ELSEVIER

European Journal of Operational Research 93 (1996) 288–299

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

A neural network model for scheduling problems

Ihsan Sabuncuoglu [a,*], Burckaan Gurgun [b]

[a] *Department of Industrial Engineering, Bilkent University, Ankara, 06533, Turkey*
[b] *Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL 32611, USA*

Received 1 July 1994; revised 1 August 1995

Abstract

Artificial neural networks (ANNs) have been successfully applied to solve a variety of problems. This paper proposes a new neural network approach to solve the single machine mean tardiness scheduling problem and the minimum makespan job shop scheduling problem. The proposed network combines the characteristics of neural networks and algorithmic approaches. The performance of the network is compared with the existing scheduling algorithms under various experimental conditions. A comprehensive bibliography is also provided in the paper.

Keywords: Scheduling; Neural networks

1. Introduction

Over the last decade Artificial Neural Networks (ANNs) have been in many areas ranging from manufacturing and military to finance and marketing. Especially, successful applications of ANNs to various classification (i.e., pattern recognition) problems have caused growing research interests in neural networks. The ability to map and solve combinatorial optimization problems using neural networks has also motivated researchers since the beginning of ANN research. As a result of these investigations, several neural network models have been developed for a variety of optimization problems (e.g., travelling salesman and graph partitioning problems). These applications have demonstrated that ANNs may not be as effective as conventional OR tools, but their inherent parallelism (i.e., parallel processing) offers some advantages in dealing with large scale optimization problems.

Detailed discussions on neural networks and their applications are beyond the scope of this paper. The reader can refer to Burke and Ignizo (1992), Zahedi (1991), and Masson and Wang (1990) for an introduction to ANNs from the OR perspective. In addition, the paper by Sharda (1994) provides a comprehensive annotated bibliography of neural networks for MS/OR professionals. There are also excellent survey papers on manufacturing applications (Udo and Gupta, 1994), combinatorial optimization problems (Looi, 1992). The focus of our paper is on scheduling problems and their solution with neural networks. Specifically, we present a survey of the ANN literature pertaining to scheduling and develop a new neural network model to solve two well known scheduling problems.

The rest of the paper is organized as follows. Section 2 provides a survey of the ANN scheduling literature. In Section 3, the detailed description of the proposed network is given. This is followed by

* Corresponding author.

PII S0377-2217(96)00041-0

implementation of the proposed network in Section 4. First, the proposed network is used to solve the single machine mean tardiness scheduling problem and then its computational results are presented for the job shop scheduling problem. Finally, concluding remarks are given and future research directions are identified in Section 5.

2. Literature review

In general, scheduling problems have received a lot of interests from ANN researchers. As a result, a number of neural networks have been developed to solve a wide range of scheduling problems. The exiting studies can be classified according to the following network structures (or types): 1) Hopfield model and other optimizing networks, 2) Competitive networks, 3) Back propagation networks.

The majority of the existing studies are based on Hopfield network or its extensions. This may be partly due to successful applications of Hopfield network to the Traveling Salesman Problem (TSP) and other optimization problems. Essentially, the Hopfield network (Hopfield, 1982) is a single layered and fully interconnected neural network model. It is an optimizer in the sense that the states of the neurons are updated in a random and asynchronous manner to minimize the energy of the network. The most important part of any neural network implementation is to find a proper map from the problem to the network. In the Hopfield case, this is accomplished by coding both the objective function (i.e., soft constraints) and hard constraints (i.e., constraints of the original problem) into a single energy function with appropriate connection weights. For example, the TSP is mapped by Hopfield and Tank (1985) to a two dimensional neuron matrix using N neurons, where N is the number of cities.

Foo and Takefuji (1988a,b) have utilized the Hopfield approach to map the $n|m$ job shop scheduling problem to an mn by $(mn + 1)$ 2D neuron matrix. In this formulation the energy function is composed of hard constraints (i.e., precedence and resource constraints) and the cost of total completion times of all jobs. The method has been applied to several 4|3 job shop scheduling problems. In their later work, Foo and Takefuji (1988c) represented the same problem as an integer linear program and developed a Hopfield based network called an integer linear programming neural network (ILPNN). With this new formulation, the total number of neurons in the network was reduced to $nm(nm + 1)/2$. Zhou et al. (1990, 1991) further improved the performance of ILPNN by unifying the indices of operations and machines to obtain simpler integer programming representation of the problem. In a later study, Van Hulle (1991a,b) replaced the single objective of ILPNN by several objectives to design a goal programming network. This network was also used for job shop scheduling problems. The further discussions on the Hopfield-based networks and their applications to scheduling problems can be found in Lo and Bavarian (1991), Gulati and Iyengar (1987), Arizono et al. (1992), Vaithyanathan and Ignizo (1992), and Johnston and Adorf (1992).

