Human-centric Computing and Information Sciences
a SpringerOpen Journal

RESEARCH Open Access

A subjective job scheduler based on a backpropagation neural network

Anilkumar Kothalil Gopalakrishnan

Correspondence: anil@scitech.au.edu
Distributed and Parallel Computing Research Laboratory, Department of Computer Science, Faculty of Science and Technology, Assumption University, Soi 24, Ramkhamheang Road, Hua Mak, Bang Kapi, Bangkok 10240, Thailand

Abstract

This paper aims to present and discuss the concept of a subjective job scheduler based on a Backpropagation Neural Network (BPNN) and a greedy job alignment procedure. The subjective criteria of the scheduler depend on the solution plan for a given job scheduling problem. When the scheduler is provided with desired job selection criteria for the problem, it generates user satisfying solution from a set of valid jobs. The job validation procedure is based on the similarity measure of the jobs with the seen dataset of the scheduler. The seen dataset is based on the subjective criteria of the scheduler. The prioritized and valid jobs are allowed to execute concurrently on the given identical machines. The satisfying criterion of the scheduler indicates the user satisfaction of the scheduler and is based on three measures: convergence test of the BPNN, job validity test and cost evaluation. The simulations presented in this paper indicate that the proposed scheduler approach is one of the most effective strategies of structuring a subjective job scheduler.

Keywords: Backpropagation neural network; Greedy task alignment procedure; Seen data; Unseen data; Subjective criteria; Satisfying criterion; Convergence test; Job validity test; Cost evaluation

Introduction

Job scheduling problems fall into a class of intractable numerical problems that are complex in nature and may not provide subjective satisfying solutions. For instance, in traditional job scheduling each job consists of m sub-jobs called *subtasks* or *tasks*, with one machine for each task. As shown in [1,2], if there are n jobs with each machine, then $(n!)^m$ solution patterns are possible. The subjective job scheduler with a satisfying criterion based on a backpropagation neural network (BPNN) [3-5] simplifies the solution complexities in job scheduling. In this context, the utilization of the parallel processing ability of the BPNN and the significance of the greedy algorithm [6,7] allow the formulation of a job scheduler which is suitable for generating user satisfying solutions. That is, the proposed scheduler has an ability to yield user satisfying solutions, especially for a problem with n independent jobs on m identical machines.

Details of the problem

In this research, a combination of a 3-layer BPNN and a greedy job alignment procedure are used to generate a “best” job scheduling result that satisfies a user for the job scheduling problem with n independent jobs on m identical machines.

It is assumed that all the jobs in a selected problem have different priorities as per the user and select the "best" possible jobs from the job queue based on the predefined job selection criteria, called *subjective criteria*. That is, prior to a scheduling process, the scheduler detects the user feasible jobs from the job queue which are supposed to be the in the valid form of the subjective criteria of the scheduler called *valid jobs* (if any). After a set of valid jobs is recognized, the scheduler allocates those jobs into the given identical machines in a concurrent manner without missing the essentiality of the top priority jobs. In this way, the scheduler maximizes the utilization of the machines to avoid any 'idle' machine states during its action. Moreover, it is assumed that all machines are operating in parallel and jobs are allowed to migrate to any available machines without violating their priority order. The greedy task alignment procedure always determines the reasonable finishing time schedule from a given scheduling problem.

The initial dataset which is generated based on the user's subjective criteria for the initial training of the BPNN called the *seen data*. These seen data and the job alignment procedure are meant to carry the details of how the job selection process happens and how the jobs are to be aligned on the machines. In this case, the user is replaced by the scheduler permanently. That is why this scheduler is named as a *subjective job scheduler*. Furthermore, this job scheduler employs the greedy algorithms which are, by their characteristics, quicker and they do not need to consider the details of all solution alternatives of the job scheduling problem.

Problem statement

In real life situations, people seek things that give them or provide optimum sense of satisfaction; therefore, all the endeavors of man are geared to finding it at all costs. On the other hand, the productions and service industry are working hard to develop goods and services that meet optimal satisfaction. In the current applications of technology, there is a need of having job schedulers that do not only schedule jobs but also provide the much needed satisfaction. Work procedures such as order of priority, time, due date and others have occupied a substantial search space unnecessarily making it almost impossible to determine whether the results are satisfactory or not [8]. Whenever a user is faced with many jobs at a time, as a human being, the user will select a set of feasible jobs subjectively and will be persuaded to complete them together without incurring any overhead. The proposed scheduler shows how a job scheduling agent would handle the above mentioned situation in an effective way.

Description of the problem and the structure of the scheduler

The proposed job scheduler is meant to solve job scheduling problems such as n independent jobs on m machines. The scheduling problem can be described as follows: denote $J = \{1, .. , j\}$ and $M = \{1,..., m\}$ as the job set and the machine set, where J and M are the number of independent jobs and identical machines, respectively. The scheduler always starts with a set of input jobs (input jobs are *unseen jobs*) which are available in the queue, also called a *job queue*. Each job in the job queue is represented by a set of parameters referred to as *job attributes*. Let's say a job, J_1 can be represented as $\{a_{11} \wedge a_{12} \wedge ....\wedge a_{1n}\}$, where a_{11}, a_{12},.., etc., are the conjunctions of the attributes of the job J_1. In this paper, a job has four attributes provided in order to estimate its *priority*, *P*. It is assumed that these four parameters of a job are known in advance and are defined as follows:

- *R,* job *release-event*, is the estimated triggering time of the job execution request [9].
- *K,* job *computation time*, is the time to complete the execution of the task.
- *D,* job relative *deadline*, is the maximum acceptable delay for its processing [9].
- *L,* job *critical type*, indicates whether the job is critically needed.

