THE SUBJECTIVE JOB SCHEDULER WITH A SATISFYING CRITERION BASED ON BACKPROPAGATION NEURAL NETWORK

Anilkumar Kothalil Gopalakrishnan[*]

Abstract: This paper aims to present and discuss the concept of a subjective job scheduler with a satisfying criterion based on a Backpropagation Neural Network (BPNN) and a greedy task alignment procedure. The BPNN is to assign priorities to the tasks of each job based on the given subjective criteria. The subjective criteria and the task alignment procedure depend on the solution plan towards a given job scheduling problem depending on the user's need. When the scheduler is provided with a desired job selection criteria and task alignment procedure for the problem, it generates user satisfying solutions for a set of jobs. The satisfying criterion of the scheduler determines the user satisfaction based on three measures: convergence test of the BPNN, validity of the input job set and cost evaluation of the solutions. The simulations and comparisons presented in this paper indicate that the proposed approach is one of the most effective strategies of structuring a subjective functional job scheduler.

Key words: *Greedy task alignment procedure, subjective criteria, satisfying criterion, convergence test, job validity test, cost evaluation.*

Received: January 31, 2011
Revised and accepted: April 30, 2012

1. Introduction

Job scheduling problems fall into a class of intractable numerical problems that are complex in nature and may not provide subjective satisfactory 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] and [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 backpropagation neural network [3]

[*]Anilkumar Kothalil Gopalakrishnan
DPL, Department of Computer Science, Assumption University, Ramkhamheang Soi-24, Huamak, Bangkok, 10240, Thailand, E-mail: anil@scitech.au.edu

simplifies the solution complexities in job scheduling. In this context, the utilization of the parallel processing ability of the BPNN and the significance of greedy algorithm can be able to formulate a job scheduler and is suitable for generating user satisfying solutions [4, 5]. That is, the combined algorithmic structure of the scheduler has an ability to yield less complex satisfactory solutions especially for a problem with n independent jobs and each job with t dependent tasks on m unit machines.

2. Background to the Problem

Since the neural networks alone cannot be adequately used to align task sequences on given machines in a job scheduling problem, the application of a proper job or task alignment procedure along with the neural network is essential. Due to this reason, a combination of a 3-layer BPNN and a greedy task alignment procedure are used to generate results that satisfy the user for a job scheduling problem with n independent jobs and t dependent tasks on m unit machines.

It is assumed that all the jobs in a selected problem have equal priorities and the precedence order of tasks of each job is assigned by the scheduler depending on its task selection criteria, called subjective criteria [6] based on the user's requirements. That is, the scheduler identifies the precedence order of each task of a job by finding its priority, which is based on the applied subjective criteria for the problem. In order to schedule the given independent jobs with their tasks on the machines, tasks of each job in a job set (the scheduler is made in such a way that it always works with a set of jobs) are allocated to the machines soon after they finish processing their current task to avoid 'idle' machine states. Moreover, it is assumed that all machines are operating in parallel and tasks of each job are allowed to migrate to any available machines without violating their precedence order. At this point, the algorithm is accurately consistent and vigilant to the integrity of the tasks precedence order. The mentioned procedure is a part of the greedy task alignment procedure which always determines the minimum finishing time schedule of the problem. The advantage of this task alignment procedure is that it maximizes the machine utilization.

The initial dataset which is generated from the user's subjective criteria for the initial training of the BPNN is called *seen data.* These seen data and the task alignment procedure are meant to carry the details of how the task selection process happens and how the tasks are to be aligned on machines. In this case, the user is replaced by the scheduler permanently. That is why this scheduler is named 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.

3. 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 and tasks 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. The subjective job scheduler with a satisfying criterion based on BPNN being presented in this paper is designed to generate user-satisfying solutions.

4. Description of the Problem and Structure of the Scheduler

The proposed scheduler is meant to solve job scheduling problems such as n independent jobs with t dependent tasks on m machines. The scheduling problem can be described as follows: denote $J=\{1, \ldots, j\}$ and $M=\{1,\ldots, m\}$ as the job set and the machine set, where j and m are the number of independent jobs and unit machines, respectively. Each job is assumed to handle n tasks and the tasks are interdependent. The scheduler always starts with a set of jobs available in a queue called *job queue*. Each task of a job is represented by a set of attributes referred to as *task parameters.* Let us say a task of job, J_1 is t_{11} and it can be represented as $\{a_{11} \wedge a_{12} \wedge \ldots \wedge a_{1n}\}$, where a_{11}, a_{12},..., etc., are the conjunctions of the attributes of task, t_{11}. In this study, a task may have four attributes provided in order to estimate its *priority*, P. Let us say t_{ix} represents a task t_x of job i, and its priority can be indicated as P_{ix}. Similarly, R_{ix}, K_{ix}, D_{ix} and L_{ix} represent the *release-event*, *computation time*, *deadline* and *critical type* of task, t_{ix}, respectively. It is assumed that these four parameters of a task are known and are defined as follows:

- R, task *release-event* is the estimated triggering time of the task execution request [7].
- K, task *computation time* is the time to complete the execution of the task.
- D, task relative *deadline* is the maximum acceptable delay for its processing [7].
- L, task *critical type* indicates whether the task is critically needed.

