A Parallel Improvement Algorithm for the Bipartite Subgraph Problem

Kuo Chun Lee, Nobuo Funabiki, and Yoshiyasu Takefuji

Abstract— **Since McCulloch and Pitts proposed an artificial neuron model in 1943, several neuron models have been investigated. This paper proposes the first parallel improvement algorithm using the maximum neural network model for the bipartite subgraph problem. The goal of this NP-complete problem is to remove the minimum number of edges in a given graph such that the remaining graph is a bipartite graph. A large number of instances have been simulated to verify the proposed algorithm, with the simulation result showing that our algorithm finds a solution within 200 iteration steps and the solution quality is superior to that of the best existing algorithm. The algorithm is extended for the *K*-partite subgraph problem, where no algorithm has been proposed.**

I. Introduction

THE goal of an optimization problem is to minimize or maximize a cost function subject to constraints. A polynomial time algorithm for finding a global minimum solution has been in great demand on the solution quality and the computation time. However, most of practical problems are of the NP-complete or NP-hard type, where there are no polynomial algorithms. Heuristics or relaxation of constraints have been used to find reasonable solutions which are approximately close to the global minimum solution.

Because the advances in VLSI technology make it possible to build a large number of processors on a single chip, parallel computation has attracted many researchers in recent years. Unfortunately, the following problems must be solved in parallel computation [1]:

1) How is a given problem partitioned into small subproblems which are independently executed on different processors?
2) How is the termination condition determined in parallel programming?
3) What is the most suitable parallel architecture for the problem?
4) How are the executions in different processors synchronized?

There is no general way to satisfy all requirements simultaneously.

The inherent parallelism in the neural network provides a promising alternative for solving these problems. Since McCulloch and Pitts proposed the simplified artificial neuron model in 1943 [2], several neuron models have been investigated. Hopfield and Tank [3] were the first to propose a neural network for solving combinatorial optimization problems. The gradient descent method seeks the local minimum of the Liapunov energy function, E, which is given by the constraints and the objective function in the problem. It is proved that the state of the neural network system converges to the local minimum. Although the final goal is the global minimum solution, the near-optimum solution is practically acceptable in NP-complete problems if the convergence time is reasonable.

This paper presents a parallel improvement algorithm for the bipartite subgraph problem [4]. The goal of this NP-complete problem is to find a bipartite subgraph with the maximum number of edges of a given graph. An efficient parallel algorithm for solving a general bipartite subgraph problem has been desired for practical purposes [5]. Within our survey, existing algorithms for the bipartite subgraph problem can deal only with either a triangle-free graph with maximum degree 3 [5] or a weakly bipartite graph [6]. A sequential heuristic algorithm for the max cut problem, which is a similar NP-complete problem, was proposed by Hsu [7]. If every edge in a graph has the same weight in the max cut problem, it becomes equivalent to the bipartite subgraph problem. The performance of our algorithm is compared with that of Hsu's algorithm on a large number of instances.

II. Three Neuron Models and the Neural Network Approach

A large number of simple processing elements are used in the neural network. The processing element is called a neuron because it performs the function of a simplified biological neuron model. An output signal generated by one neuron propagates to inputs of several neurons through synaptic links. The linear sum of the weighted input signals determines the new state of the neuron.

The first mathematical neuron model was proposed by McCulloch and Pitts and was based on the biological computation model [2]. The input/output function is given by

$$V_i = f(U_i) = 1 \quad \text{if } U_i \geq 0; \qquad 0 \text{ otherwise} \tag{1}$$

where V_i and U_i are the output and the input of the ith neuron, respectively.

Hopfield and Tank [3] used the sigmoid neuron model, which is a differentiable, continuous, and nondecreasing function. The input/output function is given by

$$V_i = g(U_i) = \frac{1}{2}(1 + \tanh(\lambda U_i)) \tag{2}$$

where the parameter λ is a constant gain which changes the steepness of the sigmoid curve.

The maximum neural network is used in this paper, which consists of M clusters of N neurons. The total number of

Manuscript received January 11, 1991; revised August 23, 1991.

K. C. Lee is with the R&D Department, Cirrus Logic Inc., Fremont, CA 94538.

N. Funabiki is with the Systems Engineering Division, Sumitomo Metal Industries, Ltd., Osaka, Japan.

Y. Takefuji is with the Department of Electrical Engineering and Applied Physics, Case Western Reserve University, Cleveland, OH 44106.

IEEE Log Number 9103812.

processing elements is $M \times N$. One and only one neuron among N neurons with the maximum input per cluster always has nonzero output. The input/output function of the ith maximum neuron in cluster m is given by

$$V_{m,i} = 1 \quad \text{if}$$
$$U_{m,i} = \max\{U_{m,1}, \cdots, U_{m,N}\}; \qquad 0 \text{ otherwise.} \quad (3)$$

If there is more than one neuron with the maximum input in any cluster, the neuron with the smallest subscript has nonzero output. The outputs of the other neurons in the same cluster become zero.

