Paper

An Efficient Algorithm for Maximum Clique Problem Using Improved Hopfield Neural Network

Rong Long Wang* Student Member
Zheng Tang* Non-member
Qi Ping Cao** Non-member

The maximum clique problem is a classic graph optimization problem that is NP-hard even to approximate. For this and related reasons, it is a problem of considerable interest in theoretical computer science. The maximum clique also has several real-world applications. In this paper, an efficient algorithm for the maximum clique problem using improved Hopfield neural network is presented. In this algorithm, the internal dynamics of the Hopfield neural network is modified to efficiently increase exchange of information between neurons and permit temporary increases in the energy function in order to avoid local minima. The proposed algorithm is tested on two types of random graphs and DIMACS benchmark graphs. The simulation results show that the proposed algorithm is better than previous works for solving the maximum clique problem in terms of the computation time and the solution quality.

Keywords: maximum clique problem, Hopfield neural network, internal dynamics, NP-complete problem

1. Introduction

A clique of an undirected graph $G(V, E)$ with a vertex set V and an edge set E is a subset of V such that all pairs of vertices are connected by an edge in E. The maximum clique problem is to find a clique of maximum size of a graph G. The maximum clique problem is highly intractable. It is one of the first problems which have been proven to be NP-complete [1]. Moreover, even its approximations with a constant factor are NP-hard [2]. In particular, Hastad [3] states that if NP $\neq$ P then no polynomial time algorithm can approximate the maximum clique to within a factor of $N^{1/2-\varepsilon}$ for any $\varepsilon > 0$, where N is the number of nodes of the graph.

The maximum clique problem has many practical applications in science and engineering [4]. These include cluster analysis, information retrieval, mobile networks, computer vision, and alignment of DNA with protein sequences. Determining maximum clique is also very useful in circuit design. The problem is to create an optimal geometric layout for different chip hardwares, such as programmable logic array and CMOS transistors. Encountered by different real life applications, many researchers have widely studied this problem using different methods.

In 1974, Johnson [5] introduced two greedy algorithms to the maximum clique problem; one selecting a vertex connected to most vertices among candidates and adds it one-by-one to a clique until no addition is possible, and another one that selecting a vertex connected to least vertices and removing it one-by-one from a graph until a clique appears. Lecky et al. [6] proposed the modified greedy algorithm by selecting each vertex of a graph in turn as the initial vertex in a clique. Tomita and Fujii proposed three algorithms using the depth-first search and preprocessing procedure [7]. Balas and Yu proposed the algorithm using the properties of triangulated graphs and the relationship between cliques and vertex colorings [8]. Carraghan and Pardalos proposed the algorithm based on a partial enumeration [9]. Pardalos and Philips formulated the maximum clique problem as a linearly constrained indefinite quadratic global optimization problem [10]. Pardalos and Rodgers proposed the algorithm using the unconstrained quadratic zero-one programming formulation [11]. Soule and Foster created a genetic algorithm to search for maximum cliques in general graphs [12]. Marchiori proposed a Heuristic based genetic algorithm for the maximum clique problem, which consists of the combination of a simple genetic algorithm and a native heuristic algorithm [13].

For solving such combinatorial optimal problems, energy-descent optimization algorithms also constitute an important avenue. The Hopfield network algorithm [14] [15] is a typical energy-descent optimization algorithm. In 1995, Jagota proposed five energy-descent optimization algorithms using the Hopfield neural network for approximating the maximum clique problem [16]. Yamada et al. proposed the energy-descent optimization algorithm called "RaCLIQUE" based on the Boltzman machine method for approximating the maximum clique problem [17]. Funabiki and Nishikawa compared the performance of four promising energy-descent optimization algorithms [18].

