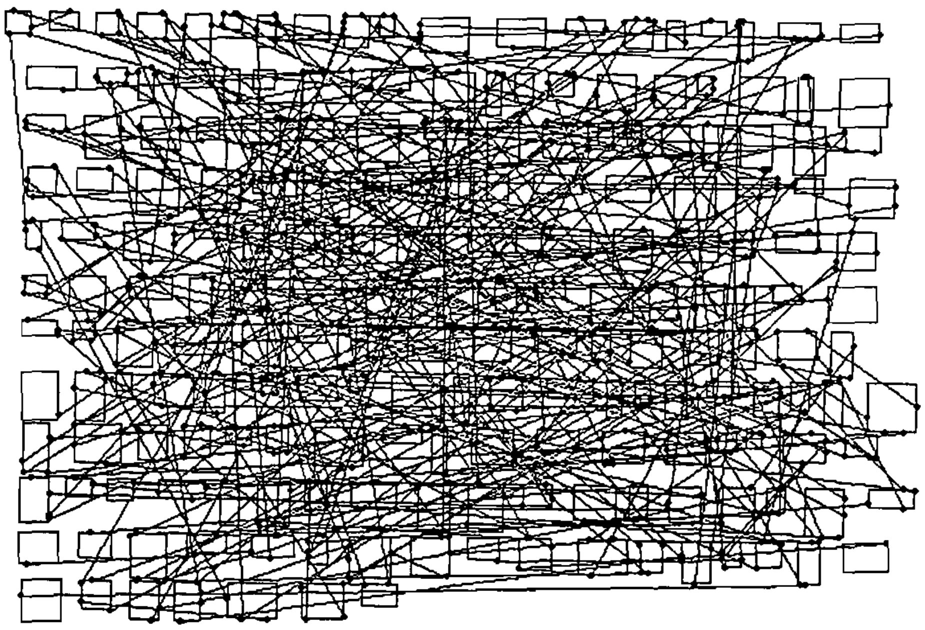


(*b*)

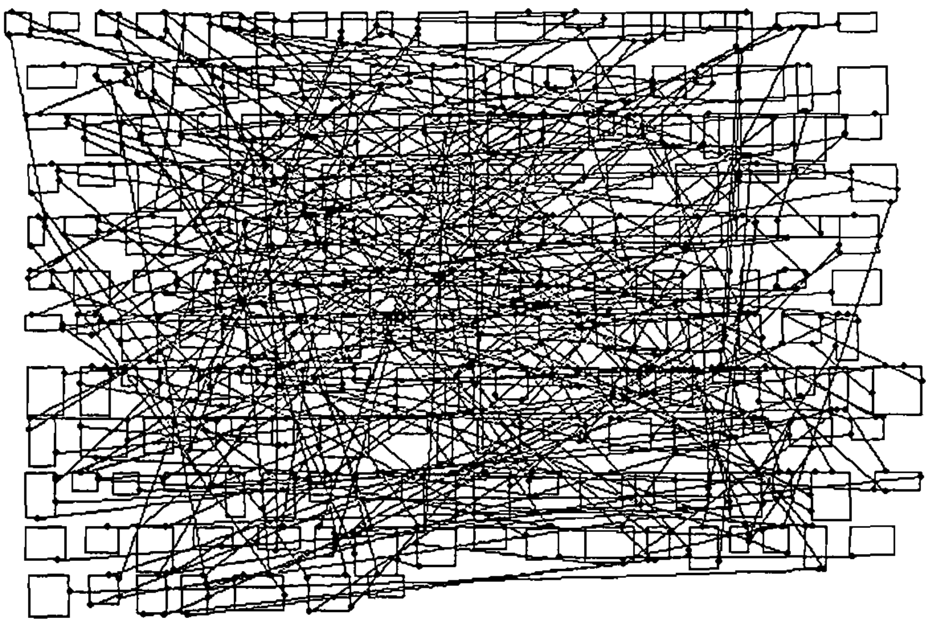


(*c*)

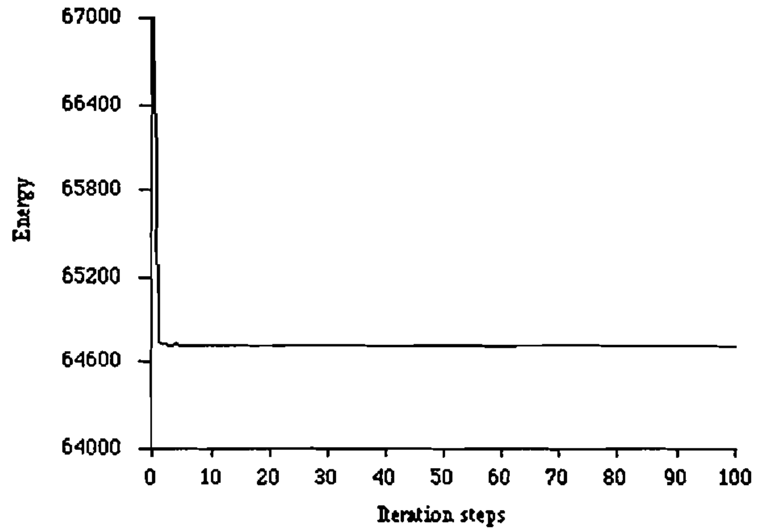
Figure 7. 100-module orientation problem: (*a*) initial state with the total length 35955·4; (*b*) final solution with the total length 34109·3; (*c*) the relationship between the energy and the number of iteration steps for the module size 100.



