Available online at www.sciencedirect.com
ScienceDirect
Expert Systems with Applications 33 (2007) 75–85
Expert Systems with Applications
www.elsevier.com/locate/eswa

Combining competitive scheme with slack neurons to solve real-time job scheduling problem

Ruey-Maw Chen [a], Shih-Tang Lo [b], Yueh-Min Huang [b,*]

[a] Department of Computer Science and Information Engineering, National Chin-yi Institute of Technology, Taichung 411, Taiwan, ROC
[b] Department of Engineering Science, National Cheng-Kung University, Tainan 701, Taiwan, ROC

Abstract

Generally, how to satisfy the deadline constraint is the major issue in solving real-time scheduling. Recently, neural network using competitive learning rule provides a highly effective method and deriving a sound solution for scheduling problem with less network complexity. However, due to the availability of resources, the machines may not reach full utilization. To facilitate the problem the extra neuron is introduced to the competitive neural network (CHNN). This study tries to impose slack neuron on CHNN with respect to process time and deadline constraints. Simulation results reveal that the competitive neural network imposed on the proposed energy function with slack neurons integrated ensures an appropriate approach of solving this class of scheduling problems of single or multiple identical machines.

Keywords: Scheduling; Slack neuron; Competitive learning; Hopfield neural network

1. Introduction

Neural networks have been widely used in a large area of applications like image processing, learning processes, identification and control, etc. But, there is a lack for their use for approximate solving real-time scheduling problems. Most problems are confirmed to be NP complete or combinatorial problems, especially for large-scale scheduling problem. 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 quite time consuming.

In general, job scheduling problems are seen as involving allocations of resources (like machines or processors) to execute a set of jobs satisfying a given type of constraints and optimizing a given criterion. Jobs are assigned timing constraints like ready time and deadline, and they need a maximum number of time units of processing time (Cardeira & Mammeri, 1996). Many different schemes have been developed for solving the scheduling problem. Liu and Layland (1973) was the pioneering paper giving an insight to real-time scheduling algorithms for mono-job or scheduling of independent and periodic tasks. Linear programming is widely approached to minimize cost function from the specific scheduling problem. Willems and Rooda (1994) translated the job-shop scheduling problem onto a linear programming format, and then mapped it into an appropriate neural network structure to obtain a solution. Furthermore, Foo and Takefuji (1998) adopted integer linear programming neural networks to solve the scheduling problem by minimizing the total starting times of all jobs by a precedence constraint. Zhang, Yan, and Chang (1991) developed a neural network algorithm derived from linear programming, in which preemptive jobs are scheduled according to their priorities and deadline. Silva, Cardeira, and Mammeri (1997) explored the multi-process or real-time scheduling with a Hopfield-type neural network. Above investigations concentrating on the preemptive jobs executed on multiple machines with job

* Corresponding author.
E-mail address: huang@mail.ncku.edu.tw (Y.-M. Huang).

0957-4174/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2006.04.017

Nomenclature

N	total number of jobs/processes to be scheduled
M	total number of machines/processors to be operated
T	deadline of the jobs
i, j, k	denotes the "job", "machine", and "time" variables, respectively
x, y, z	denotes the "job", "machine", and "time" variables, respectively
V_{ijk}, V_{xyz}	represents the binary states of neurons (i,j,k) and (x,y,z) on Hopfield neural network
P_i	denotes the total execution time required by process i
di	deadline of the process i
G_{ijk}	defined to examine that whether the time of process x finished in processor i later than the time limit
$H(G_{ijk})$	unit step function. Defined to check if the timing constraint satisfied. A non-zero value indicates the assigned schedule violating the timing constraint. On the other hand, a zero value is yield as meet the timing requirement
W_{xyzijk}	synaptic weight between neuron (x,y,z) and neuron (i,j,k)
θ_{ijk}	input bias from outside of neuron (i,j,k)
Net_{ijk}	net value of neuron (i,j,k), a neuron (i,j,k) receives a community of neuron with interconnection strength W_{xyzijk} and an input bias, θ_{ijk}, from outside
$\delta(a,b)$	*Kronecker delta* function. The value is 1 if a equals b. Otherwise, the value is zero
E	energy function
C_1, C_2, C_3, C_4, C_5	weighting factors of energy terms

transfer by a neural network. Moreover, Hanada and Ohnishi (1993) presented a parallel algorithm based on a neural network for task scheduling problems by permitting task transfer among machines. A classical local search heuristic algorithm was embedded into the TSP optimization neural network by Park et al. (1994). Most investigations have constructed the energy functions for scheduling problems in terms of timing constraint, preemption, and migration features associated with the process. Meanwhile, the neural networks were applied to solve scheduling problems extensively.

This work aims to find a feasible solution to generic scheduling problem. Most scheduling problem are concentrated on minimizing the maximum complete time (*makespan*), or minimizing the tardiness. Such problem to obtaining the optimal solution is quite time consuming. An advantage of real-time task scheduling is owing to its ability to meet task timing constraints rather than optimize a given target. Examples of real-time scheduling include nuclear power plant control system, traffic control systems, flight mission control system and embedded tactical systems for military applications. In these applications, failure to meet timing constraints of system might not only lead to system degradation, but even it may lead to a hazardous situation. To solve generic scheduling problems containing timing constraints similar to the above examples is the major consideration in this study. This work investigates a job scheduling problem involving preemptive multitasking with processing time and deadline constraints on the condition of no job migration allowed. A modified neural network with slack neuron is constructed to solve the scheduling problems.

Hopfield and Tank (1985) started the applications in using the neural network to solve optimization problems. In the Hopfield neural networks, the state input information from a community of neurons is received to decide neuron output state information. Each neuron exchanges information with other neurons in the network. These neurons apply this information to cooperatively move 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. Dixon, Cole, and Bellgard (1995) applied the Hopfield neural network with mean field annealing to solve the shortest path problem in a communication network. In our previous work also, we solved a multi-constraint schedule problem for a multiprocessor or system by the Hopfield neural network (Huang & Chen, 1999).