The subjective criteria for determining the priority of a job and also the subjective criteria used to identify valid jobs out of a set of input jobs depend on the nature and the definition of these four attributes. The job attributes of the scheduler may vary with the nature of the scheduling problems and users' preferences. Here the scheduler detects the *execution concurrency* of a set of valid jobs based on the job priorities and the priorities of the jobs depend on the subjective criteria of the scheduler as specified by the user. At this point, the priority definition of a job is informal. That means that the priority of a job cannot be described formally and it can only be detected through the subjective criteria of the scheduler. Subjective criteria for generating seen data for the scheduler for assigning priority to each valid job are described in Section Subjective criteria. Though the *deadline* attribute of a job is an important one for detecting its priority, it is considered that jobs are soft in nature such that a deadline is never missed to jeopardize the performance of the scheduler. Figure 1 shows the structure of the scheduler with a BPNN (3-layered BPNN with a network topology of four input neuron, thirty hidden neurons and one output neuron), a *job queue*, a *priority queue*, BPNN convergence test, job validation test, cost evaluation, greedy job alignment procedure and machine set. The scheduler is formulated in such a way that it works with a set of jobs simultaneously at a time.

The selected 3-layer BPNN is trained with a backpropagation algorithm with the seen data until its Mean Squared Error (MSE) is reduced to a value less than 0.001. Section Subjective criteria shows the details of generating the seen data for the BPNN.

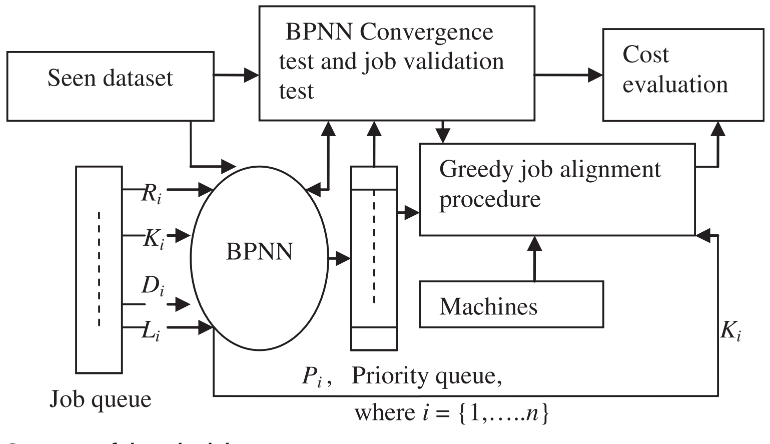
BPNN Convergence test and job validation test
Seen dataset
Cost evaluation
Greedy job alignment procedure
R_i
K_i
D_i
L_i
BPNN
Machines
K_i
Job queue
P_i, Priority queue, where $i = \{1, \ldots.. n\}$

Figure 1 Structure of the scheduler.

Description of backpropagation neural networks

The neurons are connected by links and each link has a numerical weight associated with it. Weights are the basic means of long-term memory in neural networks [4]. They express the strength, or in other words, importance of each neuron input. A neural network 'learns' through repeated adjustments of these weights. The neurons are connected to the external environment through the input and output layers. The weights are modified to bring the network input/output behavior in line with that of the environment. Figure 2 shows the structure of a typical neuron.

Each neuron in the neural network is an elementary information-processing unit. It has the means of computing its *activation level* given the inputs and numerical weights. To build an artificial neural network, first it is to be decided how many neurons are to be used and how the neurons are to be connected to form a network. Then the learning algorithm to be used is selected. Finally, the neural network is trained with the selected supervised training algorithm. Training of the neural network means that the learning algorithm initializes the weights of the network and updates the weights from a set of training examples (seen data). In this paper, the backpropagation algorithm is used to train the 3-layer neural network (hence it is called a *backpropagation neural network*). Typically, a backpropagation neural network (BPNN) is a multilayer neural network that has three or four layers. The layers are fully connected, that is, every neuron in each layer is connected to every other neuron in the adjacent forward layer. A neuron determines its output by computing its net weight input as [4]:

$$X = \sum_{i=1}^{n} x_i w_i - \theta \tag{1}$$

where the variable, X, is the net weighted input to the neuron, x_i is the value of input i, w_i is the weight of input i, n is the number of neuron inputs, and θ is the threshold applied to the neuron. The *sigmoid activation* function guarantees that the neuron output is bounded between 0 and 1. The neuron in this