In competitive networks, the inhibitory links are established as a result of competition rather than being determined initially as in the hopfield case. In designing such a network, one usually develops equations of motion for the elements of the problem and defines an appropriate energy function to show the convergence of the network. Indeed, this network type has not received enough attention from ANN scheduling research. There are only two reported applications of competitive networks (Fang and Li, 1990; Pellerin and Herault, 1994). Whereas, back-propagation networks (BP nets) have been used more frequently. Especially, their generalization property has been explored to find the relationship between problem data and optimal schedules (Sabuncuoglu and Hommertzheim, 1992; Hayes and Sayegh, 1992), to determine the proper value of a look ahead parameter of a job priority rule (Kim and Lee, 1993), and to establish adequate weights between an operational policy at the work center level and the overall performance measure of a manufacturing system (Chryssolouris et al., 1991). They have also been used together with OR and AI tools in an integrated manner for realtime scheduling systems (Rabelo et al., 1993; Yih et al., 1993).

In the ANN literature, another related topic is simulated annealing (SA). SA is not a neural network. But it has proved to be useful in some of the neural network applications. In general, the neural networks discussed in this paper minimize some function during either the learning process (e.g.,

error function in a BP network) or the recall process (e.g. energy function in a Hopfield network). The problem with the conventional search methods (e.g., gradient descent) is that once a local optimum is detected, further improvements cannot be made. As a stochastic search method, SA just tries to overcome this problem. For that reason, it has been applied successfully to various scheduling problems (Osman and Potts, 1989; Potts and Van Wassenhove, 1991). The interesting readers can also refer to Van Laarhoven et al. (1992) to see how the SA algorithm asymptotically converges to a global minimum solution.

In summary, all these studies showed that scheduling problems can be attacked by neural networks. At present, the neural network approach may not seem to be competitive with the conventional OR algorithms in terms of the quality of solutions and computational times. However, their inherent parallelism (i.e., parallel processing) offers some advantages that should be explored in the future research work. It is also observed that the majority of the scheduling applications of neural networks are based on the Hopfield network. There are only few applications of competition based networks. Although the exiting applications of ANNs yield promising results, they still inherit the problems of the Hopfield model in the TSP applications (i.e., feasibility problems and excessive computation times on the large size problems). Moreover, performances of these networks have not been adequately measured. In most of the cases, they were applied to simple instances of scheduling problems. Because there is a lack of objective comparisons between ANNs and OR based scheduling algorithms, it is hard to judge on their relative performances. Hence, there is a need for further experimental studies to test performances of ANNs under various conditions, in addition to the existing research efforts to develop new neural models for scheduling and other combinatorial optimization problems.

3. The proposed network

In this section, we describe the structure of a new neural network model developed for scheduling problems. The proposed model is similar to the Hopfield network, but yet there are some noticeable differences. The main difference is that it has an additional external processor to monitor and control evolution of the network. With this processor, the network is also forced to converge to a final solution after a certain number of iterations. In our case, the network is emulated on a traditional computer which employs sequential algorithms for ANN implementations. Here, the external processor is used to accomplish parallelism in the proposed network. The physical (i.e., electronic) equivalent of the proposed network might eliminate this processor with its parallel architecture, but our aim is not to design such a physical ANN.

In the proposed model, the hard constraints (i.e., feasibility constraints) of the problem are also dropped from the energy function and both feasibility and cost calculations are carried out by this external processor. Hence, as compared to the Hopfield model, most of the interconnections are eliminated and the network is simplified. The network also possesses a "competition" property. With this property, the neurons (representing jobs in scheduling problems) are allowed to compete with each other to get the first available position in the sequence.

In the proposed model, a sequence of jobs (tasks) on a given machine (resource) is represented by an $n \times n$ neuron matrix. In this representation, each row refers to a job and each column indicates a position in the sequence (or schedule). Since each job can occupy at most one position, the final feasible solution matrix has only one activated neuron at each row and column with the rest of the elements are zero (i.e., a permutation matrix). In what follows, we define the state of a neuron $a_{i,j} = 1$ when it is activated (i.e., job i is assigned to the jth position), $a_{i,j} = 0$ when it is deactivated (i.e., job i is not assigned to the jth position). If the activation value is between 0 and 1 (i.e., $0 < a_{i,j} < 1$) then it means that job i is assigned to the jth position with a probability of a_{ij}. In the single machine scheduling case, having a final matrix as a permutation matrix gives a solution for the problem (Fig. 1).

In the job shop scheduling problem, there are m matrices corresponding to m machines. Hence, we define a three dimensional matrix of size $m \times n \times n$. As a notation, we call such an $n \times n$ matrix that represents a machine, a *layer*. Again, each row

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 \\ 0.00 & 0.00 & 1.00 \end{bmatrix}$$

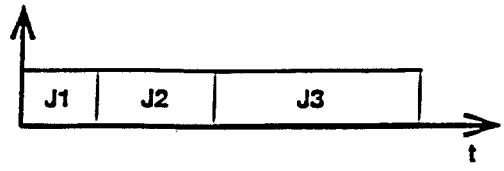

Fig. 1. A neural matrix and the resulting schedule for a three job problem.

indicates an operation of a job and each column indicates a position of this operation in the sequence of that machine.

In general, the Hopfield-based networks use inhibitory connections to achieve feasibility. In the proposed model, this is partially achieved by normalization (i.e. dividing activation values of each neuron by the sum of the activation values of its row and column in the matrix). Moreover, the activation values of the neurons are passed from a sigmoid function so that some of neurons get stronger and the rest get weaker. Hence, neurons compete with each other to win the first available position in the sequence. During this competition process, the sum of each row and column are kept constant (i.e., one) so that only one neuron with the highest activation value in each row and column reaches to 1 while get zeros. At the end of the process, the neural matrix becomes a permutation matrix and a final feasible schedule is decoded. In the formulation, the set of positions that has only one neuron with the activation value 1 and others 0 is called *Partial Sequence Positions* (PSP) set. Thus, the neuron matrix becomes a permutation matrix when all positions of the schedule are added into the PSP set.

