Paper

A Parallel Algorithm for Maximum Cut Problem Using Gradient Ascent Learning of Hopfield Neural Networks

Student Member Rong Long WANG (Toyama University)
Non-member Zheng TANG (Toyama University)
Non-member Qi Ping CAO (Tateyama Systems Institute)

Maximum cut is an important example of a class of combinatorial optimization problems. It has many important applications including the design of VLSI circuits and design of communication networks. The goal of this NP-complete problem is to partition the node set of an undirected graph into two parts in order to maximize the cardinality of the set of edges cut by the partition. In this paper, we propose a parallel algorithm using gradient ascent learning algorithm of the Hopfield neural networks for efficiently solving such optimization problems. The proposed learning algorithm is tested on a 2-variable quadratic polynomial and applied to the MAX CUT problem. Extensive simulations are performed and its effectiveness is confirmed.

Keywords: Maximum cut problem, Hopfield neural network, Gradient ascent learning, NP-complete problem

1. Introduction

The maximum cut problem (MAX CUT) [1] is a representative problem of combinatorial optimal problems. In this problem, we have a weighted, undirected garph G=(V, E) and we look for a partition of vertices of graph G into two disjoint sets $(S, \overline{S})$, such that the total weight of the edges that go from one to the other is as large as possible. Besides its theoretical importance, the maximum cut problem has applications in the design of VLSI circuits, the design of communication networks, circuit layout design and statistical physics [2], [3]. This problem is one of the Karp's original NP-complete problems [1], and has long been known to be NP-complete even if the problem is unweighted [4]. For planar graphs this problem has been shown to be polynomial solvable [5]. However, in general the weighted graph may not be planar. Because of its theoretical and practical importance and because efficient algorithms for NP-complete combinatorial optimization problems are unlikely to exist, many polynomial time approximation algorithms have been proposed to solve it.

In 1976, Sahni and Gonzales [6] presented a approximation algorithm for the MAX CUT problem. Their algorithm iterates through the vertices and decides whether or not to assign vertex i to S based on which placement maximizes the weight of the cut of vertices 1 to i. This algorithm is essentially equivalent to the randomized algorithm that flips an unbiased coin for each vertex to decide which vertices are assigned to the set S. Since 1976, a number of researchers have presented approximation algorithms for the MAX CUT problem. Hsu [7] developed a greedy algorithm to approximate the solution to the maximum cut problem on general graphs with arbitrarily weighted edges. Using semidefinite programming [8], Goemans and Williamson [9] presented randomized approximation algorithm for the maximum cut problem. Their algorithm has a very good worst case performance but it can handle efficiently only graphs of small size (vertices $\simeq 100$), while it becomes very slow for larger instances (vertices $\simeq 200$). Besides, because of its complex design it cannot easily be implemented on dedicated circuits. For this reasons, Bertoni et al. [10] presented a simple algorithm, called LORENA, which is inspired by Goemans and Williamson's idea. In [10] Bertoni et al showed that LORENA behaves better than Goemans and Williamson's algorithm.

For solving such combinatorial optimal problems, Hopfield neural networks [11]-[12] also constitute an important avenue. The Hopfield neural networks have provided a parallel and powerful method of solving difficult optimazition problems which can be described as a quadratic polynomial function without square. Using the neural network, Alberti et al. [13] presented a neural algorithm for MAX CUT problem. It has generally been found that the Hopfield networks do produce some useful results very fast, they suffer from significant drawbacks, such as local minimum problems. The rate to get the maximum cut is not as good as other local search based algorithm such as LORENA [10].

In this paper, we introduce a parallel learning algorithm for solving local minimum problems of a Hopfield network. The parallel learning algorithm has two phases, the Hopfield network updating phase and the gradient ascent learning phase. The Hopfield network updating phase finds a local minimum of energy function. The gradient ascent learning phase makes the network escape from the local minimum by modifying the weight and the thresholds of the network (or parameters of the quadratic polynomial). The proposed parallel learning algorithm is tested on a 2-variable quadratic polynomial and applied to solving the MAX CUT problem. The proposed learning algorithm is also compared with LORENA Because the LORENA gives better solution than other algorithms for solving MAX CUT problem. The experimental results shows that the proposed algorithm produces better solutions than the LORENA. Focusing on the local minimum problem of optimization techniques, some other optimization techniques have been proposed. Simulated Annealing (SA) [14] is a widely used meta-heuristic. It could be described as a randomized scheme, which reduces the risk of getting trapped in local minima by allowing moves to inferior solution. Simulated annealing is a powerful method for solving local minima, but it always requires more iterations than exhaustive search to find a good solution [15]. Baba [16] proposed a hybrid algorithm to solving the minimum problem of the back-propagation network based on random optimization method. To evaluate the performance of the proposed method, simulated annealing is also executed for comparison.

This paper is organized as follows. In Section 2, we describe the Hopfield network representation of MAX CUT problem. Section 3 describes gradient ascent learning phase. Section 4 describes procedure of the proposed learning algorithm for solving optimization problems. Section 5 presents simulation results on 2-variable quadratic polynomial and MAX CUT problems.

2. MAX CUT Problem and Its Hopfield Neural Network Representation

Let $G = (V, E)$ be an edge-weighted undirected graph, where V is the set of vertices and E is the set of edges. The edge from vertex i to vertex j is represented by $e_{ij} \in E$. $d_{ij} = d_{ji}$ defines weights on edges whose endpoints are vertex i and vertex j. The maximum cut problem is to find a partition of V into two nonempty, disjoin sets A and B, such that $A \cup B = V$ and $A \cap B = \Phi$ and $\sum_{i \in A, j \in B} d_{ij}$ is maximum. The maximum condition can also be represented as $(\sum_{ij \in A} d_{ij} + \sum_{ij \in B} d_{ij})$ is minimum.

