ELSEVIER Neurocomputing 37 (2001) 177–196

NEUROCOMPUTING

www.elsevier.com/locate/neucom

Competitive neural network to solve scheduling problems

Ruey-Maw Chen, Yueh-Min Huang*

Department of Engineering Science, National Cheng-Kung University, Tainan 701, Taiwan, ROC

Accepted 1 November 2000

Abstract

Most scheduling problems have been demonstrated to be NP-complete problems. 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. Hopfield neural networks, although providing rapid convergence to the solution, require extensive effort to determine coefficients. A competitive learning rule provides a highly effective means of attaining a sound solution and 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 problem with no process migration, time constraints (execution time and deadline), and limited resources. 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. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Scheduling; Winner-take-all; Competitive learning; Hopfield neural network

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.

* Corresponding author. Tel.: + 886-62757575 ext. 63336; fax: + 886-62766549.
E-mail address: raymond@mail.ncku.edu.tw (Y.-M. Huang).

This fact implies that an optimal solution for a large scheduling problem is rather time consuming. The traveling salesman problem (TSP) is a typical NP-complete problem, comprising a Hamiltonian cycle, 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. However, in certain multiprocessor applications, task scheduling is not merely restricted to the timing constraint. For instance, in [7], the *display system* on an advanced avionics 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. Additionally, tasks utilize shareable resources such as the *triple-channel display component* for output display data, and the *cooperative memory component* for data exchange. Tasks do not use the same resource simultaneously. 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 and resource 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. Intrinsically, the operation of the Hopfield neural network is a relaxation process that allows an energy function to reach an optimum solution with less computation. The energy function used in the Hopfield neural network is an appropriate Lyapunov function. Many researchers have recently applied this method to various applications. Gallone, Charpillet, and Alexandre presented a set of non-preemptive tasks on a single machine, scheduled by applying the *k-out-of-N* rule to the Hopfield neural network [9]. Meanwhile, Nasrabadi and Choo used the Hopfield neural network for stereo vision correspondence [10]. Furthermore, Dixon, Cole, and Belgard employed the Hopfield neural network with mean-field annealing to solve the shortest path problem in a communication network [11]. 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 et al. [12] proposed a competitive Hopfield neural network for polygonal approximation. Similarly, Lin et al. [13] also applied a competitive Hopfield neural network to demonstrate the promising results in medical image segmentation. Furthermore, Uchiyama and Arbib [14] 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, forcing 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 [15].

Previously, we conducted a series of study to solve the scheduling problem [7]. A multiconstraint scheduling problem for a multiprocessor system was investigated, in which the process is segmented into numerous subprocesses. Although these subprocesses are preemptive, no process migration is allowed. The Hopfield neural network scheme and mean-field annealing technique are utilized to obtain the adequate schedule [7]. Additionally, our earlier work [16] studied the convergence rates corresponding to the cooling procedures of the mean-field annealing technique in obtaining the scheduling results for the problem in [7]. That investigation proposed a modified cooling schedule to accelerate convergence rate for the investigated problem. Furthermore, the fuzzy c-maens method used in clustering was applied to investigate the scheduling problem, owing to the concept of the scheduling problem resembling the clustering problem. Thus, a non-preemptive task scheduling problem was resolved by incorporating the fuzzy c-maens strategy into the Hopfield neural network [17].

In light of above developments, this work investigates the job schedule problem of a multiprocess on a multiprocessor that includes timing as well as resource constraints, via CHNN. An energy function designed to illustrate the timing and resource 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, then 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, as discussed in our previous study [7] and summarized herein. 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. Additionally, although calculating execution time is often difficult, the execution time of each job can be estimated by calculating the machine cycles or employing some heuristic rule methods. The constraints imposed on the model proposed herein are a deadline and an execution time for each job with limited available system resources. Moreover, a resource instant is not permitted to migrate to any other machine. 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 and limited numbers of non-preemptive resources are interesting. This work focuses on a multiprocess in a multiprocessor system and attempts to obtain a set of schedules.

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

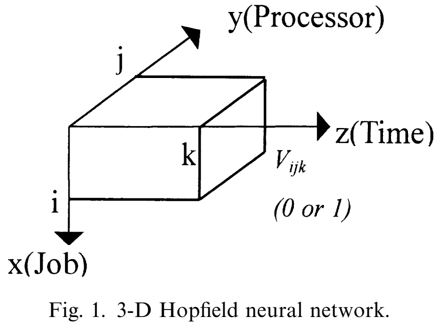


Fig. 1. 3-D Hopfield neural network.