8(*a*)

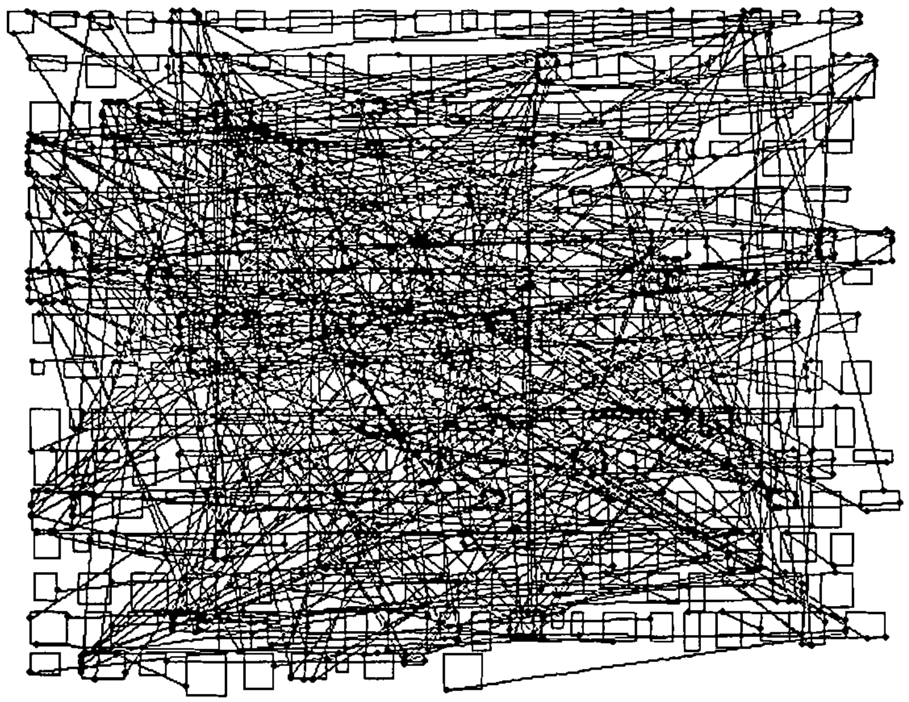


8(*b*)

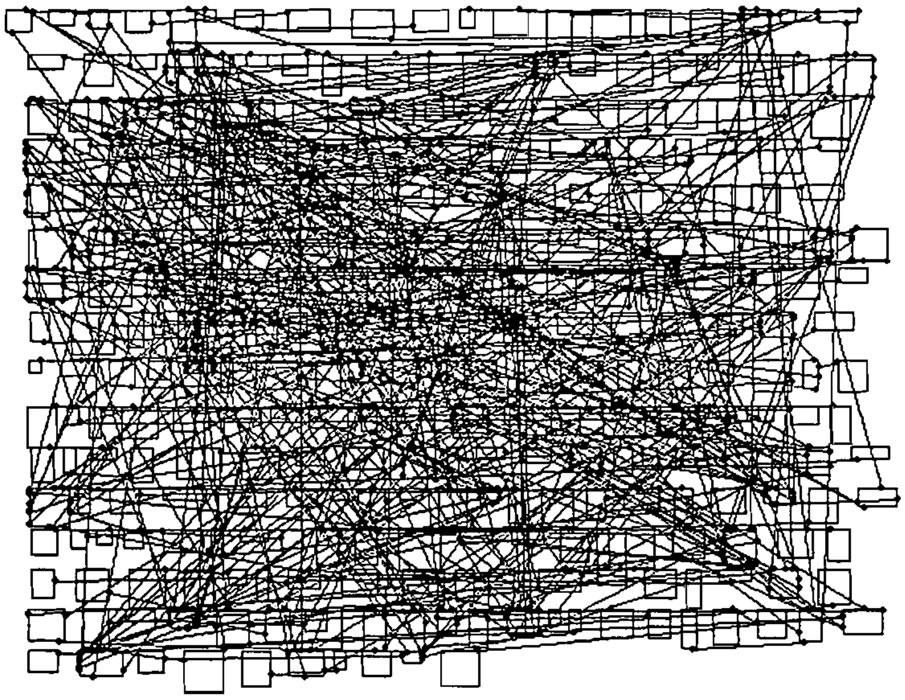


(*c*)

