Applied Mathematics and Computation 217 (2011) 6379–6389

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Reducing network and computation complexities in neural based real-time scheduling scheme

Ruey-Maw Chen

Department of Computer Science and Information Engineering, National Chin-yi University of Technology, Taichung 411, Taiwan, ROC

ARTICLE INFO

Keywords:
Scheduling
M-out-of-N competitive
Hopfield neural network
Computational complexity

ABSTRACT

Multiprocessor real-time scheduling is an important issue in many applications. A neural network provides a highly effective method to obtain good solutions for real-time scheduling problems. However, multiprocessor real-time scheduling problems include multiple variables; processor, process and time, and the neural networks have to be presented in three dimensions with these variables. Hence, the corresponding neural networks have more neurons, and synaptic weights, and thus associated network and computational complexities increase. Meanwhile, a neural network using the competitive scheme can provide a highly effective method with less network complexity. Therefore, in this study, a simplified two-dimensional Hopfield-type neural network using competitive rule is introduced for solving three-dimensional multiprocessor real-time scheduling problems. Restated, a two-dimensional network is proposed to lower the neural network dimensions and decrease the number of neurons and hence reduce the network complexity; an *M-out-of-N* competitive scheme is suggested to greatly reduce the computational complexity. Simulation results reveal that the proposed scheme imposed on the derived energy function with respect to process time and deadline constraints is an appropriate approach to solving these class scheduling problems. Moreover, the computational complexity of the proposed scheme is greatly lowered to $O(N \times T^2)$.

1. Introduction

Generally speaking, scheduling problems are seen as involving allocation of resources (like machines or processors) to execute a set of activities (like processes or tasks) satisfying given constraints and optimizing given criteria. Processes or tasks have time constraints, like ready time, execution time, precedence, and deadline. A scheduling algorithm determines a schedule for a set of processes, satisfying the prerequisite constraints. Scheduling problems differ markedly from case to case. Many different types of scheduling problems are widely studied such as job-shop, flow-shop, open-shop, task assignment, real-time, and other scheduling problems. Among these scheduling problems, the flow-shop problem (FSP), resource-constrained project scheduling problem (RCPSP), and real-time scheduling problem have recently attracted the attention of many researchers. The FSP has been studied by many researchers [1–5] and is a real world combinational optimization (CO) problem The FSP considers that a set of independent jobs has to be assigned to run on a set of different machines. Every job requires a given fixed, non-negative processing time on every machine. In a flow-shop, all jobs are to be processed on all the machines in the same order, that is, the jobs follow the same machine order in the shop starting from the first machine and finishing on the last machine. Meanwhile, RCPSP is an interesting scheduling problem which includes both resource and precedence constraints [6–9]. There are different types of resources available in a RCPSP system. The resource constraints

E-mail address: raymond@mail.ncut.edu.tw

0096-3003/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2011.01.014

assume that the total amount of a resource type required by scheduled activities cannot exceed available quantity at any time. A feasible schedule is an assignment of activities to processors such that an activity comes in the schedule when all of its predecessors have finished. Additionally, the real-time scheduling problem is concerned with all activities in the system having to be finished prior to the task-specific deadline. The real-time scheduling problem is at the heart of many scheduling problems in various applications. Examples of real-time scheduling include: nuclear power plant control systems, traffic control systems, flight mission control systems, and embedded tactical systems for military applications. Failure to meet the timing constraints in such systems not only degrades the system but is also dangerous. Therefore, the real-time scheduling algorithm is a vital component of these systems. Many applications involve scheduling notion, such as generating unit planning of power plants [10], grid computing [11,12], control systems [13], the food industry [14], network packet switching [15], museum visitor routing problem [16], transit vehicle scheduling problem [17], classroom arrangement [18] and manpower scheduling [19]. However, most scheduling problems are confirmed to be *NP*-complete problems. Hence, many schemes have been developed to solve a variety of scheduling applications. Heuristics like the (shifted) shortest processing time ((S) SPT), shortest remaining processing time (SRPT) and the earliest deadline first (EDL) policies are often used for system scheduling [20–22]. Approximation techniques [23,24], iterative search techniques such as the Tabu Search [25,26], and Genetic Algorithms [27,28] have been applied to various scheduling problems. Moreover, ant colony optimization (ACO) is frequently utilized to solve various scheduling problems [29,30]. Additionally, an immune algorithm (IA) was used to tackle complex problems and produce a reasonable manufacturing schedule within an acceptable time [31]. More recently, particle swarm optimization (PSO) based schemes have become popular algorithms for solving scheduling problems [32,33]. However, most of the above approaches require adequate parameter settings, and have difficulty providing good solutions as the problem size grows. Additionally, the advancement of hardware and software technologies enables neural networks to be applied to exploding *NP*-complete problems [34]. In one study [35], two neural network models are integrated to solve multiprocessor (parallel resources) deadline problems. Feng et al. applied generic neural networks to solve independent and non-preemptive tasks with deadline requirements for real time scheduling problems [36]. Neural networks have been widely applied to many different fields such as engineering, physics, mathematics, computer science and medicine. Moreover, neural networks have been applied to various real-world applications including: prediction and forecasting, control, clustering, speech recognition, pattern recognition, classification, grid computing, scheduling, etc. Restated, neural networks have attracted the attention of many researchers. Meanwhile, a neural scheduler has a very fast convergence rate. Hence, various neural networks related investigations are presented for solving different scheduling problems. Shen and Wang used a neural network to explore a satellite broadcast scheduling problem [37]. Moreover, Wang and Zhao [38] also presented a Hopfield type neural network for packet scheduling algorithm in the HSDPA System. Ma et al. [39] provided an efficient method for power-line communication channel based on neural networks.