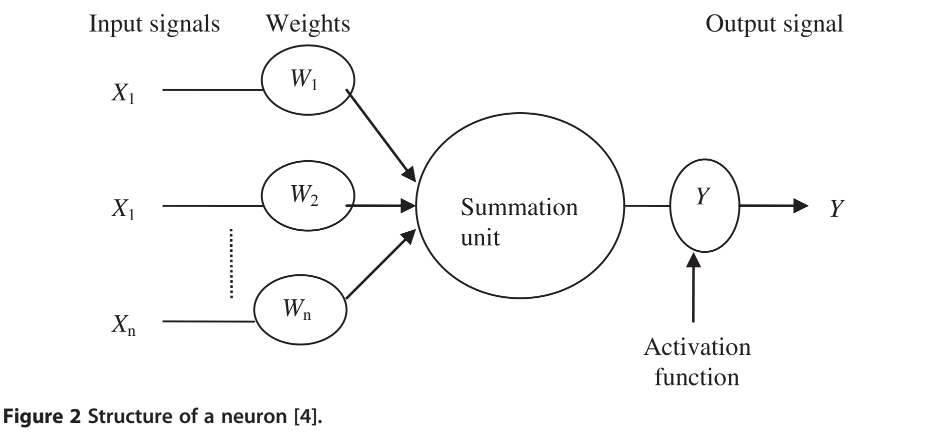

Figure 2 Structure of a neuron [4].

backpropagation network uses a sigmoid activation function. Hence the output $Y^{sigmoid}$ of the neuron is given as [4]:

$$Y^{sigmoid} = 1/\left(1 + e^{-X}\right). \tag{2}$$

The BPNN with four input variables and one output variable used in this paper is shown in Figure 3, where R, K, D, L and P are the *release-event*, *computation time*, *relative deadline*, *critical type* and *priority* of the given job. In a typical 3-layer BPNN, the computation time will be asymptotically Θ ($ih + ho$), where i, h, and o are the number of input neurons, hidden neurons and output neurons, respectively. Again, the main function of the BPNN is to assign priorities to the jobs based on the given subjective criteria.

Subjective criteria

In order to generate a seen dataset for the initial training of the BPNN of the scheduler, there are five numerical values with their proper linguistic terms applied along with the parameters of each job. The four parameters of a job with their numerical values and linguistic terms are as follows:

I. R_i is the *release-event* of job i with values: [0.1 (*very small*), 0.3 (*small*), 0.5 (*not small*), 0.7 (*long*), 0.9 (*very long*)].
II. K_i is the *computation time* of job i with values: [0.1 (*very low*), 0.3 (*low*), 0.5 (*not low*), 0.7 (*high*), 0.9 (*very high*)].
III. D_i is the *deadline* of job i with values: [0.1 (*very near*), 0.3 (*near*), 0.5 (*not near*), 0.7 (*far*), 0.9 (*very far*)].
IV. L_i is the *critical type* of job i with values: [0.1 (*very low*), 0.3 (*low*), 0.5 (*not low*), 0.7 (*high*), 0.9 (*very high*)].

The output of the BPNN, P_i, is the *priority* of job i ranging from 0.01 (*very low* priority) to 0.99 (*very high* priority).Based on the numerical values and linguistic terms of the

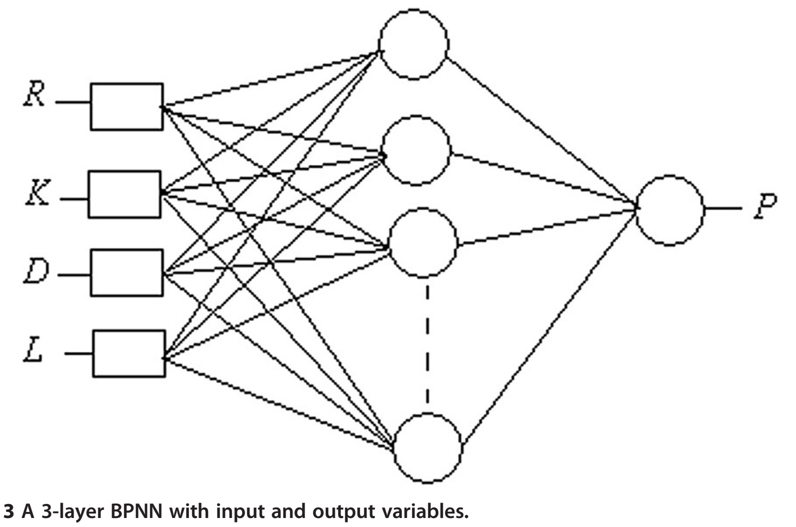

Figure 3 A 3-layer BPNN with input and output variables.

attributes of a job, the criteria for generating the seen data (including both input and output data patterns) for finding job priority are as listed below:

(a) A valid job with a *very high / high* computation time never holds a *very near* / *near* deadline (deadline must be greater than or equal to computation time).
(b) A valid job with a *very long* / *long* release-event never holds a *very near* / *near* deadline.
(c) A *very high* critical job should hold a *very near* deadline.
(d) A *very high* critical job never holds a *very far* deadline.
(e) A *very high* / *high* critical job never holds a *very high* / *high* computation time.
(f) A valid job with a *very small* / *small* release-event, a *very low* / *low* computation time, a *very near*/ *near* deadline and a *very high* / *high* critical type holds a *very high* / *high* priority.
(g) A valid job with a *not small* / *long* / *very long* release event, a *not near* / *far* / *very far* deadline, a *not low* / *high* / *very high* computation time and a *very low* / *low* / *not low* critical type can achieve only a priority value which is proportional to its critical type value.
(h) A valid job with a *very long* / *long* release-event, a *very high* / *high* computation time, a *very far* / *far* deadline and a *very low* / *low* critical type holds a *very low* / *low* priority.
(i) A valid job with a *not small* / *long* release-event, a *not near* / *far* deadline and a *not low* critical type will get a priority value which is proportional to its critical type.
(j) A valid job with a *very small* / *small* release-event, a *very high* / *high* computation time, a *not near* deadline and a *not low* critical type will hold a priority value which is proportional to its deadline.
(k) A valid job with a *very long* / *long* release-event, a *very high* / *high* computation time, a *very far* / *far* deadline and a *very low* / *low* critical type has a priority which is proportional to its critical type.