A competitive Hopfield neural network (CHNN) applies a competitive learning mechanism to update the neuron states in the Hopfield neural network. A competitive learning rule cannot only reduce the time consumed in obtaining coefficients but also obtain an effective and sound solution. CHNN has been applied in various fields, mostly on image processing such as image clustering processes and specific image segmentation. Chung, Tsai, Chen, and Sun (1994) presented a competitive Hopfield neural network for polygonal approximation. Uchiyama and Arbib (1994) used competitive learning 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 machine at a certain time, enforcing the *1-out-of-N* constraint to be held. The maximum output value neuron of the set of neurons is activated. The monotonic of the maximum neuron follows the fact that it is equivalent to a McCulloch and Pitts neuron with a dynamic threshold (Lee, Funabiki, & Takefuji, 1992). A series of studies have been done to fully utilized processors scheduling problem (Chen & Huang, 1998; Huang & Chen, 1999). Hopfield neural network scheme and mean field annealing technique are utilized to obtain an adequate schedule. The

convergence rates corresponding to the cooling procedure of the mean field annealing technique in obtaining the scheduling results for the problem in Huang and Chen (1999) was also studied (Chen & Huang, 1998). In Chen and Huang (1998), we proposed a modified cooling schedule to accelerate convergence rate for the investigated problem. A typical CHNN scheme was applied to the same problem in Chen and Huang (2001). Intrinsically, including competitive architecture into the network solves the problems, which have a unique activated neuron on each column or row of the networks. Accordingly, competitive scheme can cope with fully utilized scheduling problems. Cardeira and Mammeri investigated the multi-process or real-time scheduling to meet deadline requirements by applying the *k-out-of-N* rule, which extends slack neurons to a neural network to agree with the inequality constraints. They extended the methodology to handle real-time scheduling with precedence constraints (Cardeira & Mammeri, 1994, 1997). Tagliarini, Christ, and Page (1991) demonstrated a weapon-to-target approach for a resource allocation task problem. A slack neuron is associated with each weapon. The slack neuron activated represents the hypotheses that the associated weapon is not fired. In real-time scheduling problem, due to the capacity constraints or availability of resources, the machines may not reach full utilization. A fully utilization system involves a special situation of job scheduling. This investigation extended the neural networks by adding some extra neurons to ease this restriction. The deterministic rules of the CHNN were applied to update the states of slack neuron. However, this work determined neuron states by the competitive rule to handle inequality constraint problem. The competitive mechanism is considered as an extreme situation of *k-out-of-N*. Hence, competitive mechanism is able to solve non-fully utilized scheduling cases.

In light of above developments, this work explores the job scheduling problem on a non-fully utilized (incomplete usage) system including timing constraints. The scheduling problem is presented using three-dimensional neural network structure. Extra slack neurons are added on to the networks to meet utilized conditions. This work can extend to cases of fully utilized situations. An energy function is proposed to illustrate the timing constraints. In CHNN, the scheduling problem is aimed at minimizing the energy function. The energy change is invariably negative when using formal mathematical derivations. The competition process of the Hopfield neural network can be applied to obtain the solution. Simulations on fully and non-fully utilized case scheduling problems were investigated in this study. The simulation results show that the proposed method can solve the real-time scheduling problem.

The rest of this paper is organized as follows. Section 2 derives the corresponding energy function of scheduling problem according to the intrinsic constraints. Section 3 reviews the competitive algorithm with slack neuron and translates the derived energy functions to the proposed algorithm. The simulation examples and experimental results are presented in Section 4. The conclusion and future work is showed in Section 5. In addition, the energy convergence proof is provided in Appendix A.

2. Energy function of the scheduling problem

Scheduling problems markedly differ from case to case. The scheduling problem domain to be considered in this paper is defined as follows. Assume that there are N jobs and M machines. First, a job can be segmented, and the execution of each segment is preemptive. Second, the different segments of a job cannot be assigned to different machines, implying that no job migration is allowed between machines. Third, each job's execution time and deadline are predetermined. Moreover, the machine is allowed to be non-fully utilized. A set of jobs can be obtained according these assumptions.

The *optimization* applications of neural networks are considered to solve this problem. At the first stage, the energy function representing the scheduling problem have to be defined, and the energy function is transformed into a 3-D HNN (Fig. 1). Second, CHNN is trained based on the predefined problem. Finally, the optimization process is searching for neuron states satisfying all constraints to minimize or maximize the energy function. This scheduling problem involves three variables, job, machine, and time. These three variables V_{ijk} are displayed in Fig. 1. The "x" axis denotes the "job" variable, with i representing a specific job with a range from 1 to $N+1$, where N is the total number of jobs to be scheduled. The $(N+1)$th job is a pseudo-job, i.e., a supplementary job to fulfill *1-out-of-N* rule. That is, inequality constraints can be enforced by adding neurons to the hypothesis representation neurons. The additional neurons are analogous to slack variables that are sometimes adopted to solve optimization problems in operation research, and are therefore called "slack neurons". Herein, slack neurons are neurons in representing the pseudo-job. If one machine processes a pseudo-job, it means machine is doing nothing at this time. The "y" axis stands for the "machine" variable, and the term j on the axis represents a dedicated machine from 1 to M, where M denotes 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, where T is the job deadline. Thus, a state variable V_{ijk} is defined as representing whether or

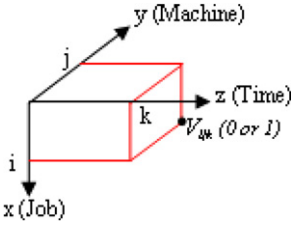

Fig. 1. 3-D Hopfield neural network.

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 run on machine j at time k; otherwise, $V_{ijk} = 0$. The activated neuron $V_{(N+1)jk} = 1$ indicates that machine j at a certain time k is free. Notably, each V_{ijk} corresponds to a neuron of the neural network; the total neurons of CHHH is $(N+1) * M * T$, which is combined with slack neuron.

Five energy terms of energy function are summarized representing the scheduling problem. The first energy term denotes the output state constraints, since machine j can only run one job at a certain time k. If job i is processed on machine j at time k ($V_{ijk} = 1$), there is no other job i_1 that can be processed on machine j at time k. This energy term is defined as

$$\sum_{i=1}^{N+1}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{\substack{i1=1\\ i1\neq i}}^{N+1} V_{ijk}V_{i1jk}, \tag{1}$$

where N, M, T, i, j, k, i_1, and V_{ijk} are as defined above, the rest of this study employs the same notations. The term has a minimum value of zero when it meets this constraint, which arises when $V_{ijk} = 0$ or $V_{i1jk} = 0$. The second term confines job migration, indicating that job i runs on machine j or $j1$. If a job is assigned on a dedicated machine, then all of its segments must be executed on the same machine. However, the $(N+1)$th job is a pseudo-job which can be processed on different machines, which is not included in this term. Accordingly, the energy term is defined as follows:

$$\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{\substack{j1=1\\ j1\neq j}}^{M}\sum_{k1=1}^{T} V_{ijk}V_{ij1k1}. \tag{2}$$