In addition to feasibility, the network searches for a minimum energy level corresponding to the value of a cost function (i.e., makespan or mean tardiness). Both starting with a suitable initial neuron matrix and evolution of neurons are required to obtain good solutions. In our case, initialization of the network is a problem specific issue and will be discussed in the applications below. During the evolution of the network, the external processor makes interchanges of two randomly selected rows corresponding to the

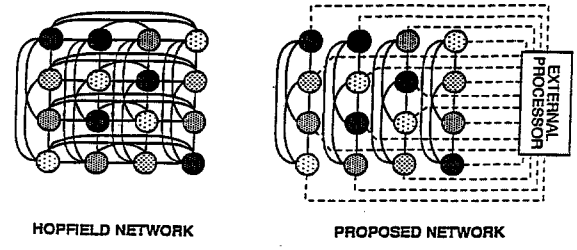

Fig. 2. A Hopfield-network and the proposed Network.

positions of two jobs in the sequence of a machine. In the job shop case, machines are also selected randomly as well as jobs. The energy (or cost) for each interchange is computed by this processor. If the energy of the network is improved after this interchange, then the new state of network is used. Otherwise, the neurons assume their previous states and the same process is repeated. In the job shop case, an annealing procedure is also applied. According to this procedure if the energy of the network does not decrease for some number of iterations, an interchange that causes an increase of energy up to 10% is accepted. But the previous minimum energy state is saved in the memory in case that a lower energy state is not reached by the network again.

As compared to the Hopfield networks, the proposed network is not a fully connected graph. There is also an external processor to support the network operations. As seen in Fig. 2 and Fig. 3, connections of the proposed network are directed edges that transfer the activation values of neurons corresponding to the positions of jobs in the sequence. Also, all neurons are connected to the external processor by directed edges to provide communication between the network and the processor.

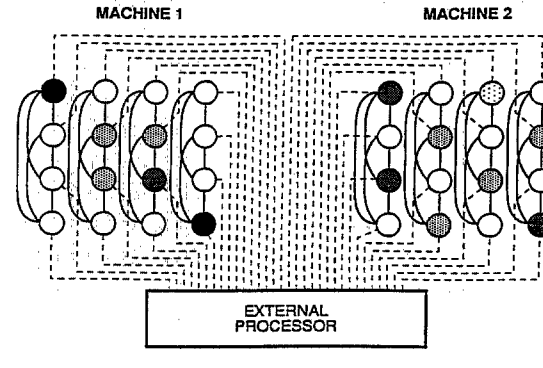

Fig. 3. The proposed network for the job shop problem.

The basic functions of this external processor are as follows:

(i) Sequentially selecting two random rows during the interchange process, where all neurons in those rows are involved in the decision process and affected; in traditional Hopfield network emulations a single neuron is randomly selected and involved in the process. As a result, the external processor provides a higher level parallelism in our implementation.

(ii) During the interchange process rows may violate the normalized matrix of neurons (i.e., feasibility). The normalization is enforced by the external process so that the feasibility constraints can be removed from the ANN model. The external processor is entitled to monitor and change the neuron values when necessary, to keep the matrix normalized.

(iii) Additionally, the external processor is involved in calculation of the expected cost (i.e., energy function) as it can monitor the overall network. It is much easier and faster for the external processor to calculate the expected cost in the emulation model.

4. Applications of the proposed approach

In this section, the structure and computational requirements of the proposed approach are discussed in detail using the single machine mean tardiness scheduling problem and the minimum makespan job shop scheduling problem. These two problems were selected because they are the most studied NP-hard problems in the scheduling literature.

4.1. Single machine mean tardiness problem

The single machine mean tardiness problem is defined as follows: a single machine is to process n jobs with known processing times (p) and due dates (d). All the jobs are ready at time zero. Tardiness is the positive lateness a job incurs if it is completed after its due date and the objective is to sequence the jobs to minimize the mean tardiness. This problem has been recently proved to be NP-hard (Du and Leung, 1990). In general, research efforts in the single machine mean tardiness problem have followed two approaches: optimization and heuristics. In the former approach, several exact methods based on implicit enumeration have been developed (Fisher, 1976; Schrage and Baker, 1978). The current limit on solvability of this problem is around 100 jobs. In the second track, various heuristic procedures have been proposed ranging from simple dispatch rules to more sophisticated algorithms (Panwalker et al., 1993; Potts and Van Wassenhove, 1991; Wilkerson and Irwin, 1971).