A sample seen dataset with input variables and their respective output data patterns based on the above subjective criteria is shown in Additional file 1: Table S1.

Greedy job alignment procedure

The greedy job alignment procedure helps to generate possible alignment patterns of valid jobs on their respective machines based on their priority values. The jobs are sorted in descending order of their priorities to allocate them on the given identical machines. The greedy procedure of the scheduler reduces the finishing time of the jobs on the machines by allowing similar priority jobs to execute concurrently on the given identical machines [10]. Assume that the job *priority queue*, q has n indices and can be represented as $q[0],\ldots, q[n\text{-}1]$ and it holds J valid jobs $\{J_1,\ldots, J_n\}$at a time. Let there are M identical machines in the scheduler and are represented as the set $\{M_1,\ldots, M_m\}$.The relationship between machines and jobs of the scheduler is $J > M$. Based on their priorities, the valid jobs are concurrently executed by the given machines as shown below:

$$\begin{gathered}\{(M_1 : \mathrm{q}[0], \mathrm{q}[n-2], \ldots)\}\\ \{(M_2 : \mathrm{q}[1], \mathrm{q}[n-3], \ldots)\}\\ \{(M_m : \mathrm{q}[n-1], \mathrm{q}[n-1]),\ ..\}.\end{gathered}$$

The greedy procedure sorts the computation times of each machine to find the finishing time, *FT,* of the schedule. The maximum value of the finishing time is considered as the feasible solution for the schedule (Section Cost evaluation shows the details of the finishing time calculation).

Convergence test of the BPNN

There is a convergence test added along with the traditional backpropagation algorithm to verify the initial training process of the BPNN. The initial training of the BPNN depends on the size of the seen data and the topology of the network. Once the BPNN is trained with the seen dataset until its MSE is 0.001, it is essential to ensure that the BPNN is free from problems such as 'over-fitting' and local maxima during its initial training process. The details of the convergence test of the BPNN are given here:

(i) Train the BPNN with the seen dataset by proper training parameters such as *learning rate* (α) and *momentum term* (β) until its MSE is reduced to a value less than 0.001.
(ii) Select the input data pattern from the seen dataset after its training.
(iii) Select the output data from the seen data after its training (say, *Q*) similarly to step (ii),.
(iv) Input the selected data pattern (from step ii) to the BPNN and find its output by the BPNN (say, *Q'*).

A *similarity measure* of *Q* and *Q'*, *S* (*Q*, *Q'*) is the convergence test of the BPNN and can be interpreted as follows: if *S* (*Q*, *Q'*) is above or equal to +0.99, then the selected BPNN is an acceptable one and is considered as *true*. Otherwise, the BPNN is considered as unacceptable (*false*) and allow the BPNN to repeat its training with different parameters and topologies until an acceptable net topology is seen.

A correlation coefficient statistics [11] was used to measure the similarity between two datasets of equal size and the results showed values between −1 and +1 on the basis of the datasets. The mathematical formulae of the correlation coefficients are included below. Let $S_{i,j}$ be the normalized similarity between two sets of attribute values X_i and X_j of datasets *i* and *j*. The analytical expression of $S_{i,j}$ is;

$$S_{i,j} = \left(\sum_{k=1}^{n}\left(X_{i,k} = \overline{X}i\right) \times \left(X_{j,k} - \overline{X_j}\right)\right) \Big/ \sum_{k=1}^{n}\left(X_{i,k} - \overline{X_i}\right)^2 \times \sum_{k=1}^{n}\left(X_{j,k} - \overline{X_j}\right)^2]^{1/2}, \tag{3}$$

where

$$\overline{X}_i = 1 \Big/ n \times \left(\sum_{k=1}^{n} X_{i,k}\right) \tag{4}$$

and

$$\overline{X}j = 1 \Big/ n \times \left(\sum_{k=1}^{n} X_{j,k}\right). \tag{5}$$

Like with the BPNN convergence test procedure, there is a job validity test procedure of the scheduler as described in the following section.

Job validity test

The job validity test, *V*, of the scheduler provides a degree of measure of the *unseen data* (input jobs) of the scheduler with respect to its seen dataset. The job validity test depends on the similarity measure of the unseen data (input jobs) and seen dataset of the scheduler for a given problem. For instance, consider each job in a problem that has a set of four attributes. Hence a set of n jobs has a size of $n \times 4$ (i.e., n rows and 4 columns) for the unseen data values. The similarity measure of an input job (unseen) is calculated with each seen data value (except its seen priority value) of the scheduler. The resulted similarities (called *correlation values*) are stored in a buffer and then the maximum value is selected as the job's correlation value. The job is valid only when its correlation value is greater than zero. Similarly, the scheduler finds the correlation values of all input jobs in the job queue. The scheduler selects only valid jobs for their priority estimation.

