Indonesian Journal of Electrical Engineering and Informatics (IJEEI)
Vol. 7, No. 2, June 2019, pp. 190~202
ISSN: 2089-3272, DOI: 10.11591/ijeei.v7i2.464

Improved time quantum length estimation for round robin scheduling algorithm using neural network

Sonia Zouaoui[1], Lotfi Boussaid[2], Abdellatif Mtibaa[3]
[1,2]Laboratory of Electronics and Micro-electronics, University of Monastir, Tunisia
[3]National Engineering School of Monastir, University of Monastir, Tunisia

Article Info

Article historys:

Received Sep 25, 2018
Revised Dec 07, 2018
Accepted Jan 18, 2019

Keywords:

Length estimation
Neural networks model
Quantum time
Round-robin scheduling algorithm
Turnaround time

ABSTRACT

In most cases, the quantum time length is taken to be fix in all applications that use Round Robin (RR) scheduling algorithm. Many attempts aim to determination of the optimal length of the quantum that results in a small average turnaround time, but the unknown nature of the tasks in the ready queue make the problem more complicated: Considering a large quantum length makes the RR algorithm behave like a First Come First Served (FIFO) scheduling algorithm, and a small quantum length cause high number of contexts switching. In this paper we propose a RR scheduling algorithm based on Neural Network Models for predicting the optimal quantum length which lead to a minimum average turnaround time. The quantum length depends on tasks burst times available in the ready queue. Rather than conventional traditional methods using fixed quantum length, this one giving better results by minimizing the average turnaround time for almost any set of jobs in the ready queue.

Corresponding Author:

Sonia Zouaoui,
Laboratory of Electronics and Micro-electronics,
University of Monastir,
Riadh city, 4000, Sousse, Tunisia.
Email: sonia.zouaoui87@gmail.com

1. INTRODUCTION

Real-time systems are increasingly used in the contemporary world. Long reserved for heavy industrial equipment (power plant, manufacturing, production line, avionics, weapons systems), their fields of use have today varied, we can find them in consumer products automotive, telephone, home automation for which development time for placing on the market should be minimized [1]. These systems must not only correctly perform actions but also realize them in a limited time. In an autopilot system of an airplane, for example, the one who must make the right decision to avoid an obstacle while respecting temporal constraints [2]. The non-respect of these constraints, which can have catastrophic consequences, are called "hard real-time constraints".

On the other hand, in a system such as a mobile telephone network, the non-respect of time constraints can lead to less serious consequences such as loss of signal or an offset in the conversation [3]. These constraints are called "flexible real-time constraints". An embedded real-time system consisting of a calculator that executes a set of programs by interacting with its physical environment [4]. The state changes are detected by the calculator which using a variety of sensors. A real-time system is called reactive since it is in permanent interaction with the environment.

Finally, most systems (automobiles, airplanes, telephones, etc.) are also embedded devices for which the dimensions, weight and consumption aspects must be taken into account [5]. The embedded aspect imposes to guarantee the respect of the real time constraints while minimizing the cost and the size of the material architecture. The programs of a real-time system are mainly based on application algorithms for

command, control, signal and images processing that require large amounts of calculations. When the latter must be carried out in a limited time, it is necessary to use high power calculator. The general problem of real-time systems is to distribute tasks on the processors composing the hardware architecture that is known as "Task Scheduling".

Scheduling is the most important service of an operating system; it gives processes access to system resources [6]. Many requirements like fast computing, multitasking (execute more than one process at a time) and multiplexing (transmit multiple flows simultaneously) arise the need of scheduling algorithms [7]. Scheduling is in the heart of an operating system. It presents a fundamental function, which selects the process to run when there are multiple runnable processes [8]. There are varieties of scheduling algorithms such as First Come First Served (FCFS), Shortest Job First (SJF), Round Robin (RR), Priority Based Scheduling, etc. [9]-[12]. In CPU scheduling, a number of assumptions are taken into consideration, which are as follows [13], [14]:

Job pool consists of runnable processes waiting for the CPU.

1. All processes are independent and compete for resources.
2. The function of the scheduler is to fairly allocate the limited resources of CPU to the different processes and in a way that optimizes some performance criteria.

The scheduler, which constitutes the heart of the kernel, plays a fundamental role in selecting the appropriate process to be run. In this context, an operating system can be characterized according to three different types of schedulers: a long term, a mid-term or medium term and a short-term scheduler Figure 1.

