Workshop on Artificial Intelligence

Title: Competitive Neural Network to Solve Real-Time Scheduling

Abstract: The Hopfield neural network is widely applied to obtain an optimal solution in a variety of different scheduling applications. A competitive learning rule provides a highly effective means of attaining a sound solution and is capable of reducing the time-consuming effort of obtaining coefficients. Restated, the competitive mechanism simplifies the network complexity. This important feature is applied to Hopfield neural network to derive a new technique, i.e. competitive Hopfield neural network. This investigation utilizes the competitive Hopfield neural network to resolve a multiprocessor real-time scheduling problem with constrained times (execution time and deadline). Simulation results demonstrate that the competitive Hopfield neural network imposed on the proposed energy function ensures an appropriate approach of solving this class of real-time scheduling problems.

Authors: Ruey-Maw Chen* and Yueh-Min Huang**
Affiliation: * Department of Electronic Engineering
National Chin-Yi Institute of Technology, Taichung, Taiwan
**Department of Engineering Science
National Cheng Kung University, Tainan, Taiwan
Address: Department of Electronic Engineering,
National Chin-Yi Institute of Technology
No. 35, Lane 215, Section 1, Chung-Shan Rd., Taiping City,
Taichung County, 411 Taiwan, ROC
Tel: 886-4-23924505 ext 2232
Fax: 886-4-23920892
Email: raymond@chinyi.ncit.edu.tw

Keywords: Scheduling, Winner-take-all, Competitive learning rule, Hopfield neural network.

Competitive Neural Network to Solve Real-time Scheduling

Ruey-Maw Chen* and Yueh-Min Huang**

*Computer Center, National Chin-yi Institute of Technology

Taichung 411, Taiwan, ROC

**Department of Engineering Science, National Cheng-Kung University

Tainan 701, Taiwan, ROC

Scheduling, winner-take all, Competitive learning, Hopfield neural network

Abstract

Most scheduling problems have been demonstrated to be NP-complete problems. Many schemes have been presented to solve the optimization problems. Among which, the Hopfield neural network is commonly applied to obtain an optimal solution in various different scheduling applications, such as the Traveling Salesman Problem (TSP), a typical discrete combinatorial problem. Instead of using deterministic rule, a competitive rule provides a highly effective means of attaining a sound solution, can reduce the effort of obtaining coefficients. Restated, the competitive mechanism reduces the network complexity. This important feature is applied to the Hopfield neural network to derive a new technique, i.e. the competitive Hopfield neural network technique. This investigation employs the competitive Hopfield neural network to resolve a multiprocessor real-time scheduling problem with no process migration, time constraints (execution time and deadline). Simulation results demonstrate that the competitive Hopfield neural network imposed on the proposed energy function ensures an appropriate approach to solving this class of scheduling

problems.

1. **Introduction**

Various applications, such as communications, routing, industrial control, operations research, and production planning employ scheduling concepts. Most problems in these applications are confirmed to be NP complete or combinatorial problems. The traveling salesman problem (TSP) is a typical NP-complete problem, which seeks a tour that has a minimum cost; obtaining the optimal solution is very time consuming.

Various schemes have been developed for solving the scheduling problem. Linear programming is a widely used scheme for determining the cost function based on the specific scheduling problem. Willems and Rooda translated the job-shop scheduling problem into a linear programming format, and then mapped it into an appropriate neural network structure to obtain a solution [1]. Furthermore, Foo and Takefuji employed integer linear programming neural networks to solve the scheduling problem by minimizing the total starting times of all jobs with a precedence constraint [2]. Meanwhile, Zhang, Yan, and Chang proposed a neural network method derived from linear programming, in which preemptive jobs are scheduled based on their priorities and deadline [3]. Additionally, Cardeira and Mammeri investigated the multi-processor real-time scheduling by applying the *k-out-of-N* rule to a neural network [4]. Above investigations concentrated on the preemptive jobs (processes) executed on multiple machines (multiprocessor) with job transfer permitted by applying a neural network. Meanwhile, Hanada and Ohnishi [5] developed a parallel algorithm based on a neural network for preemptive task scheduling problems by allowing for a task transfer among machines. Park [6] embedded a classical local

search heuristic algorithm into the TSP optimization neural network. Most investigations have constructed the energy functions for scheduling problems in terms of timing constraint, preemption, and migration features associated with the process. In our previous work [7], the *display system* on an advanced avionics system, which is a real-time system, may consist of two or more display processors. Each processor is responsible for different tasks involving timing constraints, but must not allow task migration between processors. To facilitate the control of the pilot, all tasks must be properly scheduled to provide the pilot with useful and timely information. Otherwise, danger is inevitable. This study focuses mainly on resolving generic problems resembling the above situation. Restated, this study investigates a multiprocessor scheduling problem involving preemptive multitasking with timing constraints, but not allowing migration.