Let the seen data be given index i and the unseen data be given index j and X_i, X_{i+1},..., X_{i+n}, are the n parameters of set i and X_j, X_{j+1},..., X_{j+n}, are the n parameters of set j (assuming that the sizes of sets i and j are the same). Then the validity of sets i and j, $V_{i,j}$, based on Equation (3) can be given as:

$$V_{i,j} = \left(\sum_{k=1}^{n}\left(\overline{X_{i,k}} - \overline{Y_i}\right) \times \left(\overline{X_{j,k}} - \overline{Y_j}\right)\right) \Big/ \left[\sum_{k=1}^{n}\left(\overline{X_{i,k}} - \overline{Y_i}\right)^2 \times \sum_{k=1}^{n}\left(\overline{X_{j,k}} - \overline{Y_j}\right)^2\right]^{1/2}, \tag{6}$$

where

$$\overline{X_{i,k}} = 1 \Big/ n \times \left(\sum_{i=1}^{n} X_{i,k}\right), \tag{7}$$

$$\overline{X_{j,k}} = 1 \Big/ n \times \left(\sum_{j=1}^{n} X_{j,k}\right), \tag{8}$$

$$\overline{Y_i} = 1 \Big/ n \times \left(\sum_{k=1}^{n} \overline{X_{i,k}}\right) \tag{9}$$

and

$$\overline{Y_j} = 1 \Big/ n \times \left(\sum_{k=1}^{n} \overline{X_{j,k}}\right). \tag{10}$$

Based on Equation (6), the correlation value of unseen job i, can be given as:

$$V_i = \max\left(V_{i,1}, V_{i,2}, V_{i,3}, \ldots V_{i,j}\right), \tag{11}$$

where $V_{i,1}$, $V_{i,2}$, $V_{i,3}$,..., $V_{i,j}$ are the correlation values of job i with *seen_ job_ data*$_1$, *seen_ job_data*$_2$,.... , etc. *If* $V_i > 0$, *then it is assumed that the unseen job i, is based on the scheduler's subjective criteria and is valid.* The job validity test V, is *true* only when the size of valid jobs is greater than the size of machines for a given problem.

Cost evaluation

The cost evaluation of the scheduler depends on its *cost value*, *C*, and is based on its finishing time. The cost evaluation of a multi-machine job scheduler can be expressed by the following theorem [7]:

Every feasible schedule has a finishing time which is not earlier than the time

$$T = \left(\sum_{i=1}^{n} K_i\right)/m, \tag{12}$$

where K_i $\{i = 1,\ldots,\ n\}$ is the computation time of i jobs and m is the number of machines. It is assumed that all machines are operating in parallel and each machine has an initialization time, w_{ti} $\{i = 1,\ldots,\ n\}$, in order to prepare for a job processing. Let T_{m1} is the total computation time taken by machine m_1 and is given as:

$$T_{m1} = w_{t1*}\sum_{i=1}^{n} K_{1i}, \tag{13}$$

where K_{1i} $\{i = 1,\ldots,\ n\}$ is the computation time of i jobs on machine m_1. Similarly, T_{mn} is the total computation by n^{th} machine, m_n of the scheduler and is given as:

$$T_{mn} = w_{t,mn*}\sum_{i=1}^{n} K_{ni} \tag{14}$$

For simplicity, it is assumed that w_{ti} is 1. Hence the finishing time, *FT*, of a complete schedule with n machines can be estimated as

$$FT = \max\{T_{m1}, T_{m2}, \ldots., T_{mn}\}. \tag{15}$$

During a scheduling result with a finishing time, *FT*, the m machines of the scheduler can use a total of at most ($m \times FT$) time units [7]. At this time, for executing all valid jobs requires $\sum_{i=1}^{n} K_i$ time units. Hence the cost value, *C*, of the scheduler can be derived from Equations (12) and (15) as follows:

$$C = [FT - T], \tag{16}$$

where T is the average execution time of n jobs on m machines (see Equation (12)).

Based on Equation (16), the cost term can be interpreted, as follows:

- If C is equal to 0, then it is a '*good enough*' schedule and C is *true*.
- If $C > 0$, then it is a '*reasonable*' schedule and C is *true*.

For both cases, it can be noticed that C is *true*. The reason is that the scheduler never generates a *non-reasonable* schedule due to the application of the greedy alignment procedure. Furthermore, due to the application of Equation (16), the cost, C, of a feasible schedule will never reach a value greater than or equal to 1.

Definition of satisfying criterion

The satisfying criterion, *Sat*, of the scheduler is a binary term which indicates the satisfaction of the scheduler for a given problem. The *Sat* of the scheduler can be defined in terms of three binary measures: (i) *S*; (ii) *V*; and (iii) *C*, where *S*, *V* and *C* are the binary results of convergence test, job validity test and cost evaluation of the scheduler, respectively. The propositional logic representation of *Sat* with respect to the atomic variables *S*, *V* and *C* can be expressed as [8]:

$$((S \wedge V \wedge C) \rightarrow Sat). \tag{17}$$

The interpretation of Equation (17) is that if *S*, *V* and *C* are *true*, then it is possible to say that *Sat* is *true*. Otherwise, it is not possible to claim that *Sat* is *true*. Because of the application of BPNN convergence test, job validity test and cost evaluation, the proposed scheduler always generates user-satisfactory schedules. Whenever all the jobs in the job queue are invalid, then the scheduler will indicate an unsatisfying situation. The same situation will happen when the number of valid jobs is less than the number of given machines for a problem.