The change of U_i is given by the motion equation of the ith neuron. The motion equation in the Hopfield neural network is given by [3]

$$\frac{dU_i}{dt} = -\frac{U_i}{\tau} - \frac{\partial E}{\partial V_i}. \quad (4)$$

The decay term $-U_i/\tau$ is known to disturb the convergence of the system because it increases the energy function, E, under certain conditions. Although many researchers use a large value τ to eliminate the undesirable effect, it is theoretically and empirically suggested that the decay term should be removed from the motion equation as shown in Appendix I. Neural network models without the decay term have been successfully used for several optimization problems [8]–[12].

III. Bipartite Subgraph Problem and Neural Representation

The bipartite subgraph problem is defined as follows [4]:

Bipartite Subgraph Problem

Instance: Graph $G = (V, E)$, positive $K \leq |E|$.

Question: Is there a subset $E' \subseteq E$ with $|E'| \geq K$ such that $G' = (V, E')$ is bipartite?

The goal of this problem is to remove the minimum number of edges from a given graph such that the remaining graph is a bipartite graph. The problem can be mathematically transformed into the following optimization problem.

Optimization Description:

$$\text{minimize} \sum_{x=1}^{N}\sum_{x\neq y}^{N}\sum_{i=1}^{2} \text{Connection}(x,y)V_{x,i}V_{y,i}$$
$$\text{subject} \sum_{i=1}^{2} V_{x,i} = 1 \quad \text{for } x = 1 \text{ to } N$$

where N is the number of vertices, $V_{x,i} \in \{1, 0\}$. Note that $\text{Connection}(x, y) = 1$ if there exists an edge between vertex x and vertex y; otherwise, it equals 0. If $V_{x,i} = 1$, vertex x belongs to cluster i, where the subscript i is either 1 or 2.

The N-vertex bipartite problem can be mapped onto the Hopfield neural network with $2 \times N$ neurons where it consists of N clusters of two neurons. When we follow the mapping procedure by Hopfield and Tank [3], the energy function with the McCulloch–Pitts neurons or the sigmoid neurons is given by

$$E = \frac{A}{2}\sum_{x=1}^{N}\sum_{i=1}^{2}\sum_{j\neq i}^{2} V_{x,i}V_{x,j} + \frac{B}{2}\left(\sum_{x=1}^{N}\sum_{i=1}^{2} V_{x,i} - N\right)^2 + \frac{C}{2}\sum_{x=1}^{N}\sum_{y\neq x}^{N}\sum_{i=1}^{2} \text{Connection}(x,y)V_{x,i}V_{y,i} \quad (5)$$

where $V_{x,i}$ is the output of the neuron representing vertex x in cluster i assignment. In a valid solution each vertex must be assigned to one of two clusters. In other words, either of $V_{x,1}$ or $V_{x,2}$ must be 1 for vertex x. This constraint is realized by the first and second terms in (5). However, because these two terms sometimes create invalid local minima in addition to valid local minima, this energy function cannot guarantee a valid solution even if the system converges to the local minimum. There is no systematic method to tune three coefficients except through a large number of experiments.

In order to alleviate the invalid local minimum problem, the first two terms can be replaced by the local constraint that each vertex be assigned to one of two clusters. The energy function is given by

$$E = \frac{A}{2}\sum_{x=1}^{N}\left(\sum_{i=1}^{2} V_{x,i} - 1\right)^2 + \frac{C}{2}\sum_{x=1}^{N}\sum_{y\neq x}^{N}\sum_{i=1}^{2} \text{Connection}(x,y)V_{x,i}V_{y,i}. \quad (6)$$

However, coefficients A and C still must be tuned and a valid solution cannot always be guaranteed.

A new energy function based on the maximum neural network is proposed in order to guarantee the valid solution and avoid the coefficient tuning problem. The energy function is given by

$$E = \frac{C}{2}\sum_{x=1}^{N}\sum_{y\neq x}^{N}\sum_{i=1}^{2} \text{Connection}(x,y)V_{x,i}V_{y,i} \quad (7)$$

where $V_{x,i} = 1$ if $U_{x,i} = \max\{U_{x,1}, U_{x,2}\}$; otherwise it equals 0. If $U_{x,1} = U_{x,2}$, the network sets $V_{x,1} = 1$ and $V_{x,2} = 0$. $U_{x,i}$ and $V_{x,i}$ are the input and the output of the xith neuron, representing vertex x in cluster i assignment. Because there is always one cluster assignment for each vertex, the converged state of the system is always a valid solution. Note that the energy function in (7) is exactly the same as the cost function in the bipartite subgraph problem for $C = 1$. The motion equation of the xith neuron for vertex x in cluster i assignment without the decay term is given by

$$\frac{dU_{x,i}}{dt} = -C\sum_{y\neq x}^{N} \text{Connection}(x,y)V_{y,i}. \quad (8)$$

The motion equation describes the synaptic links. The xith neuron is communicated with the yith neuron for vertex y in cluster i assignment where an edge exists between vertices x and y. The maximum neural network can guarantee the valid solutions. The proof of the convergence for the maximum neural network is given in Appendix II. The maximum neural network not only realizes the parallel computation but also solves the four problems of the parallel computation in Section I:

1) Each neuron can be assigned on a different processor.