Hopfield and Tank led the way in using the neural network to solve optimization problems. The basic operation of the Hopfield neural networks [8] cooperatively decides neuron output state information based on the state input information from a community of neurons. Each neuron exchanges information with other neurons in the network. The neurons apply this information to drive the network to achieve convergence. The energy function used in the Hopfield neural network is an appropriate Lyapunov function. Many researchers have recently applied this method to various applications. Furthermore, [9] employed the Hopfield neural network with mean field annealing to solve the shortest path problem in a communication network. Similarly, our previous work [7] also solved a multi-constraint schedule problem for a multiprocessor system using the Hopfield neural network. Neural networks seldom include competitive architecture into the network for solving the most scheduling problems.

Imposing a competitive learning mechanism to update the neuron states in the Hopfield neural network is referred to as a competitive Hopfield neural network (CHNN). A competitive learning rule can not only reduce the time consumed in obtaining coefficients but also obtains an effective and sound solution. CHNN has been applied to various fields, such as image clustering processes and specific image segmentation. Chung, Tsai, Chen, and Sun [10] proposed a competitive Hopfield neural network for polygonal approximation. Similarly, Lin, Cheng, and Mao [11] also applied a competitive Hopfield neural network to demonstrate the promising results in medical image segmentation. Furthermore, Uchiyama and Arbib [12] used competitive learning as an efficient method in color image segmentation application. The winner-take-all rule employed by the competitive learning mechanism ensures that only one job is executed on a dedicated processor at a certain time, enforcing the *1-out-of-N* constraint to be held. The maximum neuron of the Hopfield neural network is the activated neuron. The monotonicity of the maximum neuron follows from the fact that the maximum neuron is equivalent to a MacCulloch-Pitts neuron with a dynamic threshold [13]. One advantage of real-time task scheduling is that we are interesting on meeting task timing constraints rather than optimizing a given target. Therefore, we are more interested on solving a constraints satisfaction problem.

In light of above developments, this work investigates the job schedule problem of a multiprocess on a real-time multiprocessor that includes timing constraints, via CHNN. An energy function designed to illustrate the timing constraints is proposed as in [7]. According to the CHNN, the scheduling problem is considered a minimization of an energy function. Our results demonstrate that the energy change is invariably negative when using formal mathematical derivations. Therefore, HNN can be employed to obtain the weighting and threshold matrices, and the competition process

can be applied to obtain the solution.

The rest of this paper is organized as follows. Section 2 derives the corresponding energy function according to the intrinsic constraints of the scheduling problem. After this, Section 3 reviews the CHNN, and translates the derived energy functions to the CHNN algorithm. Section 4 then gives mathematical proof of the convergence of the simplified energy function of CHNN. Next, Section 5 presents the simulation examples. Finally, Section 6 includes discussion and conclusions.

2. Energy function of the scheduling problem

Job-shop scheduling problems markedly differ among cases. Our scheduling problem domain considers N jobs (or processes) and M machines (or processors). The following assumptions are made regarding the problem domain. First, a job can be segmented and the execution of each segment is preemptive. Second, different segments of a job cannot be assigned to different machines, implying that no job migration is allowed between machines. Third, the execution time of each job is predetermined. These assumptions and constraints are quite feasible, as demonstrated by the display system depicted in the previous section. Given these assumptions, the preemptive processes with deadlines in a multiprocessor real-time system are interesting.