In this paper, we introduce an improved Hopfield neu-

*Faculty of Engineering, Toyama University, Toyama-shi, Japan 930-8555
**Tateyama Systems Institute, Toyama-shi, Japan 930-0001

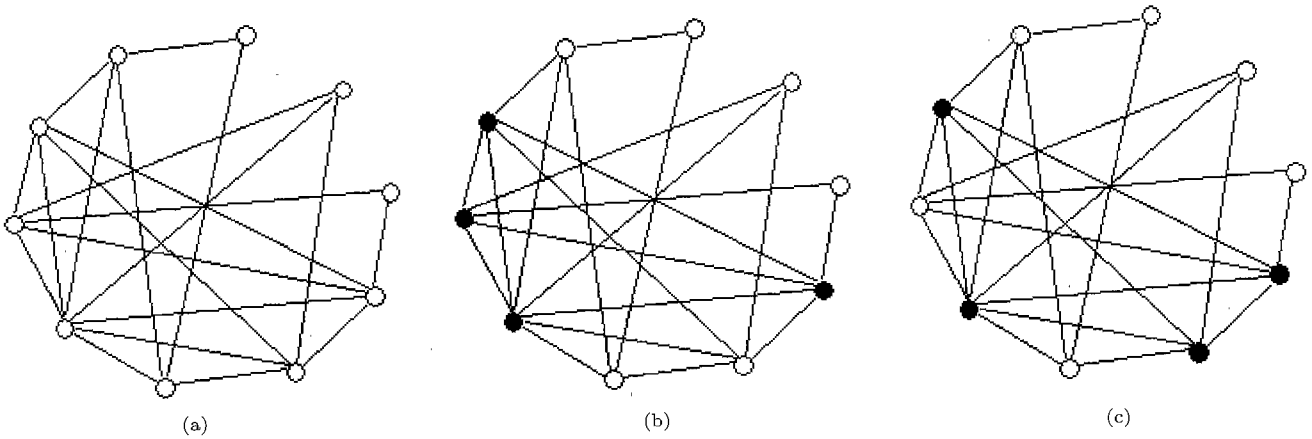


Fig. 1. (a) A 10-vertex 21-edge undirected graph. (b) first maximum clique consisted of the black vertices (c) second maximum clique consisted of the black vertices.

ral network algorithm for efficiently solving the maximum clique problem. In the proposed improved Hopfield neural network algorithm, the internal dynamics is modified to efficiently increase exchange of information between neurons and permit temporary increases in the energy function in order to avoid local minima. Two types of random graphs are simulated to verify the proposed algorithm. The performance of the proposed algorithm is compared with that of RaCLIQUE [16] and Funabiki's Binary Neural Network (BNN) [18]. The simulation results show that the proposed algorithm works well on finding a maximum or a better clique than RaCLIQUE [16] and BNN [18]. We also test the proposed algorithm on several DIMACS benchmark graphs [19]. The simulation results show that the proposed algorithm can find optimal solutions in these test graphs.

2. Maximum Clique Problem and Its Neural Representation

Let $G = (V, E)$ be an undirected graph, where $V = 1, \cdots, n$ is the set of vertices, and $E \subseteq V \times V$ is the set of edges. A clique of G is a subset of V in which every pair of vertices is connected by an edge. A clique is called maximal if no strict superset of it is also a clique. A maximum clique is a clique having largest cardinality (note that a maximal clique is not necessary a maximum one). Hence, the maximum clique problem consists of finding a clique of maximum size of a graph G. Figure 1(a) illustrates a 10-vertex 21-edge undirected graph. There exist two maximum cliques which are consisted of the black vertices shown in Fig1(b) and (c).

In general, the N-vertex maximum clique problem can be mapped onto the Hopfield neural network with N neurons. 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: goal term which is to realize the number of vertices in clique is maximum, and constrain term which is to satisfy that in clique every pair of vertices is connected by an edge. Suppose the output y_i of neuron $\#i$ represent vertex $\#i$ and constant d_{ij} represent the edge between vertex $\#i$ and vertex $\#j$, this optimization problem can be mathematically stated as finding the minimum of the following Hopfield network energy function:

$$E = A\sum_{i=1}^{N}\sum_{j\neq i}^{N}(1-d_{ij})y_iy_j - B\sum_{i=1}^{N}y_i \quad \cdots\cdots\cdots (1)$$

where the output y_i of neuron $\#i$ is 1 if vertex $\#i$ is in the clique, 0 otherwise and constant d_{ij} is 1 if vertex $\#i$ and vertex $\#j$ is connected by an edge, 0 otherwise. A and B are constant coefficients.

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

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}w_{ij}y_iy_j - \sum_{i=1}^{N}h_iy_i \quad \cdots\cdots\cdots (2)$$

where the weights and the thresholds of the Hopfield network become:

$$w_{ij} = -2A(1-d_{ij})(1-\delta_{ij}) \quad \cdots\cdots\cdots (3)$$

$$h_i = B \quad \cdots\cdots\cdots (4)$$

In Eq.(3), the notation δ_{ij} is 1 if $i = j$, 0 otherwise.