In the proposed approach, the neuron matrix may not be a permutation matrix at the beginning and intermediate stages of the evaluation process due to the random initialization of the activation values. For that reason, the external processor computes expected mean tardiness, $E[T]$, rather than exact mean tardiness. Assuming $a_{i,j}$ is the probability of assigning job i to the jth position and using p_i and d_i as the processing time and the due date of job i respectively, the expected mean tardiness (or the energy function to be minimized) is defined as follows:

$$E[T] = \sum_{i=1}^{n} \frac{\max(0, E[C_i] - d_i)}{n} \tag{1}$$

where $E[C_i]$ is the expected completion time of job i and it is calculated as

$$E[C_i] = \sum_{j=1}^{h} a_{ij} \left(\sum_{l=1}^{j-1} \sum_{k=1}^{n} a_{kl} p_i + a_{ij} p_i \right)$$

or

$$E[C_i] = \sum_{j=1}^{n} a_{ij} \sum_{l=1}^{j-1} \sum_{k=l}^{n} a_{kl} p_k + p_i \tag{2}$$

since

$$\sum_{j=1}^{n} a_{ij} = 1$$

Since the number of positions added into the PSP set increases as the network evolves, the cost is computed faster as the algorithm proceeds. The complexity of the proposed model is mainly determined by the complexity of calculating the cost. In our case, the initial neuron matrix gives a completely infeasible schedule due to random initializations of the neurons. This brings $O(n^2)$ complexity in the worst case because the double sum over n is used for calculating $E[C_i]$. However, when the neuron matrix

produces a completely feasible schedule (or permutation matrix) at the end of the process, this complexity is reduced to $O(n)$.

During early development stages of the prorosed network, pilot experiments have been performed to set the values of parameters (i.e., temperature and shift parameters of the sigmoid function). In addition, three alternative ways of network initialization have been investigated. These are: (1) randomly activated neurons, (2) randomly activated neurons biased to SPT (shortest processing times) sequence (i.e., the activation values of the neurons corresponding to the SPT sequence are higher than others), (3) randomly activated neurons biased to EDD (earliest due dates) sequence in which case the activation values of the neurons corresponding to the EDD sequence are relatively higher than others. The results of the experiments indicated that the second type of initialization performs well when the tardiness factor (TF) is smaller than 0.6, otherwise the third is a better choice. Here, tardiness factor measures the expected proportion of tardiness in a given job population. The procedure of the proposed approach is as follows:

Procedure 1.

Step 1. Initialize the neuron matrix as explained above.

Step 2. Pass the activation values of the jobs from the following sigmoid function:

$$a'_{ij} = \frac{1}{\exp\left(-\left(a_{ij} - S\right)/T\right)}$$

,where S and T are the shift and the temperature parameters, respectively.

Step 3. Normalize the neuron matrix as:

$$a'_{ij} = a_{ij} / \sum_{k=1}^{n} a_{ik}$$

for rows first and then

$$a'_{ij} = a_{ij} / \sum_{k=1}^{n} a_{kj}$$

for columns.

Step 4. Compute the value of the energy function (i.e., expected tardiness) as

$$E[T] = \sum_{i=1}^{n} \frac{\max(0, E[C_i] - d_i)}{n}.$$

Step 5. Select two rows (jobs) randomly, interchange their activation values and compute the energy function again.

Step 6. If the energy function is improved accept the new state, else return it to the previous state.

Step 7. Periodically (after a predetermined number of iterations), select a column beginning from the first position in the schedule. Assign the neuron with the highest activation value to 1 and make other neurons 0 in the selected column (i.e. one more position is included in the PSP set).

Step 8. Normalize the neuron matrix again.

Step 9. If the matrix is still infeasible go to step 2, else go to step 10.

Step 10. Even though, all positions of the neuron matrix are feasible repeat the steps 2 through 9 for some number of iterations and stop.

Step 7 is included in the procedure to reduce the excessive computations that may be required by the neurons to converge to 0 or 1 states. The number iterations at which this step is invoked has been determined as a result of the pilot simulation experiments (about 10000 randomly generated problems were solved). It was noted that this number is a function of a constant multiplied by the problem size n. During these experiments, it was also observed that the number of iterations required to obtain a permutation matrix is a square function of the number of jobs n.

The Step 10 is included to obtain further improvement in the performance of the network. Essentially, at this step we perform a simple neighborhood search with random job interchanges.

4.1.1. Performance evaluation and experimental results

The relative performance of the proposed network is measured against two phase algorithm of Wilkerson and Irwin (WI). The WI algorithm was selected for comparisons because it is generally used as a benchmark in the relevant literature. The other reason was that its code is readily available. Both the