To resolve this scheduling problem, the energy function of the problem must first be derived. The energy function, which resembles the TSP, is transformed into a 3-D HNN (Fig. 1); then the *"optimization"* process searches for solutions satisfying a set of constraints such that the energy function is minimized or maximized. Herein, scheduling involves three variables: job, machine, and time. These three variables are depicted in Fig. 1. The "x" axis denotes the "job" variable, with *i* representing a

specific job with a range from 1 to N, the total number of jobs to be scheduled. Meanwhile, the “y” axis represents the “machine” variable, and each point *j* on the axis represents a dedicated machine from 1 to M, the total number of machines to be operated. Finally, the “z” axis denotes the “time” variable, with *k* representing a specific time, which should be less than or equal to T, the deadline of the job. Based on this definition of variables, a neuron indicated by state variable V_{ijk} is defined as representing whether or not job *i* is executed on machine *j* at a certain time *k.* The activated neuron $V_{ijk}=1$ denotes that the job *i* is arranged to execute on machine *j* at the time *k*; otherwise, $V_{ijk}=0$. Notably, each V_{ijk} corresponds to a neuron of the neural network.

The derived energy function representing the neural network system is as follows:

$$\begin{aligned} E = & \frac{C_1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{\substack{i1=1 \\ ,i1\neq i}}^{N} V_{ijk}V_{i1jk} + \frac{C_2}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{\substack{j1=1 \\ ,j1\neq j}}^{M}\sum_{k1=1}^{T} V_{ijk}V_{ij1k1} \\ & + \frac{C_3}{2}\sum_{i=1}^{N}(\sum_{j=1}^{M}\sum_{k=1}^{T} V_{ijk} - P_i)^2 + \frac{C_4}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}(\sum_{k=1}^{T} V_{ijk} - 1)^2 \\ & + \frac{C_5}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T} V_{ijk}G^2{}_{ijk}H(G_{ijk}) \end{aligned} \tag{1}$$

$$\text{where } \begin{cases} G_{ijk} = k - d_i \\ H(G_{ijk}) = \begin{cases} 1, if.G_{ijk} > 0 \\ 0, if.G_{ijk} \le 0 \end{cases} \end{cases} \text{,and}$$

Where $C_1, C_2, C_3, C_4,$ and C_5 refer to weighting factors; *N* denotes the total number of processes to be scheduled; *M* is the total number of machines to be operated; *T* represents the maximum time quantum of a process. These weighting factors, *N*, *M*, and *T*, are assumed to be positive constants herein.

The C_1 energy term confines a processor *j* to executing only one process, say *i* or *i*1, at a certain time *k.* This energy term has a minimum value of zero when satisfying this constraint. The C_2 energy term indicates that a process migration is prohibited, implying that process *i* runs on processor *j* or *j1*. This term also has a minimum value of zero. In the C_3 energy term, P_i denotes the total execution time required by process *i.* This energy term means that the time consumed by process *i* must equal P_i such that $\sum\sum V_{ijk} = P_i$, i.e. this term becomes zero. Additionally, the C_4 energy term is actually a supplemental constraint to prevent no process being executed on a specific processor at a certain time. Thus, this energy item falls to a minimum of zero when satisfying this constraint. The purpose of the C_5 energy term is to meet the deadline requirement of each process *i*, where d_i is the time limitation of process *i* and $H(G_{ijk})$ is the *Heavside function*. When a process is allocated with a run time that exceeds *d*, the energy term will exceed zero, and the energy value will grow exponentially with the associated time lag between d_i and *k*. Based on the above discussion, the derived energy function has a minimum value of zero when all constraints are satisfied.

Eq.(1) can be proved to be an appropriate Lyapunov function for the system under discussion, as Section 4 illustrates.

3. Competitive HNN

In this section, the discussed scheduling problem and its energy function are mapped onto the competitive HNN to obtain solutions as described.

Hopfield and Tank originally proposed the neural network, HNN, in [14]. Essentially, the HNN algorithm is based on the gradient technique, thus providing rapid convergence. The HNN also provides potential for parallel implementation. Based on dynamic system theory, the Liapunov function [14] [15] shown in Eq. (2).,

has verified the existence of stable states of the network system. This energy function representing the scheduling problem must be in the same format as the Lyapunov function, as below:

$$E = -\frac{1}{2}\sum_{x}\sum_{y}\sum_{z}\sum_{i}\sum_{j}\sum_{k} V_{xyz} W_{xyzijk} V_{ijk} + \sum_{i}\sum_{j}\sum_{k} \theta_{ijk} V_{ijk} \tag{2}$$