Most studied scheduling problems (such as job-shop, flow-shop, MRCPSP, task assignment, and some soft real-time problems) are focused on minimizing the maximum completion time (*makespan*) or minimizing tardiness. Nevertheless, the major concern of real-time task scheduling is to meet task deadline constraints rather than optimizing a given target. Restated, any solution satisfying the system constraints (processing time, deadline, etc.) is a solution of the studied real-time scheduling problem. Moreover, most studied real-time scheduling problems consider that jobs can be executed on a single machine with one resource only [22,40,41]. Additionally, many real-time scheduling problems consider only *non-preemptive* activities within the scheduling system [24,35,36]. In some multiprocessor systems, for instance, the *display system* on an advanced avionics system may consist of two or more display processors. Each processor is responsible for different tasks containing the timing constraint without allowing task migration between processors. To facilitate the pilot's control action, all tasks must be properly scheduled to provide the pilot with useful information. Otherwise, a hazardous situation becomes inevitable. Therefore, this study focuses mainly on resolving the generic problem similar to the above situation.

In this study, a multiprocessor real-time scheduling problem involving preemptive multi-processing with processing time, deadline constraints, and no process migration allowed is investigated. An integrating competitive mechanism with modified Hopfield type two-dimensional neural network scheme leading to simplified structure and less computational complexity is constructed for solving applicable real-time scheduling problems.

Hopfield and Tank employed neural networks to solve optimization problems involving constraint satisfaction. Thus, many researchers have applied this method to various applications [42,43]. Our previous work [44] also solved multiprocessor schedule problems based on the typical Hopfield neural network.

A competitive Hopfield neural network applies a competitive learning mechanism instead of using deterministic rules to update the neuron states in the network. A competitive scheme provides a highly effective method with less network structure complexity and has been applied in various fields. A typical competitive neural network scheme was applied to the scheduling problem in Chen and Huang [45]. Including the *1-out-of-N* competitive architecture into the network to assist in problem solving has a unique activated neuron in each column or row of the network. Hence, the energy function is simplified and the network complexity is lowered. Nevertheless, the used neural networks are usually built in three or more dimensions as the application problems have multiple variables. Similarly, the studied multiprocessor real-time scheduling problems also have multiple variables: processor, processes and time. Therefore, the corresponding neural networks have three or more dimensions [44,45]. A three-dimensional neural network then has more neurons and synaptic weights as the problem size increases. Furthermore, the computation time of neuron state updates depends on the number of neurons and interconnections between them. Restated, three-dimensional neural networks yield high network complexity as well as computational complexity. Accordingly, the performance of a neural scheduler is limited on large scale problems.

In light of the above mentioned developments, this work explores the scheduling problems in multiprocessor real-time systems, including processing time and deadline constraints. A scheduling problem with multiple variables is represented by our suggested two-dimensional neural network structure. This two-dimensional neural network is then sent to an *M-out-of-N* competitive algorithm to determine the neuron states and obtain the solution. Moreover, an associated energy function is designed to illustrate the timing constraints. Hence, the studied scheduling problem is then aimed at minimizing this designed energy function. Consequently, a proposed *M-out-of-N* competitive neural network can be employed to calculate the weighting and threshold matrices, and a solution can be derived using the *M-out-of-N* competition processes. The simulation results show that the proposed method can provide a highly effective method with less network complexity and low computational complexity in solving multiprocessor real-time scheduling problems. Moreover, the proposed model can be extended to solve scheduling problems with more complex, multiple constraints.

The rest of this paper is organized as follows: Section 2 derives the corresponding energy function of the scheduling problem according to the intrinsic constraints. Section 3 reviews the *M-out-of-N* competitive algorithm and translates the derived energy functions to the proposed algorithm. The computational complexity of the presented model is analyzed in Section 4. The simulation examples and experimental results are presented in Section 5. Finally, conclusions and suggestions for future work are given in Section 6.

2. Energy formulation of two-dimensional neural networks

The scheduling problem domain to be considered in this paper is defined as follows. Assume that there are N processes and M processors in the system. First, a process can be segmented, and the execution of each process is preemptive. Second, the various segments of a process cannot be assigned to different machines. Third, each process's execution time and deadline are predetermined.

This study involves the optimization application of neural networks to solve real-time scheduling problems including three variables: job (or process), machine (or processor), and time. These three variables are the axes in three-dimensional networks and a neuron state is represented by V_{ijk} as shown in Fig. 1. This V_{ijk} state variable is defined as representing whether or not process i is executed on processor j at a specific time k. The activated neuron $V_{ijk} = 1$ denotes that the process i runs on processor j at time k; otherwise, $V_{ijk} = 0$. In this investigated scheduling problem, N is the total number of processes to be scheduled, M denotes the total number of processors to be operated, and T is the job's deadline. Thus, the total neurons in the three-dimensional neural network are $N \times M \times T$.

A three-dimensional neural network can be visualized as a neural network which is composed of M layers (M 2-dimensional neural networks); each layer contains $N \times T$ *neurons*. In this work, those M layers are overlapped and become one layer network. Restated, the total neurons in the resulting one layer two-dimensional neural network are $(N \times T)$. The compressed two-dimensional neural networks for the scheduling problem require only two variables: process and time. These two variables are the axes n the two-dimensional networks, and V_{ik} or V_{xz} represents the neuron state as shown in Fig. 2. The x-axis denotes the *process* variable, with i representing a specific process with a range from 1 to N, where N is the total number of processes to be scheduled. 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's deadline. Thus, a state variable V_{ik} is defined as representing whether or not process i is being executed at a certain time k. The activated neuron $V_{ik} = 1$ denotes that the process i runs on a processor at time k; otherwise, $V_{ik} = 0$. Each V_{ik} corresponds to a neuron of the two-dimensional neural network. The defined neuron state V_{ik} or V_{xz} is then used to construct the energy function which is made up of three energy terms stated as follows. The first energy term expresses the *M-out-of-N* constraint, since M processors can run a maximum of M processes at a certain time k. If M processes among N total processes are handled on associated M processors at time k ($V_{ik} = 1$), then there can be no other processes (N-M) handled at time k. This energy is defined as

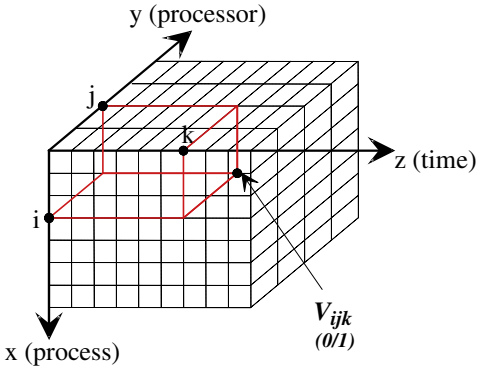

Fig. 1. 3-D neural networks.

