Solve Multiprocessor Real-Time Scheduling Using Competitive Slack Neural Networks

Ruey-Maw Chen
Department of Electronic Engineering, National Chin-yi Institute of Technology
raymond@ncit.edu.tw

Shih-Tang Lo
Department of Engineering Science, National Cheng-Kung University
edwardlo@mail.ksut.edu.tw

Yueh-Min Huang
Department of Engineering Science, National Cheng-Kung University
huang@mail.ncku.edu.tw

Chuin-Mu Wang
Department of Electronic Engineering, National Chin-yi Institute of Technology
cmwang@ncit.edu.tw

Abstract-Neural network using competitive learning rule provides a highly effective method of attaining a sound solution and is capable of simplifying the network complexity. Intrinsically, the competitive scheme is used to solve fully utilized real-time scheduling problem. This study extends the competitive Hopfield neural network to solve both non-fully and fully utilized multiprocessor real-time scheduling problems with execution time and deadline constraints. Extra "*slack neurons*" added on to the neural network topology to represent pseudo job were applied to ease the non-fully utilized situation and facilitate solving the problem. Simulation results confirm that the competitive neural network imposed on the proposed energy function corresponding neural networks with slack neurons integrated ensures an appropriate approach of solving this class of real-time scheduling problems of single processor or multiprocessor system.

Keywords: Real-time scheduling, Slack neuron, Competitive learning

1.Introduction

A scheduling algorithm is used to determine a schedule for a set of tasks so that the task's timing constraints are satisfied. Many different schemes have been developed for solving the scheduling problem. Linear programming is a widely used scheme 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 [2]. Meanwhile, the neural networks were applied to solve scheduling problems extensively. Zhang, Yan, and Chang proposed a neural network method derived from linear programming [3]. Additionally, Silva, Cardeira and Mammeri investigated the multi-processor real-time scheduling by applying the Hopfield-type neural network [4]. Honda and Ohnishi [5] developed a parallel algorithm based on a neural network for preemptive task scheduling problems.

Hopfield and Tank started the applications in using the neural network to solve optimization problems [6]. The energy function used in the Hopfield neural network (HNN) is an appropriate Lyapunov function. Many researchers have recently applied this method to various applications. Our previous work [7] solved a multi-constraint schedule problem for a multiprocessor system using the Hopfield neural network.

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 reduces the time consumed in obtaining coefficients but also obtains an effective and sound solution. CHNN has been applied to various fields Chung, Tsai, Chen, and Sun [8] proposed a competitive Hopfield neural network for polygonal approximation. Similarly, Uchiyama and Arbib [9] 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. We have conducted a series of study in solving "fully utilized" processors scheduling problem by using HNN [7,10]. Also, a typical CHNN scheme applied to the same problem was conducted in [11]. Intrinsically, including competitive architecture into the network solves the problems, which have unique activated neuron on each column or row of the networks. Accordingly, competitive scheme copes with fully utilized scheduling problems. In [12], which investigated the

multi-processor real-time scheduling to meet deadline requirement by applying the *k-out-of-N* rule to a neural network, slack neurons are extended to agree with the inequality constraints. Tagliarini, Christ and Page [13] demonstrated a weapon-to-target assign problem, a resource allocation task problem. There is a slack neuron associated with each weapon. The slack neurons are used only to assure that problem constraints are satisfied. In most situations, a fully multiprocessor utilization system is a restrictive situation. Moreover, real-time scheduling is interesting on meeting task timing constraints rather than optimizing a given target.

In light of above developments, this work investigates the job schedule problem of multi-process on a "fully" or "non-fully" utilized real-time multiprocessor system that includes timing constraints. In addition, this studied problem is presented using a 3-dimentional neural networks. Extra slack neurons are added on to the networks to content the non-fully utilized conditions. An energy function designed to illustrate the timing constraints is proposed. According to the CHNN, the scheduling problem is considered a minimization of an energy function. The simulations involve the fully utilize multiprocessor real-time scheduling problems, which were solved with competitive scheme in [11]. Furthermore, some fully and non-fully utilized real-time scheduling problems were simulated. Notably, the slack neurons should not be taken into account when retrieving the solution of the problem.