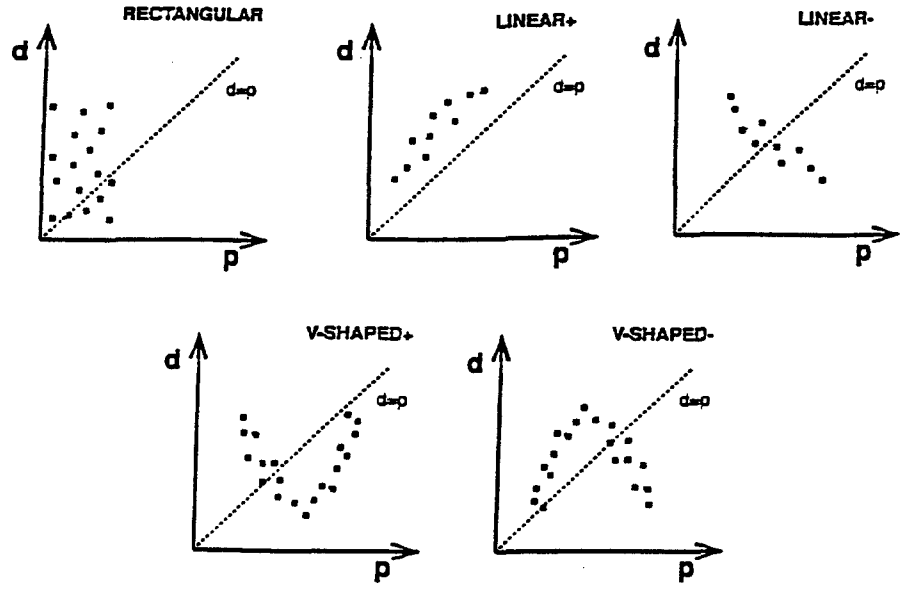

Fig. 4. The geometric shapes of the data types.

proposed network and WI were coded using C programming language and run on SUN Spark 2 workstations.

In the previous studies, test problems have been generated from the uniform distributions by using two problem parameters: TF (Tardiness Factor) and RDD (Range of Due Date). TF measures the rate of expected proportion of tardiness of jobs and is calculated as $TF = 1 - \bar{d}/(n\bar{p})$. The second parameter RDD defines the range of due date and is computed as $(d_{max} - d_{min})/n\bar{p}$ where $\bar{p}$ is the average processing time of the jobs and $\bar{d}$, d_{min}, and d_{max} are the average, minimum and maximum due dates, respectively. Thus, a geometric shape of the data when mapped onto a two dimensional graph (x-axis: processing times, y-axis: due dates) usually resembles a rectangular shape. In this study, we also used other data types such as linear +, linear −, V-shaped +, and V-shaped − Fig. 4). No research to date has studied WI or other single machine scheduling algorithms for these problem characteristics. In all of the test problems, processing times were generated from the uniform distribution with the range of 100. Due dates were also sampled from the uniform distribu-

Table 1
Comparison of WI and the proposed network on the single machine problem

Data type	Problem size	WI/T	WI/C	ANN/T	ANN/C	%Δ
Rectangular	50 jobs	235.86	0.30	232.76	51.25	1.32
Rectangular	100 jobs	913.26	68.51	899.15	626.09	1.54
Linear +	50 jobs	191.50	0.01	191.43	50.99	0.04
Linear +	100 jobs	742.56	0.05	742.45	640.14	0.01
Linear −	50 jobs	321.29	1097.1	306.96	58.14	4.46
V-shaped +	50 jobs	237.25	0.09	236.51	57.73	0.31
V-shaped +	100 jobs	921.07	15.04	917.80	624.05	0.35
V-shaped −	50 jobs	240.81	4.26	232.48	50.50	3.46

tion with the ranges specified by TF and RDD parameters.

In the experiments, we used two problem sizes (i.e., $n = 50$ and $n = 100$). For each of the five data types, values of TF and RDD were varied between 0.1 and 0.9. Ten random samples (or replications) were taken at each experimental point that resulted in total of $2 \times 5 \times 5 \times 5 \times 10 = 2500$ test problems. Table 1 summarizes the results in terms of the mean tardiness and the computation time (in seconds). WI/T and ANN/T represent the mean tardiness of WI and the proposed network. Whereas WI/C and ANN/C stand for computation times of WI and the proposed model, respectively. The symbol %Δ indicates the percentage of improvement in mean tardiness of the proposed method over WI.

The results in Table 1 showed that the proposed approach improves the mean tardiness from 0.01% to 4.46%. However, it requires more computation times than WI. Note that the largest improvement in tardiness was achieved for the linear data type. As indicated by Tansel and Sabuncuoglu (1994, 1996), this data type is one of the hard data patterns that probably makes the problem intractable. We also know from previous studies that an opposite pattern of the previous data type (i.e, linear +) defines one of the easy problem instances which qualify SPT optimality. As shown in Table 1, the smallest improvement was obtained by the network for this data type.

A Confidence Interval (CI) approach was also used to determine whether differences in mean tardiness performances of the methods are statistically significant. As shown in Table 2, the test results are significant in favor of the proposed method at $\alpha =$ 0.05. This is because none of confidence intervals constructed by using the paired-t approach contained zero. The CI approach gives more information than the corresponding hypothesis testing. It does not only indicates the significance but also gives magnitude of the observed differences. As can be noted, the 95% CI ranges in Table 2 are very wide even though the proposed network produces better results than WI in most of the test problems. This is mainly due to the variance of performance differences between the methods.

Table 2
Confidence interval (CI) tests for the relative mean tardiness difference between the proposed method and WI

Data type	Problem size	Lower limit	Upper limit
Rectangular	50 jobs	0.509	5.698
Rectangular	100 jobs	1.624	26.595
Linear+	50 jobs	0.027	0.109
Linear+	100 jobs	0.051	0.153
Linear−	50 jobs	5.146	23.508
V-shaped+	50 jobs	0.294	1.178
V-shaped+	100 jobs	0.062	6.466
V-shaped−	50 jobs	1.814	14.845

4.2. The job shop scheduling problem

After obtained the promising results from the proposed network on the single machine scheduling problem, it has been applied to the minimum makespan job shop scheduling problem. This problem is defined as follows: a shop consisting of m machines is to process n jobs with a general process flow in a minimum total completion time. Jobs consist of m sub jobs, called operations each of which requires a specified machine for an uninterrupted duration, called its processing time. All the jobs are ready at time zero. The objective is to determine a set of start and completion times of each operation such that the time required to complete all the jobs (i.e. makespan) is minimized. This problem is also one of the most studied problems in the literature that there are a number of exact algorithms and heuristics for its solution (Adams et al., 1988; Lageweg et al., 1977; Lawrence, 1984).