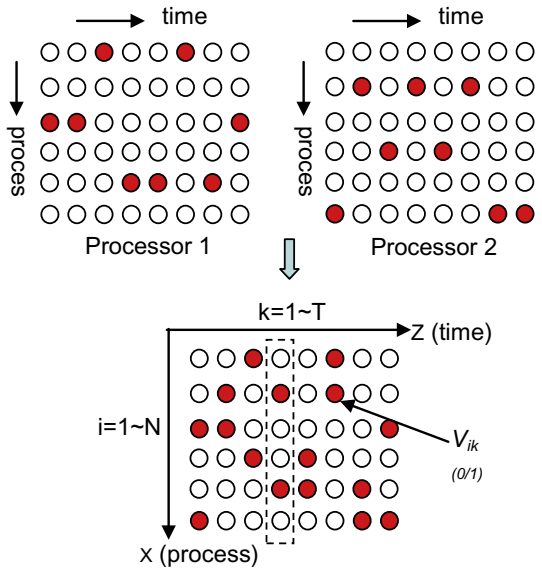

Fig. 2. A condensed 2-D neural network by overlapping multiple neural networks.

$$\sum_{k=1}^{T}\left(\sum_{i=1}^{N}V_{ik}-M\right)^2, \tag{1}$$

where N, M, T, i, k, and V_{ik} are defined as above. The rest of this study employs the same notations. This term has a minimum value of zero when it meets this *M-out-of-N* constraint, which arises when $\sum_{i=1}^{N}V_{ik}=M$. The second energy term is defined as

$$\sum_{i=1}^{N}\left(\sum_{k=1}^{T}V_{ik}-P_i\right)^2, \tag{2}$$

where P_i is the total execution time needed by process i. This energy term indicates that the time consumed by process i must equal P_i, such that $\sum_{k=1}^{T}V_{ik}=P_i$ and Eq. (2) becomes zero. The following energy term is defined to meet the real-time requirement of each process i:

$$\begin{aligned}&\sum_{i=1}^{N}\sum_{k=1}^{T}V_{ik}D_{ik}^2H(D_{ik}),\\ &H(D_{ik})=\begin{cases}1, & \text{if}\quad D_{ik}>0,\\ 0, & \text{if}\quad D_{ik}\leqslant 0,\end{cases}\quad D_{ik}=k-d_i,\end{aligned} \tag{3}$$

where d_i is the deadline of process i and $H(D_{ik})$ is the unit step function. The energy term will exceed zero when a segment of the process is assigned to run at a time later than d_i, i.e., when $V_{ik}=1$, $k-d_i>0$, and hence $H(D_{ik})>0$. Therefore, the energy value grows exponentially with the associated time lag between k and d_i, given by k-d_i. Conversely, this energy term has a value of zero if $V_{ik}=1$ and k-$d_i\leqslant 0$. Accordingly, the energy function corresponding to the studied scheduling problem with all constraints can be derived, as shown in Eq. (4).

$$E=\frac{C_1}{2}\sum_{k=1}^{T}\left(\sum_{i=1}^{N}V_{ik}-M\right)^2+\frac{C_2}{2}\sum_{i=1}^{N}\left(\sum_{k=1}^{T}V_{ik}-P_i\right)^2+\frac{C_3}{2}\sum_{i=1}^{N}\sum_{k=1}^{T}V_{ik}D_{ik}^2H(D_{ik}), \tag{4}$$

C_1, C_2, and C_3 represent weighting factors which are assumed to be positive constants. Based on the discussion above, the derived energy function has a minimum value of zero when all constraints are met.

3. *M-out-of-N* competitive algorithm

In this section, the scheduling problem and the defined energy function are mapped onto the *M-out-of-N* competitive neural networks to yield solutions.

Hopfield and Tank originally used the neural network method to solve optimization problems [46]. The Hopfield neural network (HNN) algorithm is based on the gradient technique to get the problem's solution and thus provide rapid convergence. Based on the Dynamic system theory, the Lyapunov function used in HNN as shown in Eq. (5) has verified the

existence of stable states of a network system. Restated, the derived energy function representing the scheduling problem must be in the same format as the Lyapunov function when applying HNN. The Lyapunov function used in a two-dimensional HNN model is shown as below.

$$E = -\frac{1}{2}\sum_{x}\sum_{z}\sum_{i}\sum_{k} V_{xz} W_{xzik} V_{ik} + \sum_{i}\sum_{k} \theta_{ik} V_{ik}, \tag{5}$$

V_{xz} and V_{ik} denote the neuron states; W_{xzik} represents the synaptic weight indicating the interconnection strength among neurons. The θ_{ik} denotes the bias input of the neuron (i,k). Additionally, the conventional HNN uses the deterministic rule displayed in Eq. (6) to update the neuron state. This deterministic rule is

$$V_{ik}^{n+1} = \begin{cases} 1, & \text{if} \quad Net_{ik} > 0, \\ V_{ik}^{n}, & \text{if} \quad Net_{ik} = 0, \\ 0, & \text{if} \quad Net_{ik} < 0. \end{cases} \tag{6}$$

Restated, the neuron state is decided by the Net_{ik} value. The Net_{ik} represents the net value of the neuron (i,k) obtained using the interconnection strength W_{xzik} to the other neurons (x,z) and the bias input θ_{ik}. The Net_{ik} definition is as follows.