An objective function can be formulated for this optimization problem whose minimum value corresponds to the optimal solution. In a reasonable formulation there are two components to the objective function: one which is to realize the maximum condition described above, and one which is to satisfy that every vertex is distributed into one and only one vertex set. Thus, this optimization problem can be mathematically stated as finding the minimum of the following objective function:

$$E(\mathbf{v}) = \sum_{i=1}^{N} \sum_{j \neq i}^{N} d_{ij}(v_{iA}v_{jA} + v_{iB}v_{jB}) + \sum_{i=1}^{N} | v_{iA} + v_{iB} - 1 |$$

where v_{iK} is 1 if vertex i♯ is partitioned into vertex set K, 0 otherwise.

In general, the N-vertex MAX CUT problem can be mapped onto the Hopfield neural network with $2 \times N$ neurons where it consists of N clusters of two neurons. The output y_{ij} of neuron ♯ij represents whether or not i-vertex ($i = 1, \cdots, N$) should be partitioned into vertex set j (j=1,2 which represents vertex set A and vertex set B). For example, the state of two neurons (y_{i1} =1, y_{i2} =0) indicates that the i-vertices is partitioned into set A. The following state ($y_{i1} = y_{i2} = 0$) and ($y_{i1} = y_{i2} = 1$) express no partition and double partition violation, respectively. These partition violation conditions can be expressed by follow:

$$\sum_{i}^{N}(\sum_{j}^{N} y_{ij} - 1)^2 = 0 \quad \cdots\cdots (1)$$

The maximum cut condition can be expressed by

$$Min(\sum_{i=1}^{N}\sum_{k \neq i}^{N}\sum_{j=1}^{2} d_{ik}y_{ij}y_{kj}) \quad \cdots\cdots (2)$$

where d_{ik} is weight on edges whose endpoints are vertex i and vertex k, and is the symmetric matrix. When we follow the mapping procedure by Hopfield and Tank [9], the energy function for the MAX CUT problem is given by:

$$e = A\sum_{i=1}^{N}(\sum_{j=1}^{2} y_{ij} - 1)^2 + B\sum_{i=1}^{N}\sum_{k \neq i}^{N}\sum_{j=1}^{2} d_{ik}y_{ij}y_{kj} \quad (3)$$

where A and B are constant coefficients.

In order to simplify the problem, we show a new Hopfield neural network representation in which only N neurons are used for the N-vertex MAX CUT problem. Neuron (y_i) expresses the i♯ vertices. Two neuron states ($y_i = 1$ and $y_i = 0$) express that the i♯ vertices is be partitioned into the set A and B, respectively. In this representation, the no partition and double partition violation can be avoided. Thus, the N-vertex MAX CUT problem can be simplify to

$$Min[\sum_{i=1}^{N}\sum_{j \neq i}^{N} d_{ij}(y_i y_j + (1 - y_i)(1 - y_j))] \quad \cdots (4)$$

The energy function for MAX CUT problem can be expressed by

$$\boxed{e = \frac{A}{2}\sum_{i=1}^{N}\sum_{j \neq i}^{N} d_{ij}(y_i y_j + (1 - y_i)(1 - y_j)) \quad \cdots} (5)$$

We rewrite this energy function into a standard energy function of the Hopfield network:

$$e = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}y_i y_j - \sum_{i=1}^{N} h_i y_i + constant \quad (6)$$

where, the weights and thresholds of the Hopfield network become

$$w_{ij} = -2A(1 - \delta_{ij})d_{ij} \quad \cdots\cdots (7)$$

$$h_i = A\sum_{j=1}^{N}(1-\delta_{ij})d_{ij} \cdots\cdots\cdots\cdots\cdots\cdots\cdots (8)$$

In Eq.(7) and Eq.(8), the notation δ_{ij} is 1 if $i = j$, 0 otherwise.

The equations of motion is:

$$dx_i/dt = \sum_{j=1}^{N} w_{ij}y_j + h_i \cdots\cdots\cdots\cdots\cdots\cdots (9)$$

The follow sigmoid function is used as input/output function of neurons.

$$y_i = 1/(1+e^{(-x_i/T)}) \cdots\cdots\cdots\cdots\cdots\cdots\cdots (10)$$

where T is a parameter called temperature parameter. It is proved by Hopfield [11] that such Hopfield network can be used as an approximate method for solving 0-1 optimization problems because, the network converges to a minimum of energy function provided that the weights are symmetric ($w_{ij} = w_{ji}$) and there are no self-connections ($w_{ii} = 0$).

The Hopfield network updating procedure can be viewed as seeking a minimum in a mountainous terrain. Thus, we can find the solution to the MAX CUT problem simply by observing the stable state that the Hopfield network reaches. But because of its inherent local minimum problem, the global minimum or good solution is usually difficult to be found.

3. Gradient Ascent Learning

In order to realize the global minimum convergence of the Hopfield neural network, we propose a learning method, which can help the network escape from a local minimum to the global minimum. In order to explain the learning method, we use a two-dimensional graph (Fig. 1) of energy function with a local minimum and a global minimum. The energy function value is reflected in the height of the graph. Each position on the energy terrain corresponds to a possible state of the network. For example, if the network is initialized onto the mountainous terrain A, the updating procedure of the Hopfield network makes the state of network move towards a minimum position and reach a steady state B (Fig. 1(a)).

Because the weights and the thresholds of the Hopfield network determine the energy terrain, we can change the weights and the thresholds to increase the energy at the point B so as to fill up the local minimum valley and finally drive the point B out of the valley. Here, suppose that a vector **v** corresponds to the weights and the thresholds of the Hopfield network. Since for a parameter vector **v** the learning requires the parameter change to be in the positive gradient direction, we take:

$$\Delta\mathbf{v} = \eta\Delta e(\mathbf{v}) \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (11)$$

Where η is a positive constant and Δe is the gradient of energy function e with respect to the parameter vector **v** in the state B.