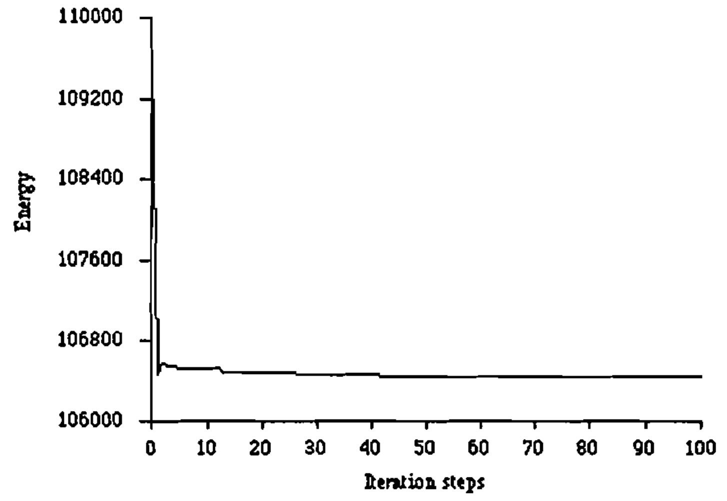
Figure 8. 200-module orientation problem: (*a*) initial state with the total length 67919·9; (*b*) final solution with the total length 64711·2; (*c*) the relationship between the energy and the number of iteration steps for the module size 200.



9(*a*)



(*b*)



(*c*)

Figure 9. 300-module orientation problem: (*a*) initial state with the total length 109748·3; (*b*) Final solution with the total length 106442·4; (*c*) the relationship between the energy and the number of iteration steps for the module size 300.

		Maximum neural network		Generalized maximum neural					
				$\lambda = 1e-8$		$\lambda = 1e-7$ network		$\lambda = 1e-6$	
Size	Optimum solution	Solution	Iteration steps	Solution	Iteration steps	Solution	Iteration steps	Solution	Iteration steps
5	1070·82	1075·66 ± 4·55	20·00 ± 0·00	1070·82 ± 0·00	20·00 ± 0·00	1073·91 ± 4·23	20·00 ± 0·00	1075·58 ± 4·60	20·00 ± 0·00
6	1021·93	1025·36 ± 5·94	20·00 ± 0·00	1021·93 ± 0·00	20·00 ± 0·00	1025·22 ± 5·86	20·00 ± 0·00	1025·49 ± 6·02	20·00 ± 0·00
7	1469·04	1474·32 ± 0·27	21·19 ± 0·81	1469·04 ± 0·00	23·00 ± 1·99	1474·37 ± 0·29	21·25 ± 0·62	1474·31 ± 0·27	21·21 ± 0·80
8	1346·26	1387·03 ± 36·43	20·00 ± 0·00	1355·03 ± 0·00	20·00 ± 0·00	1381·24 ± 34·11	20·00 ± 0·00	1386·60 ± 36·01	20·00 ± 0·00
9	1782·45	1784·60 ± 2·11	20·00 ± 0·00	1782·45 ± 0·00	20·43 ± 0·72	1783·48 ± 1·67	20·00 ± 0·00	1784·52 ± 2·12	20·00 ± 0·00
10	2176·63	2180·55 ± 5·63	20·00 ± 0·00	2176·87 ± 0·52	20·00 ± 0·00	2177·96 ± 3·06	20·00 ± 0·00	2179·38 ± 4·61	20·00 ± 0·00
11	3108·03	3120·62 ± 20·24	20·75 ± 1·25	3108·03 ± 0·00	23·00 ± 0·00	3125·56 ± 22·31	21·65 ± 1·39	3125·65 ± 22·13	21·06 ± 1·38
12	3262·56	3277·33 ± 0·77	22·51 ± 1·53	3262·56 ± 0·00	21·69 ± 3·02	3276·99 ± 0·42	22·98 ± 2·30	3277·12 ± 0·60	22·46 ± 1·84
13	3276·38	3280·66 ± 3·88	25·29 ± 3·99	3276·90 ± 0·33	25·46 ± 0·50	3279·42 ± 2·18	51·43 ± 6·82	3287·62 ± 3·76	48·81 ± 3·03
14	3103·13	3107·97 ± 3·60	20·00 ± 0·00	3108·61 ± 0·03	20·06 ± 0·42	3108·32 ± 2·73	20·00 ± 0·00	3107·68 ± 3·02	20·00 ± 0·00

Table 2. Comparisons with global optimum solutions.