$$Net_{ik} = -\frac{\partial E}{\partial V_{ik}} = \sum_{x}\sum_{z} W_{xzik} - \theta_{ik}. \tag{7}$$

Instead of applying conventional deterministic rules to update the neuron states, this study uses the competition rule to decide the active neurons among a set of neurons. Instead of using a deterministic rule, an *M-out-of-N* competitive rule is adopted in this work to meet neural network evolution while satisfying constraints. The *M-out-of-N* constraint indicates that "exactly *M* neurons among *N* neurons" should be activated when the network reaches a stable state. Restated, the number of activated neurons during each time unit has to be exactly the same as the number of processors when the neural network reaches a convergent state. This *M-out-of-N* Competitive Hopfield Neural Network is referred to as MCHNN herein.

Since at a specific time, *M* processors can at most execute *M* processes in a subject scheduling problem, the first C_1 energy term can be handled implicitly while applying the *M-out-of-N* competitive rule. Restated, the first C_1 energy term can be omitted from the energy function Eq. (4) and a simplified energy function is generated. The resulting energy function after simplification is given as follows:

$$E = \frac{C_2}{2}\sum_{i=1}^{N}\left(\sum_{k=1}^{T} V_{ik} - P_i\right)^2 + \frac{C_3}{2}\sum_{i=1}^{N}\sum_{k=1}^{T} V_{ik} D_{ik}^2 H(D_{ik}). \tag{8}$$

Therefore, the synaptic interconnection strength W_{xzik} and the bias input θ_{ik} corresponding to HNN can be obtained by comparing Eq. (8) with Eq. (5) where

$$W_{xzik} = -C_2\delta(x,i) \tag{9}$$

and

$$\theta_{ik} = -C_2 P_i + \frac{C_3}{2} D_{ik}^2 H(D_{ik}), \tag{10}$$

respectively, where

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

The MCHNN imposes an *M-out-of-N* competitive rule to update the neuron states. Neurons in the same column at a given time compete with one another to determine the activated neurons. According to net value, the *M* neurons corresponding to the largest *M* net values in a column are selected as the activated neurons of a network. Accordingly, the output states of the activated neurons are set to 1, and the output states of all the other neurons in the same column are set to 0. The neuron state update of the proposed *M-out-of-N* competitive rule for the *k*th column is illustrated as follows:

$$V_{ik} = \begin{cases} 1, & \text{if} \qquad Net_{ik} \in \text{ Ranks among the } M \text{ largest}, \\ 0, & otherwise \quad . \end{cases} \tag{11}$$

Restated, the neuron (i,k) is set to activated ($V_{ik} = 1$) if the corresponding net value Net_{jk} is one of the largest *M* net values. For example, there are four jobs to be processed in two processors as displayed in Fig. 2. Suppose that processes 2 and 3 are assigned to run at the same specific time, *k*. Then, the net values, Net_{2k} and Net_{3k} corresponding to neurons $(2,k)$ and $(3,k)$ are the 2 largest net values in the same column. Therefore, the final network state has exactly two neurons at a time. The activated neurons ($V_{ik} = 1$) are shown by solid nodes in Fig. 2. The term Net_{jk} denotes the net value of neuron (j,k).

4. Computational complexity

The main advantage of artificial neural networks is that few iterations are needed to find an optimal solution. However, an iteration involves the state update computations of all neurons in the networks. Meanwhile, the computation time of each neuron update is proportional to the number of connections to that neuron. Restated, network and computational complexities decide the execution time of each iteration. Thus, great effort has been made to seek better algorithms and thereby reduce these complexities.

Within three-dimensional neural networks, the number of neurons in the network is $(N \times M \times T)$. The suggested two-dimensional neural network in this study consist of only $(N \times T)$ neurons. Hence, the bounded computational network complexity is reduced to $O(N \times T)$ in this approach. Moreover, reducing the number of neurons in the network further decreases the number of connections to each neuron. The computation time of each neuron update is related to the interconnections among neurons. Hence, the yield computation time for each neuron is proportional to $(N \times M \times T)$ in three-dimensional neural networks. However, this work proposed two-dimensional neural networks, which lowers the interconnections to $(N \times T)$ for each neuron. Accordingly, the computation time for each iteration is equal to the total number of neurons $(N \times T)$ times the computational time of each neuron $(N \times T)$ which makes the upper bounded computational complexity $O(N^2 \times T^2)$.

Nevertheless, the scheduling algorithm may need to invest very significant computation time to determine the proper synaptic weight (interconnection) strengths and neural biases. The programming complexity is defined as the number of arithmetic operations that must be performed to determine these synaptic weight strengths and neural biases for the problem to be solved. For example, for a combinatorial optimization problem known as the Hitchcock problem to be solved on a neural network, the program must re-determine the interconnection strengths and/or neural biases each time the program computes new net values and decides the new neuron states. This is because the computation of interconnections and/or neural biases involves data terms (neuron states), and in such an environment the programming complexity becomes an important measure of the efficiency of neural computing. However, the programming complexity in this investigation is ignored, since the computation of the interconnection strengths and neural biases does not depend on the data terms as displayed in Eqs. (9) and (10). Restated, the interconnection and neural biases need not be re-determined for each iteration, thus the proposed scheme is independent of programming complexity.