TABLE I
COMPARISON BETWEEN HSU'S ALGORITHM AND THE MAXIMUM NEURAL NETWORK

Node size	5% Hsu's	5% Max Net	15% Hsu's	15% Max Net	20% Hsu's	20% Max Net	25% Hsu's	25% Max Net
10	2	2	6	6	6	6	9	10
20	8	8	24	25	26	28	41	41
30	19	20	49	50	52	56	76	80
40	36	36	90	90	96	99	140	143
50	50	53	128	135	143	149	208	216
60	78	80	191	195	210	218	296	304
70	102	107	246	254	282	282	410	417
80	125	132	311	330	363	367	536	536
90	158	162	390	405	445	459	676	683
100	185	195	478	494	553	564	823	836
110	218	227	582	595	657	686	1001	1005
120	262	267	691	695	790	801	1163	1190
130	308	317	803	819	940	943	1387	1391
140	354	371	929	947	1068	1080	1583	1593
150	403	412	1032	1060	1189	1206	1791	1818
160	462	476	1203	1218	1372	1393	2042	2064
170	512	520	1325	1348	1515	1552	2278	2314
180	566	580	1476	1502	1699	1719	2571	2578
190	647	646	1652	1681	1900	1930	2890	2902
200	698	703	1824	1860	2086	2146	3166	3177
210	770	785	2002	2037	2290	2332	3505	3505
220	842	851	2197	2228	2540	2561	3803	3837
230	909	914	2376	2400	2725	2756	4126	4193
240	975	1000	2584	2603	2954	3003	4511	4527
250	1083	1088	2817	2854	3213	3255	4859	4917
260	1147	1168	3018	3049	3461	3498	5242	5272
270	1223	1237	3239	3276	3753	3777	5602	5692
280	1287	1342	3486	3518	4010	4066	6051	6076
290	1426	1433	3735	3753	4319	4325	6487	6544
300	1494	1524	3978	4016	4548	4626	6929	6964

Each element in the table represents the number of embedded edges in a solution.

2) The algorithm can be terminated in the equilibrium state where no neuron changes the output.
3) The algorithm can be implemented either on a sequential machine or on a parallel machine with maximally $2N$ processors.
4) The algorithm does not require a rigorous synchronization procedure.

The maximum neural network improves the solution quality in a parallel computation until no further improvement can be achieved.

IV. PARALLEL ALGORITHM WITH THE MAXIMUM NEURAL NETWORK

The parallel algorithm is implemented in C on DEC3100 and in Turbo Pascal on a Macintosh SE/30. the following procedure describes the parallel algorithm based on the first-order Euler method.

0) Set $t = 0$.
1) Randomly generated numbers are assigned to the initial values of $V_{x,i}(t)$ for $x = 1$ to N and $i = 1$ to 2, where N is the number of vertices.
2) Evaluate values of $V_{x,i}(t)$ based on the maximum function for $x = 1$ to N and $i = 1$ to 2.

$$V_{x,i}(t) = 1 \quad \text{if } U_{x,i}(t) \geq U_{x,j}(t) \quad \text{for } i < j, \qquad 0 \text{ otherwise.}$$

3) Use the motion equation in (8) to compute $\Delta U_{x,i}(t)$:

$$\Delta U_{x,i} = -\sum_{y \neq x}^{N} \text{Connection}(x,y) V_{y,i}.$$

4) Compute $U_{x,i}(t)$ based on the first-order Euler method:

$$U_{x,i}(t+1) = U_{x,i}(t) + \Delta U_{x,i}(t) \quad \text{for } x = 1 \text{ to } N \quad \text{and} \quad i = 1 \text{ to } 2.$$

5) If the system reaches an equilibrium state, stop the procedure; otherwise increment t by 1 and go to step 2.