2. Energy function of the scheduling problem

The scheduling problem domain to be considered in this paper is defined as follows. Suppose there are N jobs (or processes) and M machines (or processors). 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 and the deadline of each job are predetermined. Furthermore, the processor is allowed to be non-fully-utilized. Based on these assumptions, we attempt to obtain a set of job schedules. Restated, the preemptive processes in a multiprocessor real-time system are interesting.

To resolve this problem, an energy function that represents the scheduling problem has to be defined. The defined energy function is transformed into an extended 3-D HNN; then the optimization process searches for neuron states satisfying a set of constraints such that the energy function is minimized. According to the problem, scheduling involves three variables: job (or process), machine (or processor), and time. Thus, neuron states variable V_{ijk} is defined. The job variable, i, indicates a specific job with a range from 1 to N+1. The total number of job is N. The $(N+1)^{th}$ job is a pseudo-job, which is a supplementary job used to facilitate satisfying *1-out-of-N* rule. Restate, by adding neurons to the hypothesis representation neurons, we can enforce inequality constraints. The extra neurons are referred to as "slack neurons" herein. In this investigation, slack neurons are those neurons in representing the pseudo-jobs. Meanwhile, the machine variable, j, represents a dedicated machine from 1 to M, the total number machines to be operated. Finally, the time variable, k represents a specific time, which is less than or equal to T, the deadline of the job. Thus, a state variable V_{ijk} is representing whether or not job *i* is executed on machine *j* at a certain time *k*. In addition, the activated neuron $V_{ijk}=1$ denotes that the job *i* is run on machine *j* at the time *k*; otherwise, $V_{ijk}=0$. The activated neuron $V_{(N+1)jk}=1$ indicates processor *j* at a certain time *k* is free. Notably, each V_{ijk} corresponds to a neuron of the neural network.

The correlating energy function representing the scheduling problem is as Eq. (1). In this equation C_1, C_2, C_3, C_4, C_5 and C_6 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, which occurs when V_{ijk} or V_{i_1jk} equals zero.

$$
\begin{aligned}
E = & \frac{C_1}{2} \sum_{i=1}^{N+1} \sum_{j=1}^{M} \sum_{k=1}^{T} \sum_{\substack{i1=1 \\ ,i1\neq i}}^{N+1} V_{ijk} V_{i1jk} \\
& + \frac{C_2}{2} \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{T} \sum_{\substack{j1=1 \\ ,j1\neq j}}^{M} \sum_{k1=1}^{T} V_{ijk} V_{ij1k1} \\
& + \frac{C_3}{2} \sum_{i=1}^{N+1} \Big(\sum_{j=1}^{M} \sum_{k=1}^{T} V_{ijk} - P_i \Big)^2 \\
& + \frac{C_4}{2} \sum_{i=1}^{N+1} \sum_{j=1}^{M} \Big(\sum_{k=1}^{T} V_{ijk} - 1 \Big)^2 \\
& + \frac{C_5}{2} \sum_{i=1}^{N+1} \sum_{j=1}^{M} \sum_{k=1}^{T} V_{ijk} G^2_{ijk} H(G_{ijk}) \\
& - \frac{C_6}{2} \sum_{i=N+1}^{N+1} \sum_{j=1}^{M} \sum_{k=1}^{T} \sum_{\substack{j1=1 \\ ,j1\neq j}}^{M} \sum_{k1=1}^{T} V_{ijk} V_{ij1k1}
\end{aligned}
\qquad (1)
$$

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

The C_2 energy term indicates that a process migration is prohibited, implying that process *i* runs on processor *j* or *j1*. This item also has a minimum value of zero when V_{ijk} or V_{ij1k1} is 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. The processing time of the pseudo-job (the $N+1^{th}$ job) is defined as the total available time for all machines subtracts the total processing time required by all N jobs. Additionally, the C_4 energy term is to prevent no process being executed on a specific processor at a certain time when *1-out-of-N* rule applied. Thus, this energy item falls to a minimum of zero when satisfying this constraint. Meanwhile, the C_5 energy term is real-time requirement to meet the deadline 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*. On the contrary, this energy term will have a value zero as long as V_{ijk}=1and $k\text{-}d_i \leq 0$. Finally, the C_6 term excludes the slack neurons from the 2nd term. Hence, this term allows slack neuron of the $N+1^{th}$ job to be activated on different processors (*j* and $j_1 \neq j$) at any time *k1*. Since the slack neurons are not to be taken into account when retrieving the solution of the problem. Restate, the 2nd term limits exactly one processor *j* to run process *i*, and this term allows more than one processor to run process *N+1* (i.e., pseudo-job). 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.