Applying this learning rule (Eq. (11)) to the MAX CUT problem, we can obtain the changes of the weights and the thresholds of MAX CUT problem from the partial differential of the energy function (Eq.(6)) to the weights and the thresholds at the state B:

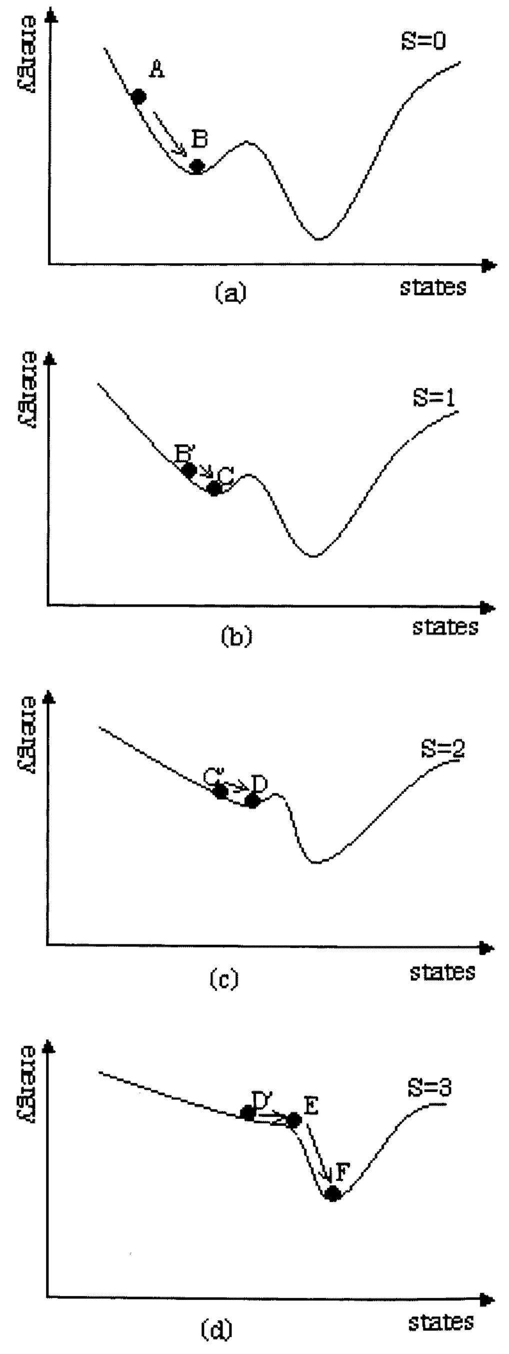

Fig. 1. The conceptual graph of the relation between energy and state transition in the learning process of the Hopfield network with two stable states.

$$\Delta w_{ij} = p\frac{\partial e}{\partial w_{ij}} = -py_iy_j \quad (for\ \ i,j = 1,...,N\ \ and\ \ j \neq j) \cdots (12)$$

$$\Delta h_i = q\frac{\partial e}{\partial h_i} = -qy_i \quad (for\ \ i = 1,...,N) \cdots (13)$$

where, p and q are small positive constants and y_i, y_j correspond to the state of B.

Now we show that after we change the weights and the thresholds according to Eq.(12) and Eq.(13), point B will be on the slope of a valley. Suppose y_{Bi} represent the state of point B, y_{Pi} represent the state of any point P of energy terrain, then the change of energy in point P by the learning rule (Eq.(12) and Eq.(13)) will be:

$$\begin{aligned}\Delta E_P &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}y_{Pi}y_{Pj} + \sum_{i=1}^{N} h_iy_{Pi} \\ &\quad -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(w_{ij} - py_{Bi}y_{Bj})y_{Pi}y_{Pj} \\ &\quad -\sum_{i=1}^{N}(h_i - qy_{Bi})y_{Pi} \\ &= \frac{p}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(y_{Bi}y_{Pi})(y_{Bj}y_{Pj}) \\ &\quad +q\sum_{i=1}^{N}(y_{Bi}y_{Pi}) \cdots\cdots\cdots\cdots\cdots\cdots (14)\end{aligned}$$

Because point B is a minimum of energy function and the output of neuron in point B is at or near 0 or 1 [11], from Eq.(14), we can know easily that the increase of energy is largest when point P is at the same point as point B, and the larger the difference of state between point P and point B, the smaller energy increases in point P. Thus, we can see that the valley will be filled up in a most effective way. In general, point B may become a point on the slope of the valley. Thus, the learning (the second phase) makes the previous stable state B becomes a point on the slope of a valley (B'). After updating of the Hopfield network with the new weights and the new thresholds in the Hopfield network updating phase again, point B' goes down the slope of the valley and reaches a new stable state C (Fig. 1(b)). Thus, the Hopfield network updating (Phase 1) and the gradient ascent learning (Phase 2) in turn may result in a movement out of a local minimum, and lead the network converge to a global minimum or a new local minimum (Fig. 1(c) and (d)).

4. Algorithm

The following procedure describes the proposed learning algorithm for solving optimization problems. Note that there are two kinds of conditions for end of the learning. One has a very clear condition, for example, the N-queen problem in which the energy is zero if the solution is the optimal. Another one has not a clear condition, for example, the travelling salesman problem and the MAX CUT problem in which the energy is not zero even when the solution is the optimal. For the latter case, we have to set a maximum numbers of the learning (*learn_limit*) in advance. Learning stops if the maximum number of learning is performed. If the *learn_limit* is supposed to be the maximum number of learning times for the system termination condition, we have,

1. Set *learn_time*=0 and set *learn_limit* and other parameters.

2. The initial value of y_i for $i = 1, \cdots, N$ are randomized in the range from 0.0 to 1.0.

3. The updating procedure is performed on the Hopfield network with original weights and thresholds until the network converges a stable state (Phase 1).

4. Record the stable state.

5. Use Eq.(12) and Eq.(13) to compute the new weights and new thresholds (Phase 2).

For $i = 1, \cdots, N$

a. Compute the Δw_{ij}, using Eq.(12) for $j = 1, \cdots, N$ and $j \neq i$.

b. Change the w_{ij} for $j = 1, \cdots, N$ and $j \neq i$.

$w_{ij} = w_{ij} + \Delta w_{ij}$

c. Compute the Δh_i, using Eq. (13).

d. Change the h_i.