This term also has a minimum value of zero when V_{ijk} or V_{ij1k1} is zero. The third energy term is defined as

$$\sum_{i=1}^{N+1}\left(\sum_{j=1}^{M}\sum_{k=1}^{T} V_{ijk} - P_i\right)^2, \tag{3}$$

where P_i is the total execution time needed by job i. This energy term means that the time consumed by job i must equal P_i such that $\sum\sum V_{ijk} = P_i$. Eq. (3) 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, another state constraint energy item is introduced as

$$\sum_{j=1}^{M}\sum_{k=1}^{T}\left(\sum_{i=1}^{N+1} V_{ijk} - 1\right)^2. \tag{4}$$

This energy term provides a supplemental constraint to ensure that no job being executed on a specific machine at a certain time when using the *1-out-of-N* rule. Therefore, this energy item should also reach a minimum value of zero. The following energy term is defined to meet the deadline requirement of each job i

$$\sum_{i=1}^{N+1}\sum_{j=1}^{M}\sum_{k=1}^{T} V_{ijk}G_{ijk}^2 H(G_{ijk}),$$
$$H(G_{ijk}) = \begin{cases} 1, & \text{if } G_{ijk} > 0, \\ 0, & \text{if } G_{ijk} \leqslant 0, \end{cases} \quad G_{ijk} = k - d_i, \tag{5}$$

where d_i denotes the deadline of job i and $H(G_{ijk})$ is the unit step function. Similarly, the maximum time limit is set to the deadline of the pseudo-job. The energy term will exceed zero when a job is allocated, the run time is greater than d, i.e., when $V_{ijk} = 1$, $k - d_i > 0$, and $H(G_{ijk}) > 0$. The energy value grows exponentially with the associated time lag between d_i and k, given by $k - d_i$. Conversely, this energy term has a value of zero if $V_{ijk} = 1$ and $k - d_i \leqslant 0$. Accordingly, the final energy function with all constraints can be induced as shown in Eq. (6)

$$\begin{aligned} E = {} & \frac{C_1}{2}\sum_{i=1}^{N+1}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{\substack{i1=1,\\ i1\neq i}}^{N+1} V_{ijk}V_{i1jk} \\ & + \frac{C_2}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{\substack{j1=1,\\ j1\neq j}}^{M}\sum_{k1=1}^{T} V_{ijk}V_{ij1k1} \\ & + \frac{C_3}{2}\sum_{i=1}^{N+1}\left(\sum_{j=1}^{M}\sum_{k=1}^{T} V_{ijk} - P_i\right)^2 \\ & + \frac{C_4}{2}\sum_{j=1}^{M}\sum_{k=1}^{T}\left(\sum_{i=1}^{N+1} V_{ijk} - 1\right)^2 \\ & + \frac{C_5}{2}\sum_{i=1}^{N+1}\sum_{j=1}^{M}\sum_{k=1}^{T} V_{ijk}G_{ijk}^2 H(G_{ijk}), \end{aligned} \tag{6}$$

C_1, C_2, C_3, C_4, and C_5 represent weighting factors which like N, M and T, are assumed to be positive constants. Based on the discussion made above, the derived energy function has a minimum value of zero when all constraints are met with the defined problem.

Eq. (6) can be proved to be an appropriate Lyapunov function for the system. The proof of convergence can refer to Appendix A.

3. Competitive algorithm

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

Hopfield and Tank originally proposed the neural network method to solve the optimization problems in Hopfield and Tank (1986). The HNN algorithm is based on the gradient technique to get the problems solution and thus provides rapid convergence. Moreover, the HNN also

provides potential for parallel implementation. In Hopfield (1982) and Hopfield and Tank (1986), a circuit composed of simple analog amplifiers that implements this type of neural networks was proposed. Based on dynamic system theory, the Lyapunov function (Cohen & Grossberg, 1983; Hopfield & Tank, 1986) shown in Eq. (7) has verified the existence of stable states of the network system. The energy function representing the scheduling problem must be in the same format as the Lyapunov function and expended to a three-dimensional model as below

$$E = -\frac{1}{2}\sum_{x}\sum_{y}\sum_{z}\sum_{i}\sum_{j}\sum_{k} V_{xyz} W_{xyzijk} V_{ijk} + \sum_{i}\sum_{j}\sum_{k}\theta_{ijk} V_{ijk}, \tag{7}$$

V_{xyz} and V_{ijk} denote the neuron states, W_{xyzijk} represents the synaptic weight and the interconnection strength among neurons, and θ_{ijk} denotes the threshold value representing the bias input of the neuron. Additionally, the conventional HNN using the deterministic rule is displayed in Eq. (8) below to update the neuron state change. This rule is

$$V_{ijk}^{n+1} = \begin{cases} 1, & \text{if } Net_{ijk} > 0, \\ V_{ijk}^{n}, & \text{if } Net_{ijk} = 0, \\ 0, & \text{if } Net_{ijk} < 0. \end{cases} \tag{8}$$

Meanwhile, Net_{ijk} represents the net value of the neuron (i,j,k) obtained using the interconnection strength W_{xyzijk}, with the other neurons (x,y,z), and the bias input θ_{ijk} which is shown as follows:

$$Net_{ijk} = -\frac{\partial E}{\partial V_{ijk}} = \sum_{x}\sum_{y}\sum_{z} W_{xyzijk} V_{xyz} - \theta_{ijk}. \tag{9}$$

Instead of applying conventional deterministic rules to update the neuron states, this study used competition rule to decide the winning neuron among the set of neurons, i.e., the active neuron. As discussed previously, a Hopfield neural network applying a winner-take-all learning mechanism is called a competitive Hopfield neural network, CHNN. The competitive rule is adopted to construct neural networks that satisfy constraints in which "exactly one neuron among N" should be activated when the network reaches a stable state, and can be regarded as a *1-out-of-N* confine rule. Hence, the number of activated neurons during each time unit has to be exactly the number of machines when the neural network reaches a stable state.