3. Competitive algorithm of neural networks

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. Moreover, the HNN also provides potential for parallel implementation. Based on dynamic system theory, the Liapunov function [14] [15] expended to a 3-dimentional model as 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.

$$E = -\frac{1}{2}\sum_x\sum_y\sum_z\sum_i\sum_j\sum_k V_{xyz}W_{xyzijk}V_{ijk} + \sum_i\sum_j\sum_k \theta_{ijk}V_{ijk} \qquad (2)$$

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:

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

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:

$$Net_{ijk} = -\frac{\partial E}{\partial V_{ijk}} = \sum_x\sum_y\sum_z W_{xyzijk}V_{xyz} - \theta_{ijk} \qquad (4)$$

Instead of applying conventional deterministic rules to update neuron states, competition among neurons is used to decide the winning neuron, i.e. the active neuron in competitive scheme. Restated, the competitive rule is to build neural networks satisfying constraints of the type of "exact one neuron among N" should be activated when the network reaches a stable state, and can be considered as a *1-out-of-N* confine rule.

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 implicitly. The resulting energy function for CHNN is highlighted as follows:

$$\begin{aligned} E = & \frac{C2}{2}\sum_{i=1}^{N+1}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{\substack{j1=1 \\ j1\neq j}}^{M}\sum_{k1=1}^{T} V_{ijk}V_{ij1k1} \\ & + \frac{C3}{2}\sum_{i=1}^{N+1}\Big(\sum_{j=1}^{M}\sum_{k=1}^{T} V_{ijk} - P_i\Big)^2 \\ & + \frac{C5}{2}\sum_{i=1}^{N+1}\sum_{j=1}^{M}\sum_{k=1}^{T} V_{ijk}G_{ijk}^2 H(G_{ijk}) \\ & - \frac{C6}{2}\sum_{i=N+1}^{N+1}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{\substack{j1=1 \\ ,j1\neq j}}^{M}\sum_{k1=1}^{T} V_{ijk}V_{ij1k1} \end{aligned} \qquad (5)$$

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

$$\begin{aligned} W_{xyzijk} = & -C2*\delta(x,i)*(1-\delta(y,j)) - C3*\delta(x,i) \\ & + C6*\delta(x,N+1)*\delta(i,N+1)*(1-\delta(y,j)) \end{aligned} \qquad (6)$$

and

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

respectively, where

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

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:

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

where Net_{xjk} is the maximum total neuron input as in Eq.(4). The *ith* column consists of jobs 1 ~ N+1 (slack neuron). Restated, the slack neuron can be the wining neuron while satisfying the constraints.

4. Convergence of the energy function

This section demonstrates a mathematical proof of convergence in the CHNN for the investigated problem.

According to Eq. (4), 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, we separate this energy function into two parts. The first one is for the processor *m* at given time *n*, that is, E_{mn}. The second part is the remainder, that is, E_{other}. In other word, E_{mn} is the summation of the energy term corresponding to neuron state V_{imn}. The energy function can then be represented as follows (Eq. 10):