Where V_{xyz} and V_{ijk} denote the neuron states, W_{xyzijk} represents the synaptic weight indicating the interconnection strength among neurons, and θ_{ijk} is the threshold value representing the bias input of the neuron. Additionally, the HNN employs the deterministic rule to update the neuron state change. This deterministic rule is displayed in Eq. (3) below:

$$V_{ijk}^{n+1} = \begin{cases} 1, & if \quad Net_{ijk} > 0 \\ V_{ijk}^{n}, & if \quad Net_{ijk} = 0 \\ 0, & if \quad Net_{ijk} < 0 \end{cases} . \tag{3}$$

Meanwhile, Net_{ijk} represents the total input or net value of the neuron *(i, j, k)* obtained using the interconnection strength, W_{xyzijk}, and the bias input, θ_{ijk} displayed as follows:

$$Net_{ijk} = -\frac{\partial E}{\partial V_{ijk}} = \sum_{x}\sum_{y}\sum_{z} W_{xyzijk} V_{xyz} - \theta_{ijk} \tag{4}$$

Instead of applying conventional deterministic rules to update neuron states, competition among neurons is used to determine the winning neuron, i.e. the active neuron. As discussed previously, applying a winner-take-all learning mechanism to a Hopfield neural network is frequently referred to as a competitive Hopfield neural network, CHNN. The concept of the competitive Hopfield neural network resembles the special case of the *k-out-of-N* rule proposed by Carderia and Mammeri [4]. Restated, the competitive Hopfield neural network can be considered a *1-out-of-N* confine rule. The proposed competitive HNN neural network converges during

network evolutions, and Section 4 provides detailed proof of this convergence.

Since a processor can only execute one job at a time in subject scheduling problems, omitting the C1 and C4 energy terms from the HNN energy function (Eq. 1) yields a simplified energy function and satisfies the competitive constraint. Restated, the C1 and C4 energy terms are handled explicitly. The resulting energy function for CHNN is highlighted as follows:

$$E = \frac{C2}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{\substack{j1=1 \\ j1\neq j}}^{M}\sum_{k1=1}^{T}V_{ijk}V_{ij1k1} + \frac{C3}{2}\sum_{i=1}^{N}(\sum_{j=1}^{M}\sum_{k=1}^{T}V_{ijk} - P_i)^2 + \frac{C5}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T}V_{ijk}G_{ijk}^2 H(G_{ijk}) \tag{5}$$

The resulting energy function makes it apparent that this must be an appropriate Lyapunov function. Comparing Eq.(2) with Eq.(5) makes it possible to determine synaptic interconnection strength, W_{xyzijk}, and the bias input, θ_{ijk}, as illustrated below:

$$W_{xyzijk} = -C2 * \delta(x,i) * (1-\delta(y,j)) - C3 * \delta(x,i) \tag{6}$$

and

$$\theta_{xyz} = -C3P_i + \frac{C5}{2} * G^2 * H(G) \tag{7}$$

respectively, where

$\delta(a,b) = \begin{cases} 1 & if.a = b \\ 0 & if.a \neq b \end{cases}$ is the *Kronecker delta* function.

In the CHNN, a competitive winner-take-all rule is imposed to update the neuron states. The neurons on the same column of a dedicated processor at a certain time compete with one another to decide which specific job should be the winning neuron. The neuron that receives the maximum net value is the winning neuron. Accordingly, the output of the winner neuron is set to 1, and the output states of all the other

neurons on the same column are set to 0. The winner-take-all update rule of the neuron for the *ith* column is illustrated as follows:

$$V_{xjk} = \begin{cases} 1 & if \quad Net_{xjk} = \underset{i=1\sim N}{Max}\, Net_{ijk} \\ 0 & otherwise \end{cases}, \tag{8}$$

where Net_{xjk} is the maximum total neuron input, and is equivalent to the dynamic threshold on a MaCulloch-Pitts neuron [13].