Since a machine can only execute one job at a certain time in a subject scheduling problems, omitting the first C_1 and the fourth C_4 energy terms from the HNN energy function Eq. (6) generates a simplified energy function that satisfies the competitive constraint. Restated, the first C_1 and the fourth C_4 energy terms are handled implicitly in *1-out-of-N* competitive rule. The resulting energy function after simplification is given as follows:

$$E = \frac{C_2}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{\substack{j1=1,\\ j1\neq j}}^{M}\sum_{k1=1}^{T} V_{ijk} V_{ij1k1} + \frac{C_3}{2}\sum_{i=1}^{N+1}\left(\sum_{j=1}^{M}\sum_{k=1}^{T} V_{ijk} - P_i\right)^2 + \frac{C_5}{2}\sum_{i=1}^{N+1}\sum_{j=1}^{M}\sum_{k=1}^{T} V_{ijk} G_{ijk}^2 H(G_{ijk}). \tag{10}$$

The resulted energy function makes it apparent that this must be an appropriate Lyapunov function. The synaptic interconnection strength W_{xyzijk} and the bias input θ_{ijk} can be obtained by comparing Eq. (10) with Eq. (7) where

$$W_{xyzijk} = -C_2 * \delta(x,i) * (1 - \delta(y,j)) - C_3 * \delta(x,i) \tag{11}$$

and

$$\theta_{xyz} = -C_3 P_i + \frac{C_5}{2} * G^2 * H(G), \tag{12}$$

respectively, where

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

The CHNN imposed a competitive winner-take-all rule to update the neuron states. Neurons on the *same column* of a dedicated machine at a given time compete with one another to determine the winning neuron. The neuron that receives the highest 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. For example, there are four jobs to be processed in two machines as displayed in Fig. 2. If jobs 2 and 3 are assigned to machine 1, then the neuron activated ($V_{ijk} = 1$) is shown by a solid node. Therefore, the final network state has exactly one neuron at a time for each machine. The winner-take-all update rule of the neuron for the ith column is illustrated as follows:

$$V_{xjk} = \begin{cases} 1 & \text{if } Net_{xjk} = \underset{i=1\sim N+1}{\text{Max}}\, Net_{ijk}, \\ 0 & \text{otherwise}, \end{cases} \tag{13}$$

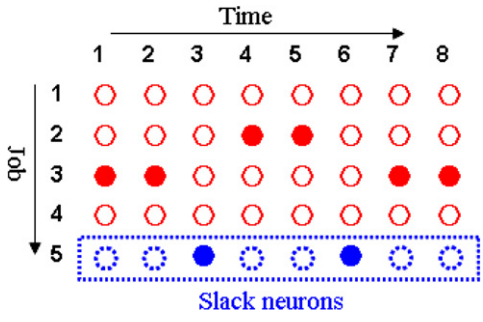

Fig. 2. Neuron states example for 4 jobs with pseudo-job activated.

where Net_{xjk} denotes the maximum total neuron input which is equivalent to the dynamic threshold on a McCulloch and Pitts neuron (Lee et al., 1992).

4. Experimental simulations

The simulations consider classes of scheduling problems with timing constraints. Several different timing constraints and various weighting factors were applied to the simulations. Table 1 shows the weighting factors constants of the energy function in Eq. (10) in simulation. Tables 2–5 show the different timing constraints of simulation cases, respectively. The simulation involves scheduling four or five jobs in two machine systems. In addition, a more complicated simulation case with 10 jobs in three machine systems is included as well. The simulation results were displayed by using a Gantt chart to graphically represent the job schedules. Cases 1 and 3 are the same simulation examples as in Chen and Huang (2001). These cases were included to facilitate the full machine utilization system scheduling study. Figs. 3 and 4 illustrate the resulting schedules of case 1 for the proposed algorithm and the scheme in Chen and Huang (2001), respectively. Moreover, different initial neuron states were simulated to understand better the response of the neural network to the scheduling problem. Figs. 5 and 6 display two of the resulting schedules correlating with different initial neuron states for case 2. This simulation example is the case for those machines which were under full usage. Meanwhile, Figs. 7 and 8 illustrate the resulting schedule of case 3 for different initial

Table 1
Weighing factors

Constants for CHNN		
C_2	C_3	C_5
1.35	0.55	1.3

Table 2
Timing matrix of 4 jobs on 2 machines (case 1)

	Time required	Time limit
Job 1	4	6
Job 2	3	4
Job 3	3	6
Job 4	2	3

Table 3
Timing matrix of 4 jobs on 2 machines (case 2)

	Time required	Time limit
Job 1	5	8
Job 2	4	8
Job 3	3	6
Job 4	2	3

Table 4
Timing matrix of 5 jobs on 2 machines (case 3)

	Time required	Time limit
Job 1	2	3
Job 2	5	8
Job 3	3	4
Job 4	4	8
Job 5	2	5

Table 5
Timing matrix of 10 Jobs on 3 processors (case 4, $A = 2$; case 5, $A = 5$)

	Time required	Time limit
Job 1	A	10
Job 2	3	5
Job 3	3	9
Job 4	2	5
Job 5	3	9
Job 6	2	6
Job 7	3	10
Job 8	2	5
Job 9	3	9
Job 10	4	10

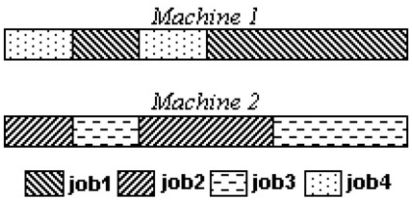

Fig. 3. Job assignment for case 1.

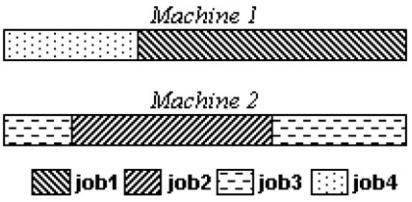

Fig. 4. Job assignment for case 1 using scheme in Chen and Huang (2001).

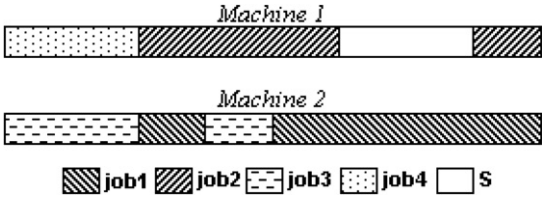

Fig. 5. Job assignment for case 2.

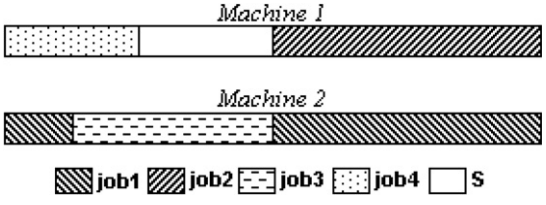

Fig. 6. Job assignment for case 2 with different initial states.

conditions with full machine usage. Figs. 8 and 9 display the full usage schedules of case 3 for this work and the method proposed in Chen and Huang (2001). Finally, Figs. 10 and 11 represent the resulting schedules correlating with the cases 4 and 5, respectively. Case 4 is the example of a machine which is not in full usage, while case 5 is the example of a machine which is in full usage. The job assignment of S as displayed in Figs. 5, 6, and 10 indicates the active slack neurons. That is, the machine does nothing at that time. To minimize the completion time of a machine, the jobs behind the slack neurons can be shifted forward if there are no other constraints. Additionally, Figs. 12 and 13 show the significant parts of the energy curves during neural network evolution. Fig. 14 demonstrates that the initial states are all set to all 0 or 1 for cases 2 and 4.