The subjective criteria for determining the priority of a task depend on the nature and the definition of these four task parameters,. The task parameters can vary with the nature of the scheduling problems and users' preferences. Here the scheduler detects the *precedence order* of each task of a job based on their priorities; the priorities of the tasks depend on the subjective criteria of the scheduler as specified by the user. At this point, the priority definition of a task is informal. That means the priority of a task cannot be described formally and it can only be detected through the subjective criteria of the scheduler. Task selection criteria for generating seen data for finding task priority are described in Section 5. Though the *deadline* attribute of a task is an important one for detecting a task priority, it is considered that jobs are soft in nature such that a deadline is never missed to jeopardize the performance of the scheduler. Fig. 1 shows the structure of the scheduler with a BPNN (3-layer one with a topology 4-20-1), a *job queue*, a *priority queue*, BPNN convergence test, job validation test, cost evaluation and machine set. The scheduler is formulated in such a way that it works with a set of jobs at a time rather than a single job.

The selected 3-layer BPNN is trained with a backpropagation algorithm [3] with the seen data until its Mean Squared Error (MSE) is reduced to a value less than 0.001. The BPNN with four input variables and one output variable is shown in Fig. 2. 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 the output neurons, respectively [3]. Again, the main function of the BPNN is to assign priorities to the tasks based on the given subjective criteria.

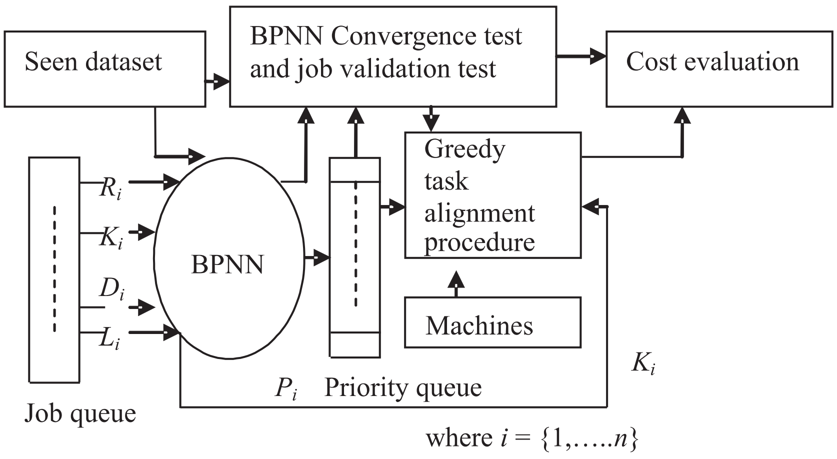
Seen dataset
BPNN Convergence test and job validation test
Cost evaluation
Greedy task alignment procedure
R_i
K_i
D_i
L_i
BPNN
Machines
K_i
P_i Priority queue
Job queue
where $i = \{1,\ldots..n\}$

Fig. 1 *Structure of the scheduler.*

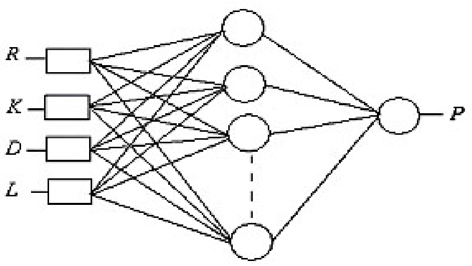

Fig. 2 *A 3-layer BPNN with input and output variables.*

In order to prove the applicability of the scheduler, the applied cost evaluation measures the feasibility of the scheduler as described in Section 9.

5. 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 task. The four parameters of a task with their numerical values and linguistic terms are as follows:

(i) R_i is the *release-event* of task 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 task 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 task 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 task 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 task i ranging from 0.01 (*very low* priority) to 0.99 (*very high* priority).

Based on the numerical values and linguistic terms of the parameters of a task, sample task selection criteria for generating the seen data (including both input and output data patterns) for finding task priority are as listed below:

(a) A task with a *very high / high* computation time never holds a *very near / near* deadline.

(b) A task with a *very long / long* release-event never holds a *very near / near* deadline.

(c) A *very high* critical type task should hold a *very near* deadline.

(d) A *very high* critical type task never holds a *very far* deadline.

(e) A *very high / high* critical type task never holds a *very high / high* computation time.

(f) A task 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 task 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 task 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 task 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 task 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 task 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 forty inputs and their respective output data patterns based on the above subjective criteria is shown in Appendix A.

6. Greedy Task Alignment Procedure

The flowchart of the greedy task alignment procedure described in this section is shown in Fig. 3. The tasks of each job are sorted in descending order of their priorities to get the tasks precedence order. A predefined *task-path* [8] helps to generate possible alignment patterns of tasks on their respective machines without violating the task precedence order. The application of the task-path reduces the solution complexity of the scheduler by allowing tasks of a job to migrate into various machines by carrying (?) their precedence order. Details of the task-path are described below:

(i) Based on the array indices of the task priority queue (where tasks are stored in their descending order of their priorities) and the size of the machines, the alignment procedure generates all legally possible scheduling patterns of tasks of each job on machines without violating their precedence order.