$h_i = h_i + \Delta h_i$

6. The updating procedure is taken on the Hopfield network with the new weights and thresholds until the network reaches a stable state.

7. Because the weights and the thresholds of the Hopfield network determine the energy terrain, once the weights and the thresholds are changed, the energy terrain may be changed and the position of global minimum in energy terrain may also be changed. In order to avoid the shift of the state of the global minimum to a specific problem, the updating procedure on the Hopfield network may be re-performed with original weights and thresholds until the network reaches a new stable state.

8. If the new stable state obtained from step 7 is better than the recorded stable state, then the recorded stable state is replaced by the new stable state obtained from step 7.

9. Increment the *learn_time* by 1. If *learn_time* = *learn_limit* then terminate this procedure, otherwise using the stable state obtained from step 6 go to the step 5.

5. Simulation Results

5.1 Simulation Results on 2-Variable Quadratic Polynomial

In order to test the efficiency of the proposed learning method, firstly we simulat the following minimum problem of a 2-variable quadratic polynomial function without square term in [0,1] topological space.

$$E = w_{12}x_1x_2 + w_{21}x_2x_1 + h_1x_1 + h_2x_2 \cdots (15)$$

This optimization problem can be performed using a 2-neuron Hopfield network. We set $w_{12} = w_{21} = -2.0$, $h_1 = 1.0$ and $h_2 = 0.9$. In simulations, the learning constants are $p = 0.001, q = 0.01$, and the temperature parameter $T = 0.24$. Fig. 2 is a 3-dimensional contour line figure of the energy of the network. The two horizontal axes correspond to the states of the two neurons, the vertical axis is the energy of the network. It can be seen that the relation between the energy function and the states of the two neurons becomes a shape of a saddle. As shown in Figs.2(a), the network has a local minimum at the corners of $y_1 = 0, y_2 = 1$ and a global minimum at the corner of $y_1 = 1, y_2 = 0$. Fig.3 shows the energy of the Hopfield network, and its state change during the learning intuitively. First, in an initial state A at time s=0 (y_1= 0.95, y_2= 0.95), the energy takes very large value -0.475287. From the initial state, the network goes down to lower the energy as shown in the figure, and converges to -0.930935 by the energy and $y_1 = 0.118270, y_2 = 0.940722$ by the state B. Next if the learning using the Eq.(11) is performed, the energy in state $B(y_1 = 0.118270, y_2 = 0.940722)$ will go up to $E = -0.911195$ and becomes a point B' on the slope of the valley again. This is called the first learning ($s = 1$). If the network updates using the new weights and the new thresholds in time domain from B' again, it will converge to $E = -0.911226$ by the energy and $y_1 = 0.124065, y_2 = 0.932768$ by the state C. In this way, the updating phase and the learning phase are repeated on the Hopfield network. After the $4 - th$ learning ($s = 4$), the network goes up to the energy $E = -0.8546609$ and the state E' (y_1=0.142568, y_2=0.902425). Thus, the local minimum valley of the energy of the network disappeares, and the network updates to a state G (y_1=0.996728, y_2=0.007408) from the state E' through F, thus resulting in an escape from the local minimum valley. The energy also decreased abruptly from $E = -0.8546609(E')$ to $E = -1.389895(G)$ after a small decrease from $E = -0.8546609(E')$ to $E = -0.868840(F)$. The details of the energy change during the learning are shown in Fig.4. Fig.5 shows the changes of the weights and the thresholds during the whole learning process. Fig.2(b) shows the 3-dimensional contour line figure of the energy of the network after learning. By the above simu-

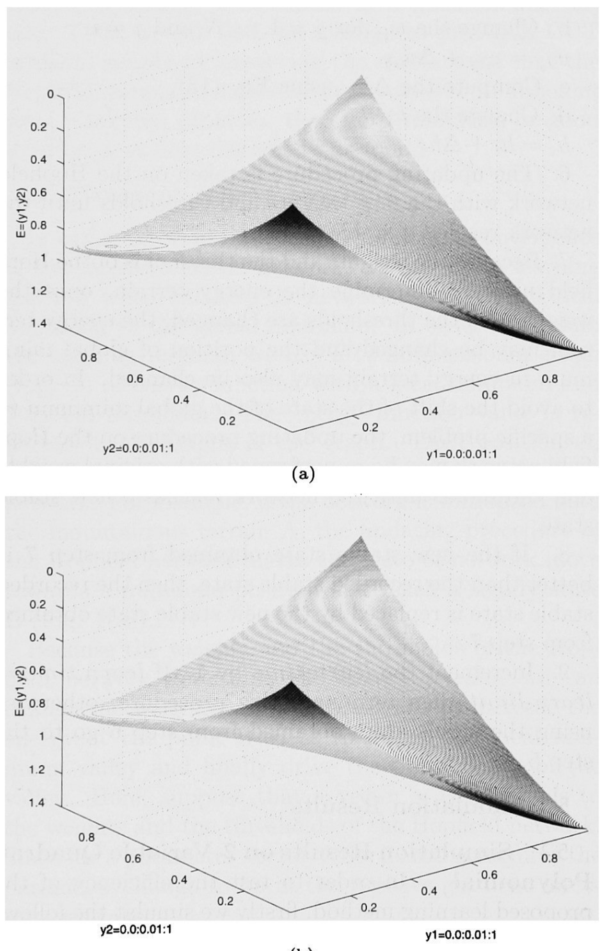

Fig. 2. 3-dimensional energy contour map for a two-neuron Hopfield network: (a) before learning, (b) after learning.