4. Convergence of the CHNN

This section provides a mathematical proof of convergence in the CHNN for the investigated problem. The simplified CHNN energy function is shown in Eq. 5.The neuron (*i, j, k)* obtains the total input, i.e. net value, which is as follows (Eq. 9):

$$\begin{aligned} Net_{ijk} &= -\frac{\partial E}{\partial V_{ijk}} \\ &= -\frac{C2}{2}\sum_{\substack{j1=1 \\ j1\neq j}}^{M}\sum_{k1=1}^{T} V_{ij11k1} - C3(V_{ijk} - P_i) - \frac{C5}{2} G_{ijk}^2 H(G_{ijk}) \end{aligned} \tag{9}$$

For clarity, this energy function is separated into two parts, E_{mn} and E_{other}. The energy function can then be represented as follows (Eq. 10):

$$\begin{aligned} E &= \frac{C2}{2}\Big(\sum_{i=1}^{N}\sum_{\substack{j1=1 \\ j1=m}}^{M}\sum_{\substack{k1=1 \\ k1\neq n}}^{T} V_{imn}V_{ij1k1} + \sum_{i=1}^{N}\sum_{\substack{j=1 \\ j\neq m}}^{M}\sum_{\substack{k=1 \\ k\neq n}}^{T}\sum_{\substack{j1=1 \\ j1\neq j,\, j1\neq m}}^{M}\sum_{k1=1}^{T} V_{ijk}V_{ij1k1}\Big) \\ &+ \frac{C3}{2}\Big(\sum_{i=1}^{N}(V_{imn} - P_i)^2 + \sum_{i=1}^{N}\Big(\sum_{\substack{j=1 \\ j\neq m}}^{M}\sum_{\substack{k=1 \\ k\neq n}}^{T} V_{ijk} - P_i\Big)^2\Big) \\ &+ \frac{C5}{2}\Big(\sum_{i=1}^{N} V_{imn}G_{imn}^2 H(G_{imn}) + \sum_{i=1}^{N}\sum_{\substack{j=1 \\ j\neq m}}^{M}\sum_{\substack{k=1 \\ k\neq n}}^{T} V_{ijk}G_{ijk}^2 H(G_{ijk})\Big) \\ &= E_{mn} + E_{other} \end{aligned} \tag{10}$$

The above four V_{imn} terms related to the neuron represent a process *i* being executed at a specific processor *m* and specific time *n*. Restated, V_{imn} is the neuron on the i^{th} row (job) and n^{th} column (time) for the specific processor *m*. The first energy part, E_{mn},

is the summation of the energy term correlating with neuron state V_{imn}. The second part is the remainder, that is, E_{other}. Focusing on these terms at the $(t)^{th}$ iteration, the V_{lmn} is supposed to be the only active neuron (l,m,n) in the n^{th} column on processor m before updating, that is,

$$\begin{cases} V_{lmn}^{(t)} = 1 & , and \\ V_{imn}^{(t)} = 0 & , for \quad i \neq l. \end{cases}$$

Moreover, the neuron (q,m,n) at the $(t+1)^{th}$ iteration is supposed to be the only neuron activated with the largest total input value following updating, namely,

$$\begin{cases} V_{qmn}^{(t+1)} = 1 & , and \\ V_{imn}^{(t+1)} = 0 & , for \quad i \neq q. \end{cases}$$

The active neuron, based on the winner-take-all update rule, as in Eq.(10), is the one with the maximum net value on each column in each update, that is

$$Net_{qmn} = \underset{i=1\sim N}{Max}\, Net_{imn}.$$

This equation implies that

$$Net_{qmn} > Net_{lmn}, \tag{11}$$

where Net_{qmn} and Net_{lmn} are derived from Eq.(9) as follows:

$$Net_{qmn} = -\frac{C2}{2}\sum_{\substack{j1=1 \\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{qj1k1} - C3(V_{qmn} - P_q) - \frac{C5}{2} G_{qmn}^2 H(G_{qmn}) \tag{12}$$

and

$$Net_{lmn} = -\frac{C2}{2}\sum_{\substack{j1=1 \\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{lj1k1} - C3(V_{lmn} - P_l) - \frac{C5}{2} G_{lmn}^2 H(G_{lmn}). \tag{13}$$

Investigating Eq.(10), the total energy difference of the neural network, ΔE, between the $(t+1)^{th}$ and $(t)^{th}$ iteration is the same as the E_{mn} change between the $(t+1)^{th}$ and $(t)^{th}$ iteration. ΔE is displayed as below:

$$\Delta E = \frac{C2}{2}\Big(\sum_{\substack{j1=1\\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{qj1k1} - \sum_{\substack{j1=1\\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{lj1k1}\Big)$$
$$+\frac{C3}{2}\Big((1-P_q)^2 - (1-P_l)^2 + \sum_{\substack{i=1\\ i\neq q}}^{N} P_i^{\,2} - \sum_{\substack{i=1\\ i\neq l}}^{N} P_i^{\,2}\Big) \qquad (14)$$
$$+\frac{C5}{2}\left(G_{qmn}^2 H(G_{qmn}) - G_{lmn}^2 H(G_{lmn})\right)$$

After some algebraic manipulation and re-arrangement, the energy changes between the neuron update equals the net value change minus *C3*. That is

$$\Delta E = Net_{lmn} - Net_{qmn} - C3\,.$$

Clearly, the above equation implies that the energy difference in the update is negative, that is $\Delta E < 0$. Restated, the energy function decreases with each iteration. Hence, the system is convergent during network evolution. Apparently, this energy function is an appropriate Lyapunov function.