We have tested the algorithm with 1000 randomly generated examples of up to 300-vertex graph problems. We have used exhaustive search up to 30-vertex graph problems on the $O(10^9)$ searching space, where it is shown that the algorithm can find the optimum solution. Table I shows a comparison of the solution quality between Hsu's algorithm and our algorithm where the 5%, 15%, 20%, and 25% density graph problems with up to 300 vertices have been examined. Note that the density of an N-vertex graph is defined by the ratio between the number of given edges and $N(N-1)/2$. For each of the instances, 100 simulation runs were performed. Table I shows that the proposed algorithm can find a better solution than Hsu's algorithm in any problem. The simulation result also shows that the algorithm always converged to a

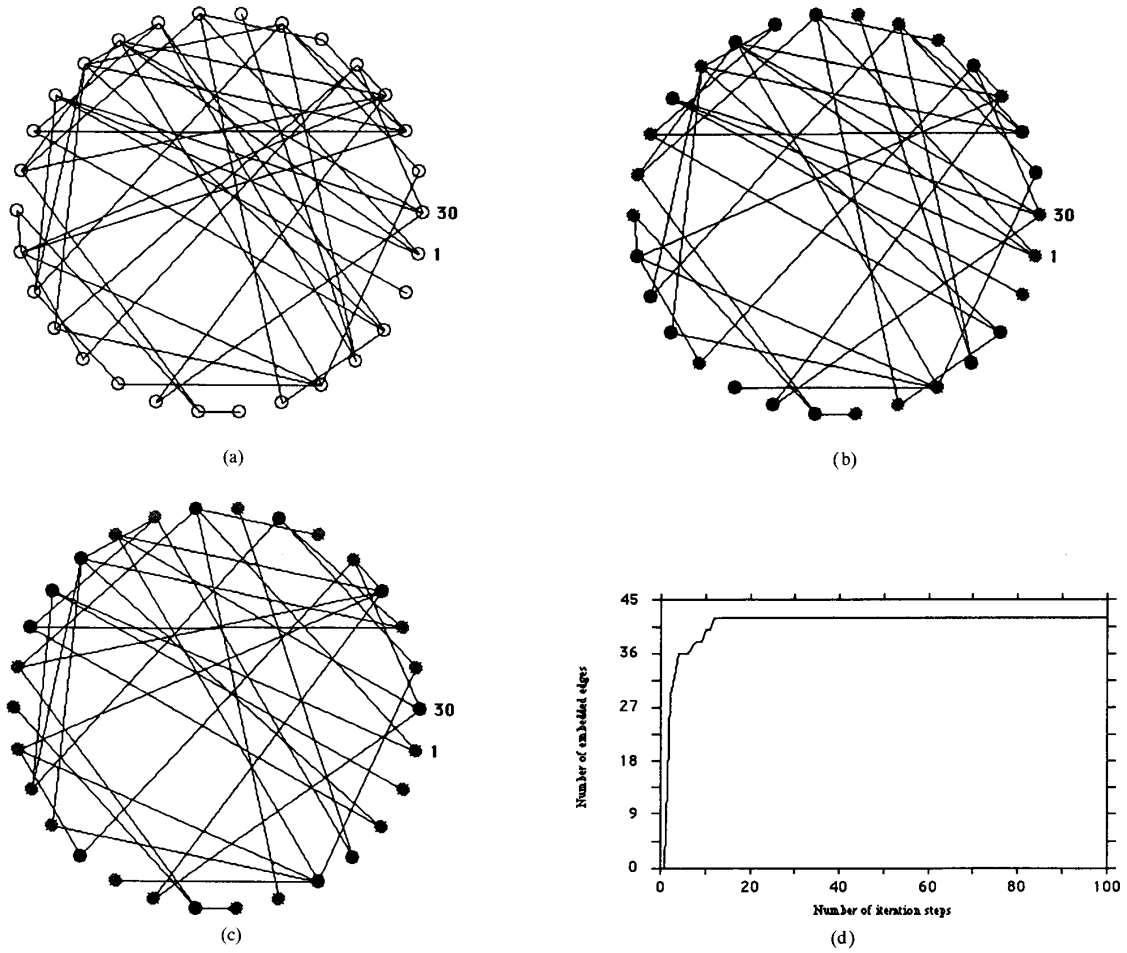

Fig. 1. Thirty-vertex bipartite subgraph problem. (a) The original graph with 50 edges. (b) Hsu's solution with 38 edges remaining. (c) One of the maximum neural network solutions with 42 edges embedded. (d) Relationship between number of embedded edges and the number of iteration steps.

solution within 200 iteration steps, which suggests that the computation time is not sensitive to the problem size. The proposed parallel algorithm is superior to Hsu's algorithm in terms of solution quality and computation time.

Fig. 1(a) shows the original graph of the 30-vertex graph problem. Parts (b) and (c) of the figure show the solutions by Hsu's algorithm and by our algorithm, respectively, where black circles and gray circles represent two disjoint sets for vertices. Fig. 1(d) shows the typical transition pattern of the number of edges embedded in a solution by our algorithm. Figs. 2 and 3 show, respectively, a comparison of the solution quality of the two algorithms and the frequency distribution of the convergence by our algorithm in the 30-vertex graph problem.