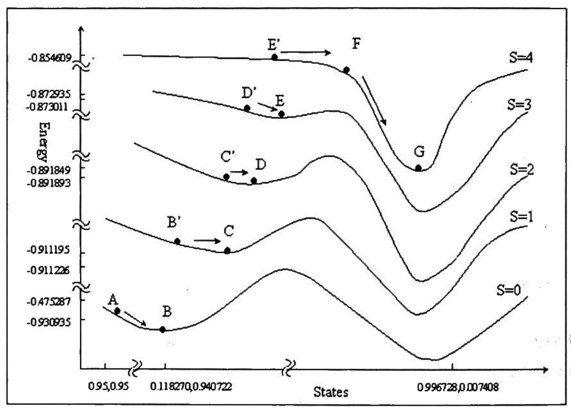

Fig. 3. Simulated energy of Hopfield network and its state change during learning.

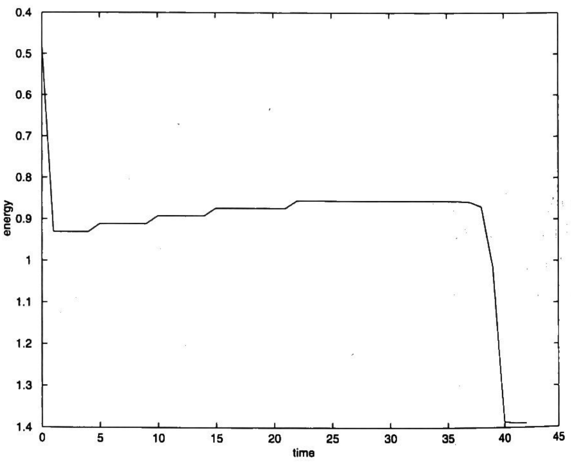

Fig. 4. Details of energy change with time during learning.

lation results, the valley (the local minimum) of energy is extinguished, namely, is understood that the learning method proposed in this paper is effective in making a Hopfield network escape from a valley (a local minimum). Furthermore, according to Fig.3 and 4, by updating phase after the 4-th learning, the energy decreases from E' to F gently first and fall§ in the minimum G from F abruptly. This is because the energy has an extraordinary loose hill as shown in Fig.2(b). These simulation results are in agreement with our learning algorithm. Furthermore, the information such as the structure of the energy and the feature may be acquired from the energy change during the learning by this learning method.

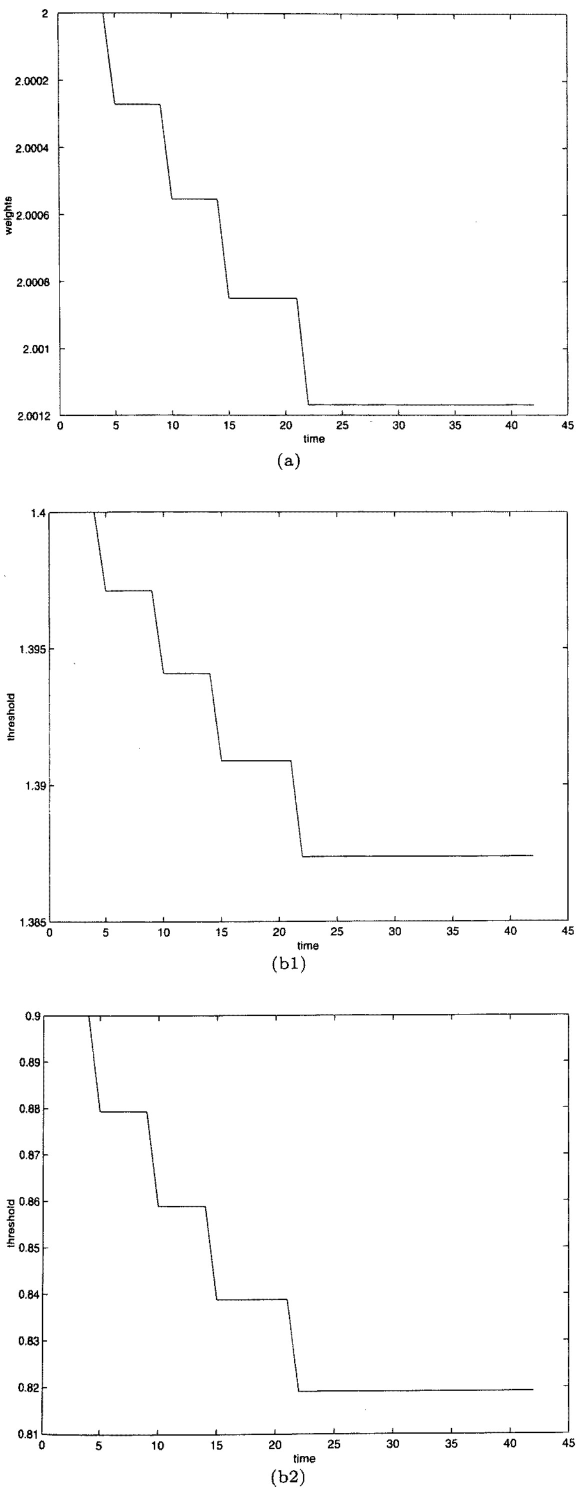

Fig. 5. Changes of the weight (a) and the thresholds (b1 and b2) during learning.

5.2 Simulation Results on MAX CUT Problem Using the proposed learning algorithm, we simulat MAX CUT problem on IBM NetVista A40 (PentiumIII 733MHz). The Hopfield network updating phase uses the weights and thresholds matrix defined in