$$\begin{aligned} E = {} & \frac{C2}{2}\Big(\sum_{i=1}^{N+1}\sum_{\substack{j1=1 \\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{imn}V_{ij1k1} \\ & + \sum_{i=1}^{N+1}\sum_{\substack{j=1 \\ j\neq m}}^{M}\sum_{\substack{k=1 \\ k\neq n}}^{T}\sum_{\substack{j1=1, \\ j1\neq j, \\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{ijk}V_{ij1k1}\Big) \\ & + \frac{C3}{2}\Big(\sum_{i=1}^{N+1}(V_{imn} - P_i)^2 + \sum_{i=1}^{N+1}\Big(\sum_{\substack{j=1 \\ j\neq m}}^{M}\sum_{\substack{k=1 \\ k\neq n}}^{T} V_{ijk} - P_i\Big)^2\Big) \\ & + \frac{C5}{2}\Big(\sum_{i=1}^{N+1} V_{imn}G_{imn}^2 H(G_{imn}) \\ & + \sum_{i=1}^{N+1}\sum_{\substack{j=1 \\ j\neq m}}^{M}\sum_{\substack{k=1 \\ k\neq n}}^{T} V_{ijk}G_{ijk}^2 H(G_{ijk})\Big) \\ & + \frac{C_6}{2}\sum_{i=N+1}^{N+1}\sum_{\substack{j1=1 \\ ,j1\neq j}}^{M}\sum_{k1=1}^{T}(-V_{ijk})V_{ij1k1} \end{aligned} \tag{10}$$

$E = E_{mn} + E_{other}$.

V_{imn} is the neuron on the i^{th} row (job) and the n^{th} column (time) for the specific processor *m*. 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 the 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 $(t+1)^{th}$ iteration is supposed to be the only neuron activated with the largest total input value after updating, that is,

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

The active neuron, based on the winner-take-all update rule as in Eq.(8), is the one with maximum net value on each column in each update, that is $Net_{qmn} = \underset{i=1\sim N+1}{Max} Net_{imn}$. This implies that $Net_{qmn} > Net_{lmn}$. Net_{qmn} and Net_{lmn} are obtained based on Eq.(9) as follows:

$$\begin{aligned} Net_{qmn} = {} & -\frac{C2}{2}\sum_{\substack{j1=1 \\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{qj1k1} - C3(V_{qmn} - P_q) \\ & - \frac{C5}{2}G_{qmn}^2 H(G_{qmn}) \end{aligned} \tag{11}$$

and

$$\begin{aligned} Net_{lmn} = {} & -\frac{C2}{2}\sum_{\substack{j1=1 \\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{lj1k1} - C3(V_{lmn} - P_l) \\ & - \frac{C5}{2}G_{lmn}^2 H(G_{lmn}) \end{aligned} . \tag{12}$$

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

Since $V_{qmn}^{(t+1)} = 1$, $V_{imn}^{(t+1)} = 0(i \neq q)$, $V_{lmn}^{(t)} = 1$, and $V_{imn}^{(t)} = 0(i \neq l)$. Thereby, the energy change difference cab be rewritten as follows:

$$\begin{aligned} \Delta E &= E_{mn}^{t+1} - E_{mn}^{t} \\ &= \frac{C2}{2}\Big(\sum_{\substack{j1=1 \\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{qj1k1} - \sum_{\substack{j1=1 \\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{lj1k1}\Big) \\ &+ \frac{C3}{2}\Big((1-P_q)^2 - (1-P_l)^2 + \sum_{\substack{i=1 \\ i\neq q}}^{N} P_i^2 - \sum_{\substack{i=1 \\ i\neq l}}^{N} P_i^2\Big) \\ &+ \frac{C5}{2}(G_{qmn}^2 H(G_{qmn}) - G_{lmn}^2 H(G_{lmn})) \end{aligned} \tag{13}$$

Accordingly, the energy changes between neuron update equals the net value change minus *C3*. That is $\Delta E = Net_{lmn} - Net_{qmn} - C3$.

Obviously, the above equation implies that the energy difference in the update is negative, i.e. $\Delta E < 0$ .Hence, the system is convergent during network evolution. Apparently, this energy function is an appropriate Lyapunov function.