5. Simulation examples and results

The simulations consider classes of scheduling problems with timing constraints. Additionally, significant portions of the energy curves during neural network evolution were also shown. Table 1 defines the constants of the energy function in Eq.(5). Two sets of timing constraints and various different initial neuron states were applied to the simulations. Tables 2 and 3 list timing constraints for two simulation examples, respectively. This simulation involves scheduling four processes (jobs) in two processors (machines). Figs. 3 and 4 illustrate the resulting schedules for distinct initiate states of example 1, respectively. Meanwhile, Fig. 2 displays the energy revolution for example 1. Additionally, a second scheduling problem with five processes and two processors was also simulated. Moreover, different initial conditions are simulated to better understand the response of the neural network to the

scheduling problem. Figs. 5 and 6 present the scheduling results for different initial states of the second example.

6. Discussion and conclusions

HNN uses the quadratic energy function, which results in the quadratic cost of the interconnection network and hence poor scaling property. The winner-take-all mechanism eliminates the constraint terms in the energy function, simplifying the network by reducing the interconnections among neurons [10]. Hence, CHNN can help overcome the scaling problem. This work illustrated an approach to mapping the problem constraint into the energy function of the competitive Hopfield neural network so as to resolve the timing constraints schedule problem. The energy function proposed herein works efficiently and can be applied to investigating certain scheduling problems.

Simulation results demonstrate some significant consequences for CHNN and the features of CHNN when applied to the scheduling domain examined herein, as follows.

(1) Randomly assigning the initial states can obtain feasible schedules for the investigated scheduling problem.

(2) The rate of convergence is initial-state dependent;

(3) The entailed synaptic weight matrix in Eq.(6), although symmetric (i.e. $W_{xyzijk} = W_{ijkxyz}$), has a self-feedback interconnection, implying $W_{xyzijk} \neq 0$. Thereby, the network may oscillate during network evolution [14][16]. Consequently, a solution is not guaranteed, causing inevitable oscillation. In [17], Takefuji and Lee proposed a hystersis binary neuron model to effectively suppress the oscillatory behavior of neural dynamics for solving combinatorial optimization problems.

Moreover, weighting factor determination is an intrinsic shortcoming of HNN, and the simulations also encounter this drawback. However, the set of weight matrixes used in our simulation as listed in Table 1 is not unique. Different sets of weighting factors produce different neural network revolutions.

The Hopfield neural network is known to frequently trap to a local minimum after the network is stabilized. The simulated annealing technique can effectively obtain a global minimum capable of escaping from the local minimum. However, this approach requires more iteration. An important feature of a scheduling algorithm is its efficiency or performance, i.e., how its execution time grows with problem size. The parameter most relevant to the time a neural network takes to find a solution is the number of iterations needed to converge to a solution. According to simulation results, this CHNN requires an average of 5 ~ 20 iterations to converge. Each iteration involves updating every column of the competitive Hopfield neural network. The number of neurons for the proposed neural network is $(N * M * T)$, and the computational time for each iteration is equal to the total neurons $(N * M * T)$ multiplied by the computation time for each neuron (proportional to $(N * M)$). Consequently, this algorithm results in a $O(N^2 * M^2 * T)$ complexity. Restated, the execution time required for each iteration is proportional to $O(N^2 * M^2 * T)$. Furthermore, finding the solution for a very large-scale system (very large N and/or very large M) might require an unacceptably long time. Thus, for large scaled cases, the scaling problem will become a drawback of the proposed model. A future work should examine how to reduce the complexity in solving the scheduling problems.