(ii) The alignment procedure stores these pre-generated task scheduling patterns in the scheduler's memory (in the form of array elements).

With the help of these pre-defined task patterns, the greedy alignment procedure will generate all feasible schedules for a given problem. At the end of a scheduling process, the greedy task alignment procedure returns a best feasible schedule (a schedule with minimum value of finishing time) from the various feasible schedules.

7. Convergence Test 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 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 0.01.

(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*, $S(Q,Q')$ 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, therefore, you should repeat its training with different parameters and topologies until an acceptable net topology is seen.

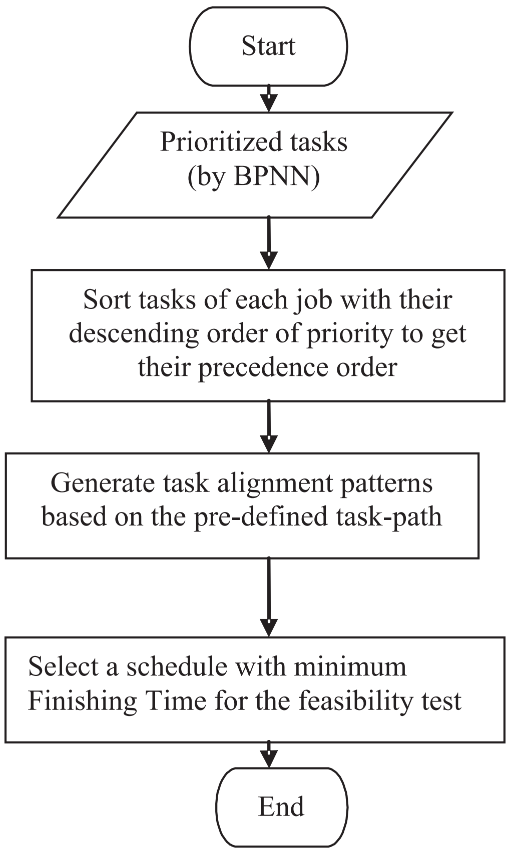
Start
Prioritized tasks (by BPNN)
Sort tasks of each job with their descending order of priority to get their precedence order
Generate task alignment patterns based on the pre-defined task-path
Select a schedule with minimum Finishing Time for the feasibility test
End

Fig. 3 *A flow chart of the task alignment procedure.*

A correlation coefficient statistics [9] 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;

$$S_{i,j} = (\sum_{k=1}^{n}(X_{i,k} - \overline{X_i}) * (X_{j,k} - \overline{X_j}))/\sum_{k=1}^{n}(X_{i,k} - \overline{X_i})^2 * \sum_{k=1}^{n}(X_{j,k} - \overline{X_j})^2]^{1/2}, \quad (1)$$

where

$$\overline{X_i} = 1/n * (\sum_{k=1}^{n} X_{i,k}) \quad (2)$$

and

$$\overline{X_j} = 1/n * (\sum_{k=1}^{n} X_{j,k}). \quad (3)$$

Like with the BPNN convergence test procedure, there is a job validity test procedure of the scheduler, as described in the following Section 8.

8. Job Validity Test

The job validity test of the scheduler provides a degree of measure of the *unseen data* of the scheduler with respect to the seen data. Here unseen data are the tasks attributes of jobs which are inputted to the scheduler in the form of numerical values, as shown in Section 5. That is, the job validity test of the scheduler depends on both seen and unseen data of the scheduler for a given problem. For instance, consider each job in a problem that has a set of four tasks and each task has four parameter values. Hence, a set of n jobs has a size of $4n$ x $4n$ (i.e., $4n$ rows and $4n$ columns) for the unseen data set. Similarly, a set of seen data with the same size as the given unseen data is considered as an ideal dataset for measuring the similarity of the given problem. Let the seen data be given index i and the unseen data be given index j and X_i, $X_{i+1}, \ldots$, X_{i+n}, are the n parameters of set i and X_j, $X_{j+1}, \ldots$, 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 Eq. (1) can be given as:

$$V_{i,j} = ((\sum_{k=1}^{n}(\overline{X_{i,k}} - \overline{Y_i}) * (\overline{X_{j,k}} - \overline{Y_j}))/[\sum_{k=1}^{n}\overline{(X_{i,k}} - \overline{Y_i})^2 * \sum_{k=1}^{n}\overline{(X_{j,k}} - \overline{Y_j})^2]^{1/2}, \quad (4)$$

where

$$\overline{X_{i,k}} = 1/n * (\sum_{i=1}^{n} X_{i,k}), \quad (5)$$

$$\overline{X_{j,k}} = 1/n * (\sum_{j=1}^{n} X_{j,k}), \quad (6)$$

$$\overline{Y_i} = 1/n * (\sum_{k=1}^{n} \overline{X_{i,k}}) \tag{7}$$

and

$$\overline{Y_j} = 1/n * (\sum_{k=1}^{n} \overline{X_{j,k}}). \tag{8}$$

Based on Eq. (4), the validity of the unseen input job set of the scheduler is given the following interpretation: *if* $(V \geq +0.5)$*, then it is assumed that the generated unseen job set is based on the scheduler's subjective criteria and is considered as valid and true.* Without lack of generality, the descriptions of the multitude of invalid job sets are not shown here due to size limitations.