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:

$$\begin{aligned} E = {} & \frac{C_1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{\substack{i1=1,\\ i1\neq i}}^{N} V_{ijk}V_{i1jk} + \frac{C_2}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{\substack{j1=1,\\ j1\neq j}}^{M}\sum_{k1=1}^{T} V_{ijk}V_{ij1k1} \\ & + \frac{C_3}{2}\sum_{i=1}^{N}\left(\sum_{j=1}^{M}\sum_{k=1}^{T} V_{ijk} - P_i\right)^2 + \frac{C_4}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\left(\sum_{k=1}^{T} V_{ijk} - 1\right)^2 \\ & + \frac{C_5}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T} V_{ijk}G_{ijk}^2 H(G_{ijk}) \\ & + \frac{C_6}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{\substack{i1=1,\\ i1\neq i}}^{N}\sum_{\substack{j1=1,\\ j1\neq j}}^{M}\sum_{s=1}^{F} V_{ijk}R_{is}V_{i1j1k}R_{i1s}, \end{aligned} \qquad (1)$$

where

$$G_{ijk} = k - d_i,$$

$$H(G_{ijk}) = \begin{cases} 1 & \text{if } G_{ijk} > 0, \\ 0 & \text{if } G_{ijk} \le 0, \end{cases}$$

and where 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, and F denotes the quantity of shareable resources. These weighting factors, N, M, T, and F, are assumed to be positive constants herein.

The C_1 energy term confines a processor j to executing only one process, say i or $i1$, 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. Since a processor prohibits simultaneous resource preemption and resource sharing, the C_6 energy term is included to provide this inhibition. In this term, F denotes the quantity of available resource instances, while R_{is} and $Ri1s$ represent process i and $i1$ requests for resource s, respectively, as shown in Table 2. $R_{is} = 1$ means that process i requires resource s, so $Ri1s = 1$ for process $i1$. Inspecting this energy term, when two distinct jobs are scheduled (say $V_{ijk} = 1$ and $V_{i1j1k} = 1$) to be executed at time k on different machines j and $j1$, then because machines j and $j1$ cannot utilize the same resource at time k, either R_{is} or $Ri1s$ must be zero. Implying that this energy term value becomes zero. 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. Therefore, the energy function leads to convergence during network evolution, and competitive HNN can adequately solve scheduling problems, as depicted in the following section.

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 [18]. Essentially, the HNN algorithm is based on the gradient technique, thus providing rapid convergence. The HNN also provides potential for parallel implementation. This scheme is basically an optimization-based relaxation process; in which the state evolutions of the HNN are based on the energy decrease. Hence, the HNN is suitable for solving optimization problems, and has been extensively used to solve various optimization problems. Based on dynamic system theory, the Liapunov function [16] [19] shown in Eq. (2), has verified the existence of stable states of the network system. This function is used in HNN to ensure the convergence of the network system. Restated, the 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. This neuron state has binary value 0 or 1, in accordance with the problem requirements. Additionally, the HNN employs the deterministic rule to update the neuron state change. This deterministic rule is displayed in Eq. (3) below:

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

where V_{xyz}^{n} and V_{xyz}^{n+1} denote the state values for the nth and $(n+1)$th iteration, respectively. 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 [19]. 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 C_1 and C_4 energy terms from the HNN energy function (Eq. (1)) yields a simplified energy function and satisfies the 1-*out-of-N* rule, i.e. the competitive constraint. Restated, the C_1 and C_4 energy terms are handled explicitly. The resulting energy function for CHNN is highlighted as follows:

$$\begin{aligned} E = {} & \frac{C_2}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{\substack{j1=1 \\ j1\neq j}}^{M}\sum_{k1=1}^{T} V_{ijk}V_{ij1k1} + \frac{C_3}{2}\sum_{i=1}^{N}\left(\sum_{j=1}^{M}\sum_{k=1}^{T} V_{ijk} - P_i\right)^2 \\ & + \frac{C_5}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T} V_{ijk}G_{ijk}^2 H(G_{ijk}) \\ & + \frac{C_6}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{T}\sum_{\substack{i1=1 \\ i1\neq i}}^{N}\sum_{\substack{j1=1 \\ j1\neq j}}^{M}\sum_{s=1}^{F} V_{ijk}R_{is}V_{i1j1k}R_{i1s}. \end{aligned} \tag{5}$$

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:

$$\begin{aligned} W_{xyzijk} = {} & -C_2\delta(x,i)(1-\delta(y,j)) - C_3\delta(x,i) \\ & - C_6(1-\delta(x,i))(1-\delta(y,j))\delta(z,k)\sum_{s} R_{xs}R_{is}, \end{aligned} \tag{6}$$

and

$$\theta_{xyz} = -C_3 P_i + \frac{C_5}{2} G^2 H(G) + C_t k \tag{7}$$

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.

Similarly, the total input to the neuron (i, j, k) is obtained based on Eq. (4), rewritten as follows:

$$Net_{ijk} = -\frac{\partial E}{\partial V_{ijk}} = \sum_x \sum_y \sum_z W_{xyzijk} V_{xyz} - \theta_{ijk}.$$

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 total input 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:

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

where Net_{xjk} is the maximum total neuron input, and is equivalent to the dynamic threshold on a MaCulloch–Pitts neuron [15].

The algorithm of the competitive HNN is summarized as follows:

(1) randomly set the initial neuron states;
(2) define the memory synaptic weights W_{ij} and threshold values θ_j according to Eqs. (6) and (7);
(3) apply Eq. (4) to calculate the total neuron input. Impose the winner-take-all rule as depicted in Eq. (8) to decide the output neuron state based on the assumed initial value; and
(4) replace the random initial states with the output neuron states obtained in step (3). Repeat the iterations in steps (3) and (4) iteration until no state change occurs in any iteration.