5. Simulation examples and results

A number of sets of timing constraints and various weighting factors were applied to the simulations. The constants in Eq.(11) are set to C_2=1.355, C_3=0.55, C_5=1.355 and used in simulation. The C_6 term exclude the slack neurons from C_2 term. Consequently, C_6 value is equal to the C_2 value. Tables 1 through 4 list timing constraints of simulation cases, respectively. The simulation involves scheduling four or five processes (jobs) in

two processors (machines) system. Moreover, more complicated simulation case with10 processes in three processors system is included as well. Case 1a and case 3a are the same conditions as in [11]. They are added to facilitate the full processor utilization system scheduling study by applying the proposed scheme herein. Figs. 1 and 2 illustrate the resulting schedules of case 1a and case 1b for proposed algorithm. Meanwhile, Figs 4 and 5 display the simulation results of case 3a and case 3b.. Finally, Figs. 3 and 6 represent the resulting schedules correlating with the case 2 and 4 respectively. The process assignment of S as displayed in figures 2, 3, 5, and 6, indicates the active "slack" neurons. Restate, the machine at that time does nothing. Additionally, figure 7 displays the significant portions of the energy curves of all cases during neural network evolution.

6. Discussion and conclusions

The competitive mechanism eliminates the constraint terms in the energy function, simplifying the network by reducing the interconnections among neurons [8], as displayed in Eq. (5). This investigation illustrated an approach to mapping the problem constraints into the energy function of the competitive neural networks containing slack neurons, they are involved so as to resolve the timing constraints schedule problem for both "non-fully utilized" and "fully utilized" system. The conventional deterministic rule in determining the neuron state was replaced by winner-take-all or competitive rule.

The weighting factor determination is an intrinsic shortcoming of Hopfield neural network and the simulations also encounter this drawback. However, the set of weighting matrix used in our simulation is not unique. Different sets of weighting factors may produce different neural network revolutions.

The energy function proposed herein works efficiently and can be applied to the similar cases of investigated scheduling problems. The competitive scheme combined with slack neurons suggests the way to be applied to this kind of scheduling, which has inequality constraints. Restated, solving "fully" or "non-fully" utilized real-time scheduling problems can apply proposed competitive slack neural networks.

This work focuses mainly on solving real-time scheduling without ready or setup time consideration. For more practical implementation, different and more complicated scheduling problems can be further investigated in future work by applying this proposed algorithm. Such problems include preemptive multi-process scheduling on multiprocessor system with multi-constraints such as deadline and resource constraints. The problem can be further extended to involve the temporal relationship of ready time or priority for each job. A future work should address these issues more thoroughly.

Acknowledgements: This work is supported by National Science Council (NSC93-2213-E167-012).

Reference

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

[2] Foo YPS, Takefuji Y. Integer linear programming neural networks for job-shop scheduling. In: IEEE Int. Conf. on Neural Networks, vol. 2, 1998, pp. 341-348.

[3] Zhang CS, Yan PF, and Chang T. Solving Job-Shop Scheduling Problem with Priority Using Neural Network. In: IEEE Int. Conf. on Neural Networks, 1991, pp. 1361-1366.

[4] Silva MP, Cardeira C, and Mammeri Z. Solving real-time scheduling problems with Hopfield-type neural networks. In: EUROMICRO 97 'New Frontiers of Information Technology, Proceedings of the 23rd EUROMICRO Conference, 1-4 Sept. 1997, pp. 671 –678.

[5] Hanada A, Ohnishi K. 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] Hopfield JJ, Tank DW. Neural computation of decision in optimization problems. Biological Cybernetics 1985; 52: 141-152.

[7] Huang YM, Chen RM. Scheduling multiprocessor job with resource and timing constraints using neural network. IEEE Trans. on System, Man and Cybernetics, part B, Aug.1999; 29(4): 490-502.

[8] Chung PC, Tsai CT, Chen EL, Sun YN. Polygonal approximation using a competitive Hopfield neural network. Pattern Recognition 1994; 27: 1505-1512.

[9] Uchiyama T, Arbib MA. Color Image Segmentation Using Competitive Learning. IEEE Trans. Pattern Analysis Machine Intelligence. 1994; 16(12): 1197-1206.

[10] Chen RM, Huang YM. Multi-constraint task scheduling in multiprocessor system by neural network. In: Proc. IEEE Tenth Int. Conf. on Tools with Artificial Intelligence, Taipei, 1998, pp. 288-294.

[11] Chen RM, Huang YM. Competitive Neural Network to Solve Scheduling Problem. Neurocomputing, April 2001; 37(1-4):177-196.