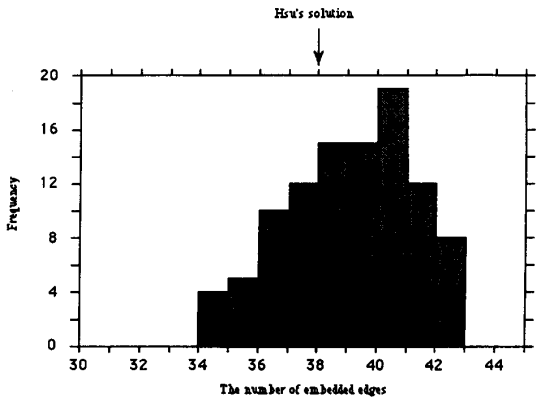

Fig. 2. The relation between the frequency and the number of embedded edges for the problem in Fig. 1.

V. The K-Partite Subgraph Problem

Another advantage of the proposed algorithm is that it can be easily applied to the K-partite subgraph problem. Unfortunately, no algorithm for the K-partite subgraph problem has been reported. The K-partite graph is a graph where the vertices are partitioned into K-disjoint sets and no edge exists between two vertices in the same set. The goal of the K-

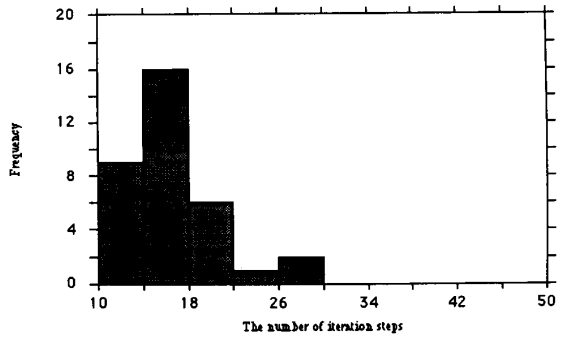

Fig. 3. The distribution of the convergence iteration steps for the problem in Fig. 1.

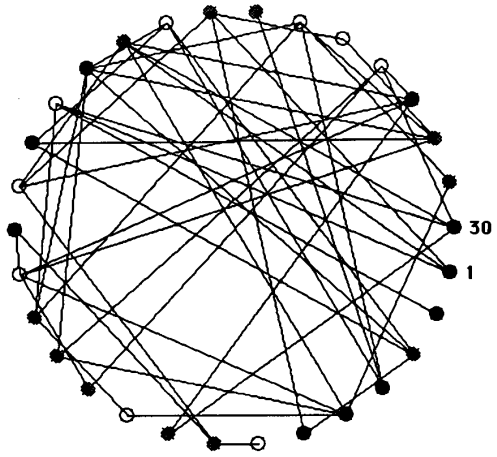

Fig. 4. One of the tripartite subgraph solutions with 50 edges embedded.

partite subgraph problem is to remove the minimum number of edges for a given graph such that the remaining graph is a K-partite graph. The bipartite subgraph is a special case of the K-partite subgraph problem ($K = 2$).

The energy function of the K-partite subgraph problem is given by

$$E = \frac{C}{2} \sum_{x=1}^{N} \sum_{y \neq x}^{N} \sum_{i=1}^{K} \text{Connection}(x, y) V_{x,i} V_{y,i} \qquad (9)$$

and the motion equation for the xith neuron of vertex x in cluster i assignment is given by

$$\frac{dU_{x,i}}{dt} = -C \sum_{y \neq x}^{N} \text{Connection}(x, y) V_{y,i}. \qquad (10)$$

Note that the (10) is the same as (8). The parallel algorithm in Section IV can be used for the K-partite subgraph problem where the cluster size is to be changed from 2 to K. Figs. 4 and 5 show the 3-partite and 4-partite subgraph solutions for the 30-vertex graph problem in Fig. 1 where all the edges are embedded.

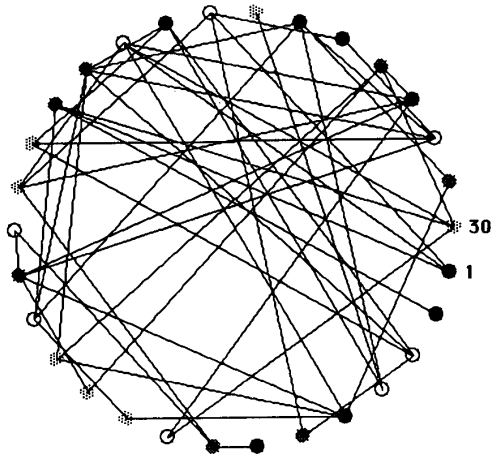

Fig. 5. One of the quadripartite subgraph solutions with 50 edges embedded.