Eq. (4) is the energy function for the two-dimensional HNN. The synaptic weight yielded based on Eq. (4) is $W_{xzik} = -C_1\delta(x,i) - C_2\delta(z,k)$. Thus, the number of interconnections of each neuron is $(N+T)$, and the computational complexity is $O((N \times T)(N+T))$. However, the synaptic weight of the proposed scheme is $W_{xzik} = -C_2\delta(z,k)$ as displayed in Eq. (9), which is derived from the simplified energy function of Eq. (8). Restated, the method developed in this study further reduces the interconnections to T, and therefore the computational complexity is $O(N \times T^2)$ instead of the upper bounded value of $O(N^2 \times T^2)$. One iteration involves updating every neuron's state within the neural network. Moreover, an important feature of an investigated scheduling algorithm is its efficiency or performance, i.e., how its calculation time increases with the problem size. Finding the solution for a large-scale (very large N and/or very large M) scheduling problem is very time-consuming when a three-dimensional neural network is applied. The computational complexity for a three-dimensional network is proportional to $O((M^2 \times N \times T)(N+T))$. However, this algorithm resulted in a significantly smaller computational complexity of $O(N \times T^2)$, an obviously more effective method especially for large-scale scheduling problems.

Table 1
Three simulation cases.

	Time required	Time limit
(a) Case1		
Process1	4	6
Process2	3	4
Process3	3	6
Process4	2	3
(b) Case2		
Process1	2	3
Process2	5	8
Process3	3	4
Process4	4	8
Process5	2	5
(c) Case3		
Process1	5	10
Process2	3	5
Process3	3	9
Process4	2	5
Process5	3	9
Process6	2	6
Process7	3	10
Process8	2	5
Process9	3	9
Process10	4	10

5. Experimental simulations

The simulations involve classes of scheduling problems with timing constraints. Several different timing constraints and various weighting factors were applied. Table 1 shows the different timing constraints of the simulation examples. Cases 1 and 3 are the same simulation examples as performed in [45]. Finally, the resulting overlapped neural network is decomposed into M neural networks. To decompose overlapped neural networks, a job sequence set is maintained for the jobs to be scanned. Restated, scanning the active neuron of a job in the job sequence set by time step identifies a process to be allocated to a processor. A job is identified as exactly M processes at specific time. Then, this identified job number is put into the end of the job sequence set for further scanning. Meanwhile, a scanned job will be removed from the job sequence set. The job scanning process for identification stops when the job sequence set is empty. The simulation results are displayed by using Gantt charts to graphically represent the process schedules. Figs. 3–5 illustrate the resulting schedules

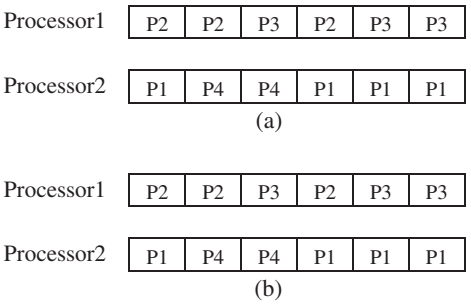

Fig. 3. Simulation results of case 1 with different initial states.

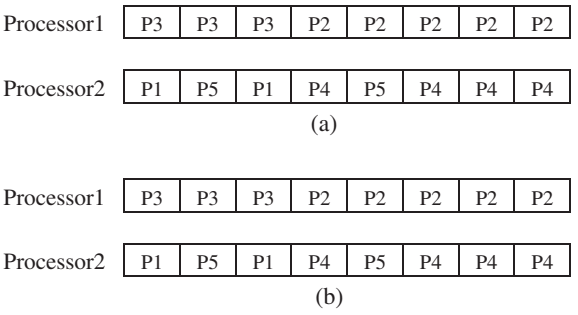

Fig. 4. Simulation results of case 2 with different initial states.

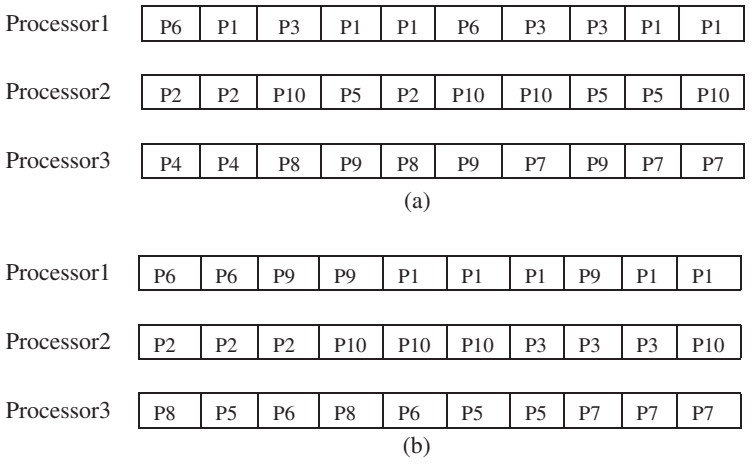

Fig. 5. Simulation results of case 3 with different initial states.