[12] Cardeira C, Mammeri Z. Neural networks for multiprocessor real-time scheduling. In: IEEE Proc. Sixth Euromicro Workshop on

Real-Time Systems, 1994, pp. 59-64.
[13] Tagliarini GA, Christ JF, and Page EW. Optimization Using Neural Networks. IEEE Transaction on Computers Sep. 1991; 40(12): 1347-1358.
[14] Hopfield JJ, Tank DW. Computing with neural circuits: A model. Science 1986; 233: 625-633.
[15] Cohen M, Grossberg S. Absolute stability of goal pattern formation and parallel memory storage by competitive neural network. IEEE transaction on system, Man, and Cybernetics Sep/Oct 1983; 13: 815-26.

Table 1. Timing constraints matrix (case 1a: A=4,B=3; case 1b:A=3,B=2)

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

Table 2. Timing constraints (case 2)

	Time Required	*Time Limit*
Process 1	**5**	**8**
Process 2	**4**	**8**
Process 3	**3**	**6**
Process 4	**2**	**3**

Table 3. Timing constraints (case 3a: A=5,B=4; case2b: A=3,B=3)

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

Table 4. Timing constraints (case 4)

	Time Required	*Time Limit*
Process 1	**2**	**10**
Process 2	**3**	**5**
Process 3	**3**	**9**
Process 4	**2**	**5**
Process 5	**3**	**9**
Process 6	**2**	**6**
Process 7	**3**	**10**
Process 8	**2**	**5**
Process 9	**3**	**9**
Process 10	**4**	**10**

Machine 1					
P4	**P1**	**P4**	**P1**	**P1**	**P1**

Machine 2					
P2	**P3**	**P2**	**P2**	**P3**	**P3**

Figure 1. Simulation results of case 1a (fully utilized)

Machine 1					
P4	**P1**	**P4**	**P1**	**P1**	**S**

Machine 2					
P2	**P2**	**P3**	**P2**	**P3**	**S**

Figure 2. Simulation results of case 1b (non-fully utilized)

Machine 1							
P4	**P4**	**S**	**S**	**P2**	**P2**	**P2**	**P2**

Machine 2							
P1	**P3**	**P3**	**P3**	**P1**	**P1**	**P1**	**P1**

Figure 3. Simulation results of case 2

Machine 1							
P4	**P1**	**P1**	**P4**	**P4**	**P4**	**P5**	**P5**

Machine 2							
P3	**P3**	**P3**	**P2**	**P2**	**P2**	**P2**	**P2**

Figure 4. Simulation results of case 3a (fully utilized)

Machine 1							
P1	**P4**	**P1**	**P4**	**S**	**P4**	**S**	**S**

Machine 2							
P3	**P3**	**P3**	**P2**	**P5**	**P2**	**P5**	**P2**

Figure 5. Simulation results of case 3b (non-fully utilized)

Machine 1									
P9	**P8**	**P6**	**P8**	**P6**	**P9**	**S**	**P9**	**S**	**S**

Machine 2									
P10	**P2**	**P7**	**P2**	**P2**	**P7**	**P10**	**P7**	**P10**	**P10**

Machine 3									
P3	**P3**	**P5**	**P4**	**P4**	**P1**	**P5**	**P5**	**P3**	**P1**

Figure 6. Simulation results of case 4

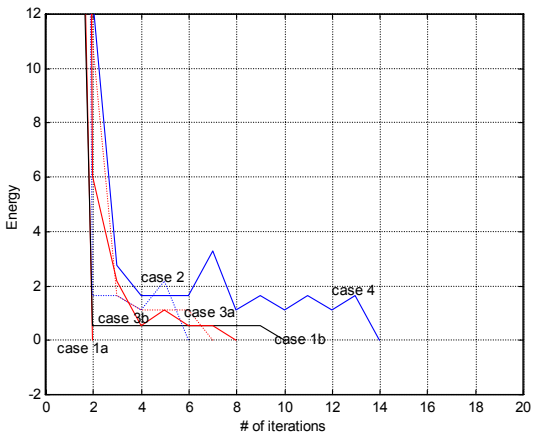

Figure 7. Energy function of case 1-4.