9. 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 [4]:

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

$$T = (\sum_{i=1}^{n} K_i)/m, \tag{9}$$

where K_i $\{i = 1, \ldots, n\}$ is the computation time of i tasks and m is the number of machines. It is assumed that all machines are operating in parallel and tasks migration is allowed but task parallelism is forbidden. Each machine has an initialization time, wt_i $\{i = 1, \ldots, n\}$, in order to prepare for a task processing. Let T_{m1} be the total computation time taken by machine m_1 and it is given as:

$$T_{m1} = w_{t1*} \sum_{i=1}^{n} T_{m1}. \tag{10}$$

Similarly, T_{mn} is the total task computation by n^{th} machine, m_n and is given as:

$$T_{m1} = w_{t,mn} * \sum_{i=1}^{n} T_{mn}. \tag{11}$$

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:

$$FT = \max\{T_{m1}, T_{m2}, \ldots, T_{mn}\}. \tag{12}$$

The best finishing time, FT_{min}, is the minimum value of finishing times of all legally possible schedules of a problem and can be given as:

$$FT_{\min} = \min\{FT_1, FT_2, \ldots, FT_n\}, \tag{13}$$

where FT_1, FT_2, $\ldots$, FT_n are the finishing times of n feasible schedules of a problem. Therefore, the cost value, C, of the scheduler can be given as follows:

$$C = [FT_{\min} - (\sum_{i=1}^{n} K_i)/m)]. \tag{14}$$

Based on Eq. (14), the cost terms can be interpreted, as follows:

- If C is equal to 0, then it is a '*good enough*' schedule and C is *true*.
- If C is greater than 0 and less than 1, 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* or a violated schedule due to the application of the greedy alignment procedure. Furthermore, due to the application of Eq. (14), the cost, C, of a feasible schedule never reaches a value greater than or equal to 1.

10. Satisfying Criterion

The satisfying criterion, *Sat*, of the scheduler indicates a value of satisfaction of the scheduler for a given problem. The *Sat* of the scheduler depends on the 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:

$$((S \wedge V \wedge C) \rightarrow Sat). \tag{15}$$

The interpretation of Eq. (15) 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*.

11. 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 unit machines, where $\forall M$ =0 (initialize all machines). The selection criteria of both job and machine can be given as $n/k > M$ and n must be a proper divisor of k, where n is the number of total tasks in the job queue and k is the number of tasks in a job.

(iii) *Backpropagation algorithm*: The backpropagation algorithm trains the BPNN for assigning priorities to tasks of each job. Let P be a set of priorities of n tasks and P can be denoted as $\{P_1, P_2, \ldots, P_n\}$. The convergence test measures the acceptability of the BPNN.

(iv) *Tasks precedence order*: Let t be the set of n tasks and be ordered according to their priority order, $\{P_1 \quad \geq P_2 \quad \geq \ldots\ldots \geq P_n\}$, where the precedence order of the tasks can be given as $\{t_1 \rightarrow \quad t_2 \quad \rightarrow \ldots \rightarrow t_n\}$.

(v) *Job set validation test*: Checks whether any invalid task(s) in the input job set and a job set with invalid task(s) will be exempted by the scheduler. Thus, the presence of an invalid task can make the whole job set invalid.

(vi) *Greedy procedure*: This is a task alignment procedure which aligns prioritized tasks of each job to a set of unit machines without violating the task precedence order. The alignment process returns a minimum finishing time schedule.

(vii) *Cost evaluation*: The cost value evaluates a lowest finishing time schedule to ensure either a *good enough* or a *reasonable* schedule on the basis of its cost value.

(viii) *Satisfying criterion*: This criterion supports the satisfactory or fulfilling conditions of the scheduler.

(ix) Go to step (i).

12. 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, two such kinds are shown: first, a problem with three jobs (each job has four tasks and a total of twelve tasks) on two machines. Second, a problem with six jobs (each job with four tasks and a total of twenty four tasks) on four machines. In order to simplify the complexity of the simulation, the task size of each job is fixed with a value of four, for example, a job J_1 can be represented as a set of four tasks, $\{t_{11}, t_{12}, t_{13}, t_{14}\}$, as discussed in the following subsections.

12.1 Scheduling problem with three jobs (twelve tasks) on two machines (M_1 and M_2)

The unseen dataset of twelve tasks of three jobs with their respective priority values (P) on two machines are shown in Tab. I. As per Eqs. (1) and (4), the selected 3-layer BPNN is acceptable with a similarity value of +0.9978 (i.e., S is *true*) and a validity, V, of the unseen job set being *true* with a value of +0.65.