The neural network proposed for this problem is an extension of the previous network used for the single machine scheduling problem. Again, the schedule is determined by using the neural matrix with $n \times n$ nodes. But this time, there are m matrices corresponding to m machines (or layers). In the job shop case, having a permutation matrix at each layer (i.e. satisfying resource constraints) is not sufficient for the feasibility of the overall problem. Any feasible schedule must also satisfy precedence constraints.

In the present approach, in order to satisfy the precedence constraints the network (m of $n \times n$ neuron matrices) is initialized using the following procedure: first, all of the activation values are set to

0 and the first operation of each job is activated at the first position of its corresponding layer (or machine represented by an $n \times n$ matrix) based on its job routing. Since there may be more than one operation activated in that position (i.e., a machine has to process more than one part at the same time), we may produce an infeasible schedule. In order to overcome this problem, when two or more jobs demand for the same position on a machine, additional positions are made on that machine so that each demand from an operation is fulfilled by a single position; hence each position in the sequence receives a single demand. By this way, the number of positions allocated on the machine will be equal to the number of operations assigned to that particular machine. When the above procedure is applied to each each layer, the positions in the layers contain at least one activated neuron as each of the n jobs visits each machine once. For that reason, this initial state eliminates the possibility of scheduling more than one operation to a machine at the same time. Then the network is normalized and evolution of the network proceeds. Note that the proposed network only provides the sequence of operations for each machine. As a result, it is not possible for the network to propose an infeasible schedule in terms of precedence constraints. These sequences are further processed by the external processor to determine starting and ending times of each operation.

In the proposed method, the PSP set is again used to compute the energy of the network. Initially, the PSP set is an empty set. In order to place new positions in the PSP set, the following procedure is applied periodically and sequentially to every position beginning from the first position: select a column that corresponds to the position to be included in the PSP set for each layer. The neuron with the highest activation value in the column is determined and its state is set to 1 while the states of others are set to 0. If there are two or more neurons having the same activation values, one of the neurons is randomly selected. In order to calculate the makespan of the partial schedule at any intermediate stage, starting time of an operation is assigned to the maximum of the completion times of operations preceding it. These starting times are updated until no overlaps and any local shifts exist. The largest completion time of the scheduled operations is the energy of the network (i.e. the makespan). At the end of the evolution of the network, each layer becomes a permutation matrix (i.e., all the positions are included in the PSP set) and the external processor produces a schedule (i.e., a Gantt chart is developed by using the sequence on each machine). The energy of the network corresponds to the makespan of the schedule generated.

The evolution of the network is carried out similar to the single machine case (i.e., based on interchanges of the operations). However, machines are also selected randomly. Moreover, an annealing procedure is also applied to improve the performance of the network. According to this procedure, if the subsequent interchanges do not reduce the energy of the network for some number of iterations, cost increase up to 10% is permitted while keeping the minimum current schedule in the memory. This helps the network to escape from a possible local minima. If the cost is not reduced as a result of annealing, the network is returned back to the previous state.

Although the evolution of the network is basically similar to the single machine case, there are some differences. These are: (1) initialization procedure, (2) computation of the energy function, (3) the additional machine selection procedure, and (4) application of an annealing procedure. The proposed network uses the following procedure to obtain a feasible solution.

Procedure 2.

Step 1. Initialize the neuron matrix as explained above.

Step 2. Add the first two positions to the PSP set.

Step 3. Normalize the neuron matrix of each layer and compute the energy function.

Step 4. Select a machine and two rows (jobs) within the PSP set randomly, interchange their activation valugs and compute the energy function.

Step 5. If the energy function is improved by the new state assume the new state, else return back to the previous state.

Step 6. Periodically (after some number of iterations), select a column from every layer beginning from the first position of the job routing. Add this position in the PSP set, i.e., the neuron with the highest activation value is assigned to 1 and other

neurons are assigned to 0 in the selected column. Normalize the neuron matrix again.

Step 7. If there are still operations not included to the PSP set, go to step 4, else go to step 9.

Step 8. Even though, all operations are added into PSP set, repeat steps 4 through 8 for some number of iterations and stop.

As in the case of single machine case, pilot experiments were also performed to determine values of some of the parameters. These were: (1) the duration of the period after which a position is included in the PSP set, (2) the percentage deviation that is allowed during annealing, and (3) the number of iterations. The first parameter is set automatically by the network in terms of a time period during which the energy of the network decreases. As specified previously, the second parameter is set to 10%. During initial experiments it was also observed that the number of iterations is directly proportional to size of the problem. Hence, after several trials this parameter was set to $L \times m \times n$ (m is the number of machines, n is the number of jobs, L is a constant). The current value of L is 5000 because it yields satisfactory results with the problems tried in this study. Similarly, when the makespan of the schedule is computed at the beginning of procedure, the complexity of computing cost is directly proportional to m. At the end of the evolution, however, the complexity becomes $O(mn^2)$ due to m updates of n operations on m machines.