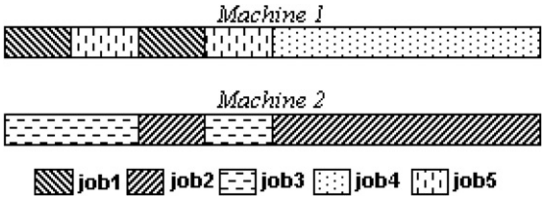

Fig. 7. Job assignment for case 3.

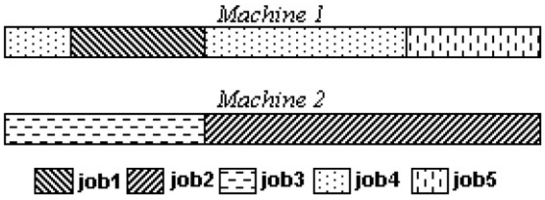

Fig. 8. Job assignment for case 3 with different initial states.

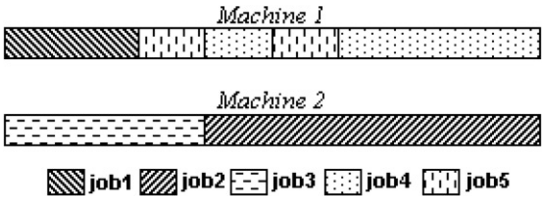

Fig. 9. Jobs assignment for case 3 using scheme in Chen and Huang (2001).

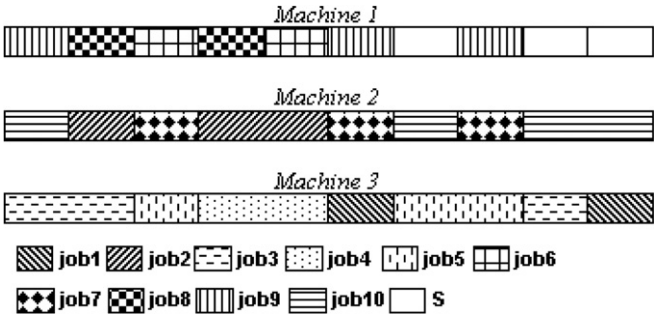

Fig. 10. Job assignment for case 4.

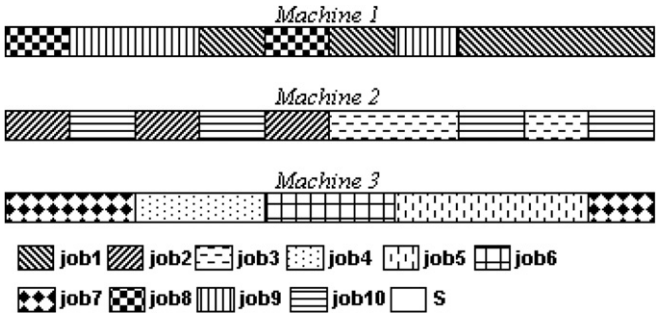

Fig. 11. Job assignment for case 5.

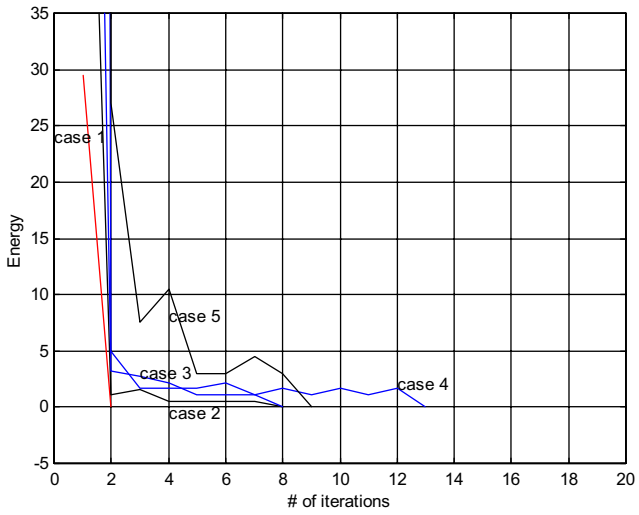

Fig. 12. Energy evolution of cases 1–5.

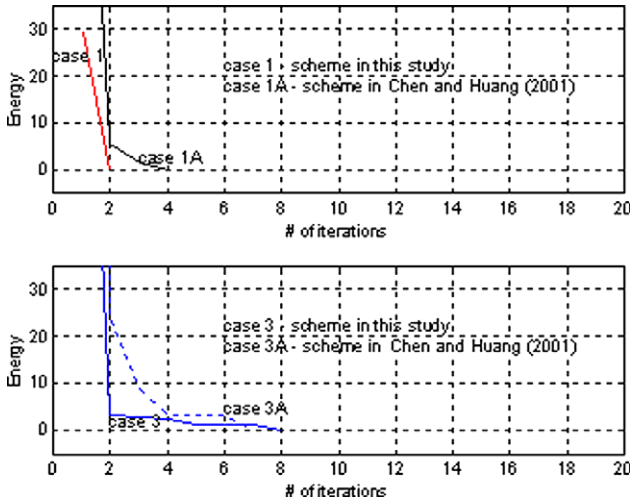

Fig. 13. Energy evolution of case 1 and 3 using different schemes.

Different initial states of neurons will generate a feasible solution. Notably, to guarantee convergence to a minimum, the neuron state update was performed sequentially with complete neuron update each time in the simulation (Hopfield, 1984; Takeda & Goodman, 1986).