Notably, the second step is referred to as the storage (learning) phase, while the third step is the recalling (searching) phase.

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 (Eq. (5))

is shown below for reference:

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

$$+ \frac{C_5}{2} \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{T} V_{ijk} G_{ijk}^2 H(G_{ijk})$$

$$+ \frac{C_6}{2} \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{k=1}^{T} \sum_{\substack{i1=1 \\ i1 \neq i}}^{N} \sum_{\substack{j1=1 \\ j1 \neq j}}^{M} \sum_{s=1}^{F} V_{ijk} R_{is} V_{i1j1k} R_{i1s}.$$

The neuron (i, j, k) obtains the total input, i.e. net value, which is as follows (Eq. (9)):

$$\begin{aligned} \mathrm{Net}_{ijk} &= -\frac{\partial E}{\partial V_{ijk}} \\ &= -\frac{C_2}{2} \sum_{\substack{j1=1 \\ j1 \neq j}}^{M} \sum_{k1=1}^{T} V_{ij1k1} - C_3(V_{ijk} - P_i) \\ &\quad - \frac{C_5}{2} G_{ijk}^2 H(G_{ijk}) - \frac{C_6}{2} \sum_{\substack{i1=1 \\ i1 \neq i}}^{N} \sum_{\substack{j1=1 \\ j1 \neq j}}^{M} \sum_{s=1}^{F} R_{is} V_{i1j1k} R_{i1s}. \end{aligned} \tag{9}$$

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)):

$$\begin{aligned} E = {} & \frac{C_2}{2} \Bigg(\sum_{i=1}^{N} \sum_{\substack{j1=1 \\ j1=m}}^{M} \sum_{\substack{k1=1 \\ k1 \neq n}}^{T} V_{imn} V_{ij1k1} \\ & + \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq m}}^{M} \sum_{\substack{k=1 \\ k \neq n}}^{T} \sum_{\substack{j1=1 \\ j1 \neq j, j1 \neq m}}^{M} \sum_{k1=1}^{T} V_{ijk} V_{ij1k1} \Bigg) \\ & + \frac{C_3}{2} \left(\sum_{i=1}^{N} (V_{imn} - P_i)^2 + \sum_{i=1}^{N} \left(\sum_{\substack{j=1 \\ j \neq m}}^{M} \sum_{\substack{k=1 \\ k \neq n}}^{T} V_{ijk} - P_i \right)^2 \right) \\ & + \frac{C_5}{2} \left(\sum_{i=1}^{N} V_{imn} G_{imn}^2 H(G_{imn}) + \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq m}}^{M} \sum_{\substack{k=1 \\ k \neq n}}^{T} V_{ijk} G_{ijk}^2 H(G_{ijk}) \right) \\ & + \frac{C_5}{2} \Bigg(\sum_{i=1}^{N} \sum_{\substack{i1=1 \\ i1 \neq i}}^{N} \sum_{\substack{j=1 \\ j \neq m}}^{M} \sum_{s=1}^{F} V_{imn} R_{is} V_{i1j1n} R_{i1s} \\ & + \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq m}}^{M} \sum_{k=1k \neq n}^{T} \sum_{\substack{i1=1 \\ i1 \neq i}}^{N} \sum_{\substack{j1=1 \\ j1 \neq m, j1 \neq m}}^{M} \sum_{s=1}^{F} V_{ijk} R_{is} V_{i1j1k} R_{i1s} \Bigg), \end{aligned} \tag{10}$$

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

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 ith row (job) and nth 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 tth iteration, the V_{lmn} is supposed to be the only active neuron (l, m, n) in the nth column on processor m before updating, that is

$$V_{lmn}^{(t)} = 1$$

and

$$V_{imn}^{(t)} = 0 \quad \text{for } i \neq l.$$

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

$$V_{qmn}^{(t+1)} = 1$$

and

$$V_{imn}^{(t+1)} = 0 \quad \text{for } i \neq q.$$

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

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

This equation implies that

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

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

$$\begin{aligned} Net_{qmn} = & -\frac{C_2}{2}\sum_{\substack{j1=1\\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{qj1k1} - C_3(V_{qmn} - P_q) \\ & -\frac{C_5}{2}G_{qmn}^2 H(G_{qmn}) - \frac{C_6}{2}\sum_{\substack{i1=1\\ i1\neq q}}^{N}\sum_{\substack{j1=1\\ j1\neq m}}^{M}\sum_{s=1}^{F} R_{qs}V_{i1j1n}R_{i1s} \end{aligned} \tag{12}$$

and

$$\begin{aligned} Net_{lmn} = & -\frac{C_2}{2}\sum_{\substack{j1=1\\ j1\neq m}}^{M}\sum_{k1=1}^{T} V_{lj1k1} - C_3(V_{lmn} - P_l) \\ & -\frac{C_5}{2}G_{lmn}^2 H(G_{lmn}) - \frac{C_6}{2}\sum_{\substack{i1=1\\ i1\neq l}}^{N}\sum_{\substack{j1=1\\ j1\neq m}}^{M}\sum_{s=1}^{F} R_{ls}V_{i1j1n}R_{i1s}. \end{aligned} \tag{13}$$