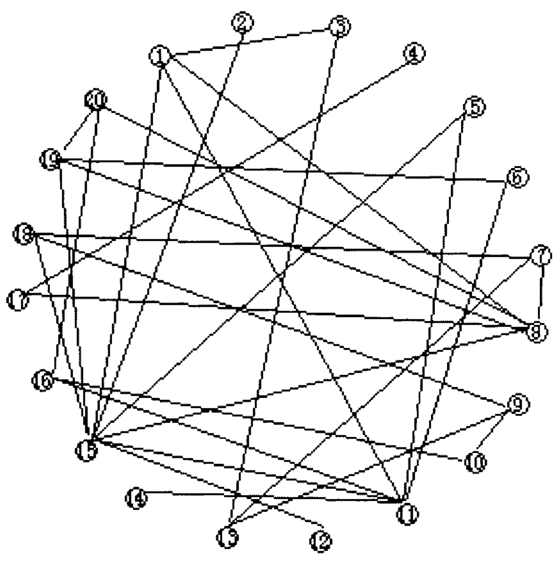

Fig. 6. A 20-vertex 30-edge MAX CUT problem

i

0	0	1	0	0	0	0	2	0	0	-1	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	2	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	5	0	0	0	0	3	0	0	0	0	4	0	0
2	0	0	0	0	0	5	0	0	0	0	0	0	0	1	0	2	0	-1	1
0	0	0	0	0	0	0	0	0	5	0	0	3	0	0	0	0	4	0	0
0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	1	0	0	0	0
-1	0	0	0	1	-1	0	0	0	0	0	0	0	2	-1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0
0	0	3	0	0	0	3	0	3	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
1	5	0	0	2	0	0	1	0	0	-1	5	0	0	0	0	0	3	4	0
0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1
0	0	0	4	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	4	0	4	0	0	0	0	0	3	0	0	0	0	0
0	0	0	0	0	1	0	-1	0	0	0	0	0	0	4	0	0	0	0	2
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	2	0

j

Fig. 7. Fig.7 The matrix of weight on edge of Fig.6

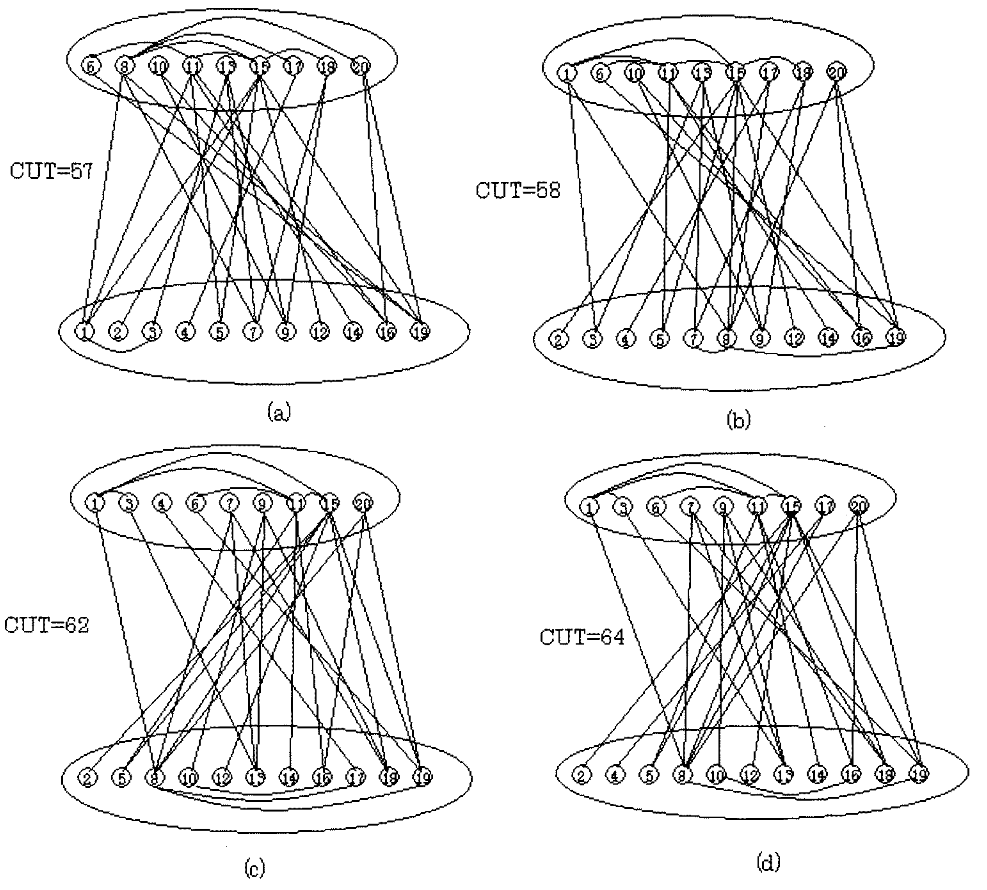

Fig. 8. The solution of Fig.6 (a) before learning (b) after first learning (c)after 4th learning (d)final solution of Fig.6 found by proposed algorithm

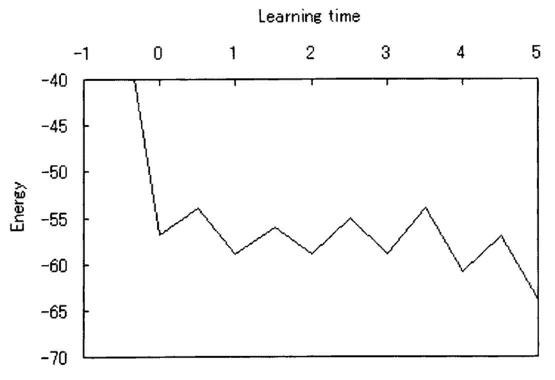

Fig. 9. The variation process of the energy during learning

Eq(7) and (8) and the gradient ascent learning phase uses the rules defined in Eqs(12) and (13). Simulations refer to parameter set at $A = 1.0$. In the experiments *learn_limit* is set to 20. The initial values of neurons are randomized in the range of 0.0 to 1.0. The temperature parameter T in Eq.(10) was set to 0.25

The first problem that we test is a graph with 20 vertices and 30 edges which is shown in Fig.6. The matrix of weights on edges is shown in Fig.7. Fig8 shows the results of a simulation on the MAX CUT problem of Fig.6, which illustrates a typical progressive intermediate partition of vertices during the Hopfield network updating phase (Phase 1) and the gradient ascent learning phase (Phase 2). Initially the Hopfield network converges to a time independent state (Fig. 8(a)). The value of CUT of this partition is 57. It is obviously not an maximum CUT. After the first, *4th* and *5th* learning, the value of CUT increases to 58, 62 and 64, and generates the partition of vertices (b), (c) and finally (d). In order to gain some insight into the optimization process, the energy can be studied by plotting the energy as time. Fig.9 shows the variation of the energy during the Hopfield network updating and the gradient ascent learning. It can be seen from the figure that the *2nd* and *3rd* learnings do not lead to a movement out of local minimum, and therefore no new partition of vertices is generated. The first, *4th* and *5th* learning do yield a movement out of local minimum, thus resulting in new partition of vertices (Fig.8 (b), (c) and (d)). To see if our solution is a global optimal solution, we use exhaustive search to this graph on the full searching space, and found that our solution is a global optimal solution. In order to widely verify the proposed algorithm, we also test the

Table 1. Comparison with LORENA.