Procedure of the scheduler

The implementation of the proposed procedure includes the following distinct steps:

(i) Generate jobs and machines randomly.
(ii) *Declarations:* Let M be the set of m identical machines, where $\forall\, M = 0$ (initialize all machines). The selection criteria of both job and machine can be given as $J > M$, where J is the total number of jobs in the job queue.
(iii) *Job validation test:* Checks whether any invalid job(s) available in the job queue. If so, such jobs will be exempted from the job queue.
(iv) *Backpropagation algorithm:* The backpropagation algorithm trains the BPNN for assigning priorities to each valid job. Let P is a set of priorities of n jobs and P can be denoted as $\{P_1, P_2,\ldots,P_n\}$. The convergence test measures the acceptability of the BPNN before its selection.

Table 1 Unseen dataset of twelve jobs with their respective correlation values

Job	*R*	*K*	*D*	*L*	Correlation
J_1	0.32	0.65	0.10	0.54	−0.34015
J_2	0.76	0.54	0.65	0.76	−0.31479
J_3	0.98	0.76	0.54	0.54	−0.69395
J_4	0.32	0.21	0.87	0.32	0.50865
J_5	0.32	0.87	0.98	0.10	0.128618
J_6	0.65	0.54	0.65	0.32	−0.37259
J_7	0.32	0.98	0.98	0.54	0.28644
J_8	0.65	0.21	0.21	0.21	−0.83024
J_9	0.10	0.32	0.54	0.65	0.92431
J_{10}	0.21	0.54	0.98	0.10	0.33508
J_{11}	0.21	0.98	0.76	0.87	0.43142
J_{12}	0.76	0.32	0.76	0.54	−0.20152

Table 2 Order of valid jobs with their priority values by BPNN in twelve jobs on three machines problem

Valid job	*P*
J_9	0.488
J_{11}	0.265
J_4	0.237
J_7	0.119
J_{10}	0.029
J_5	0.007

(v) *Greedy job alignment procedure:* The job alignment procedure aligns valid jobs into a set of identical machines for their concurrent execution based on their priorities. The alignment procedure returns a best finishing time, *FT*, of the schedule which is either 'good enough' or a reasonable one.
(vi) *Cost evaluation:* The cost evaluation of the scheduler evaluates result of a schedule to either a *good enough* or a *reasonable* schedule on the basis of its finishing time, *FT*.
(vii) *Satisfying criterion:* Satisfying criterion defines the satisfying conditions of the scheduler. Whenever the number of valid jobs is zero and the number of valid jobs is less than the machines, then satisfying criterion of the scheduler fails.
(viii) Go to step (i).

Simulation results

The subjective scheduler is written in C++ and supportive simulations are made to show the satisfying nature of the scheduler for several given scheduling problems of different kinds. For the purposes of this study, three such kinds are shown: first, a problem with twelve jobs on three machines and second, a problem with sixteen jobs on four machines. Third, a problem with twenty one jobs on seven machines. In this section, the situations, such as the scheduler with only invalid jobs or when the size of the valid jobs is less than the size of the machines, are omitted. Details of the simulations carried out are given below.

Scheduling problem with twelve jobs on three machines

The unseen dataset of twelve jobs with their respective correlation values (as per Equation (6)) are shown in Table 1. As per Equation (3), the 3-layer BPNN is acceptable with a similarity value of +0.9978 (i.e., *S* is *true*). Table 1 shows that there are six valid jobs (J_4, J_5, J_7, J_9, J_{10} and J_{11}) in the job queue as per Equation (11) and the invalid jobs are

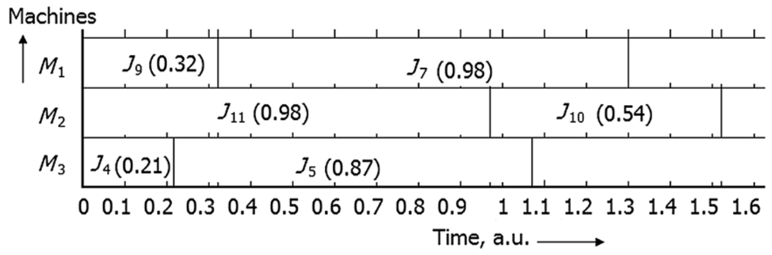

Figure 4 Scheduling of six valid jobs on three machines (*FT* = 1.52).

Table 3 Unseen dataset of sixteen jobs with their respective correlation values

Job	*R*	*K*	*D*	*L*	Correlation
J_1	0.32	0.43	0.65	0.54	0.93713
J_2	0.76	0.32	0.87	0.21	−0.11176
J_3	0.76	0.32	0.76	0.32	−0.21439
J_4	0.21	0.98	0.10	0.43	−0.01612
J_5	0.87	0.32	0.98	0.32	−0.11865
J_6	0.21	0.76	0.32	0.54	0.19192
J_7	0.98	0.21	0.32	0.76	−0.71251
J_8	0.98	0.87	0.21	0.10	−0.68731
J_9	0.65	0.21	0.21	0.43	−0.79364
J_{10}	0.98	0.21	0.87	0.10	−0.31559
J_{11}	0.21	0.32	0.76	0.10	0.50818
J_{12}	0.32	0.10	0.98	0.87	0.46068
J_{13}	0.65	0.65	0.10	0.98	−0.39876
J_{14}	0.21	0.98	0.87	0.76	0.63493
J_{15}	0.65	0.32	0.65	0.98	0.00138
J_{16}	0.43	0.10	0.54	0.87	0.17097

discarded. Validity, *V*, of the selected jobs is *true* because the size of the valid jobs is greater than the size of the identical machines (M_1, M_2 and M_3) for the problem. Priority order of the six valid jobs with their priority value, *P*, by the BPNN is shown in Table 2.