7. Conclusion

In this paper a generalized maximum neural network for parallel computing is proposed and the module orientation problem is used to test the capability of the artificial neural network. The circuit diagrams of the generalized maximum neural network and the maximum neural network are also presented and compared with that of the Liu's Hopfield neural network. The simulation results for the proposed system are quire encouraging. As Yu (1989) pointed out, the artificial neural network cannot work well if the original Hopfield neural network is used. However, it does not mean that the artificial neural network approach is not suitable for VLSI/CAD problems. As long as the proper neural model and the proper representation are used, the artificial neural network gives a better solution than the best known algorithms. Future research will focus upon finding a proper representation and processing element model for other CAD problems.

References

Carpenter, G., and Grossberg, S., 1987, A massively parallel architecture for a self-organization neural pattern recognition machine. *Computer Vision, Graphical Image Processing*, **37**, 54-15.

Funabiki, N., and Takefuji, Y., 1991, A parallel algorithm for spare allocation problems. *I.E.E.E. Transactions on Reliability*, **40**, 338-346.

Garey, M. R., and Johnson, D. S., 1979, *Computers and Intractability: a Guide to the Theory of NP-Completeness* (New York: W. H. Freeman).

Hanan, M., Wolff, P. K., Sr., and Agule, B. J., 1976, A Study of Placement Techniques. *Journal of Design Automation and Fault Tolerant Computin*, **1**, 28-61.

Hopfield, J. J., 1984, Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences of the U.S.A.*, **81**, 3088-3092.

Hopfield, J. J., and Tank, D. W., 1985, Neural Computation of Decisions in Optimization Problems. *Biological Cybernetics*, **52**, 141-152.

Kohonen, T., 1984, *Self-Organization and Associated Memory* (New York: Springer-Verlag).

Lee, K. C., and Takefuji, Y., 1991, A parallel algorithm using the maximum neural network for the max cut problem. *Case Western Reserve University, CAISR Tech. Rep.* TR91-104.

Lee, K. C. *et al.*, 1991, A parallel algorithm for the graph partition problem. *Case Western Reserve University, CAISR Tech. Rep.* TR91-003; A parallel improvement algorithm for the bipartite subgraph problem. *Case Western Reserve University, CAISR Tech. Rep.* TR91-105.

Libeskind-Hadas, Ran, and Liu, C. L., 1989, Solutions to the module orientation and rotation problems by neural computation networks. *Proceedings of 26th Design Automation Conference*, pp. 400-405.

McCulloch, W. S., and Pitts, W. H., 1943, A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, **5**, 115.

Preas, B. T., and Karger, P. G., 1988, Placement, assignment, and floorplanning. *Physical Design Automation of VLSI Systems*, edited by B. Preas and M. Lorenzetti (Menlo Park, Calif: Benjamin-Cummings), Chap. 4, pp. 87-155.

Rosenblatt, F., 1958, The perceptron: a probabilistic model for information storage and organization in the brain. *Phychological Review*, **65**, 386-408.

Rumelhart, D. E., and McClelland, J. L., 1986, *Parallel Distributed Processing*, Vol. 1: *Foundations* (Cambridge, Massachusetts: MIT Press).

Takefuji, Y., and Lee, K. C., 1989, A near-optimum parallel planarization algorithm. *Science*, **245**, 1221-1223; 1990*a*, A parallel algorithm for tiling problems. *I.E.E.E. Transactions on Neural Networks*, **1**, 143-145; 1990*b*, A super parallel sorting algorithm based on neural networks. *I.E.E.E. Transactions on Circuits and Systems*, **37**, 1425-1429; 1991, Artificial neural networks for four-coloring map problems and k-colorability problems. *I.E.E.E. Transactions on Circuits and Systems*, **38**, 326-333.

Takefuji, Y., Li-lin Chen, Lee, K. C., and Huffman, J., 1990, Parallel algorithm for finding a near-maximum independent set of a circle graph. *I.E.E.E. Transactions on Neural Networks*, **1**, 263-267.

Takefuji, Y., Lin, C. W., and Lee, K. C., 1990, A parallel algorithm for estimating the secondary structure in Ribonucleic acids. *Biological Cybernetics*, **63**, 337-340.

Widrow, B., and Hoff, M. E., 1960, Adaptive switching circuits. *IRE Convention Record: Part 4—Computers; Man-Machine Systems*, Los Angeles, pp. 96-104.

Yamada, M., and Liu, C. L., 1988, An analytical method for optimal module orientation. *Proceedings of the 1988 International Symposium on Circuits and Systems*, pp. 1679-1682.

Yao, X., Yamada, M., and Liu, C. L., 1989, A new approach to the pin assignment problem. *I.E.E.E. Transactions on Computer-Aided Design*, **8**, 999-1006.

Yu, M. L., 1989, A study of the applicability of Hopfield decision neural nets to VLSI CAD. *Proceedings of the 26th Design Automation Conference*, pp. 412-417.