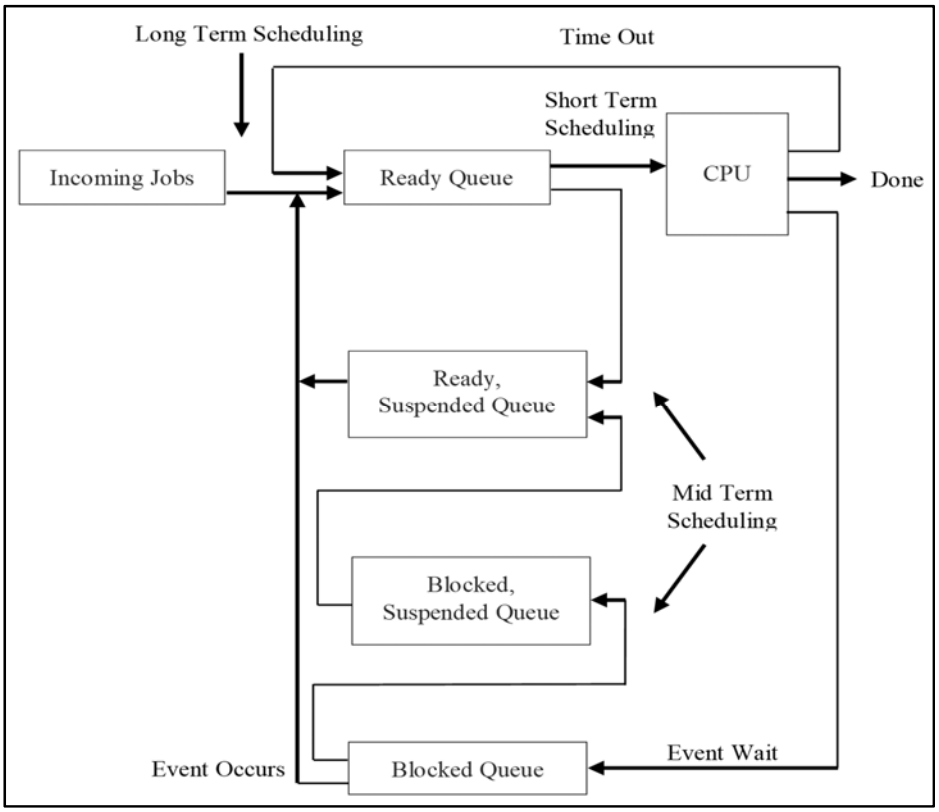

Figure 1. Various types of schedulers

For long-term scheduler, it loads processes in memory for execution after selecting them from the job pool. Concerning the short-term scheduler, it allocates the CPU to one process from those ready to be executed [12], [15]. For medium-term scheduler, it takes off processes from main memory and place them on secondary memory (such as a disk drive) or vice versa. This is commonly referred to as “Swapping” of processes in memory [9], [16].

Due to their poor performance, the majority of these algorithms are rarely used in real time operating systems (RTOS) except for the Round Robin scheduling. The efficiency of the Round Robin

Scheduling Algorithm depends on the time quantum size. Firstly, if the quantum of time (QT) is extremely large, it decreases the response time and it behaves similar to FCFS algorithm. In the other hand, if the QT is extremely small this causes many context switches, which decreases the CPU efficiency [17], [18]. As a good remedy to this problem Artifical Neural Network model (ANN) is used to estimate the time quantum has the advantage of varying the time quantum according to the variation of the service time of tasks in the ready queue [19]-[21]. Instead of using fixed quantum time, neural network model estimates and predict the time quantum that minimizes the average turnaround time.

In this paper, first, we discuss some mechanism deployed to improve RR scheduling algorithm performance. Second, we detail RR scheduling algorithm and its drawbacks. Finally, we introduce the Improved Time Quantum length estimation for round robin scheduling algorithm using Neural Network and the simulation results

2. RELATED WORKS

In the recent years, a number of CPU scheduling mechanisms have been developed for predictable allocation of processor. Extracting advantages of each algorithm and try to mix them to lead to the perfect algorithm according to a specific situation.