VI. Conclusion

This paper has presented a parallel improvement algorithm for the bipartite subgraph problems based on the maximum neural network. The simulation result shows that the algorithm finds better solutions than the best existing algorithm. With the advantage of the parallelism the algorithm always finds a near-optimum solution in a nearly constant time. The algorithm can be easily applied to the K-partite subgraph problem, where no algorithm has yet been proposed.

Appendix I
Removal of Decay Term

In the Hopfield neural network the symmetric conductance matrix W with zero diagonal elements must be used to guarantee the convergence to the local minimum. The Liapunov energy function, E, is given by

$$\begin{aligned} E(t) &= -\frac{1}{2} \sum_i \sum_j w_{ij} V_i(t) V_j(t) - \sum_i I_i V_i(t) \\ &\quad + \frac{1}{\lambda} \sum_i \frac{1}{R_i} \int_0^{V_i(t)} g^{-1}(V(t))\, dV(t) \\ &= E_D(t) + \frac{1}{\lambda} \sum_i \frac{1}{R_i} \int_0^{V_i(t)} g^{-1}(V(t))\, dV(t) \end{aligned} \qquad (A1)$$

where $E_D(t) = E_D(V_1(t), V_2(t), \cdots, V_n(t))$. The motion equation of the ith neuron is given by

$$\frac{dU_i(t)}{dt} = \sum_j w_{i,j} V_j(t) + I_i - \frac{U_i(t)}{\tau} = -\frac{\partial E_D(t)}{\partial V_i} - \frac{U_i(t)}{\tau} \qquad (A2)$$

where τ is a time constant. For a given problem, the energy function, E, is constructed by the necessary and sufficient constraints and the cost function. The energy function is

mapped onto the Hopfield neural network. Note that the energy function is not unique for a problem. The energy function determines synaptic strengths w_{ij} between neurons and the constant bias I_i in (A2). Hopfield proved that the function E is a Liapunov function based on the motion equation in (A2). Because of the last term,

$$\frac{1}{\lambda}\sum_i \frac{1}{R_i}\int_0^{V_i(t)} g^{-1}(V(t))\,dV(t),$$

in (A1), a mapping becomes very difficult and the local minimum solution is displaced slightly by the constant λ. It is assumed that a very high gain is used to simplify the mapping. In the high-gain limit $\lambda \to \infty$, the Liapunov function becomes the function E_D without the last term in (A1), and the motion equation of the ith neuron is given by $dU_i(t)/dt = -\partial E_D(t)/\partial V_i(t)$. When the function E_D is used, the decay term $-U_i(t)/\tau$ must be excluded in the motion equation. If the gain λ is not infinite, the Liapunov function must include the last term,

$$\frac{1}{\lambda}\sum_i \frac{1}{R_i}\int_0^{V_i(t)} g^{-1}(V(t))\,dV(t),$$

in (A1). If the high-gain limit $\lambda \to \infty$ is used, the decay term $-U_i(t)/\tau$ must be excluded in the motion equation. Theorem 1 shows that the decay term increases the energy E_D under certain conditions. In other words, E_D is not the Liapunov function of the system.

Theorem 1: The decay term $-U_i(t)/\tau$ in the motion equation of the ith neuron increases the energy $E_D(t)$ when

$$\left|\sum_i \frac{dV_i(t)}{dt}\frac{U_i(t)}{\tau}\right| > \left|\sum_i \left(\frac{dV_i(t)}{dU_i(t)}\right)\left(\frac{dU_i(t)}{dt}\right)^2\right|$$

and if either ($U_i(t) > 0$ and $dV_i(t)/dt < 0$) or ($U_i(t) < 0$ and $dV_i(t)/dt > 0$) is satisfied.

Proof: Consider the derivatives of the energy E_D with respect to time t:

$$\frac{dE_D(t)}{dt} = \sum_i \frac{dV_i(t)}{dt}\frac{\partial E_D(t)}{\partial V_i(t)} = \sum_i \frac{dV_i(t)}{dt}\left(-\frac{U_i(t)}{\tau} - \frac{dU_i(t)}{dt}\right)$$

where $\partial E_D(t)/\partial V_i(t)$ is replaced by

$$\begin{aligned}&\left(-\frac{U_i(t)}{\tau} - \frac{dU_i(t)}{dt}\right)\\ &= -\sum_i \frac{dV_i(t)}{dt}\frac{U_i(t)}{\tau} - \sum_i \frac{dV_i(t)}{dt}\frac{dU_i(t)}{dt}\\ &= -\sum_i \frac{dV_i(t)}{dt}\frac{U_i(t)}{\tau} - \sum_i \left(\frac{dU_i(t)}{dt}\frac{dV_i(t)}{dU_i(t)}\right)\left(\frac{dU_i(t)}{dt}\right)\\ &= -\sum_i \frac{dV_i(t)}{dt}\frac{U_i(t)}{\tau} - \sum_i \left(\frac{dV_i(t)}{dU_i(t)}\right)\left(\frac{dU_i(t)}{dt}\right)^2.\end{aligned}$$