3. The Improvement of the Internal Dynamics of the Hopfield Network

For the Hopfield neural network, the motion equation is composed of the partial derivation term of the energy function as the gradient descent method.

$$dx_i/dt = \sum_{j\neq i}^{N}w_{ij}y_j + h_i \quad \cdots\cdots\cdots (5)$$

There are two kinds of modes to update the internal potential x_i of neuron:

(1) Time-independent one in which the internal potential of neuron at time $t+1$ does not directly depend on its value at time t [20][21].

$$x_i(t+1) = \frac{dx_i(t)}{dt} \quad \cdots\cdots\cdots (6)$$

(2) Time-dependent one in which the internal potential of neuron at time $t+1$ depends on its value at time t [22]:

$$x_i(t+1) = x_i(t) + \frac{dx_i(t)}{dt} \quad \cdots\cdots\cdots\cdots\cdots\cdots (7)$$

The neuron state y_i (output) is updated from x_i using a non-linear function called neuron model. The following two neuron models have been used for optimization problems:

(1) The McCulloch-Pitts neuron model [23]

$$y_i = \begin{cases} 1 & if\ x_i > 0 \\ 0 & otherwise \end{cases} \quad \cdots\cdots\cdots\cdots\cdots\cdots (8)$$

(2) The sigmoid function.

$$y_i = 1/(1 + e^{(-x_i/T)}) \quad \cdots\cdots\cdots\cdots\cdots\cdots (9)$$

We now propose a new method of modifying the internal dynamics of the Hopfield neural network to efficiently increase exchange of information between neurons and permit temporary increases in the energy function in order to avoid local minima. The motivation for this modification is that:

(1) In the time-independent updating mode of the internal potential, if the McCulloch-Pitts neuron model is used to update the neuron state, energy of the network decrease at each updating step. Thus, the network will attempt to take the best path to the nearest minimum, whether global or local. But if a local minimum is reached, the network will fail to update. Moreover, it is usually difficult for the network to get out of the local minimum and find the global minimum.

(2) Also in the time-independent updating mode of the internal potential, if the sigmoid function is used as input/output function, because the neuron state at time $t+1$ does not directly depend on its value at time t, this kind of updating mode of the internal potential cannot guarantee energy of the network decrease at each updating step. The proof is given in Appendix. Thus, the network cannot always converge to a stable state.

(3) In the time-dependent updating mode of the internal potential, the Hopfield neural network converges to the first local minimum it encounters and fail to update. Particularly, in this updating mode, the internal potential $x_i(t)$ is updated according to Eq.(7) unconditionally. The internal potential $x_i(t)$ may become so large or so small that the neuron state is insensitive to its internal potential. Once it happens, the convergence will become very slow and the states of network may be very difficult to be changed. The solution formed under this condition is less likely to have high quality.

In order to overcome the above difficulty, we modify the updating mode of the internal potential as following:

$$x_i(t+1) = \alpha_i(y_i, t) \cdot x_i(t) + \frac{dx_i(t)}{dt} \quad \cdots\cdots\cdots (10)$$

where $0 \leq \alpha_i(y_i, t) \leq 1$. This modified internal dynamic behavior means that the internal potential change in any neuron is now controlled by a new parameter $\alpha_i(y_i, t)$ which represents the stabilization of neuron $\#i$. When the neuron state is far from 0 and 1 or the network is in the initial stage (far from stable state), the stabilization of neuron is very low ($\alpha_i(y_i, t)$ is near 0). Thus, the internal potential of neuron ($x_i(t+1)$) is mainly decided by the weight states of other neurons (second term of Eq.(10)). Clearly, this internal dynamic behavior is similar to that in time-independent mode. As time proceeds and the neuron state approaches to 0 or 1, the stabilization of neuron increases. Finally the stabilization of neuron ($\alpha_i(y_i, t)$) is near 1, and the internal dynamic behavior of neuron ($x_i(t+1)$) tends toward that in time-dependent updating mode. This kind of variation behavior of the internal potential is more similar to that of actual neuron than that in the simple time-independent and simple time-dependent modes. This variation of the internal potential may cause an efficient exchange of information between neurons and result in better performance for the network. In this improved Hopfield neural network, the sigmoid function is used to update the neuron state. We define the parameter $\alpha_i(y_i, t)$ as following:

$$\alpha_i(y_i, t) = 1 - e^{-\frac{(0.5 - y_i(t))^2}{\varepsilon} \cdot \frac{t}{\lambda}} \quad \cdots\cdots\cdots\cdots (11)$$

where y_i is the state of neuron $\#i$, t is updating iteration, ε and λ are constant which can decide the growing speed of neuron and the convergent speed of network. The smaller these constant are, the faster the network converges to a stable state. For practical purpose we choose these constants that are as smaller as possible. This offers the most rapid convergence. But too smaller values will cause the network fall into local minima easily. In general, λ is selected at or near maximum number of iteration step allowed by user. About the value of ε, because at stable state, y_i is at or near 0 or 1, from Eq.(11) we can see that ε must be selected at a positive value smaller than 0.25. Although the optimal value of ε sometimes depends on the size of problem, in our simulations we found the ε around 0.1 worked very well in the maximum clique problem.

We have described the modification of the internal dynamics of the Hopfield network and analyzed the internal dynamics behavior of neuron. This kind of dynamics behavior of neuron can guide the network avoids some local minima. In the initial stage, the internal potential of each neuron is mainly decided by the weight states of other neurons. In other words, in this stage the network has the similar nature with the network working in time-independent mode. Furthermore due to the sigmoid function is used as input/output function, we can know that while descent of the energy function is always permitted, ascent of the energy function is permitted temporary in this stage. The ascent of the energy function can make the network avoids some local minima. Thus, we can say that the improved Hopfield neural network provides a mechanism for avoiding local minima.

Table 1. Simulation results on p-random graphs.

Notes	Edges	Probability	Propose Method		BNN		RaCLIQUE	
			Time(S)	Clique	Time(S)	Clique	Time(S)	Clique
100	2475	0.5	0.11	9	0.23	9	0.87	9
100	3960	0.8	0.12	20	0.4	19	0.68	19
100	4455	0.9	0.11	31	0.51	30	0.59	30
200	9950	0.5	0.47	12	9.65	10	6.21	11
200	15920	0.8	0.48	25	10.01	21	10.09	21
200	17910	0.9	0.45	40	12.18	35	9.41	39
300	22425	0.5	1.28	12	45.63	10	26.6	11
300	35880	0.8	1.35	28	39.15	25	28.34	27
300	40365	0.9	1.25	46	56.45	45	26.71	46
500	62375	0.5	3.94	12	95.31	10	123.03	11
500	99800	0.8	3.38	31	115.01	29	127.64	31
500	112275	0.9	3.86	56	101.76	50	126.19	56

Table 2. simulation results on k-random clique graphs ($k = 20$).

Notes	Edges	Probability	Propose Method		BNN		RaCLIQUE	
			Time(S)	Clique	Time(S)	Clique	Time(S)	Clique
200	16291	0.82	0.49	94	3.34	94	6.26	94
200	17453	0.87	0.51	103	4.36	101	7.02	102
200	15578	0.78	0.57	96	14.55	94	7	94
200	14114	0.71	0.54	86	5.07	83	6.64	80
200	16927	0.85	0.49	96	11.81	95	6.96	93
200	17973	0.9	0.50	113	6.33	111	6.82	111
200	13822	0.69	0.58	88	10.23	88	6.87	88
200	16100	0.81	0.51	85	13.86	84	6.75	84
200	14704	0.73	0.54	88	9.56	83	6.81	83
200	15728	0.79	0.54	95	8.07	95	7.54	94
400	65030	0.81	3.14	179	16.11	175	70.87	177
400	72140	0.9	2.58	192	7.45	192	76.95	188
400	68327	0.86	2.62	180	14.01	177	78.12	176
400	70425	0.88	2.51	194	19.37	192	73.5	190
400	67527	0.85	2.72	185	9.87	175	70.08	181
400	73417	0.92	2.35	198	19.03	197	78.03	196
400	65518	0.82	2.65	199	15.11	199	77.15	197
400	71376	0.89	2.46	188	13.02	188	72.45	185
400	62997	0.79	1.99	187	14.31	187	71.33	187
400	62481	0.78	2.67	180	16.02	180	71.05	180

4. Algorithm

In generally, the algorithm procedure of solving the maximum clique problem of an N-vertex graph using the proposed improved Hopfield neural network in asynchronous mode can be described as following:

(1) Set constant A, B, T, ε and λ, and set iteration step t=1.