4.2.1. The performance of the proposed network on job shop problems

The performance of the proposed method has been measured on a number of job shop scheduling problems. The problems given by Applegate and Cook (1990) were used in the experiments. In Table 3, the item "problem reference" indicates the identity of the problem. For example, the problem which is named, MT06 is the problem number 6 cited by Muth and Thompson (1963). Similarly, ABZ's are cited by Adams et al. (1988) and LA's are cited by Lawrence (1984).

In all these problems, the number of operations of each job equal the number of machines and each job has exactly one operation on each machine. The results in Table 3 showed that the proposed approach gives very promising results. It found the optimal solutions in 18 out of 25 problems. The problems for which optimal solutions were found are the problem sizes 6|6, 15|5, 20|5 and 15|10. In one of the five 10|5 job shop problems, a near optimal solution was found with 1.2% deviation from the optimality. The near optimal solutions were also found in six of the seven 10|10 job shop problems with the average 0.04 percentage deviation from the optimality. It was also observed that the computational requirements of the algorithm increases polynomial with respect to the size of the problem. In our approach, the number of iterations and the computational requirements per iteration are the same for any problem with the same size (i.e., same m and n). However, there might be some differences in the computational times for the problems that have the same size due to some random iterations added to recover from local minimums. Unfortunately, the computational times presented in Table 3 have the accuracy 10 seconds

Table 3
Performance of the proposed method on job shop scheduling problems

Problem reference	Problem size	Optimum	ANN result	ANN/T (s)
MT06	6|6	55	55	280
LA01	10|5	666	666	920
LA02	10|5	655	655	920
LA03	10|5	597	604	920
LA04	10|5	590	590	920
LA05	10|5	593	593	920
LA06	15|5	926	926	2300
LA07	15|5	890	890	2300
LA08	15|5	863	863	2300
LA09	15|5	951	951	2300
LA10	15|5	958	958	2300
LA11	20|5	1222	1222	3700
LA12	20|5	1039	1039	3700
LA13	20|5	1150	1150	3700
LA14	20|5	1292	1292	3700
LA15	20|5	1207	1207	3700
MT20	20|5	1165	1165	3700
MT10	10|10	930	940	6800
ABZ5	10|10	1234	1239	6800
ABZ6	10|10	943	947	6800
LA16	10|10	945	946	6800
LA17	10|10	784	786	6800
LA19	10|10	842	842	6800
LA20	10|10	902	907	6800
LA29	15|10	1032	1032	19172

except the last experiment which has the accuracy of 2 seconds. Hence, the fluctuations of the computational times between the same size problems are not observable in the experimental results. But those fluctuations are not more than 10 seconds for the problems of the same size.

5. Concluding remarks

In this paper, ANN scheduling literature has been reviewed and a new neural network model has been proposed. The performance of the proposed network has also been measured on various test problems. In general, the results are very encouraging. It generates better solutions than WI for the single machine problem and finds the optimum solutions for most of the selected problems of the shop job scheduling. Continuing research is now being directed toward improving the performance of the network and developing a hybrid OR/ANN methodology that incorporates both qualitative and quantitative aspects of scheduling problems.

Acknowledgements

The authors wish to thank the referees for their detailed comments and constructive criticisms of the initial draft.

References

Adams, J., Balas, E. and Zawack, D. (1988), "The shifting bottleneck procedure for job shop scheduling", *Management Science* 34, 391–401.

Arizono, I., Yamamoto, A. and Ohto, H. (1992), "Scheduling for minimizing total actual flow time by neural networks", *International Journal of Production Research* 30, 503–511.

Applegate, D. and Cook, W. (1990), "A Computational study of job shop scheduling", Technical paper:CMU–CS–90–145, Carnegie Mellon University, Pittsburgh.

Burke, L.I. and Ignizo, J.P. (1992), "Neural networks and operations Research: an overview", *Computers and Operations Research* 19, 179–189.

Chryssolouris, G., Lee, M. and Domroese, M. (1991),"The use of neural networks in determining operational policies for manufacturing systems", *Journal of Manufacturing Systems* 10, 166–175.

Du, J. and Leung, J.Y.T. (1990), "Minimizing on one processor is NP-Hard", *Mathematics of Operations Research* 15, 483–495.

Fang, L. and Li, T. (1990), "Design of competition based neural networks for combinatorial optimization", *International Journal of Neural Systems* 1, 221–235.

Fisher, M.L. (1976), "A dual algorithm for the one machine scheduling problem", *Mathematical Programming* 11, 229–251.

Foo, Y.P.S. and Takefuji, Y. (1988a), "Stochastic neural networks for solving job shop scheduling: part 1. Problem presentation", *Proceedings of the International Conference on Neural Networks* 2, 275–282.

Foo, Y.P.S. and Takefuji, Y. (1988b), "Stochastic neural networks for solving job shop scheduling: part 2. Architecture and simulations", *Proceedings of the International Conference on Neural Networks* 2, 283–290.