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

$$\begin{aligned}
\Delta E &= E_{mn}^{(t+1)} - E_{mn}^{(t)} \\
&= \frac{C_2}{2}\left(\sum_{\substack{j1=1 \\ j1 \neq m}}^{M} \sum_{k1=1}^{T} V_{qmn}^{(t+1)} V_{qj1k1} - \sum_{\substack{j1=1 \\ j1 \neq m}}^{M} \sum_{k1=1}^{T} V_{lmn}^{(t)} V_{lj1k1} \right. \\
&\quad \left. + \sum_{\substack{i=1 \\ i \neq q}}^{N} \sum_{\substack{j1=1 \\ j1 \neq m}}^{M} \sum_{k1=1}^{T} V_{imn}^{(t+1)} V_{ij1k1} - \sum_{\substack{i=1 \\ i \neq l}}^{N} \sum_{\substack{j1=1 \\ j1 \neq m}}^{M} \sum_{k1=1}^{T} V_{imn}^{(t)} V_{ij1k1} \right) \\
&\quad + \frac{C_3}{2}\left((V_{qmn}^{(t+1)} - P_q)^2 - (V_{lmn}^{(t)} - P_l)^2 \right. \\
&\quad \left. + \sum_{\substack{i=1 \\ i \neq q}}^{N} (V_{imn}^{(t+1)} - P_i)^2 - \sum_{\substack{i=1 \\ i \neq l}}^{N} (V_{imn}^{(t)} - Pi)^2 \right) \\
&\quad + \frac{C_5}{2}\left(V_{qmn}^{(t+1)} G_{qmn}^2 H(G_{qmn}) - V_{lmn}^{(t)} G_{lmn}^2 H(G_{lmn}) \right. \\
&\quad \left. + \sum_{\substack{i=1 \\ i \neq q}}^{N} V_{imn}^{(t+1)} G_{imn}^2 H(G_{imn}) - \sum_{\substack{i=1 \\ i \neq l}}^{N} V_{imn}^{(t)} G_{imn}^2 H(G_{imn}) \right) \\
&\quad + \frac{C_6}{2}\left(\sum_{\substack{i1=1 \\ i1 \neq q}}^{N} \sum_{\substack{j1=1 \\ j1 \neq m}}^{M} \sum_{s=1}^{F} V_{qmn}^{(t+1)} R_{qs} V_{i1j1n} R_{i1s} \right. \\
&\quad - \sum_{\substack{i1=1 \\ i1 \neq l}}^{N} \sum_{\substack{j1=1 \\ j1 \neq m}}^{M} \sum_{s=1}^{F} V_{lmn}^{(t)} R_{ls} V_{i1j1n} R_{i1s} \\
&\quad + \sum_{\substack{i=1 \\ i \neq q}}^{N} \sum_{\substack{i1=1 \\ i1 \neq i, i1 \neq q}}^{N} \sum_{\substack{j1=1 \\ j1 \neq m}}^{M} \sum_{s=1}^{F} V_{imn}^{(t+1)} R_{is} V_{i1j1n} R_{i1s} \\
&\quad \left. - \sum_{\substack{i=1 \\ i \neq l}}^{N} \sum_{\substack{i1=1 \\ i1 \neq i, i1 \neq l}}^{N} \sum_{\substack{j1=1 \\ j1 \neq m}}^{M} \sum_{s=1}^{F} V_{imn}^{(t)} R_{is} V_{i1j1n} R_{i1s} \right). \qquad (14)
\end{aligned}$$

Since $V_{qmn}^{(t+1)} = 1$, $V_{imn}^{(t+1)} = 0 (i \neq q)$, $V_{lmn}^{(t)} = 1$, and $V_{imn}^{(t)} = 0 (i \neq l)$, the energy change difference demonstrated in Eq. (14), is rewritten as follows:

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

$$+\frac{C_5}{2}(G_{qmn}^2 H(G_{qmn}) - G_{lmn}^2 H(G_{lmn}))$$
$$+\frac{C_6}{2}\left(\sum_{\substack{i1=1\\ i1\neq q}}^{N}\sum_{\substack{j1=1\\ j1\neq m}}^{M}\sum_{s=1}^{F} R_{qs}V_{i1j1n}R_{i1s}\right.$$
$$\left.-\sum_{\substack{i1=1\\ i1\neq l}}^{N}\sum_{\substack{j1=1\\ j1\neq m}}^{M}\sum_{s=1}^{F} R_{ls}V_{i1j1n}R_{i1s}\right). \tag{15}$$

Examining Eq. (15), the C_3 term is rearranged as follows:

$$\frac{C_3}{2}\left(1-2P_q+\left(P_q^2+\sum_{\substack{i=1\\ i\neq q}}^{N}P_i^2\right)-1+2P_l-\left(P_l^2+\sum_{\substack{i=1\\ i\neq l}}^{N}P_i^2\right)\right)$$
$$= C_3(P_l-P_q). \tag{16}$$