(2) Randomly initialize the internal potential x_i for $i = 1, \cdots, N$.

(3) Update the neuron state y_i for $i = 1, \cdots, N$ using sigmoid function (Eq.(9)).

(4) Set $loop\_time = 1$.

(5) Loop until $loop\_time >= N$, where N is the number of vertices.

(a)Randomly select a neuron $\#i$.

(b) Use Eq.11 and Eq.10 to update the internal potential x_i of neuron $\#i$.

(c) Use Eq.9 to update neuron state y_i.

(d) Increment the $loop\_time$ by 1.

(6) Increment the t by 1.

(7) If the system reaches an equilibrium state go to step 8, else go to step 4.

(8) Compute the maximum clique using the stable state of the network.

5. Simulation Results

In order to assess the effectiveness of the proposed algorithm, extensive simulations were carried out over randomly generated graphs of various sizes on digital computer (PentiumIII 733MHz) in asynchronous. Simulations referred to parameter set at $A = 1.0$ $B = 1.0$. In the experiments, ε and λ were selected at 0.1 and 135 respectively. The temperature parameter T in Eq.(9) was set to 0.64. The RaCLIQUE [16] and Funabiki's Binary neural network (BNN) [18] were also executed for comparison. The reason to compare the proposed algorithm to RaCLIQUE and BNN is that in [18] Funabiki et al. indicated that RaCLIQUE and BNN were the efficient energy-descent algorithm in solving the maximum clique problem.

The first type of graphs we tested was p-random graph [16] where each of vertex pairs is connected by an edge with the probability of p independently to other edges. The size of this type of graph in our simulation were N=100, 200, 300, 500 and the probability were p=0.5, 0.8, 0.9. Because in general, the solution quality depend on the initial state selection of neuron input potential, all methods were executed by 100 simulation runs with different initial state on each graph. Because the objec-

tive of the maximum clique problem is to find a clique of maximum size of a graph, the best solution among 100 runs was used for the evaluation. It is worth to note that the rates to find the best solution of the proposed method were more than 90 percent for each graph. RaCLIQUE also had high rate ($> 90\%$) to find its best solution for each graph. Comparing with the proposed method and RaCLIQUE, BNN was very sensitive to the initial state selection of neuron input potential. The results of simulations were summarized in Table 1. The first three columns indicate the graph size (i.e., the number of notes and edges) and probability. The other columns indicate the average computation time to find the best cliques, and the best cliques using the proposed algorithm, BNN and RaCLIQUE. From this Table, we can see that the proposed algorithm outperformed BNN and RaCLIQUE for solving such kind of graphs.

The second type of graphs we tested was so called k-random clique graph [16]. This kind of graphs is generated by generating k cliques of varying size at random and taking their union. Such graphs have a wide range of clique sizes, much wider than in p-random graphs. This suggests that such graphs should be hard for approximating maximum clique and might separate the poor algorithm from the good ones. In our simulation we generated 20 such graphs to test the proposed algorithm. BNN and RaCLIQUE were also executed for comparison. The best solutions of 100 simulation runs on each graph were summarized in Table 2. From Table 2, we can note that the proposed algorithm was also the best algorithm for solving k-random clique graph.

We also evaluated experimentally the performance of the proposed algorithm on the DIMACS benchmark graphs [19]. These graphs provide a valuable source for testing the performance of algorithms for the maximum clique problem, because they arise from different areas of application. For example the Hamming graphs are from coding theory problem [24], the Keller graphs are based on Keller's conjecture on tilings using hypercubes [25]. We tested the proposed algorithm on Hamming8-4 and Keller4 [19]. The maximum clique size of Hamming8-4 and Keller4 generated by the proposed algorithm were 16 and 11, respectively which were as same as the best clique size which were found by all fifteen heuristic algorithms presented at the second DIMACS Challenge [19] and have been proven to be global optimality.