In [22], an improved Round Robin scheduler is developed named Improved Round Robin (IRR) CPU scheduling algorithm. It works similar to Round Robin (RR) with a small improvement. IRR picks the first process from the ready queue and allocate the CPU to it for a time interval of up to one QT. Every time a process accomplishes its QT, it checks the remaining CPU burst time of the currently running process. If the remaining CPU burst time of the currently running process is less than one QT, the CPU again allocated to the currently running process for remaining CPU burst time.

To evaluate the performance, let's consider the following ready queue Table 1. Performances of the two algorithms are resumed in Table 2 regarding to Average WT and Average TT.

Table 1. Process's Id and Burst Time

Process ID	Burst Time (ms)
P1	5
P2	12
P3	23
P4	26
P5	34

Table 2. Comparison of RR and IRR

Algorithm	Average WT (ms)	Average TT (ms)
RR	38.4	57.8
IRR	30.4	49.8

For simple Round Robin algorithm, the Gantt chart is shown in Figure 2 for QT=10 ms. Concerning Improved Round Robin algorithm (IRR), the Gantt chart will be as follow Figure 3.

	P1	P2	P3	P4	P5	P2	P3	P4	P5	P4	P5	P5
0	5	15	25	35	45	47	57	67	77	83	93	97

Figure 2. Gantt chart for simple RR

	P1	P2	P3	P4	P5	P2	P3	P4	P5	P4	P5	P5
0	5	15	25	35	45	47	57	67	77	83	93	97

Figure 3. Gantt chart for IRR

The proposed IRR in CPU scheduling algorithm is giving better performances than RR. After improvement in RR, it has been found that the WT and TT have been reduced drastically [22].

A modified round robin algorithm, proposed by [23], consists of a mixture between shortest job first and round robin algorithms. In this new algorithm, the QT takes the burst time of mid process when the number of processes is odd else, it takes the average time of burst time of all processes.

To evaluate the performance, let's consider the following ready queue Table 3. Performances of the two algorithms are resumed in Table 4 regarding to average WT and average TT.

Table 3. Process's Id and burst time

Process ID	Burst Time (ms)
P1	14
P2	45
P3	36
P4	25
P5	77

Table 4. Comparison of RR and MRR

Algorithm	Average WT (ms)	Average TT (ms)
RR	70.2	109.6
MRR	56.8	96.2

For simple Round Robin algorithm, the Gantt chart is shown in Figure 4 with QT=25 ms. Concerning Modified Round Robin algorithm (MRR), the Gantt chart will be as follow Figure 5.

	P1	P2	P3	P4	P5	P2	P3	P5	P5
0	14	39	64	89	114	134	145	170	195

Figure 4. Gantt chart for simple RR

	P1	P2	P3	P4	P5	P2	P5	P5
0	14	50	86	111	147	156	192	197

Figure 5. Gantt chart for MRR

It is concluded from the above experiments that the proposed algorithm MRR performs better than simple RR in terms of performance metrics such as average WT and average TT.

In the work of [24], the proposed algorithm is based on the integration of Round Robin and priority scheduling algorithm and also implements the concept of aging by attributing new process priorities. First, the CPU is allocated to every process in round robin fashion with a given priority and quantum. Then, processes are sorted in increasing order according to their remaining CPU burst time in the ready queue. Consequently, new priorities are assigned; the process with shortest remaining CPU burst is assigned with highest priority. Each process gets the control of the CPU until they finished their execution.

This new approach improves the performance of CPU in real time operating system. To evaluate the performance, let's consider the following ready queue Table 5. Performances of the two algorithms are resumed in Table 6 regarding to average WT and Average TT.

Table 5. Process's Id and burst time

Process ID	Burst Time (ms)	Priority
P1	22	4
P2	18	2
P3	9	1
P4	10	3
P5	4	5

Table 6. Comparison of RR and PRR

Algorithm	Average WT (ms)	Average TT (ms)
RR	33.2	45.8
PRR	26.2	38.8

For simple Round Robin algorithm with QT =5ms, we obtain the following Gantt chart Figure 6. Concerning Priority based Round Robin algorithm (PRR), tasks are executed according to their priority, and the Gantt chart will be as follow Figure 7.

	P1	P2	P3	P4	P5	P1	P2	P3	P4	P1	P2	P1	P2
0	5	10	15	20	24	29	34	38	43	48	53	58	61

Figure 6. Gantt chart for simple RR

	P3	P2	P4	P1	P5	P3	P4	P2	P1
0	5	15	25	35	45	47	57	67	77

Figure 7. Gantt chart for PRR

The proposed algorithm improves all the drawbacks of round robin CPU scheduling algorithm. It retains the advantage of round robin in reducing starvation and also integrates the advantage of priority scheduling.