The precedence order of the tasks of each job is based on their priorities, as shown below:

For job J_1: $t_{14} \rightarrow t_{12} \rightarrow t_{11} \rightarrow t_{13}$;
For job J_2: $t_{22} \rightarrow t_{24} \rightarrow t_{21} \rightarrow t_{23}$;
For job J_3: $t_{33} \rightarrow t_{31} \rightarrow t_{34} \rightarrow t_{32}$.

Job	Task	R	K	D	L	P
J_1	t_{11}	0.2	0.3	0.6	0.4	0.50312
J_1	t_{12}	0.5	0.2	0.5	0.6	0.62116
J_1	t_{13}	0.1	0.4	0.7	0.3	0.30189
J_1	t_{14}	0.4	0.2	0.2	0.9	0.92231
J_2	t_{21}	0.6	0.4	0.8	0.4	0.36785
J_2	t_{22}	0.2	0.4	0.2	0.9	0.93474
J_2	t_{23}	0.7	0.5	0.8	0.2	0.15355
J_2	t_{24}	0.3	0.4	0.2	0.5	0.77127
J_3	t_{31}	0.5	0.4	0.6	0.4	0.39116
J_3	t_{32}	0.8	0.2	0.7	0.3	0.22379
J_3	t_{33}	0.1	0.2	0.3	0.8	0.92432
J_3	t_{34}	0.6	0.5	0.7	0.3	0.28947

Tab. I *Unseen dataset of three jobs and their respective output P, by the BPNN.*

On the other hand, Tab. II shows the four most feasible schedules (a), (b), (c) and (d) by the greedy procedure of the scheduler, where M_1 and M_2 are two unit machines and the computation times, K, of the tasks are shown in brackets along with each task. Even though all the given schedules are reasonable as per Eq. (14), the one with minimum finishing time (FT_{min} is 2.1) and cost value (C is 0.05) is selected as a satisfying one. Therefore, schedules (a), (c) and (d) are the satisfying schedules.

12.2 Scheduling problem with six jobs (twenty four tasks) on four machines ($\mathbf{M_1}$, $\mathbf{M_2}$, $\mathbf{M_3}$ and $\mathbf{M_4}$)

Similarly to the previously described scheduling scenario, a simulation of the scheduler with an unseen dataset of six jobs (twenty four tasks) on four unit machines (M_1, M_2, M_3 and M_4) is illustrated below.

Tab. III shows an unseen dataset of twenty four tasks with their respective priority values, P. The validity, V, of the unseen job set is *true* with +0.75. Therefore, the given input job set is valid. Furthermore, the precedence orders of the tasks of six jobs are detected on the basis of their priorities, as shown below:

For job J_1: $t_{14} \rightarrow t_{12} \rightarrow t_{11} \rightarrow t_{13}$;
For job J_2: $t_{22} \rightarrow t_{24} \rightarrow t_{21} \rightarrow t_{23}$;
For job J_3: $t_{33} \rightarrow t_{31} \rightarrow t_{34} \rightarrow t_{32}$;
For job J_4: $t_{43} \rightarrow t_{44} \rightarrow t_{42} \rightarrow t_{41}$;
For job J_5: $t_{53} \rightarrow t_{52} \rightarrow t_{54} \rightarrow t_{51}$;
For job J_6: $t_{63} \rightarrow t_{62} \rightarrow t_{61} \rightarrow t_{64}$.

Tab. IV shows the three most feasible schedules (schedule (a), schedule (b), and schedule (c)) are generated by the task alignment procedure. Where M_1, M_2, M_3, Schedule (a): $FT_a = 2.1$

M_1	t_{14} (0.2)	t_{33} (0.2)	t_{24} (0.4)	t_{11} (0.3)	t_{34} (0.5)	t_{23} (0.5)
M_2	t_{22} (0.4)	t_{12} (0.2)	t_{31} (0.4)	t_{21} (0.4)	t_{13} (0.4)	t_{32} (0.2)

Schedule (b): $FT_b = 2.2$

M_1	t_{14} (0.2)	t_{22} (0.4)	t_{31} (0.4)	t_{11} (0.3)	t_{21} (0.4)	t_{32} (0.2)
M_2	t_{33} (0.2)	t_{12} (0.2)	t_{24} (0.4)	t_{34} (0.5)	t_{13} (0.4)	t_{23} (0.5)

Schedule (c) – $FT_c = 2.1$

M_1	t_{33} (0.2)	t_{22} (0.4)	t_{34} (0.5)	t_{12} (0.2)	t_{21} (0.4)	t_{13} (0.4)
M_2	t_{14} (0.2)	t_{31} (0.4)	t_{24} (0.4)	t_{32} (0.2)	t_{11} (0.3)	t_{23} (0.5)

Schedule (d) – $FT_d = 2.1$

M_1	t_{33} (0.2)	t_{14} (0.2)	t_{24} (0.4)	t_{34} (0.5)	t_{11} (0.3)	t_{23} (0.5)
M_2	t_{22} (0.4)	t_{31} (0.4)	t_{12} (0.2)	t_{21} (0.4)	t_{32} (0.2)	t_{13} (0.4)

Tab. II *Four feasible schedules (a, b, c and d) with their FT for the problem of three jobs on two machines.*

and M_4 are four unit machines and the computation time, K of each task is shown in brackets along with the tasks. Even though the given schedules are reasonable as per Eq. (14), schedule (c) with minimum finishing time (FT_{min} is 2.9) and cost value (C is 0.58) is selected as the satisfying schedule.