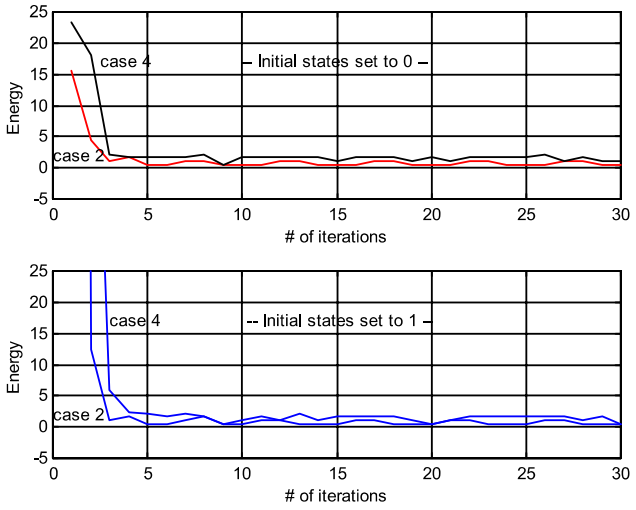

Fig. 14. Energy evolution of case 2 and 4 with initial states set to 0 and 1.

This study proposed an approach for solving timing constraint problems with full or not full machine usage problems or with different initial states of neurons. From these simulations, each job has a process time and deadline which were given in advance. The proposed method can solve the real-time job scheduling problem by addressing the problem constraint.

5. Conclusions

Hopfield used the quadratic energy function which resulted in the quadratic cost of the interconnection network and hence a poor scaling property. The competitive mechanism eliminated the constraint terms in the energy function, simplifying the network by reducing the interconnections among neurons (Chung et al., 1994), and this is shown in Eq. (10). Hence, the competitive scheme can help overcome the scaling problem.

This investigation illustrated an approach to map the problem constraint into the energy function of the competitive neural network containing slack neurons which were involved so as to resolve the timing constraints schedule problem for both non-fully utilized and fully-utilized systems. The simulation results demonstrated some significant consequences for this study, specifically the features of this work, when applied to the scheduling domain examined. These were as follows:

(1) The extra slack neurons which were added facilitated solving the studied problems with inequality constraints. The proposed method is workable and feasible in real-time job scheduling problems, even in fully or non-fully utilized scheduling problems.
(2) Convergence is initially state dependent, as displayed in Fig. 14. In many researches, they experienced unstable revolutions and produced no solutions. Distributing the initial states randomly can generally produce feasible schedules for the investigated scheduling problem.
(3) The entailed synaptic weight matrix in Eq. (11) has a symmetric (i.e., $W_{xyzijk} = W_{ijkxyz}$) property, but nevertheless has a self-feedback interconnection, indicating that $W_{xyzijk} \neq 0$. Therefore, the network may oscillate when it is updated (Hopfield & Tank, 1986; Takeda & Goodman, 1986). Consequently, a solution is not guaranteed, causing an inevitable oscillation procedure. In Takefuji and Lee (1991), Takefuji and Lee proposed a hysteresis binary neuron model to effectively suppress the oscillatory behaviors of neural dynamics for solving combinatorial optimization problems.

Moreover, weighting factor determination is a laborious work. This study did not employ a unique set of weighting matrices in our simulation, as listed in Table 1. Various sets of weighting factors were investigated. C_2, C_3, and C_5 can be set to 1.27–1.44, 0.51–0.56, and >0.54 in these simulations, respectively. C_2 and C_3 were tightly coupled since they dominate the synaptic of the network. Different sets of weighting factors may produce different neural network revolutions. However, the reduction of energy terms in this work also assisted in easing this annoying affair.

The HNN is frequently caught in a local minimum. The simulated annealing technique can effectively obtain a global minimum capable of escaping the local minimum. However, this approach requires a higher time complexity and is not solvable by analog circuits (Tank & Hopfield, 1986). An important feature of a scheduling algorithm is its efficiency or performance, i.e., how its execution time grows with the problem size. The parameter most relevant to the time that a neural network takes to find a solution is the number of iterations needed to converge to a solution. According to the simulation results, the proposed algorithm required an average of 5–20 epochs to converge. An epoch involves updating every column of the competitive Hopfield neural network. Consequently, this algorithm resulted in a $O((N+1)^2 * M^2 * T)$ complexity. Restated, the execution time was proportional to $O(N^2 * M^2 * T)$ for each epoch. Furthermore, finding the solution for a very large-scale system (very large N and/or very large M) is a drawback of the proposed model. Future works should examine how to reduce the complexities of solving the scheduling problems.

The energy function proposed herein works efficiently and can be applied to similar cases of investigated scheduling problems. Among which, the jobs are independent without requiring communication or utilizing memory resources to exchange data. However, the required network implementation depends on the intended applications. The competitive scheme combined with slack neurons suggests that the way to apply this kind of scheduling has inequality constraints.

This work concentrated mainly on solving job scheduling without ready time consideration or resource constraints.

For more practical implementations, different and more complicated scheduling problems can be further investigated in future researches by applying the proposed algorithm. Such problems include preemptive multi-job scheduling on multi-machine systems with multi-constraints such as deadline and resource constraints, or jobs with precedence relationship and synchronized consideration. The problem can be further extended to involve the temporal relationship of ready time or priority for each job. Correspondingly, the energy function in our work can be modified by using additional energy terms to satisfy extra requirements. Future research endeavors should address these issues more thoroughly.

Appendix A. Convergence of the energy function

This appendix proves the convergence of the derived energy function mathematically for the investigated problem. The simplified energy function (Eq. (10)) is re-listed below

$$E = \frac{C_2}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{\substack{j1=1,\\ j1\neq j}}^{M}\sum_{k1=1}^{T} V_{ijk}V_{ij1k1} + \frac{C_3}{2}\sum_{i=1}^{N+1}\left(\sum_{j=1}^{M}\sum_{k=1}^{T} V_{ijk} - P_i\right)^2 + \frac{C_5}{2}\sum_{i=1}^{N+1}\sum_{j=1}^{M}\sum_{k=1}^{T} V_{ijk}G_{ijk}^2 H(G_{ijk}). \tag{A.1}$$

For clearness, the energy function can be represented as follows (Eq. (A.2)):

$$E = \frac{C_2}{2}\left(\sum_{i=1}^{N}\sum_{\substack{j1=1,\\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{imn}V_{ij1k1} + \sum_{i=1}^{N}\sum_{\substack{j=1,\\ j\neq m}}^{M}\sum_{\substack{k=1,\\ k\neq n}}^{T}\sum_{\substack{j1=1,j1\neq j,\\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{ijk}V_{ij1k1}\right) + \frac{C_3}{2}\left(\sum_{i=1}^{N+1}\left(V_{imn} + \left(\sum_{\substack{j=1,\\ j\neq m}}^{M}\sum_{\substack{k=1,\\ k\neq n}}^{T} V_{ijk} - P_i\right)\right)^2\right) + \frac{C_5}{2}\left(\sum_{i=1}^{N+1} V_{imn}G_{imn}^2 H(G_{imn}) + \sum_{i=1}^{N+1}\sum_{\substack{j=1,\\ j\neq m}}^{M}\sum_{\substack{k=1,\\ k\neq n}}^{T} V_{ijk}G_{ijk}^2 H(G_{ijk})\right). \tag{A.2}$$