Subtracting Eq. (12) from Eq. (13) yields the following:

$$Net_{lmn}-Net_{qmn}=\frac{C_2}{2}\left(\sum_{\substack{j1=1,\\ j1\neq m}}^{M}\sum_{k1=1}^{T}V_{qj1k1}-\sum_{\substack{j1=1,\\ j1\neq m}}^{M}\sum_{k1}^{T}V_{lj1k1}\right)$$
$$+C_3(P_l-P_q)+C_3+\frac{C_5}{2}(G_{qmn}^2H(G_{qmn})-G_{lmn}^2H(G_{lmn}))$$
$$+\frac{C_6}{2}\left(\sum_{\substack{i1=1,\\ i1\neq q}}^{N}\sum_{\substack{j1=1,\\ j1\neq m}}^{M}\sum_{s}^{F}R_{qs}V_{i1j1n}R_{i1s}\right.$$
$$\left.-\sum_{\substack{i1=1,\\ i1\neq l}}^{N}\sum_{\substack{j1=1,\\ j1\neq m}}^{M}\sum_{s}^{F}R_{ls}V_{i1j1n}R_{i1s}\right). \tag{17}$$

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

$$\Delta E = Net_{lmn} - Net_{qmn} - C_3.$$

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

Small-scale shared-memory multiprocessors are commonly used in a workgroup environment where multiprocesses are executed concurrently while sharing processors

Table 1
Weighting factor of competitive HNN

Constants for HNN			
C_2	C_3	C_5	C_6
4.0	1.0	3.0	1.0

Table 2
Resource requested matrix (cases 1 and 2)

	R_1	R_2	R_3
Process 1	1	0	0
Process 2	0	0	1
Process 3	0	1	0
Process 4	1	0	0

Table 3
Timing constraints matrix (cases 1 and 2)

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

Table 4
Initial states for CHNN neural network (case 1)

	1	2	3	4	5	6
Machine 1						
Process 1	1	0	1	0	0	0
Process 2	0	1	0	1	0	1
Process 3	1	0	0	0	1	1
Process 4	0	1	1	0	1	0
Machine 2						
Process 1	1	0	1	0	0	0
Process 2	1	0	1	0	0	1
Process 3	0	1	0	1	0	1
Process 4	1	0	0	0	0	0

and other system resources. The simulations consider classes of scheduling problems such as the *display system* described in the introduction. 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 resource and timing constraints and various different initial neuron states were applied to the simulations. Tables 2 and 3 list the resource and timing constraints, respectively. This simulation involves scheduling four processes (jobs) in two processors (machines). Tables 4 and 5 list the initial states of cases 1 and 2, and Figs. 2 and 3 illustrate the resulting schedule of cases 1 and 2, respectively. Meanwhile, Fig. 4 displays the energy revolution for these two cases. Additionally, a second scheduling problem with five processes and two processors was also simulated. Tables 6 and 7 display the corresponding resource and timing constraint matrices used for this second scheduling problem. Moreover, different initial conditions are simulated to better understand the response of the neural network to the scheduling problem. Tables 8, 9 and 10 display the different initial conditions of cases 3–8, while Figs. 5 and 6 present the scheduling results for cases 3 and 4. Fig. 7 illustrates the same schedule result from cases 7 and 8. Furthermore, Fig. 8(a) demonstrates the energy curves during neural network revolution for cases 3–8. Additionally, Fig. 8(b) represents a significant portion of energy curves for cases 3 and 4, while Fig. 8(c) contains the interesting portion of the energy curves for cases 5–8.

Table 5
Initial states for CHNN neural network (case 2)

	1	2	3	4	5	6
Machine 1						
Process 1	0	0	0	1	0	1
Process 2	1	0	1	0	1	0
Process 3	1	1	0	0	0	1
Process 4	0	1	0	1	1	0
Machine 2						
Process 1	0	0	0	1	0	1
Process 2	1	0	0	1	0	1
Process 3	1	0	1	0	1	0
Process 4	0	0	0	0	0	1

Table 6
Resource requested matrix (cases 3–8)

	R_1	R_2	R_3	R_4
Process 1	1	0	0	0
Process 2	0	1	0	0
Process 3	0	0	1	0
Process 4	0	0	0	1
Process 5	1	0	0	1

Table 7
Timing constraints matrix (cases 3–8)

	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 8
Initial states for CHNN neural network (case 3)

	1	2	3	4	5	6	7	8
Machine 1								
Process 1	1	0	1	0	0	0	1	0
Process 2	0	1	0	1	0	1	0	0
Process 3	1	0	0	0	1	1	1	1
Process 4	0	1	1	0	1	0	0	0
Process 5	0	1	0	1	0	1	0	1
Machine 2								
Process 1	1	0	1	0	1	0	1	1
Process 2	1	0	1	0	0	1	0	0
Process 3	0	1	0	1	0	1	0	1
Process 4	1	0	0	0	0	0	1	0
Process 5	0	0	0	0	1	1	1	1

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 [11]. 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 multi-constraint schedule problem. The energy function proposed herein works efficiently and can be applied to investigating certain scheduling problems, such as problems in which processes are executed concurrently with asynchronous shared-memory communication.