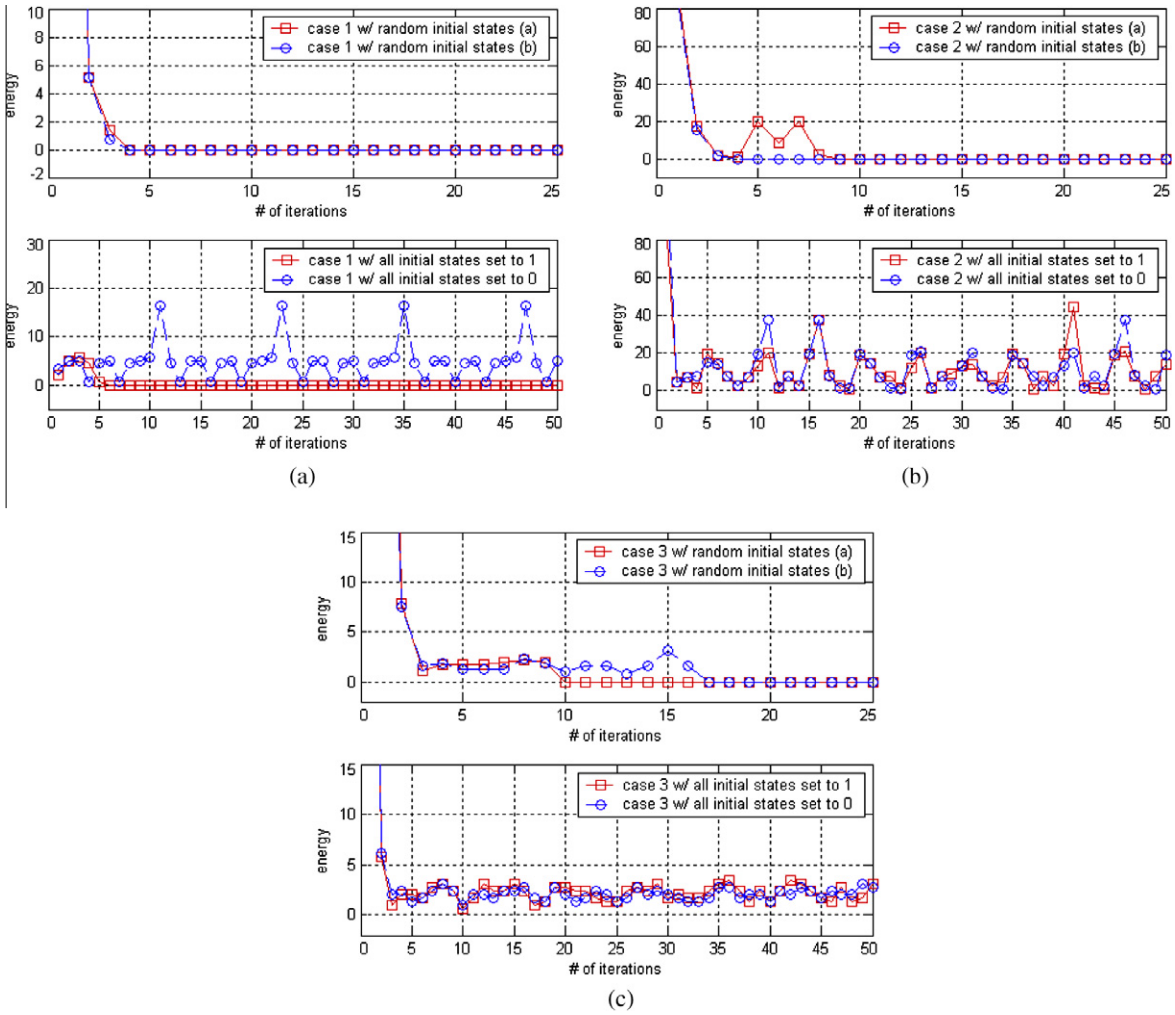

Fig. 6. Energy evolutions of the simulation cases.

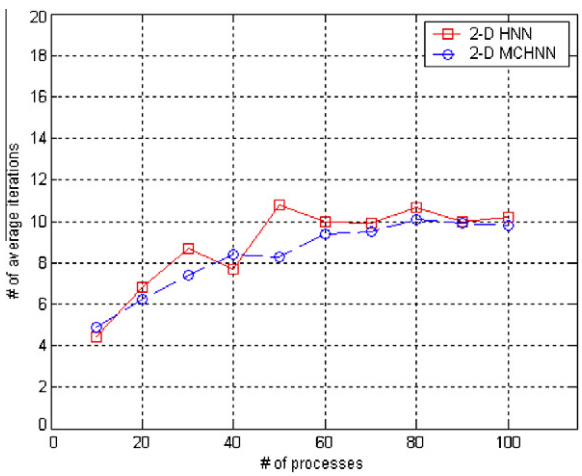

Fig. 7. Simulation results of the two studied methods.

Table 2
Computational complexity comparison between different methods.

	3-D HNN in [44]	3-D CHNN in [45]	2-D HNN	2-D *M-out-of-N* competitive HNN
Energy terms	6	5	3	2
Neurons	$N \times M \times T$	$N \times M \times T$	$N \times T$	$N \times T$
Interconnections	$O(N \times M \times T)$	$O(N \times M \times T)$	$O(N+T)$	$O(T)$
Computational complexity	$O((N \times M^2 \times T)(N+T))$	$O((N \times M^2 \times T)(N+T))$	$O((N \times T)(N+T))$	$O(N \times T^2)$

of cases 1 to 3 for the proposed algorithm. In these figures, P1 represents process 1, P2 indicates process 2, and so on. Moreover, different initial neuron states were simulated to better understand the response of the neural network to the scheduling problem, and Figs. 3–5 also display such cases. Additionally, Fig. 6 shows the significant parts of the energy curves during neural network evolution, as well as that the initial states are all set to all 0 or 1 for the three studied cases. Different initial states of neurons will all still generate feasible solutions. Moreover, in order to probe the feasibility and computational efficiency of the proposed scheme for large-scale problems, the simulation cases are set from 10 processes to 100 processes with intervals of 10 processes, on two processors. Each case is carried out 10 times to calculate its average iteration number. The same cases are also examined with HNN. Fig. 7 shows the comparison of simulation results between two methods which are two-dimensional HNN and the proposed *M-out-of-N* competitive HNN in the study; they are denoted as 2-D HNN and 2-D MCHNN respectively in the figure. Finally, Table 2 lists the comparison among the number of neurons, connections and computational complexity in different schemes.

In the simulations, each process has the constraints of a processing time and deadline, which were given in advance. These results show that the proposed scheme can solve real-time process scheduling problems with multiple constraints.