The first term,

$$-\sum_i \frac{dV_i(t)}{dt}\frac{U_i(t)}{\tau},$$

can be positive, negative, or zero. The second term,

$$-\sum_i \left(\frac{dV_i(t)}{dU_i(t)}\right)\left(\frac{dU_i(t)}{dt}\right)^2,$$

is always negative or zero because the output $V_i(t) = f(U_i(t))$ is a nondecreasing function. The following condition is true:

$$-\sum_i \frac{dV_i(t)}{dt}\frac{U_i(t)}{\tau} - \sum_i \left(\frac{dV_i(t)}{dU_i(t)}\right)\left(\frac{dU_i(t)}{dt}\right)^2 > 0$$

when

$$\left|\sum_i \frac{dV_i(t)}{dt}\frac{U_i(t)}{\tau}\right| > \left|\sum_i \left(\frac{dV_i(t)}{dU_i(t)}\right)\left(\frac{dU_i(t)}{dt}\right)^2\right|$$

and if one of the following condition is satisfied: ($U_i(t) > 0$ and $dV_i(t)/dt < 0$) or ($U_i(t) < 0$ and $dV_i(t)/dt > 0$). Under such a condition the derivatives of $E_D(t)$ with respect to time t must be positive: $dE_D(t)/dt > 0$. Therefore, the $-U_i(t)/\tau$ term increases the energy function under such conditions, which means $E_D(t)$ is not a Liapunov function for the system. Q.E.D.

Appendix II
Convergence Property of the Maximum Neural Network

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

Lemma 1: $dE/dt \le 0$ is satisfied under two conditions, such as $dU_i/dt = -\partial E/\partial V_i$ and $V_i = f(U_i)$, where $f(U_i)$ is a nondecreasing function.

Proof:

$$\frac{dE}{dt} = \sum_i \frac{dU_i}{dt}\frac{dV_i}{dU_i}\frac{\partial E}{\partial V_i} = -\sum_i \left(\frac{dU_i}{dt}\right)^2 \frac{dV_i}{dU_i}$$

where $\partial E/\partial V_i$ is replaced by dU_i/dt (condition 1) ≤ 0 where $dV_i/dU_i > 0$ (condition 2). Q.E.D.

Lemma 1 points out that the state of the system finally reaches an equilibrium state. The input/output function of a maximum McCulloch–Pitts neuron actually behaves like a binary neuron with dynamically changing the threshold. The function follows a nondecreasing function requirement. In Lemma 2, the convergence of the maximum neural network for the bipartite subgraph problem is given.

Lemma 2: $dE/dt \le 0$ is satisfied under two conditions, namely

$$\frac{dU_{x,i}}{dt} = -\frac{\partial E}{\partial V_{x,i}}$$

and

$$V_{x,i} = 1 \quad \text{if } U_{x,i} = \max\{U_{x,1}, U_{x,2}\}$$

and

$$U_{x,i} \geq U_{x,j} \quad \text{for } i > j; \qquad 0 \text{ otherwise.}$$

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

$$\begin{aligned}\frac{dE}{dt} &= \sum_x \sum_i \frac{dU_{x,i}}{dt} \frac{dV_{x,i}}{dU_{x,i}} \frac{\partial E}{\partial V_{x,i}} \\ &= -\sum_m \sum_i \left(\frac{dU_{x,i}}{dt}\right)^2 \frac{dV_{x,i}}{dU_{x,i}}\end{aligned}$$

where $\partial E/\partial V_{x,i}$ is replaced by $-dU_{x,i}/dt$ (first condition). Let $dU_{x,i}/dt$ be

$$\frac{U_{x,i}(t+dt) - U_{x,i}(t)}{dt}.$$

Let $dV_{x,i}/dU_{x,i}$ be

$$\frac{V_{x,i}(t+dt) - V_{x,i}(t)}{U_{x,i}(t+dt) - U_{x,i}(t)}.$$

Let us consider the term

$$\sum_i \left(\frac{dU_{x,i}}{dt}\right)^2 \frac{dV_{x,i}}{dU_{x,i}}$$

for each module separately. Let $U_{x,a}(t+dt)$ be the maximum at time $t+dt$ and $U_{x,b}(t)$ be the maximum at time t for the vertex x:

$$\begin{aligned}U_{x,a}(t+dt) &= \max\{U_{x,1}(t+dt),\ U_{x,2}(t+dt)\}, \\ U_{x,b}(t) &= \max\{U_{x,1}(t),\ U_{x,2}(t)\}.\end{aligned}$$