Figure 4 shows the scheduling of six valid jobs on three identical machines (M_1, M_2 and M_3). The computation time, *K*, of each job is shown in brackets. The greedy job alignment procedure allocates valid jobs on the given identical machines based on their order of the priority. The finishing time, *FT*, is 1.52 and cost value, *C*, is 0.22 and is *true* as per Equation (16) and the resulted schedule is a *reasonable* one. Because of the *true* values of *S*, *V* and *C* (as per Equation (17)), the schedule from scheduler for the given problem is a satisfying one.

Scheduling problem with sixteen jobs on four machines

A simulation of the scheduler with an unseen dataset of sixteen jobs on four identical machines (M_1, M_2, M_3 and M_4) is illustrated.

Table 3 shows an unseen dataset of sixteen jobs with their respective correlation values. From the Table 3, it can be noticed that there are seven valid jobs (J_1, J_6, J_{11}, J_{12},

Table 4 Order of valid jobs with their priority values by BPNN in sixteen jobs on three machines problem

Valid job	*P*
J_1	0.491
J_{16}	0.469
J_{15}	0.452
J_{14}	0.195
J_{12}	0.057
J_{11}	0.035
J_6	0.007

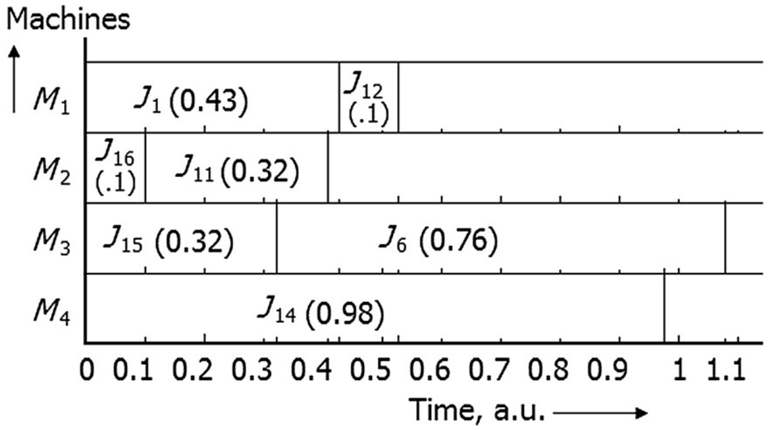

Figure 5 Scheduling result of seven valid jobs on four machines (*FT* = 1.08).

J_{14}, J_{15} and J_{16}) in the job queue. Validity, *V*, of the selected jobs is *true* because the size of the valid jobs is greater than the size of the identical machines (M_1, M_2, M_3 and M_4) for the problem. Priority order of the seven valid jobs with their priority values, *P*, by the BPNN is shown in Table 4.

Figure 5 shows the scheduling result of seven valid jobs on four identical machines (M_1, M_2, M_3 and M_4). The computation time, *K*, of each job is shown in brackets. The finishing time, *FT*, is 1.08 and cost value, *C*, is 0.33 and is *true* as per Equation (16). As

Table 5 Unseen dataset of twenty one jobs with their respective correlation values

Job	*R*	*K*	*D*	*L*	Correlation
J_1	0.76	0.10	0.54	0.10	−0.30535
J_2	0.76	0.65	0.87	0.65	0.01423
J_3	0.32	0.43	0.65	0.21	0.45152
J_4	0.43	0.54	0.43	0.65	0.19109
J_5	0.65	0.43	0.43	0.21	−0.48479
J_6	0.87	0.87	0.87	0.43	−0.16129
J_7	0.76	0.10	0.32	0.65	−0.47652
J_8	0.10	0.87	0.21	0.65	0.16139
J_9	0.21	0.65	0.21	0.87	0.12139
J_{10}	0.65	0.76	0.32	0.76	−0.35715
J_{11}	0.87	0.98	0.32	0.10	−0.54791
J_{12}	0.10	0.32	0.10	0.87	0.19261
J_{13}	0.65	0.43	0.54	0.87	−0.08543
J_{14}	0.43	0.76	0.87	0.87	0.55939
J_{15}	0.87	0.54	0.32	0.65	−0.64674
J_{16}	0.32	0.21	0.65	0.98	0.44291
J_{17}	0.98	0.32	0.76	0.1	−0.29859
J_{18}	0.32	0.43	0.87	0.21	0.48435
J_{19}	0.65	0.54	0.65	0.98	0.15131
J_{20}	0.43	0.32	0.32	0.65	−0.01266
J_{21}	0.76	0.32	0.98	0.65	0.10433

Table 6 Order of valid jobs with their priority values by BPNN in twenty one jobs on seven machines problem

Valid job	*P*
J_{19}	0.801
J_4	0.544
J_{12}	0.510
J_{16}	0.357
J_{14}	0.287
J_2	0.213
J_9	0.195
J_3	0.184
J_{21}	0.167
J_{18}	0.078
J_8	0.003

per the cost value, the resulted schedule is a *reasonable* one. As per Equation (17), the result from the scheduler for the given problem is a satisfying one.

Scheduling problem with twenty one jobs on seven machines

Similarly a simulation of the scheduler with an unseen dataset of twenty one jobs on seven identical machines (M_1, M_2, M_3, M_4, M_5, M_6 and M_7) is illustrated in this section. Table 5 shows an unseen dataset of twenty one jobs with their respective correlation values.