Table 9
Initial states for CHNN neural network (case 4)

	1	2	3	4	5	6	7	8
Machine 1								
Process 1	0	1	0	0	0	1	0	1
Process 2	0	0	1	0	1	0	1	0
Process 3	1	1	1	1	0	0	0	1
Process 4	0	0	0	1	0	1	1	0
Process 5	1	0	1	0	1	0	1	0
Machine 2								
Process 1	1	1	0	1	0	1	0	1
Process 2	0	0	1	0	0	1	0	1
Process 3	1	0	1	0	1	0	1	0
Process 4	0	1	0	0	0	0	0	1
Process 5	1	1	1	1	0	0	0	0

Table 10
Initial states for CHNN neural network: case 5, $a = 0$, $b = 0$; case 6, $a = 1$, $b = 0$; case 7, $a = 0$, $b = 1$; case 8, $a = 1$, $b = 1$

	1	2	3	4	5	6	7	8
Machine 1								
Process 1	a	a	a	a	a	a	a	a
Process 2	a	a	a	a	a	a	a	a
Process 3	a	a	a	a	a	a	a	a
Process 4	a	a	a	a	a	a	a	a
Process 5	a	a	a	a	a	a	a	a
Machine 2								
Process 1	b	b	b	b	b	b	b	b
Process 2	b	b	b	b	b	b	b	b
Process 3	b	b	b	b	b	b	b	b
Process 4	b	b	b	b	b	b	b	b
Process 5	b	b	b	b	b	b	b	b

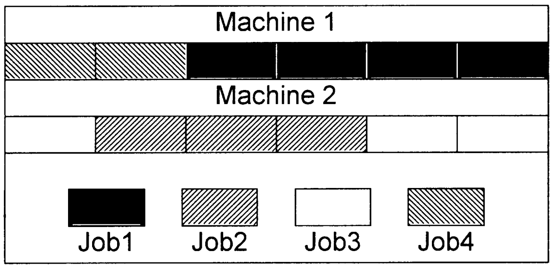


Fig. 2. Simulation results of case 1.

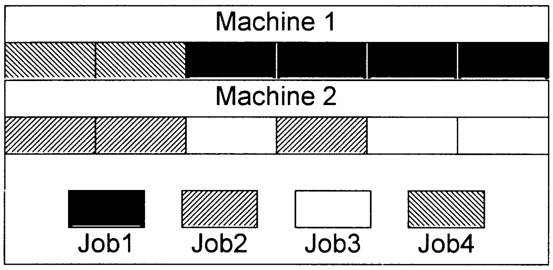


Fig. 3. Simulation results of case 2.

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.

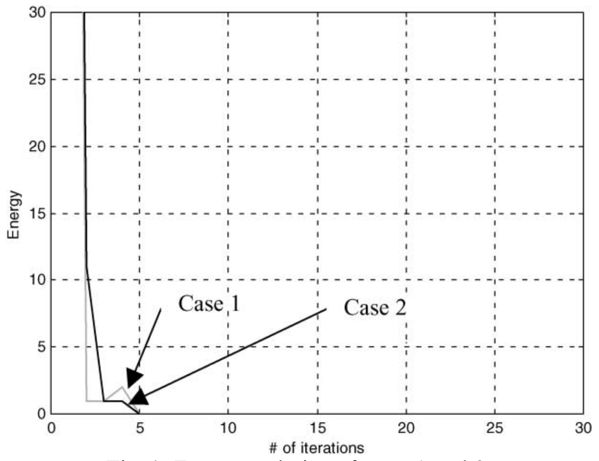


Fig. 4. Energy evolution of cases 1 and 2.

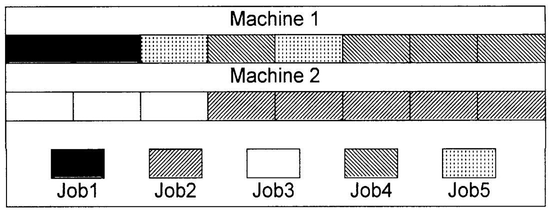


Fig. 5. Simulation results of case 3.

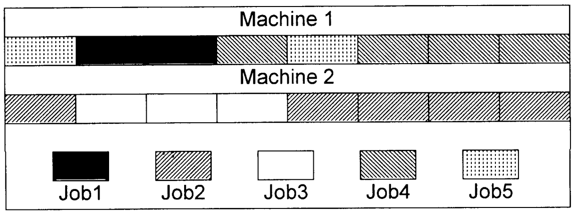


Fig. 6. Simulation results of case 4.