6. Conclusions

We have proposed an efficient algorithm for efficiently solving the maximum clique problem using the improved Hopfield neural network, and showed its effectiveness by simulation experiments. In the proposed neural network algorithm, the internal dynamics of the Hopfield network is modified to efficiently increase exchange of information between neurons and permit temporary increases in the energy function in order to avoid local minima. The proposed algorithm was tested on two types of random graphs and DIMACS benchmark graphs. The simulation results showed that the proposed algorithm is better than the previous work for solving the maximum clique problem in terms of the computation time and the solution quality. Because the proposed method is problem independent, the method can be applied to other combinatorial optimization problems. As the future works we plan to apply the proposed method to other NP-hard problems.

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 December 21, 2001, revised April 12, 2002)

References

(1) R.M. Karp: "Reducibility among combinatorial problems", *Complexity of Computer Computations*. pp.85-103, Plenum Press, New York (1972)

(2) U. Feiges, S. Goldwasser, S. Safra, L. Lovasz, and M. Szegedy: "Approximating clique is almost NP-complete", *In Proc. 32nd Annual IEEE Symposium on the Foundations of Computer Science (FOCS)*, pp.2-12 (1991)

(3) J. Hastad: "Clique is hard to approximate within n1-ε", *In Proc. 37th Annual IEEE Symposium on the Foundations of Computer Science (FOCS)* pp.627-636 (1996)

(4) P. M. Pardalos and J. Xue: "The maximum clique problem", *J. Global Opt.*, Vol.4, pp.301-328 (1994)

(5) D. S. Johnson: "Approximation algorithms for combinatorial problem", *J. Comput. Syst. Sci.*, Vol.9, pp.256-278 (1974)

(6) J. E. Lecky, O. J. Murphy, and R. G. Absher: "Graph theoretic algorithms for the PLA folding problem", *IEEE Trans. Computer-Aided Design*, Vol.8, No.9, pp.1014-1021 (1989)

(7) E. Tomita and T. Fujii: "Efficient algorithms for finding a maximum clique and their experimental evaluation", *Trans. IEICE*, Vol.J68-D, No.3, pp.221-228 (1985)

(8) E. Balas and C. S. Yu: "Finding a maximum clique in an arbitrary graph", *SIAM. J. Comput.*, Vol.15, No.4, pp.1054-1068 (1986)

(9) R. Carraghan and P. M. Pardalos: "An exact algorithm for the maximum clique problem", *Oper. Res. Lett.*, Vol.9, pp.375-382 (1990)

(10) P. M. Pardalos and A. T. Phillips: "A global optimization approach for solving the maximum clique problem", *Int. J. Comp. Math.*, Vol.33, pp.209-216 (1990)

(11) P. M. Pardalos and G. P. Rodgers: "A branch and bound algorithm for the maximum clique problem", Comp. Oper. Res., Vol.19, No.5, pp.363-375 (1992)

(12) T. Soule and J. A. Foster: "Using genetic algorthms to find maximum cliques", Technical Report LAL 95-12, Department of Computer Science, University of Idaho, (1995)

(13) E. Marchiori: "A simple heuristic based genetic algorithm for the maximum clique problem", *Proc. ACM Symp. Appl. Comput.*, pp.366-373 (1998)

(14) 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)

(15) J.J. Hopfield and D.W.Tank: " 'Neural' computation of decisions in optimization problems", *Biol. Cybern.*, No.52, pp.141-152 (1985)

(16) A. Jagota: "Approximating maximum clique with a Hopfield network", *IEEE Trans. Neural Networks*, Vol.6, No.3, pp.724-735 (1995)

(17) Y. Yamada, E. Tomita, and H. Takahashi: "A randomized algorithm for finding a near-maximum clique and its experimental evaluations", *Trans. IEICE*, Vol.J76-D-I, No.2, pp.46-53 (1993)

(18) N. Funabiki and S. Nishikawa: "Comparisons of energy-descent optimization algorithms for maximum clique problems", *Trans. IEICE* Vol.E79-A, No.4 (1996)