13. Comparison of the Proposed Scheduler with Earliest Deadline First (EDF) and Least Laxity First (LLF) Algorithms

The performance of the proposed subjective scheduler is compared with two well known scheduling algorithms, EDF and LLF [7, 10].

Job	Task	R	K	D	L	P
J_1	t_{11}	0.2	0.3	0.6	0.4	0.50312
J_1	t_{12}	0.5	0.2	0.5	0.6	0.62116
J_1	t_{13}	0.1	0.4	0.7	0.3	0.30189
J_1	t_{14}	0.4	0.2	0.2	0.9	0.92231
J_2	t_{21}	0.6	0.4	0.8	0.4	0.36785
J_2	t_{22}	0.2	0.4	0.2	0.9	0.93474
J_2	t_{23}	0.7	0.5	0.8	0.2	0.15355
J_2	t_{24}	0.3	0.4	0.2	0.5	0.77127
J_3	t_{31}	0.5	0.4	0.6	0.4	0.39116
J_3	t_{32}	0.8	0.2	0.7	0.3	0.22379
J_3	t_{33}	0.1	0.2	0.3	0.8	0.92432
J_3	t_{34}	0.6	0.5	0.7	0.3	0.28947
J_4	t_{41}	0.8	0.5	0.9	0.2	0.09292
J_4	t_{42}	0.7	0.5	0.7	0.3	0.25584
J_4	t_{43}	0.4	0.3	0.4	0.5	0.60375
J_4	t_{44}	0.6	0.2	0.5	0.4	0.48962
J_5	t_{51}	0.9	0.4	0.8	0.2	0.08051
J_5	t_{52}	0.2	0.2	0.4	0.5	0.60651
J_5	t_{53}	0.1	0.1	0.2	0.8	0.93634
J_5	t_{54}	0.4	0.2	0.6	0.4	0.48545
J_6	t_{61}	0.8	0.4	0.9	0.3	0.19689
J_6	t_{62}	0.4	0.8	0.8	0.4	0.32166
J_6	t_{63}	0.3	0.8	0.2	0.7	0.81685
J_6	t_{64}	0.9	0.8	0.9	0.1	0.02106

Tab. III *Unseen dataset of six jobs (24 tasks) and their respective output, P, by the BPNN.*

13.1 Comparison of the proposed scheduler with the EDF algorithm

The EDF scheduling algorithm considers independent jobs and jobs are ordered in their increasing order of deadlines. The algorithm is applicable to both single- and multi-machine problems. The optimality criterion of the EDF algorithm is to minimize the lateness (FT of the last job – deadline of the last job), that is, a *negative* lateness value can indicate an *optimal* schedule. In order to compare the proposed scheduler with the EDF algorithm, a sample scheduling problem with six jobs (J_1, J_2, J_3, J_4, J_5 and J_6) on two machines (M_1 and M_2) is considered, hence, the six independent jobs are assumed to be equal to that of the six independent tasks.

Like what has been described above, each job is represented by four attributes: K_i– *computation time* of job i, D – *deadline* of job i, R_i – *release-event* of job i, and L_i – *critical nature* of job i. Tab. V shows the six jobs and their assigned attribute values. Fig. 4 shows the possible schedule of the problem with six jobs

Schedule (a) – $FT_a = 3$

M_1	t_{53} (0.1)	t_{63} (0.8)	t_{12} (0.2)	t_{51} (0.4)	t_{61} (0.4)	t_{32} (0.2)
M_2	t_{33} (0.2)	t_{43} (0.3)	t_{54} (0.2)	t_{34} (0.5)	t_{21} (0.4)	t_{13} (0.4)
M_3	t_{14} (0.2)	t_{52} (0.2)	t_{44} (0.2)	t_{62} (0.8)	t_{11} (0.3)	t_{23} (0.5)
M_4	t_{22} (0.4)	t_{31} (0.4)	t_{24} (0.4)	t_{42} (0.5)	t_{41} (0.4)	t_{64} (0.9)

Schedule (b) – $FT_b = 3.2$

M_1	t_{14} (0.2)	t_{53} (0.1)	t_{62} (0.8)	t_{44} (0.2)	t_{13} (0.4)	t_{41} (0.4)
M_2	t_{22} (0.4)	t_{63} (0.8)	t_{52} (0.2)	t_{61} (0.4)	t_{42} (0.5)	t_{64} (0.9)
M_3	t_{33} (0.2)	t_{12} (0.2)	t_{21} (0.4)	t_{54} (0.2)	t_{34} (0.5)	t_{23} (0.5)
M_4	t_{43} (0.3)	t_{24} (0.4)	t_{11} (0.3)	t_{31} (0.4)	t_{51} (0.4)	t_{32} (0.2)