6. Conclusions and discussion

The simulation results demonstrated some significant consequences when applied to the scheduling domain. These were as follows:

(1) The proposed *M-out-of-N* competitive scheme simplifies the three-dimensional neural network problem into a two-dimensional neural network. Restated, the two-dimensional MCHNN is able to solve multi-variable real-time system scheduling problems. Additionally, weighting factor determination is a laborious task, and the reduction of energy terms by the *M-out-of-N* competitive rule can assist in easing this burden.
(2) The competitive scheme eliminated the *M-out-of-N* constraint term in the energy function, simplifying the network structure by reducing the interconnections among neurons. Hence, the competitive scheme can help overcome scaling problems.
(3) Convergence is almost independent of the large number of processes as shown in Fig. 7. Thus, the model adopted in this approach is more effective for large scale problems.
(4) Convergence is initially state dependent, as displayed in Fig. 6. Distributing the initial states randomly can generally produce feasible schedules for the investigated scheduling problem.
(5) The entailed synaptic weight matrix in Eq. (9) has a symmetric (i.e., $W_{xzik} = W_{ikxz}$) property, and nevertheless has a self-feedback interconnection, indicating that $W_{xzik} \neq 0$. Therefore, the network may oscillate during evolution.
(6) Moreover, the computational complexity of the proposed scheme for the investigated scheduling problem in this work is greatly reduced to $O(N \times T^2)$ for one iteration.

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 the simulation results, the proposed algorithm required an average of $5 \sim 25$ iterations to converge. The energy function proposed herein works efficiently and can be applied to similar cases of scheduling problems as long as the processes are independent, without requiring communication or utilizing memory resources to exchange data. However, the required network implementation depends on the intended application.

This work focuses on full processor utilization scheduling, in that the number of activated processes at a specific time unit is equal to the total number of processors. However, full process utilization is a limited situation. To apply the proposed scheme to non-full utilization cases, some extra neurons can be added to the networks to satisfy this inequality constraint. The added neurons are called slack neurons. Restated, the presented *M-out-of-N* competitive scheme combined with slack neurons suggests that the way to apply the scheduling problems have inequality constraints [7]. However, the proposed scheme concept allows expansion in any of the two-dimensional neural networks, and it can readily be adopted for any number of processes and any number of processors. Moreover, this work concentrated mainly on solving process scheduling without ready time consideration or resource constraints. For more practical implementations, different and more complicated scheduling problems can be further investigated in future research by applying the proposed algorithm. Such problems include preemptive multi-process scheduling on multiprocessor systems with multiple constraints, such as deadline and resource constraints, or processes with precedence relationships and synchronization considerations. The problem can be further extended to involve the temporal relationship of ready time, priority for each process or temporal relationship of

required resources for each process. Correspondingly, it is possible to modify the energy function in our work by using additional energy terms to satisfy any further requirements. Future research endeavors should address these issues more thoroughly.

Acknowledgements

This work was partly supported by the National Science Council, Taiwan (ROC), under contract NSC 97-2511-S-167-004-MY3.

References

[1] R. Ruiz, T. Stützle, A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem, European Journal of Operational Research 177 (2007) 2033–2049.
[2] L. Wang, L. Zhang, D.Z. Zheng, An effective hybrid genetic algorithm for flow shop scheduling with limited buffers, Computers and Operations Research 33 (2006) 2960–2971.
[3] C. Zhang, J. Sun, X. Zhu, Q. Yang, An improved particle swarm optimization algorithm for flowshop scheduling problem, Information Processing Letters 108 (4) (2008) 204–209.
[4] Z. Lian, X. Gu, B. Jiao, A similar particle swarm optimization algorithm for permutation flowshop scheduling to minimize makespan, Applied Mathematics and Computation 175 (1) (2006) 773–785.
[5] G. Easwaran, L.E. Parten, R. Moras, P.X. Uhlig, Makespan minimization in machine dominated flowshop, Applied Mathematics and Computation 217 (1) (2010) 110–116.
[6] X.G. Luo, D.W. Wang, J.F. Tang, Y.L. Tu, An improved PSO algorithm for resource-constrained project scheduling problem, in: The Sixth World Congress on Intelligent Control and Automation (WCICA 2006) 1, 2006, pp. 3514–3518.
[7] R.M. Chen, S.T. Lo, Y.M. Huang, Combining competitive scheme with slack neurons to solve real-time job scheduling problem, Expert Systems with Applications 33 (1) (2007) 75–85.
[8] D. Merkle, M. Middendorf, H. Schmeck, Ant colony optimization for resource-constrained project scheduling, IEEE Transactions on Evolutionary Computation 6 (4) (2002) 333–346.
[9] M. Ranjbar, Solving the resource-constrained project scheduling problem using filter-and-fan approach, Applied Mathematics and Computation 201 (1–2) (2008) 313–318.
[10] T. Saksornchai, W. Lee, K. Methaprayoon, J.R. Liao, R.J. Ross, Improve the unit commitment scheduling by using the neural-network-based short-term load forecasting, IIEEE Transactions on Industry Applications 41 (1) (2005) 169–179.
[11] Z. Hou, X. Zhou, Y. Wang, An application-oriented on-demand scheduling approach in the computational grid environment, in: The Fifth International Conference on Grid and Cooperative Computing (GCC 2006), 2006, pp. 66–70.
[12] L Liu, Y. Yang, W. Shi, W. Lin, L. Li, A dynamic clustering heuristic for jobs scheduling on grid computing systems, in: First International Conference on Semantics, Knowledge and Grid (SKG '05), 2005.
[13] H.S. Park, H.O. Kim, D.S. Kim, W.H. Kwon, A scheduling method for network-based control systems, IEEE Transactions on Control Systems Technology 10 (3) (2002) 318–330.
[14] S. Simonov, J. Simonovov'a, Simulation scheduling in food industry application, Czech Journal on Food Science 20 (1) (2002) 31–37.
[15] K.J. Symington, A.J. Waddie, M.R. Taghizadeh, J.F. Snowdon, A neural-network packet switch controller: scalability, performance, and network optimization, IEEE Transactions on Neural Networks 14 (1) (2003) 28–34.
[16] V.F. Yu, S.W. Lin, S.Y. Chou, The museum visitor routing problem, Applied Mathematics and Computation 216 (3) (2010) 719–729.
[17] H. Wang, J.S. Shen, Heuristic approaches for solving transit vehicle scheduling problem with route and fueling time constraints, Applied Mathematics and Computation 190 (2) (2007) 1237–1249.
[18] Z. Vejzovic, E. Humo, A software solution for a mathematical model of classroom-period schedule defragmentation, in: EUROCON: The International Conference on Computer as a Tool, 2007, pp. 2034–2038.
[19] M. Ohki, A. Morimoto, K. Miyake, Simulated annealing algorithm for scheduling problem in daily nursing cares, in: 2008 IEEE International Conference on Systems, Man and Cybernetics, 2008, pp. 1681–1687.
[20] X. Lu, R.A. Sitters, L. Stougie, A class of on-line scheduling algorithms to minimize total completion time, Operations Research Letters 31 (3) (2003) 232–236.
[21] Nikhil Bansal, Mor Harchol-Balter, Analysis of SRPT scheduling: investigating unfairness, in: 2001 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, 2001, pp. 279–290.
[22] M. Litoiu, M.V. Litoiu, Some Results on Real Time Systems. WRTP 92, Bruges, Belgium, June 1992.
[23] C. Chekuri, R. Motwani, B. Natarajan, C. Stein, Approximation techniques for average completion time scheduling, Journal on Computing 31 (2001) 146–166.
[24] J.D. Orozco, R.M. Santos, J. Santos, Hybrid rate-monotonic/reward-based scheduling of real-time embedded systems, in: IEEE Real-Time Embedded System Workshop, 2001.
[25] H. Qi, Z.Z. Qiu, A hybrid algorithm based on Tabu search for no-wait flow shop, in: 2008 Chinese Control and Decision Conference (CCDC 2008), 2008, pp. 1645–1648.
[26] R. Shanmugapriya, S. Padmavathi, S.M. Shalinie, Contention awareness in task scheduling using Tabu search, in: 2009 IEEE International Advance Computing Conference, 2009, pp. 272–277.
[27] S.C. Cheng, Y.M. Huang, Scheduling multi-processor tasks with resource and timing constraints using genetic algorithm, in: The 5th IEEE International Symposium on Computational Intelligence in Robotics and Automation, 2003, pp. 624-629.
[28] M. Sakawa, R. Kubota, Fuzzy programming for multi-objective job shop scheduling with fuzzy processing time and fuzzy due date through genetic algorithms, European Journal of Operational Research 120 (2000) 393–407.
[29] C.C. Lo, G.F. Deng, Using ant colony optimization algorithm to solve airline crew scheduling problems, Third International Conference on Natural Computation (ICNC 2007) 4 (2007) 797–804.
[30] S.T. Lo, R.M. Chen, Y.M. Huang, C.L. Wu, Multiprocessor system scheduling with precedence and resource constraints using an enhanced ant colony system, Expert Systems With Applications 34 (3) (2008) 2071–2081.
[31] M. Zandieh, S.M.T. Fatemi Ghomi, S.M. Moattar Husseini, An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times, Applied Mathematics and Computation 180 (1) (2006) 111–127.
[32] J.Y. Qiu, J. Yin, D. Zhou, An adaptive repulsive particle swarm optimization for makespan and maximum lateness minimization in the permutation flowshop scheduling problem, in: International Workshop on Intelligent Systems and Applications (ISA 2009), May 2009, pp. 1–4.
[33] D.Y. Sha, H.H. Lin, A multi-objective PSO for job-shop scheduling problems, in: International Conference on Computers and Industrial Engineering (CIE 2009), July 2009, pp. 489–494.
[34] S. Jain, S. Meeran, Job shop scheduling using neural networks, International Journal of Production Research 36 (1998) 1249–1272.