It is necessary and sufficient to consider the following two cases: 1) $a = b$ and 2) $a \neq b$. If the case 1) is satisfied, then there is no state change for the vertex x. Consequently,

$$\sum_i \left(\frac{dU_{x,i}}{dt}\right)^2 \frac{dV_{x,i}}{dU_{x,i}}$$

must be zero. If the case 2) is satisfied, then

$$\begin{aligned}&\sum_i \left(\frac{dU_{x,i}}{dt}\right)^2 \frac{dV_{x,i}}{dU_{x,i}} \\ &= \left(\frac{U_{x,a}(t+dt) - U_{x,a}(t)}{dt}\right)^2 \frac{V_{x,a}(t+dt) - V_{x,a}(t)}{U_{x,a}(t+dt) - U_{x,a}(t)} \\ &\quad + \left(\frac{U_{x,b}(t+dt) - U_{x,b}(t)}{dt}\right)^2 \frac{V_{x,b}(t+dt) - V_{x,b}(t)}{U_{x,b}(t+dt) - U_{x,b}(t)} \\ &= \left(\frac{U_{x,a}(t+dt) - U_{x,a}(t)}{dt}\right)^2 \frac{1}{U_{x,a}(t+dt) - U_{x,a}(t)} \\ &\quad + \left(\frac{U_{x,b}(t+dt) - U_{x,b}(t)}{dt}\right)^2 \frac{-1}{U_{x,b}(t+dt) - U_{x,b}(t)} \\ &= \frac{U_{x,a}(t+dt) - U_{x,a}(t)}{(dt)^2} - \frac{U_{x,b}(t+dt) - U_{x,b}(t)}{(dt)^2} \\ &= \frac{1}{(dt)^2}\{U_{x,a}(t+dt) - U_{x,a}(t) - U_{x,b}(t+dt) + U_{x,b}(t)\} \\ &= \frac{1}{(dt)^2}\{U_{x,a}(t+dt) - U_{x,b}(t+dt) + U_{x,b}(t) \\ &\quad - U_{x,a}(t)\} > 0\end{aligned}$$

because $U_{x,a}(t+dt)$ is the maximum at time $t+dt$ and $U_{x,b}(t)$ is the maximum at time t for the vertex x. The contribution from each term is either 0 or positive; therefore

$$\sum_i \left(\frac{dU_{x,i}}{dt}\right)^2 \frac{dV_{x,i}}{dU_{x,i}} \geq 0 \quad \text{and} \quad -\sum_x \sum_i \left(\frac{dU_{x,i}}{dt}\right)^2 \frac{dV_{x,i}}{dU_{x,i}} \leq 0 \Rightarrow \frac{dE}{dt} \leq 0.$$

Q.E.D.

REFERENCES

[1] P. J. Denning and W. F. Tichy, "Highly parallel computation," *Science,* vol. 250, pp. 1217–1222, Nov. 1990.

[2] W. S. McCulloch and W. H. Pitts, "A logical calculus of ideas immanent in nervous activity," *Bull. Math. Biophys.,* vol. 5, p. 115, 1943.

[3] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biol. Cybern.,* vol. 52, pp. 141–152, 1985.

[4] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York: W. H. Freeman, 1979.

[5] J. A. Bondy and S. C. Locke, "Largest bipartite subgraph in triangle-free graphs with maximum degree three," *J. Graph Theory,* vol. 10, pp. 477–504, 1986.

[6] F. Barahona, "On some weakly bipartite graphs," *Oper. Res. Lett.,* vol. 2, no. 5, pp. 239–242, 1983.

[7] C.-P. Hsu, "Minimum-via topological routing," *IEEE Trans. Computer-Aided Design,* vol. 2, pp. 235–246, Oct. 1983.

[8] Y. Takefuji and K. C. Lee, "A near-optimum parallel planarization algorithm," *Science,* vol. 245, pp. 1221–1223, Sept. 1989.

[9] Y. Takefuji and K. C. Lee, "A parallel algorithm for tiling problems," *IEEE Trans. Neural Networks,* vol. 1, pp. 143–145, Mar. 1990.

[10] Y. Takefuji, L. L. Chen, K. C. Lee, and J. Huffman., "Parallel algorithm for finding a near-maximum independent set of a circle graph," *IEEE Trans. Neural Networks,* vol. 1, pp. 263–267, Sept. 1990.

[11] Y. Takefuji and K. C. Lee, "Artificial neural networks for four-coloring map problems and k-colorability problems," *IEEE Trans. Circuits Syst.,* vol. 38, pp. 326–333, Mar. 1991.

[12] N. Funabiki and Y. Takefuji, "A parallel algorithm for spare allocation problems," *IEEE Trans. Reliability,* vol. 40, no. 3, pp. 338–346, 1991.