Expanding the C_3 terms as follows:

$$E = \frac{C_2}{2}\left(\sum_{i=1}^{N}\sum_{\substack{j1=1,\\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{imn}V_{ij1k1} + \sum_{i=1}^{N}\sum_{\substack{j=1,\\ j\neq m}}^{M}\sum_{\substack{k=1,\\ k\neq n}}^{T}\sum_{\substack{j1=1,j1\neq j,\\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{ijk}V_{ij1k1}\right) + \frac{C_3}{2}\left(\sum_{i=1}^{N+1}\left((V_{imn})^2 + 2V_{imn}\left(\sum_{\substack{j=1,\\ j\neq m}}^{M}\sum_{\substack{k=1,\\ k\neq n}}^{T} V_{ijk} - P_i\right) + \left(\sum_{\substack{j=1,\\ j\neq m}}^{M}\sum_{\substack{k=1,\\ k\neq n}}^{T} V_{ijk} - P_i\right)^2\right)\right) + \frac{C_5}{2}\left(\sum_{i=1}^{N+1} V_{imn}G_{imn}^2 H(G_{imn}) + \sum_{i=1}^{N+1}\sum_{\substack{j=1,\\ j\neq m}}^{M}\sum_{\substack{k=1,\\ k\neq n}}^{T} V_{ijk}G_{ijk}^2 H(G_{ijk})\right). \tag{A.a}$$

The energy function can be written as following two parts:

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

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}. Restated, E_{mn} is the summation of the energy term corresponding to neuron state V_{imn}. These V_{imn} terms related to the neuron represents a process i being executed at a specific processor m at a certain time n. Restated, V_{imn} is the neuron on the ith row (job) and the nth 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 nth column on the processor m before updating, that is,

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

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,

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

According to Eq. (9), the total input of the neuron (i,j,k) is obtained, i.e., net value, which is as follows (Eq. (A.3)):

$$\begin{aligned} Net_{ijk} &= -\frac{\partial E}{\partial V_{ijk}} \\ &= -\frac{C_2}{2}\sum_{\substack{j1=1,\\ j1\neq j}}^{M}\sum_{k1=1}^{T} V_{ij1k1} - C_3(V_{ijk} - P_i) \\ &\quad - \frac{C_5}{2}G_{ijk}^2 H(G_{ijk}). \end{aligned} \tag{A.3}$$

The active neuron, based on the winner-take-all update rule as in Eq. (A.3), is the one with the maximum net value on each column in each update, that is

$$Net_{qmn}^{(t+1)} = \operatorname*{Max}_{i=1\sim N+1} Net_{imn}^{(t+1)}.$$

This equation implies that

$$Net_{qmn}^{(t+1)} > Net_{lmn}^{(t+1)}, \tag{A.4}$$

where Net_{qmn} and Net_{lmn} are obtained based on Eq. (A.3) as follows:

$$Net_{qmn} = -\frac{C_2}{2}\sum_{\substack{j1=1,\\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{qj1k1} - C_3(V_{qmn} - P_q) - \frac{C_5}{2}G_{qmn}^2 H(G_{qmn}) \tag{A.5}$$

and

$$Net_{lmn} = -\frac{C_2}{2}\sum_{\substack{j1=1,\\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{lj1k1} - C_3(V_{lmn} - P_l) - \frac{C_5}{2}G_{lmn}^2 H(G_{lmn}). \tag{A.6}$$

Investigating Eq. (A.2), 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. Restated, the E_{other} is canceled out, ΔE is displayed as follows:

$$\begin{aligned}\Delta E &= E_{mn}^{(t+1)} - E_{mn}^{(t)} \\ &= \frac{C_2}{2}\left(\sum_{\substack{j1=1,\\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{qmn}^{(t+1)} V_{qj1k1} - \sum_{\substack{j1=1,\\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{lmn}^{(t)} V_{lj1k1} + \sum_{\substack{i=1,\\ i\neq q}}^{N}\sum_{\substack{j1=1,\\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{imn}^{(t+1)} V_{ij1k1} - \sum_{\substack{i=1,\\ i\neq l}}^{N}\sum_{\substack{j1=1,\\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{imn}^{(t)} V_{ij1k1}\right) \\ &\quad + \frac{C_3}{2}\left(((V_{qmn}^{(t+1)})^2 - 2V_{qmn}^{(t+1)}P_q) - ((V_{lmn}^{(t)})^2 - 2V_{lmn}^{(t)}P_l) + 2\sum_{i=1}^{N+1} V_{imn}^{(t+1)}\sum_{\substack{j=1,\\ j\neq m}}^{M}\sum_{\substack{k=1,\\ k\neq n}}^{T} V_{ijk} - 2\sum_{i=1}^{N+1} V_{imn}^{(t)}\sum_{\substack{j=1,\\ j\neq m}}^{M}\sum_{\substack{k=1,\\ k\neq n}}^{T} V_{ijk}\right) \\ &\quad + \frac{C_5}{2}\left(V_{qmn}^{(t+1)}G_{qmn}^2 H(G_{qmn}) - V_{lmn}^{(t)}G_{lmn}^2 H(G_{lmn}) + \sum_{\substack{i=1,\\ i\neq q}}^{N+1} V_{imn}^{(t+1)}G_{imn}^2 H(G_{imn}) - \sum_{\substack{i=1,\\ i\neq l}}^{N+1} V_{imn}^{(t)}G_{imn}^2 H(G_{imn})\right) \\ &\quad + \sum_{\substack{i=1,\\ i\neq q}}^{N+1} V_{imn}^{(t+1)}G_{imn}^2 H(G_{imn}) - \sum_{\substack{i=1,\\ i\neq l}}^{N+1} V_{imn}^{(t)}G_{imn}^2 H(G_{imn}).\end{aligned} \tag{A.7}$$

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 demonstrated in Eq. (A.7), is rewritten as follows:

$$\begin{aligned}\Delta E &= \frac{C_2}{2}\left(\sum_{\substack{j1=1,\\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{qj1k1} - \sum_{\substack{j1=1,\\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{lj1k1}\right) + \frac{C_3}{2}\left((1-P_q)^2 - (1-P_l)^2 + \sum_{\substack{i=1,\\ i\neq q}}^{N+1} P_i^2 - \sum_{\substack{i=1,\\ i\neq l}}^{N+1} P_i^2\right) \\ &\quad + \frac{C_5}{2}(G_{qmn}^2 H(G_{qmn}) - G_{lmn}^2 H(G_{lmn})).\end{aligned} \tag{A.8}$$