No.vertex	Probability	No.edges	LORENA			Proposed algorithm			
			Av Cut	Av Time(S)	Best Cut	Av Cut	Av Time(S)	Av Learning	Best Cut
20	0.05	10	21.2	0.02	23	24	0.01	2.3	24
20	0.15	30	61.5	0.01	63	64	0.01	2.1	64
20	0.25	50	95.1	0.01	97	96.6	0.01	3.4	98
50	0.05	61	128.5	0.04	130	131	0.03	4.7	132
50	0.15	183	312.6	0.1	320	319.8	0.07	4.3	323
50	0.25	305	486.3	0.18	491	497.3	0.09	8.6	499
80	0.05	158	311.6	0.26	316	315.2	0.3	4	317
80	0.15	474	766.4	0.58	768	769.4	0.4	5.6	772
80	0.25	790	1192.5	0.85	1196	1198.7	0.33	5.3	1201
100	0.05	247	479.2	0.97	483	483.1	0.73	6	484
100	0.15	742	1163.9	0.56	1172	1175.4	0.4	3.7	1178
100	0.25	1235	1789.6	0.61	1798	1800.9	0.7	5.3	1802
150	0.05	558	982.5	0.94	987	989.1	0.81	12	992
150	0.15	1676	2480.8	0.58	2489	2493.7	0.47	3	2496
150	0.25	2790	3887	2.15	3888	3895	0.85	7.6	3898
200	0.05	995	1669.4	0.78	1675	1675.9	0.65	4.1	1680
200	0.15	2985	4278.8	0.8	4284	4294.5	0.87	3.2	4296
200	0.25	4975	6745.2	2.32	6748	6752	0.79	6.7	6755
250	0.05	1556	2493.5	1.02	2499	2502.8	0.45	3.4	2506
250	0.15	4668	6527.6	1.54	6534	6539.5	0.74	4.5	6541
250	0.25	7780	10217.7	1.98	10223	10228.1	1.26	6.3	10231
300	0.05	2242	3612.4	0.89	3615	3526.7	0.83	4.3	3530
300	0.15	6727	9138.5	1.45	9147	9162	1.36	5.6	9165
300	0.25	11212	14465.6	2.78	14491	14512.6	1.98	7.7	14516

Table 2. Comparison with the simulated annealing.

No.vertex	Probability	No.edges	SA			Proposed algorithm		
			Av Cut	Av Time(S)	Best Cut	Av Cut	Av Time(S)	Best Cut
20	0.05	10	23.76	0.59	24	24	0.01	24
20	0.15	30	63.35	0.81	64	64	0.01	64
20	0.25	50	96.96	0.49	98	96.6	0.01	98
50	0.05	61	130.13	0.92	132	131	0.03	132
50	0.15	183	314.27	0.61	323	319.8	0.07	323
50	0.25	305	495.51	0.51	499	497.3	0.09	499
80	0.05	158	314.15	2.29	317	315.2	0.3	317
80	0.15	474	760.82	1.34	770	769.4	0.4	772
80	0.25	790	1190.39	1.15	1200	1198.7	0.33	1201
100	0.05	247	479.84	2.91	482	483.1	0.73	484
100	0.15	742	1162.08	2.17	1175	1175.4	0.4	1178
100	0.25	1235	1783.98	2.21	1802	1800.9	0.7	1802
150	0.05	558	979.56	4.72	992	989.1	0.81	992
150	0.15	1676	2468.40	3.23	2491	2493.7	0.47	2496
150	0.25	2790	3854.71	3.47	3891	3895	0.85	3898
200	0.05	995	1655.60	6.45	1679	1675.9	0.65	1680
200	0.15	2985	4255.20	5.45	4285	4294.5	0.87	4296
200	0.25	4975	6686.20	5.33	6736	6752	0.79	6755
250	0.05	1556	2481.50	9.7	2506	2502.8	0.45	2506
250	0.15	4668	6447.70	8.90	6532	6539.5	0.74	6541
250	0.25	7780	10160.4	8.57	20215	10228.1	1.26	10231
300	0.05	2242	3492.88	14.23	3523	3526.7	0.83	3530
300	0.15	6727	9105.91	12.66	9143	9162	1.36	9165
300	0.25	11212	14440.3	12.73	14497	14512.6	1.98	14516

algorithm with a large number of randomly generated graphs [17] defined in terms of two parameters, n and p. The parameter n specifies the number of vertices in the graph; the parameter p, $0 < p < 1$, specifies the probability that any given pair of vertices constitutes an edge. Integer numbers are given randomly on edges as weight. The range for weight was from -1 to 5. To evaluate our results, we compare our results with the results of LORENA [10]. For each of instances, 100 simulation runs are performed. Information on the test graphs as well as all results are shown in Table 1. The results that we record for each graph are the average cut, average computational time and best cut produced by the LORENA [10], and by the proposed algorithm. The average learning epochs of 100 runs for every graphs are also shown in Table 1. From Table 1 we can see that the proposed parallel algorithm is superior to the LORENA [10] in terms of solution quality and computational time.