This work focuses on the problem of real-time scheduling without ready time consideration. For practical implementation, the problem can be extended to involve the temporal relationship of ready time or priority for each job. Restated, each job

would be ready to run at different time. Correspondingly, the constructed energy function in this work can be modified by adding additional energy terms to satisfy extra requirement. A notion resembling the priority scheduling constraint may be involved. A future work should address this issue more thoroughly.

Reference

[1] T. M. Willems and J. E. Rooda, "Neural Networks for Job-shop Scheduling," *Control Eng. Practice*, 2(1) (1994) 31-39.

[2] Y. P. S. Foo and Y. Takefuji, "Integer linear programming neural networks for job-shop scheduling," in: *IEEE Int. Conf. on Neural Networks*, vol. 2, 1998, pp. 341-348.

[3] Chang-shui Zhang, Ping-fan Yan, Tong Chang, "Solving Job-Shop Scheduling Problem with Priority Using Neural Network," in: *IEEE Int. Conf. on Neural Networks*, 1991, pp. 1361-1366.

[4] C. Cardeira and Z. Mammeri, "Neural networks for multiprocessor real-time scheduling," in: *Proc. Sixth Euromicro Workshop on Real-Time Systems*, 1994, pp. 59-64.

[5] A. Hanada, and K. Ohnishi, "Near optimal jobshop scheduling using neural network parallel computing," in: *Int. Conf. on Proc.Industrial Electronics, Control, and Instrumentation*, vol. 1, 1993, pp. 315-320.

[6] Jeon Gue Park, Jong Man Park, Dou Seok Kim, Chong Hyun Lee, Sang Weon Suh, and Mun Sung Han, "Dynamic neural network with heuristic," in: *IEEE Int. Conf. on Neural Networks*, vol. 7, 1994, pp. 4650-4654.

[7] Yueh-Min Huang and Ruey-Maw Chen, "Scheduling Multiprocessor Job with Resource and Timing Constraints Using Neural Network," *IEEE Trans. on System, Man and Cybernetics*, part B, 29(4) (1999).

[8] J. J. Hopfield and D. W. Tank, “Neural Computation of Decision in Optimization Problems,” *Biological Cybernetics*, 52 (1985) 141-152.

[9] M. W. Dixon, G. R. Cole, and M. I. Bellgard, “Using the Hopfield network with mean field annealing to solve the shortest path problem in a communication network,” in: *Int. Conf. on Neural Networks*, vol. 5, 1995, pp. 2652-2657.

[10] P. C. Chung, C. T. Tsai, E. L. Chen, and Y. N. Sun, “Polygonal approximation using a competitive Hopfield neural network,” *Pattern Recognition*, 27 (1994) 1505-1512.

[11] J. S. Lin, K. S Cheng, and C. W. Mao, “A fuzzy Hopfield neural network for medical image segmentation,” *IEEE Trans. Nuclear Science*, 43(4) (1996) 2389-2398.

[12] Toshio Uchiyama and Michael A. Arbib, “Color Image Segmentation Using Competitive Learning,” *IEEE Trans. Pattern Analysis Machine Intelligence*, 16(12) (1994) 1197-1206.

[13] K. Lee, Y. Takefuji and N. Funabiki, “A parallel improvement algorithm for the biparties subgraphy problem,” Case Western Reserve Univ., CAISR Tech. Rep. TR91-105. 1991.

[14] J. J. Hopfield and D. W. Tank, “Computing with neural circuits: A model,” *Science*, 233 (1986) 625-633.

[15] G. Bilbro, R. Mann, T. Miller W. Snyder, D. E. Van den Bout, and M. White, “Mean field annealing and neural networks,” in: *Advances in Neural Information Processing System*, 1989, pp. 91-98.

[16] M. Takeda, J. W. Goodman, “Neural networks for computation: number representation and programming complexity,” *Applied Optics*, 251986 3033-3046.

[17] Y. Takefuji and K. C. Lee, “An artificial hystersis binary neuron: a model suppressing the oscillatory behaviors of neuron dynamics,” *Biological Cybernetics*,

64 (1991) 353-356.

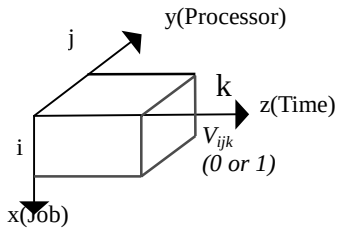

Fig.1. 3-D Hopfield neural network.