Schedule (c) – $FT_c = 2.9$

M_1	t_{43} (0.3)	t_{14} (0.2)	t_{31} (0.4)	t_{41} (0.4)	t_{23} (0.5)	t_{51} (0.4)
M_2	t_{53} (0.1)	t_{63} (0.8)	t_{12} (0.2)	t_{21} (0.4)	t_{34} (0.5)	t_{64} (0.9)
M_3	t_{22} (0.4)	t_{44} (0.2)	t_{62} (0.8)	t_{52} (0.2)	t_{61} (0.4)	t_{13} (0.4)
M_4	t_{33} (0.2)	t_{24} (0.4)	t_{42} (0.5)	t_{11} (0.3)	t_{54} (0.2)	t_{32} (0.2)

Tab. IV *Three feasible schedules (a, b, and c) for the problem with six jobs on four machines.*

on two machines (M_1 and M_2) by the EDF algorithm. It is noticed that the EDF algorithm fails to deliver true *Sat* to the user, which means that the EDF algorithm fails to fulfill the satisfying criterion of the scheduler, as discussed earlier in Section 10. But the schedule is *optimal* as per its lateness criterion which equals -0.2.

Fig. 5 shows the scheduling result by the subjective scheduler. The said scheduler finds the priority of the jobs based on the given subjective criteria similarly to the cases in Subsections 12.1 and 12.2. It is noticeable that the result is *not optimal* with the EDF algorithm due to its *positive* lateness value (2.0), but the schedule is *reasonable* with a cost value 0.1.

Attributes	J_1	J_2	J_3	J_4	J_5	J_6
K	0.1	0.1	0.1	0.3	0.2	0.2
D	0.3	0.99	0.7	0.8	0.5	0.4
E	0.2	0.4	0.2	0.3	0.6	0.5
L	0.8	0.2	0.4	0.3	0.6	0.5

Tab. V *Six jobs and their four attribute values.*

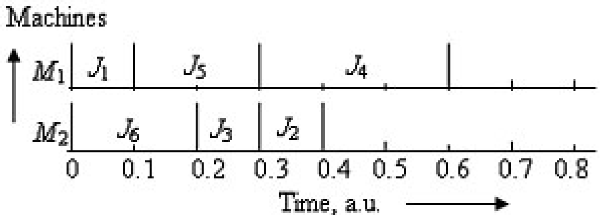

Fig. 4 *Scheduling result with the EDF algorithm (FT = 0.6).*

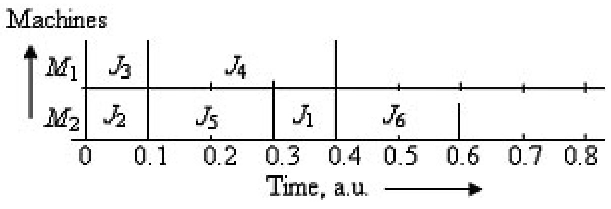

Fig. 5 *Scheduling result with the subjective scheduler (FT = 0.6).*

It can be concluded from the aforesaid simulation that even though the proposed scheduler is not optimal with the EDF algorithm, its true *Sat* indicates, as per Eq. (15), that the result is a satisfying one.

13.2 Comparison of the proposed scheduler with the LLF algorithm

The LLF scheduling algorithm assigns priority to jobs according to their relative laxity (deadline – computation time). Therefore, jobs with smallest laxity will be executed at the highest priority. A schedule with a *zero* lateness value indicates an optimal one. Preemption is allowed and in order to compare the performance of the subjective scheduler with the LLF algorithm, the same problem described in the previous section is considered (see Tab. V). Fig. 6 shows the schedule with the LLF algorithm. Though the LLF algorithm generates *optimal* schedule (with lateness 0), it fails to show a true *Sat*, thus the LLF algorithm fails to fulfill the satisfiability conditions.

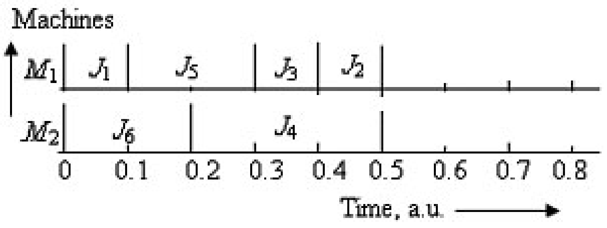

Fig. 6 *Scheduling result with the LLF algorithm (FT = 0.5).*

Fig. 7 shows the schedule by the subjective scheduler. It can be noticed that the result is not optimal with the LLF algorithm due to its *non-zero* lateness value (+1.0), but the schedule is reasonable with a cost equal to 0.1.

Even though the result with the subjective scheduler is not optimal as with the LLF algorithm, it is a satisfying one as per the Eq. (15).