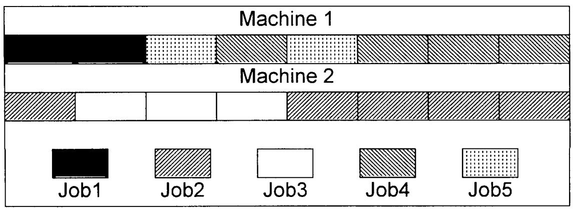
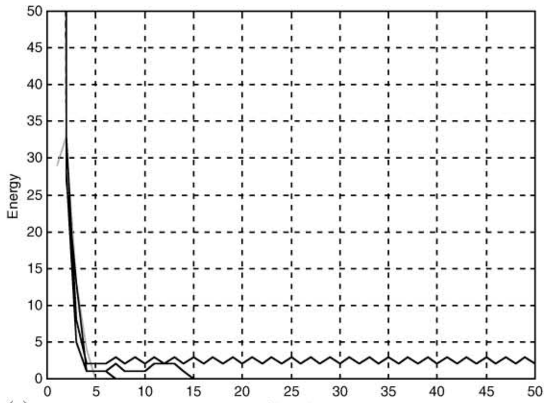
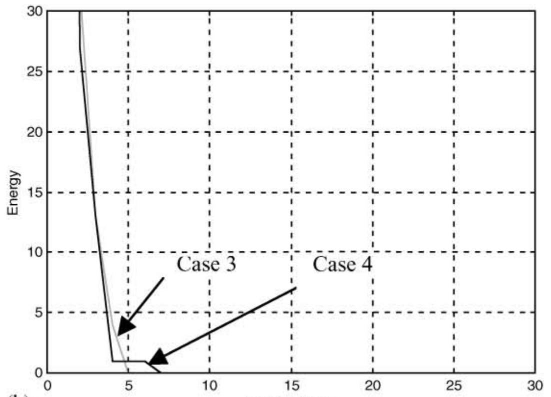


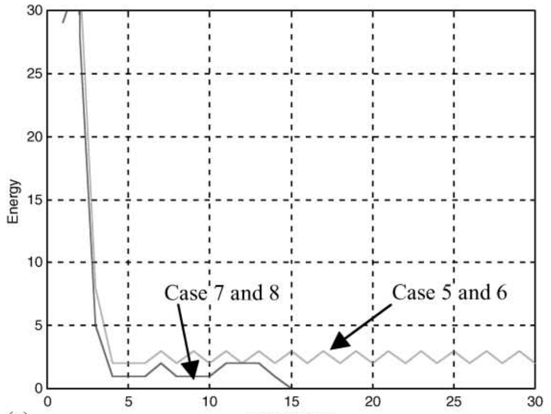
Fig. 7. Simulation results of cases 7 and 8.



(a)



(b)



(c)

Fig. 8. Energy function of cases 3–8.

(2) Although larger setting values of initial states imply larger initial energy values, those energies (errors) decrease dramatically to nearly the same energy value once the CHNN neural network starts updating. Again, random initial state assignment is recommended.

(3) The rate of convergence is initial-state dependent, as shown in cases 3–8 of Tables 8, 9 and 10. Generally, the initial states with a random distribution arrive at a rapid and sound solution, as in cases 1, 2, 3 and 4

(4) 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 [18,20]. Consequently, a solution is not guaranteed, causing inevitable oscillation. Simulation results illustrate that cases 5 and 6 involve unstable revolutions and no solutions are obtained. Meanwhile, Figs. 1, 7 and 8 oscillated to the convergence, as displayed in Figs. 4 and 8. In [21], 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 matrices used in our simulation as listed in Table 1 is not unique. Meanwhile, another valid weight matrix containing $C_2 = 1.3$, $C_3 = 0.3$, $C_5 = 1.5$, and $C_6 = 1.2$ was also simulated. 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 iterations. 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 5–15 iterations to converge. When initial values are randomly set, a minimum of 5 iterations is needed to converge to a solution. Each iteration involves updating every column of the competitive Hopfield neural network. The number of neurons for the proposed neural network is (NMT), and the computational time for each iteration is equal to the total neurons (NMT) multiplied by the computation time for each neuron (proportional to (NM)). Consequently, this algorithm results in a $O(N^2M^2T)$ complexity. Restated, the execution time required for each iteration is proportional to $O(N^2M^2T)$. For example, in some case, one machine contains 20 jobs and each job requires one unit time. Then, 20! distinct sequences may exist, where 20! = 2432902008176640000. Finding the solution using exhaustive search is impractically slow on an Intel celeron 450 desktop PC used in our simulation. Instead, using CHNN, this CHNN consists of $20 \times 1 \times 20$ neurons and 15 iterations, the most required to find the solution. The computation time of each neuron is proportional to 20 times one-state computation (interconnection, net value, and others computation). $20 \times 20 \times 20 \times 15 = 120{,}000$ state computations may exist for the CHNN algorithm, and 10 more minutes may be required to converge on our simulation platform. Furthermore, when the problem involves 100 jobs on five processors, then 375,000,000 state computations are necessary, meaning it will take around 1.5 months to find the solution. Restated, 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 mainly on the problem of resource utilization. For practical implementation, the problem can be extended to involve the temporal relationship of the resources required for each job. Restated, each job would require different resources in a specific order at different times. Correspondingly, the constructed energy function in this work can be modified by adding additional energy terms to satisfy additional requirements. A notion resembling the priority scheduling constraint may be involved. A future work should address this issue more thoroughly.

Acknowledgements

The authors would like to gratefully acknowledge helpful comments and valuable assistance from both anonymous reviewers.