Investigating Eq. (A.8), rearrange the C_3 term as follows:

$$\frac{C_3}{2}\left(1 - 2P_q + \left(P_q^2 + \sum_{\substack{i=1,\\ i\neq q}}^{N+1} P_i^2\right) - 1 + 2P_l - \left(P_l^2 + \sum_{\substack{i=1,\\ i\neq l}}^{N+1} P_i^2\right)\right) = C_3(P_l - P_q). \tag{A.9}$$

Subtracting Eq. (A.5) from Eq. (A.6) at $(t+1)$th iteration yields the following:

$$\begin{aligned}Net_{lmn}^{(t+1)} - Net_{qmn}^{(t+1)} &= \frac{C_2}{2}\left(\sum_{\substack{j1=1,\\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{qj1k1} - \sum_{\substack{j1=1,\\ j1\neq m}}^{M}\sum_{k1}^{T} V_{lj1k1}\right) + C_3(P_l - P_q) + C_3(V_{qmn}^{(t+1)} - V_{lmn}^{(t+1)}) \\ &\quad + \frac{C_5}{2}(G_{qmn}^2 H(G_{qmn}) - G_{lmn}^2 H(G_{lmn})).\end{aligned} \tag{A.10}$$

Accordingly, the energy changes between neuron update equals the net value change minus C_3. That is

$$\Delta E = Net_{lmn}^{(t+1)} - Net_{qmn}^{(t+1)} - C_3. \tag{A.11}$$

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

References

Cardeira, C., & Mammeri, Z. (1994). Neural networks for multiprocessor real-time scheduling. In *IEEE proceedings of the sixth Euromicro workshop on real-time systems* (pp. 59–64).

Cardeira, C., & Mammeri, Z. (1996). Neural network versus max-flow algorithms for multi-processor real-time scheduling, real-time systems. In *Proceedings of the eighth Euromicro workshop* (pp. 175–180).

Cardeira, C., & Mammeri, Z. (1997). Handling precedence constraints with neural network based real-time scheduling algorithms. In *Proceedings of ninth Euromicro workshop on real-time systems* (pp. 207–214).

Chen, R. M., & Huang, Y. M. (1998). Multiconstraint task scheduling in multiprocessor system by neural network. In *Proceedings of the IEEE*

tenth international conference on tools with artificial intelligence, Taipei (pp. 288–294).

Chen, R. M., & Huang, Y. M. (2001). Competitive neural network to solve scheduling problem. *Neurocomputing, 37*(1–4), 177–196.

Chung, P. C., Tsai, C. T., Chen, E. L., & Sun, Y. N. (1994). Polygonal approximation using a competitive Hopfield neural network. *Pattern Recognition, 27*, 1505–1512.

Cohen, M., & Grossberg, S. (1983). Absolute stability of goal pattern formation and parallel memory storage by competitive neural network. *IEEE Transaction on System, Man, and Cybernetics, 13*, 815–826.

Dixon, M. W., Cole, G. R., & Bellgard, M. I. (1995). Using the Hopfield model with mean-field annealing to solve the routing problem in a communication network. In *International conference on neural networks* (vol. 5, pp. 2652–2657).

Foo, Y. P. S., & Takefuji, T. (1998). Integer linear programming neural networks for job-shop scheduling. In *IEEE international conference on neural networks* (vol. 2, pp. 341–348).

Hanada, A., & Ohnishi, K. (1993). Near optimal jobshop scheduling using neural network parallel computing. In *Proceedings of the international conference on industrial electronics, control, and instrumentation* (vol. 1, pp. 315–320).

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Science, 79*, 2554–2558.

Hopfield, J. J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Science, 81*, 3088–3092.

Hopfield, J. J., & Tank, D. W. (1985). Neural computation of decision in optimization problems. *Biological Cybernetics, 52*, 141–152.

Hopfield, J. J., & Tank, D. W. (1986). Computing with neural circuits: A model. *Science, 233*, 625–633.

Huang, Y. M., & Chen, R. M. (1999). Scheduling multiprocessor job with resource and timing constraints using neural network. *IEEE Transactions on System, Man and Cybernetics, Part B, 29*(4), 490–502.

Lee, K. C., Funabiki, N., & Takefuji, Y. (1992). A parallel improvement algorithm for the bipartite subgraph problem. *IEEE Transactions on Neural Networks, 3*(1), 139–145.

Liu, C., & Layland, J. (1973). Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM, 20*(l), 46–61.

Park, J. G., Park, J. M., Kim, D. S., Lee, C. H., Suh, S. W., & Han, M. S. (1994). Dynamic neural network with heuristic. In *IEEE international conference on neural networks* (vol. 7, pp. 4650–4654).

Silva, M. P., Cardeira, C., & Mammeri Z. (1997). Solving real-time scheduling problems with Hopfield-type neural networks. In *EUROMICRO'97 new frontiers of information technology, proceedings of the 23rd EUROMICRO conference* (pp. 671–678).

Tagliarini, G. A., Christ, J. F., & Page, E. W. (1991). Optimization using neural networks. *IEEE Transaction on Computers, 40*(12), 1347–1358.

Takeda, M., & Goodman, J. W. (1986). Neural networks for computation: number representation and programming complexity. *Applied Optics, 25*, 3033–3046.

Takefuji, Y., & Lee, K. C. (1991). An artificial hysteresis binary neuron: a model suppressing the oscillatory behaviors of neuron dynamics. *Biological Cybernetics, 64*, 353–356.

Tank, D., & Hopfield, J. J. (1986). Simple neural optimization networks: an A/D converter, signal decision circuit and a linear programming circuit. *IEEE Transactions on Circuits and Systems, 33*(5), 533–541.

Uchiyama, T., & Arbib, M. A. (1994). Color image segmentation using competitive learning. *IEEE Transactions on Pattern Analysis Machine Intelligence, 16*(12), 1197–1206.

Willems, T. M., & Rooda, J. E. (1994). Neural networks for job-shop scheduling. *Control Engineering Practice, 2*(1), 31–39.

Zhang, C. S., Yan, P. F., & Chang, T. (1991). Solving job-shop scheduling problem with priority using neural network. In *IEEE international conference on neural networks* (pp. 1361–1366).