Constants for CHNN		
C2	*C3*	*C5*
2.0	**1.0**	**3.0**

Table 1. Weighting factors

	Time Required	*Time Limit*
Process 1	**4**	**6**
Process 2	**3**	**4**
Process 3	**3**	**6**
Process 4	**2**	**3**

Table 2. Timing Constraints Matrix. (simulation example 1)

	Time Required	*Time Limit*
Process 1	**2**	**3**
Process 2	**5**	**8**
Process 3	**3**	**4**
Process 4	**4**	**8**
Process 5	**2**	**5**

Table 3. Timing Constraints Matrix. (simulation example 2)

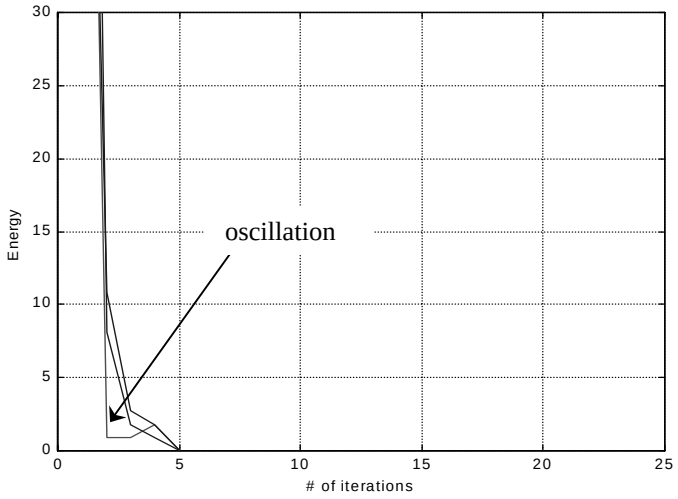

Fig. 2. Energy evolution of example 1.

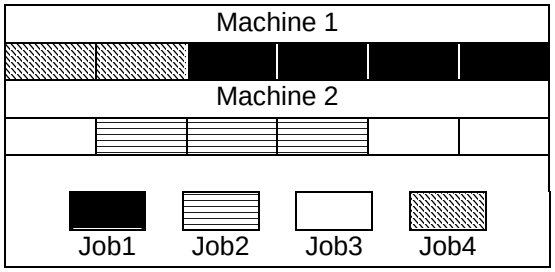

Fig. 3. Simulation results of example 1.

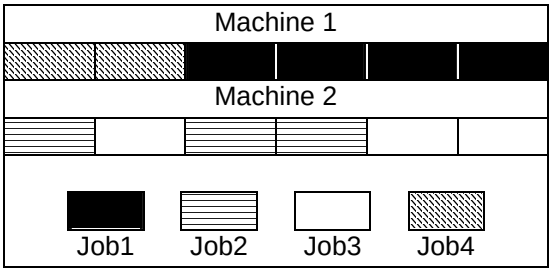

Fig. 4. Simulation results of example 1.

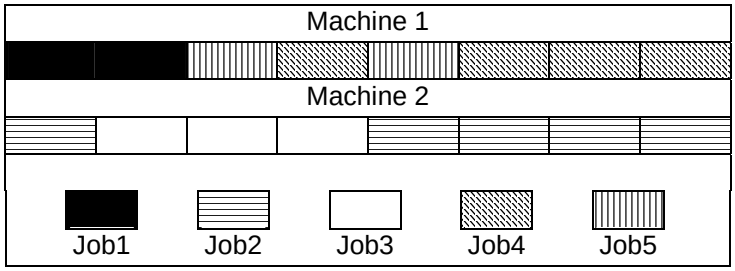

Fig. 5. Simulation results of example 2.

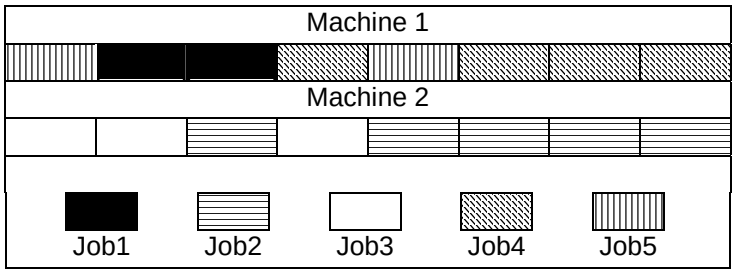

Fig. 6. Simulation results of example 2.