References

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

[2] Y.P.S. Foo, Y. Takefuji, Integer linear programming neural networks for job-shop scheduling, IEEE International Conference 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, IEEE International Conference on Neural Networks, 1991, pp. 1361–1366.

[4] C. Cardeira, Z. Mammeri, Neural networks for multiprocessor real-time scheduling, Proceedings of the Sixth Euromicro Workshop on Real-Time Systems, 1994, pp. 59–64.

[5] A. Hanada, K. Ohnishi, Near optimal jobshop scheduling using neural network parallel computing, International Conference on Proceedings of the 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, Mun Sung Han, Dynamic neural network with heuristic, IEEE International Conference on Neural Networks, Vol. 7, 1994, pp. 4650–4654.

[7] Yueh-Min Huang, Ruey-Maw Chen, Scheduling multiprocessor job with resource and timing constraints using neural network, IEEE Trans. System Man Cybernet. Part B 29 (4) (1999) 490–502.

[8] J.J. Hopfield, D.W. Tank, Neural computation of decision in optimization problems, Biol. Cybernet. 52 (1985) 141–152.

[9] J.M. Gallone, F. Charpillet, F. Alexandre, Anytime scheduling with neural networks, Proceedings of the INRIA/IEEE Symposium on Emerging Technologies and Factory Automation, Vol. 1, 1995, pp. 509–520.

[10] N.M. Nasrabadi, C.Y. Choo, Hofield network for stereo vision correspondence, IEEE Trsns. Neural Networks 3 (1992) 5–13.
[11] M.W. Dixon, G.R. Cole, M.I. Bellgard, Using the Hopfield network with mean field annealing to solve the shortest path problem in a communication network, International Conference on Neural Networks, Vol. 5, 1995, pp. 2652–2657.
[12] P.C. Chung, C.T. Tsai, E.L. Chen, Y.N. Sun, Polygonal approximation using a competitive Hopfield neural network, Pattern Recognition 27 (1994) 1505–1512.
[13] J.S. Lin, K.S. Cheng, C.W. Mao, A fuzzy Hopfield neural network for medical image segmentation, IEEE Trans. Nucl. Sci. NS-43 (4) (1996) 2389–2398.
[14] T. Uchiyama, M.A. Arbib, Color image segmentation using competitive learning, IEEE Trans. Pattern Anal. Mach. Intell. 16 (12) (1994) 1197–1206.
[15] K. Lee, Y. Takefuji, N. Funabiki, A parallel improvement algorithm for the biparties subgraphy problem, Case Western Reserve University, CAISR Technical Report TR91-105, 1991.
[16] R.M. Chen, Y.M. Huang, Multiconstraint task scheduling in multiprocessor system by neural network, Proceedings of the IEEE Tenth International Conference on Tools with Artificial Intelligence, 1998, pp. 288–294.
[17] R.M. Chen, Y. M. Huang, Multitask scheduling by fuzzy Hopfield neural Network, Sixth National Conference On Fuzzy Theory and Its Applications, 1998, pp. 231–236.
[18] J.J. Hopfield, D.W. Tank, Computing with neural circuits: a model, Science 233 (1986) 625–633.
[19] G. Bilbro, R. Mann, T. Miller, W. Snyder, D.E. Van den Bout, M. White, Mean field annealing and neural networks, in: D.S. Touretzky (Ed.), Advances in Neural Information Processing System, Morgan Kaufmann, San Mateo, CA, 1989, pp. 91–98.
[20] M. Takeda, J.W. Goodman, Neural networks for computation: number representation and programming complexity, Appl. Opt. 25 (1986) 3033–3046.
[21] Y. Takefuji, K.C. Lee, An artificial hystersis binary neuron: a model suppressing the oscillatory behaviors of neuron dynamics, Biol. Cybernet. 64 (1991) 353–356.

Ruey-Maw Chen was born in Taiwan, R.O.C., in 1960. He received the B.S. and the M.S. degrees in engineering science from National Cheng Kung University, Taiwan, R.O.C., in 1983 and 1985, respectively. Currently, he is a Ph.D. student in the Department of Engineering Science of the same university.

From 1985 to 1994, he was a senior engineer on avionics system design at Chung Shan Institute of Science and Technology (CSIST). Since 1994, he is a technical staff at National Chin-Yi Institute of Technology (NCIT). His research interests include scheduling, digital image process, and neural network.

Yueh-Min Huang was born in Taiwan, R.O.C., in 1960. He received the B.S. degree in engineering science from National Cheng-Kung University, Taiwan, R.O.C., in 1982, and both the M.S. and Ph.D degrees in electrical engineering from the University of Arizona, Tucson, AZ, in 1988 and 1991, respectively.

Since 1991, he has been with the Department of Engineering Science, National Cheng-Kung University, where he is a professor. His research interests include distributed multimedia systems, data mining, and real-time systems.

Dr. Huang is a member of IEEE Computer Society, the American Association for Artificial Intelligence, and the Chinese Fuzzy Systems Association. He was a winner of the 1996 Acer Long-Term Award for Best M.S. Thesis Supervision.