[35] R. Thawonmas, N. Shiratori, S. Noguchi, A real time scheduler using neural networks for scheduling independent and nonpreemptable tasks with deadline and resource requirements, IEICE Transactions on Information and Systems E76-D (8) (1993) 947–955.
[36] X. Feng, L. Tang, H.F. Leung, A real time scheduler using generic neural network for scheduling with deadlines, International Conference on Neural Networks and Brain (ICNN& B '05) 1 (2005) 504–508.
[37] Y.J. Shen, M.S. Wang, Optimizing satellite broadcast scheduling problem using the competitive hopfield neural network, in: 2007 Wireless Telecommunications Symposium (WTS 2007), 2007, pp. 1–6.
[38] H. Wang, Q. Zhao, A packet scheduling algorithm in hsdpa system based on hopfield neural network, in: 2008 IEEE International Symposium on Knowledge Acquisition and Modeling Workshop, 2008, pp. 92–95.
[39] Y.T. Ma, K.H. Liu, Y.N. Guo, Artificial neural network modeling approach to power-line communication multi-path channel, in: 2008 International Conference on Neural Networks and Signal Processing, 2008, pp. 229–232.
[40] C. Koulamas, G.J. Kyparisis, Single machine scheduling with release times, deadlines and tardiness objectives, European Journal of Operational Research 133 (2001) 447–453.
[41] H. Chetto, M. Chetto, Some results of the earliest deadline scheduling algorithm, IEEE Transactions on Software Engineering 15 (10) (1989) 1261–1269.
[42] C. Wang, H. Wang, F. Sun, Hopfield neural network approach for task scheduling in a grid environment, in: 2008 International Conference on Computer Science and Software Engineering, vol. 4, 2008, pp. 811–814.
[43] G. Xue, Z. Zhao, M. Ma, M. Ma, Task scheduling by neural network with mean field annealing improvement in grid computing, Canadian Conference on Electrical and Computer Engineering (2006) 554–557.
[44] Y.M. Huang, R.M. Chen, Scheduling multiprocessor job with resource and timing constraints using neural network, IEEE Transactions on System, Man and Cybernetics, Part B 29 (4) (1999) 490–502.
[45] R.M. Chen, Y.M. Huang, Competitive neural network to solve scheduling problem, Neurocomputing 37 (1–4) (2001) 177–196.
[46] J.J. Hopfield, D.W. Tank, Neural computation of decision in optimization problems, Biological Cybernetics 52 (1985) 141–152.