Foo, Y.P.S. and Takefuji, Y.(1988c),"Integer linear programming neural networks for job shop scheduling", *Proceedings of the International Conference on Neural Networks* 2, 341–348.

Gulati, S. and Iyengar, S.S.(1987), "Nonlinear networks for deterministic scheduling", *Proceedings of the International Conference on Neural Networks* 4, 745–752.

Hayes, P.V. and Sayegh, S.I. (1992),"A supervised neural network approach to optimization as applied to the N-Job, M-Machine job sequencing problem", *Proceedings of ANNIE'92*.

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

Hopfield, J.J. and Tank, D.W. (1985), "Neural Ccmputation of decisions in optimization problems", *Biological Cybernetics* 52, 141–152.

Johnston, M.D. and Adorf, H.M. (1992), "Scheduling with neural networks — The case of Hubble Space Telescope", *Computers and Operations Research* 19, 179–189.

Kim, S. and Lee, Y. (1993), "Enhancement of a job sequencing rule using an artificial neural network", *2nd Industrial Engineering Research Conference Proceedings* 842–846.

Lageweg, B.J., Lenstra, J.K. and Rinnooy Kan, A.H.G. (1977), "Job shop scheduling by implicit enumeration", *Management Science* 24, 441–450.

Lawrence, S. (1984), "Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques", Technical Report, Carnegie Mellon University, Pittsburgh.

Lo, Z.P. and Bavarian, B. (1991), "Scheduling with neural networks for fexible manufacturing systems", *Proceedings of the IEEE International Conference on Robotics and Automation* California, 818–823.

Looi, C. (1992), "Neural network methods in combinatorial optimization", *Computers and Operations Research* 19, 191–208.

Masson, E. and Wang, Y. (1990), "Introduction to computation and learning in artificial neural networks", *European Journal of Operational Research* 47, 1–28.

Muth, J.F. and Thompson, G.L. (1963), *Industrial Scheduling*, Prentice-Hall, Englewood Cliffs, New Jersey.

Osman, I.H. and Potts, C.N. (1989), "Simulated annealing for permutation flow shop scheduling", *Omega* 17, 551–557.

Panwalker, S.S., Smith, M.L. and Koulamas, C.P. (1993),"A heuristic for the single machine tardiness problem", *European Journal of Operational Research* 70, 304–310.

Potts, C.N. and Van Wassenhove, L.N. (1991), "Single machine tardiness sequencing heuristics", *IIE Transactions* 23, 346–354.

Rabelo, L., Jones, A. and Tsai, J. (1993),"Using hybrid systems for FMS scheduling", *2nd Industrial Engineering Research Conference Proceedings* 471–475.

Sabuncuoglu, I. and Hommertzheim, D. (1992),"Artificial neural networks: investigations and developments of neural networks for scheduling problems", *TIMS/ORSA Joint national Meeting* USA.

Sharda, R.(1994), "Neural networks for the MS/OR analyst: an application bibliography", *Interfaces* 24, 116–130.

Schrage, L. and Baker, K.R. (1978),"Dynamic programming solution of sequencing problem with precedence constraints", *Operations Research* 26, 444–449.

Tansel, B. and Sabuncuoglu, I. (1994b), "Geometry based analysis of single machine tardiness problem and implications on solvability", Technical Report, Department of Industrial Engineering, Bilkent University, Turkey.

Tansel, B. and Sabuncuoglu, I. (1994a), "New insights on the single machine total tardiness scheduling problem", *Journal of the Operational Research Society* (forthcoming).

Udo, G.J. and Gupta, Y.P. (1994), "Applications of neural networks in manufacturing management systems", *Production Planning and Control* 5, 258–270.

Vaithyanathan, S. and Ignizo, J.P. (1992),"Stochastic neural network for resource constrained scheduling", *Computers and Operations Research* 19, 241–254.

Van Hulle, M.M.(1991a), "A goal programming network for linear programming", *Biological Cybernetics* 65, 243–252.

Van Hulle, M.M.(1991b), "A goal programming network for mixed integer linear programming: a case study for the job shop scheduling problem", *International Journal of Neural Systems* 2/3, 201–209.

Van Laarhoven, P.J.M., Aarts, E.H.L. and Lenstra, J.K. (1992), "Job Shop Scheduling by Simulated Annealing", *Operations Research* 40/1, 113–125.

Yih, Y., Liang, T. and Moskowitz, H. (1993),"Robot scheduling in a circuit board production line:a hybrid ORR/ANN approach", *IIE Transactions* 5, 26–33.

Wilkerson, L.j. and Irwin, J.D. (1971), "An improvement method for scheduling independent tasks", *AIIE Transactions* 3, 113–125.

Zahedi, F. (1991), "An introduction to neural networks and a comparison with artificial intelligence and expert systems", *Interfaces* 21, 25–38.

Zhou, D.N., Cherkassky, V., Baldwin, T.R. and Olson, D.E. (1991), "A neural approach to job shop scheduling", *IEEE Transactions on Neural Networks* 2, 175–179.

Zhou, D.N., Cherkassky, V., Baldwin, T.R. and Hong, D.W. (1990), "Scaling neural network for job shop scheduling", *Proceedings of the International Conference on Neural Networks* 3, 889–893.