(19) D. Johnson and M. Trick (Eds.): "Cliques, Coloring, and Sat-

isfiability", AMS, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol.26 (1996)
(20) J. Mandziuk and B. Macukow: "A neural network designed to solve the N-Queens problem", *Boil. Cybern*, Vol 66, pp.375-379 (1992)
(21) J. Mandziuk: "Solving the N-Queens problem with a binary Hopfield-type network", *Boil. Cybern*, Vol 72, pp.439-445 (1995)
(22) Y. Takefuji and K. C. Lee: "Artificial neural networks for four-coloring map problems and K-colorability problems", *IEEE Trans. Circuits Syst.*, Vol.38, No.3, pp.326-333 (1991-3)
(23) W. S. McCulloch and W. H. Pitts: "A logical calculus of ideas immanent in nervous activity", *Bull. Math. Biophys*, Vol.5 pp.115-133 (1943)
(24) N. J. A. Sloane: "Unsolved problem in graph theory arising from the study of codes", Graph Theory Notes of New York XVIII, (1989)
(25) J. C. Lagarias and P. W. Shor: "Keller's cube-tiling conjecture is false in high dimensions", *Bullettin AMS*, Vol.27, No.2, pp.279-283 (1992)

Appendix

Theorem : If sigmoid function is used as input/output function, the time-independent updating mode of the internal potential cannot guarantee energy of the Hopfield network to decrease at each updating step.

Proof : We define energy at time t to be:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N} w_{ij}y_i y_j - \sum_{i=1}^{N} h_i y_i$$

The change in energy caused by the change in the states of neurons is:

$$\Delta E = -\frac{1}{2}\sum_{i=1}\sum_{j\neq i} w_{ij}[y_i(t+1)y_j(t+1) - y_i(t)y_j(t)] - \sum_{i=1}^{N} h_i[y_i(t+1) - y_i(t)]$$

Adding and subtracting $y_i(t+1)y_j(t)$, and simplifying, we get:

$$\Delta E = -\frac{1}{2}\sum_{i}\sum_{j\neq i} w_{ij}y_i(t+1)\Delta y_j(t+1) - \frac{1}{2}\sum_{i}\sum_{j\neq i} w_{ij}\Delta y_i(t+1)y_j(t) - \sum_{i} h_i\Delta y_i(t+1) \quad \cdots\cdots\cdots\cdots\cdots\cdots \text{(A1)}$$

Suppose that at time t, the state of the kth neuron is changed. define the change in the state of the ith unit as following:

$$\Delta y_i(t+1) = y_i(t+1) - y_i(t)$$

We have $\Delta y_k(t+1) \neq 0$ and $\Delta y_i(t+1) = 0$ for $i \neq k$. Thus, Eq.(A1) can be reduced to:

$$\Delta E = -\frac{1}{2}\sum_{i\neq k} w_{ik}y_i(t+1)\Delta y_k(t+1) - \frac{1}{2}\sum_{i\neq k} w_{ki}\Delta y_k(t+1)y_i(t) - h_k\Delta y_k(t+1) \quad \cdots\cdots\cdots\cdots\cdots\cdots \text{(A2)}$$

Because of the facts that $\Delta y_i(t+1) = 0$, i.e., $y_i(t+1) = y_i(t)$ for $i \neq k$ and $w_{ij} = w_{ji}$, Eq.(A2) can be rewritten to:

$$\Delta E = -\Delta y_k(t+1)[\sum_{i\neq k} w_{ki}y_i(t) + h_k] \quad \cdots\cdots \text{(A3)}$$

Using the Eq.(10) and Eq.(9), we have:

$$\Delta E = -\Delta y_k(t+1)x_k(t+1) \quad \cdots\cdots\cdots\cdots \text{(A4)}$$

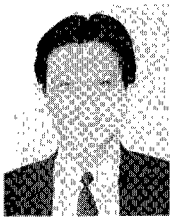
If the sigmoid function is used as input/output function, the following change in state of the kth neuron is possible:

$$y_k(t) > y_k(t+1) > 0.5 \quad \cdots\cdots\cdots\cdots\cdots\cdots \text{(A5)}$$

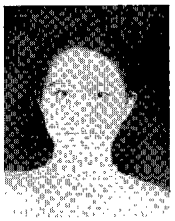
Using sigmoid function (Eq.(13)), from the Eq.(A5), we can know that $\Delta y_k(t+1) < 0$, and $x_k(t+1) > 0$. In this case, it is very clear that $\Delta E > 0$, in other words, energy of the network increase at this updating step.

Q.E.D.

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 an Instructor 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) received 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.