Table 5 shows that there are eleven valid jobs (J_2, J_3, J_4, J_8, J_9, J_{12}, J_{14}, J_{16}, J_{18}, J_{19} and J_{21}) in the job queue. The validity, *V*, of the selected jobs is *true*. Priority order of the eleven valid jobs with their priority values, *P*, by the BPNN is shown in Table 6.

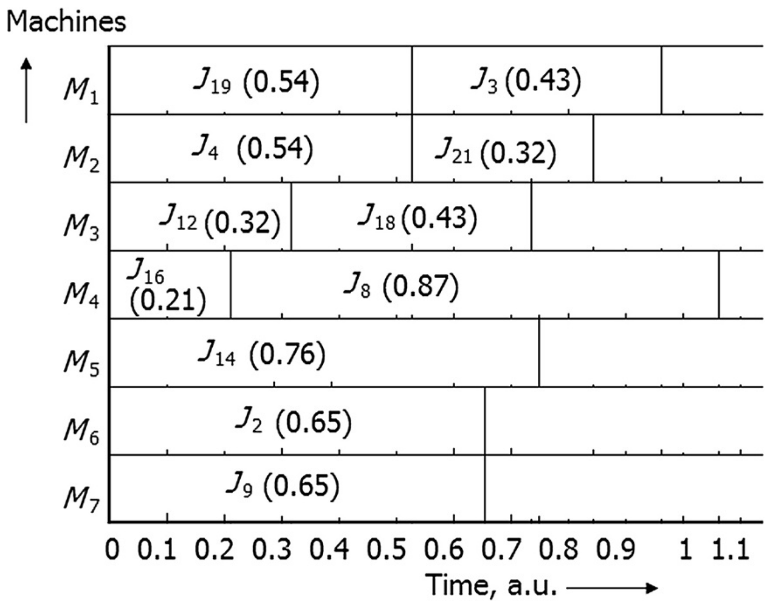

Figure 6 **Scheduling result of eleven valid jobs on seven machines (*FT* = 1.08).**

Figure 6 shows the scheduling of eleven valid jobs on seven identical machines (M_1, M_2, M_3, M_4, M_5, M_6 and M_7). The computation time, K, of each job is shown in brackets. The finishing time, FT, is 1.08 and cost value, C, is 0.263 and is *true* as per Equation (16). As per the cost value, the resulted schedule is a *reasonable* one and as per Equation (17), the result from the scheduler is a satisfying one.

Conclusions

The presented subjective job scheduler shows its ability in generating user satisfying schedules by establishing proper neural net training paradigm, exempting invalid jobs from the job queue and evaluating its results with a cost evaluation. The scheduler utilizes the customizable nature of the BPNN and the feature of the greedy algorithm.

The term 'job priority' of the scheduler cannot be described formally, that is, it is not possible to define the priority of a job in a normal way because that depends only on the given subjective influence. Therefore, the results of the scheduler are biased towards certain objective based on its subjective criteria.

The proposed scheduler is flexible enough to adopt views of various users for a given problem and it functions like an intelligent job scheduling agent for providing user satisfied results.

Additional file

Additional file 1: Table S1. sample seen dataset with four inputs and their respective output data patterns.

Competing interests
The author declares that he has no competing interests.

Received: 20 February 2012 Accepted: 17 September 2013
Published: 22 September 2013

References

1. Cook SA (1971) The complexity of theorem proving procedures. Proceedings of the Third Annual ACM Symposium on the Theory of Computing 151:158
2. Garey MR, Johnson DS (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman and Co, New York
3. Rao VB, Rao HV (1996) Neural Networks & Fuzzy logic. BPB Publications, Delhi
4. Negnevitsky M (2005) Artificial Intelligence- A Guide to Intelligent Systems. Addison Wesley, Europe
5. Russell S, Norvig P (2004) Artificial Intelligence A Modern Approach. Pearson Education, New Jersey
6. Cormen TH, Leiserson CE, Rivest RL, Stein C (2001) Introduction to Algorithms. McGraw-Hill, Cambridge, Massachusetts London
7. Stinson S (1980) An Introduction to the Design and Analysis of Algorithms. Cambridge University Press, Cambridge
8. Anilkumar KG (2012) The Subjective Job Scheduler with a Satisfying Criterion Based a Backpropagation Neural Network. Network World, 2/2012, Technology Academy of Sciences of the Czech Republic (ASCR); Faculty of Transport, Czech Polytechnic University, Prague, 195–213
9. Cottet F, Delacroix J, Kaiser C, Mammeri Z (2002) Scheduling in Real-Time Systems. John Wiley & Sons Ltd, England
10. Anilkumar KG, Tanprasert T (2007) Generalized Job-shop Scheduler Using Feed Forward Neural Network and Greedy Alignment Procedure. In: Proc. IASTED Conference on Artificial Intelligence and Applications, IASTED (International Association of Science and Technology for Development). ACTA Press, Austria, pp 115–120
11. Johnson RA, Wichern DW (2002) Applied Multivariate Statistical Analysis. Prentice Hall, NJ

doi:10.1186/2192-1962-3-17
Cite this article as: Gopalakrishnan: A subjective job scheduler based on a backpropagation neural network. *Human-centric Computing and Information Sciences* 2013 **3**:17.