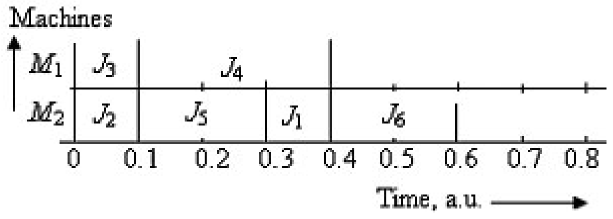

Fig. 7 *Schedule with the subjective scheduler (FT = 0.6).*

14. Conclusion

The presented subjective job scheduler shows its ability in generating user satisfied schedules by establishing proper neural net training paradigm, exempting invalid job inputs and evaluating the schedules with its cost evaluation. The scheduler utilizes the customizable nature of the BPNN and the quick solution feature of the greedy algorithm.

The term ‘task priority’ of the scheduler cannot be described formally; that is, it is not possible to define the priority of a task in a normal way because that depends only on the given subjective influence. That is 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 scheduling agent for providing user satisfied schedules. The said scheduler is most suitable in machine automation and robotics applications.

Acknowledgements

The support of the Montfort Brothers of St. Gabriel, Assumption University of Thailand, is gratefully appreciated.

References

[1] Cook S. A.: The complexity of theorem proving procedures. In: Proc. of the Third Annual ACM Symposium on the Theory of Computing., NY, 1971, pp. 151-158.

[2] Garey M. R., Johnson D. S.: Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman and Co., 1979.

[3] Rao V. B., Rao H. V.: Neural Networks & Fuzzy Logic, BPB Publications, Delhi, 1996.

[4] Stinson S.: An Introduction to the Design and Analysis of Algorithms, Cambridge University Press, 1980, pp. 70-92.

[5] Cormen T. H., Leiserson C. E., Rivest R. L., Stein C.: Introduction to Algorithms, MIT Press: McGraw-Hill, 2001, pp. 370-399.

[6] Anilkumar K. G., Tanprasert T.: A Subjective Scheduler Based on Neural Network for Job Routing in a Generalized Job-Shop Problem, GESTS International Transactions on Computer Science and Engineering, 45, 2008, pp. 79-96.

[7] Cottet F., Delacroix J., Kaiser C., Mammeri Z.: Scheduling in Real-Time Systems, John Wiley & Sons, Ltd., England, 2002, pp. 93-102.

[8] Anilkumar K. G., Tanprasert T.: Generalized Job-shop Scheduler Using Feed Forward Neural Network and Greedy Alignment Procedure. In: Proc. of IASTED Conference on Artificial Intelligence and Applications., Austria, 2007, pp. 115-120.

[9] Johnson R. A., Wichern D. W.: Applied Multivariate Statistical Analysis, 5$^{\text{th}}$ edition, NJ: Prentice Hall, NJ, 2002, pp. 668-719.

[10] Buttazzo G. C.: Hard Real-Time Computing Systems – Predictable Scheduling Algorithms and Applications, 2$^{\text{nd}}$ Edition, Springer, 2005, pp. 45-70.

APPENDIX A

R	K	D	L	P
0.1	0.1	0.1	0.9	0.01
0.1	0.5	0.1	0.9	0.03
0.1	0.3	0.3	0.7	0.1
0.1	0.3	0.3	0.9	0.04
0.1	0.3	0.5	0.7	0.08
0.1	0.5	0.5	0.7	0.06
0.1	0.7	0.7	0.3	0.7
0.1	0.3	0.5	0.7	0.12
0.3	0.5	0.3	0.9	0.2
0.3	0.5	0.5	0.7	0.45
0.3	0.5	0.3	0.7	0.35
0.3	0.9	0.7	0.5	0.68
0.3	0.1	0.9	0.3	0.72
0.3	0.1	0.1	0.9	0.03
0.3	0.1	0.7	0.5	0.64
0.3	0.1	0.3	0.7	0.32
0.3	0.5	0.9	0.3	0.74
0.5	0.7	0.7	0.5	0.58
0.5	0.1	0.9	0.3	0.74
0.5	0.3	0.7	0.5	0.55
0.5	0.3	0.5	0.7	0.55
0.5	0.7	0.9	0.3	0.76
0.5	0.7	0.7	0.5	0.46
0.5	0.3	0.9	0.1	0.78
0.5	0.1	0.7	0.3	0.73
0.5	0.5	0.7	0.1	0.76
0.5	0.3	0.7	0.3	0.7
0.7	0.3	0.7	0.5	0.65
0.7	0.1	0.9	0.1	0.82
0.7	0.7	0.7	0.3	0.78
0.7	0.7	0.9	0.1	0.85
0.7	0.5	0.9	0.1	0.83
0.7	0.1	0.9	0.7	0.66
0.7	0.1	0.5	0.5	0.54
0.7	0.5	0.7	0.3	0.77
0.7	0.1	0.7	0.3	0.67
0.9	0.3	0.9	0.1	0.95
0.9	0.9	0.9	0.1	0.99
0.9	0.7	0.9	0.1	0.97
0.9	0.1	0.9	0.1	0.89

Tab. VI *A seen dataset with forty inputs and their respective output data patterns.*