To evaluate the performance of the proposed method, simulated annealing is also executed on MAX CUT problems for comparison. We use the annealing algorithm given by Johnson et al [17]. Starting with a randomly generated assignment, it repeatedly picks a random variable, and computes the change (Δ) of energy when the state of that variable is flipped. If $\Delta \geq 0$, make the flip. Otherwise, flip the variable with probability $e^{-\Delta/T}$. We slowly decrease the temperature (T) from 3.0 to 0.00001. 100 simulation runs are performed for each graphs. To compare the proposed method with the simulated annealing, the average cut, average computational time and best cut produced by the proposed learning algorithm and simulated annealing are shown in Table 2. From this table, we can see that the proposed learning algorithm works better than the simulated annealing in terms of the solution quality and the computation time.

6. Conclusions

We have proposed a near-optimum parallel algorithm for solving optimization problems, and showed its effectiveness by simulation experiments. This parallel algorithm has two phases, the Hopfield network updating phase and the gradient ascent learning phase. In the first phase we implemented the Hopfield network for optimizing the energy function in state domain. In the second phase we intentionally increased the energy of the Hopfield network by modifying parameters in weight domain in a gradient ascent direction, thus making the network escape from local minimum. The proposed learning algorithm was tested on 2-variable quadratic polynomial and applied to the MAX CUT problems. Extensive simulations were performed and its effectiveness was confirmed.

7. Acknowledgment

This work was supported in part by the grant-in-aid for Science Research of the Ministry of Education, Science and Culture of Japan under Grant (C)(2)12680392.
(Manuscript received November 7, 2001, revised June 27, 2002)

References

(1) R.M. Karp: "Reducibility among combinatorial problems," Complexity of Computer Computations. pp.85-103, Plenum Press, New York (1972)
(2) K.Chang and D. Du: "Efficient algorithm for the layer assignment problem," *IEEE T. CAD* vol. 6, pp.67-78 (1987)
(3) F. Barahona, M. Grotschel, M. Junger, and G. Reinelt. "An application of combinatorial optimization to statistical physics and circuit layout design," *Operations Research*, vol.36 pp.493-513 (1988)
(4) M.R. Garey, D.S. Johnson, and L.J. Stockmeyer: "Some simplified NP-complete graph problem," Theor. Comput. Sci. Vol.1 pp.237-267 (1976)
(5) F. Hadlock: "Finding a maximum cut of a planar graph in polynomial time," SIAM J. Comp., vol.4 no.3 pp.221-225 (1975)
(6) S. Sahni and T. Gonzalez: "P-complete approximation problems" J. ACM, vol.23, p555-565 (1976)
(7) C.-P. Hsu, "Minimum-via topological routing," *IEEE T. CAD*, vol.2, pp.235-246 Oct (1983)
(8) F. Alizadeh: "Interior point method in semidefinite programming with applications to combinatorial optimization," *SIAM J. Optim.*, vol.5, pp.13-51 (1995)
(9) M. X. Goemans and D. P. Williamson: "Improved approximation algorithms for the maximum cut and satisfiability problems using semidefinite programming", *J. ACM*, vol.42, pp.1115-1145 (1995)
(10) A. Bertoni, P. Campadelli, and G. Grossi: "An Approximation Algorithm for the Maximum Cut Problem ans its Experimental Analysis", Proceedings of algorithms and experiments, Ternto, Italy, Feb 9-11, pp.137-143 (1998)
(11) J.J. Hopfield: "Neurons with graded response have collective computation properties like those of two-state neurons", Proc. Nat. Acad. Sci. U.S., Vol.81, pp.3088-3092 (1982)
(12) J.J. Hopfield and D.W.Tank: " 'Neural' computation of decisions in optimization problems", *Bio. Cybern.*, No.52, pp.141-152 (1985)
(13) M. A. Alberti, A. Bertoni, P. Campadelli, G. Grossi, and R. Posenato: "A neural algorithm for MAX-2SAT: Performance analysis and circuit implementation", *Neural Networks*, vol.10 no.3, pp.555-560 (1997)
(14) S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi: "Optimisation by Simulated Annealing", *Science*, Vol.220, pp.671-680 (1983)
(15) P. J. M. van Laarhoven and E. H. L. Aarts: Simulated Annealing: Theory and Applications. Kluwer, Dordrecht, (1988)
(16) N. Baba: "A Hybrid algorithm for finding the global minimum of error function of neural networks", Proc. Of Int. Joint Conderence on Neural Networks, vol.1, pp.585-588, Washington, D. C. (1990)
(17) D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, "Optimization by simulated annealing: An experimental evaluation; Part 1, graph partitioning", Operations Research, vol.37, no.6, pp.865-892 (1989)

Rong Long Wang (Student Member) received a B.S. degree from Hangzhe teacher's college, Zhejiang, China and an M.S. degree from Liaoning University, Liaoning, China in 1987 and 1990, respectively. Since 1990 he has been a lecturer in Benxi University, Liaoning, China. Now he is working toward the Ph.D degree at Toyama University, Japan. His main research interests are neural networks and optimization problems.

Zheng Tang (Non-member) received a B.S. degree from Zhejiang University, Zhejiang, China in 1982 and an M.S. degree and a D.E. degree from Tsinghua University, Beijing, China in 1984 and 1988, respectively. From 1988 to 1989, he was an Instructor in the Institute of Microelectronics at Tsinhua University. From 1990 to 1999, he was an Associate Professor in the Department of Electrical and Electronic Engineering, Miyazaki University, Miyazaki, Japan. In 2000, he joined Toyama University, Toyama, Japan, where he is currently a Professor in the Department of Intellectual Information Systems. His current research interests include intellectual information technology, neural networks, and optimizations.

Qi Ping Cao (Non-member) eceived a B.S. degree from Zhejiang University, Zhejiang, China and an M.S. degree from Beijing University of Posts and Telecommunications, Beijing, China in 1983 and 1986, respectively. From 1987 to 1989, she was an Assistant Professor in the Institute of Microelectronics at Shanghai Jiaotong University, Shanghai, China. From 1989 to 1990, she was a Research Student in Nagoya University, Nagoya, Japan. From 1990 to 2000, she was a Senior Engineer in Sanwa Newtech Inc., Japan. In 2000, she joined Tateyama Systems Institute, Japan. Her current research interests include multiple-valued logic, neural networks, and optimizations.