In [25], a New Improved Round Robin (NIRR) was developed. This algorithm is an improved version of Improved Round Robin (IRR) algorithm mentioned above in [22]. This algorithm holds processes according to their arrival times in a queue named ARRIVE queue. In the other hand, other processes are in a queue called REQUEST queue waiting their turn to occupy the CPU. The time quantum is considered as the average of burst times of the processes in the REQUEST queue.

To evaluate the performance, let's consider the following ready queue Table 7. Performances of the two algorithms are resumed in Table 8 regarding to the average WT and the average TT.

Table 7. Process's with its Id and burst time

Process ID	Burst Time (ms)
P1	23
P2	75
P3	93
P4	48
P5	2

Table 8. Comparison of RR and NIRR

Algorithm	Average WT (ms)	Average TT (ms)
RR	113	161.2
NIRR	53.8	102

For simple Round Robin algorithm with QT=50ms, we obtain the following Gantt chart Figure 8. Concerning New Improved Round Robin algorithm (NIRR), the Gantt chart is as follows Figure 9.

	P1	P2	P3	P4	P5	P2	P3
0	23	73	123	171	173	198	241

Figure 8. Gantt chart for simple RR

	P1	P5	P4	P2	P3	P3
0	23	25	75	148	203	241

Figure 9. Gantt chart for NIRR

In [26], an algorithm called “Modulo Based Round Robin Algorithm” is proposed. This algorithm is based on Round Robin fashion an intelligent time quantum and then assign priority to the processes.

This algorithm starts by calculating the average of CPU burst of all the processes (P) and then compute for each process the burst time modulo P which is named (M). After that processes are sorted according to the value of M and then the time quantum (QT) is considered equal to P.

To evaluate the performance, let's consider the following ready queue Table 9 (All processes arrive at t=0 and QT=10 ms). Performances of the two algorithms are resumed in Table 10 regarding to the average WT and the average TT.

Table 9. Process's ID and burst time

Process ID	Burst Time (ms)
P1	10
P2	20
P3	30
P4	40
P5	50

Table 10. Comparison of RR and "Modulo Based Round Robin Algorithm"

Algorithm	Average WT (ms)	Average TT (ms)
RR	60	90
NIRR	52	82

For simple Round Robin algorithm, we obtain the following Gantt chart Figure 10. For "Modulo Based Round Robin Algorithm", the Gantt chart is as follows Figure 11.

	P1	P2	P3	P4	P5	P2	P3	P4	P5	P3	P4	P5	P4	P5	P5
0	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150

Figure 10. Gantt chart for simple RR

	P3	P1	P2	P4	P5	P4	P5
0	30	40	60	90	120	130	150

Figure 11. Gantt chart for "Modulo Based Round Robin Algorithm"

In the other hand, the intelligent algorithms are more and more concerned, like genetic algorithm (GA), swarm optimization (PSO) algorithm, ant colony optimization (ACO) algorithm, bee colony optimization (BCO) algorithm etc. Intelligent algorithm is integrated to Round robin scheduling algorithm in order to construct better algorithm performances.

Intelligent algorithms combined with Round Robin scheduling algorithm are spatially used to compute a suitable time quantum for a given CPU scheduling scenario. In the work of [27], the Fuzzy Inference System (FIS) is used to find the optimum time quantum. The FIS has got 2 inputs and one output. First input is N which presents the number of user/ processes in the system and the second input is the average burst time of the processes in the ready queue. As output of FIS we have the Time quantum. Memberships are distributed as follows:

Membership Function for N

Type- Triangular, Range:1-10, low- [0,2,4], medium [3,5.5,8] High [7,8.5,10]

Membership Function the Average Burst Time (ABT)

Type-Triangular, Range:0-10, low- [-4,0,4], medium- [3,5,7] High: - [6,10,16]

Membership Function for Time Quantum

Type- Triangular, Range:1-5, low- [0,1,2], medium-
[1,2.5,4] High: - [3,5,7]

Taking into consideration basic rules of FIS mentioned in the table below (Table 11):

Table 11. Rules base for FIS

S. No	N	ABT	Time quantum
1	low	low	low
2	low	medium	medium
3	low	high	high
4	medium	low	medium
5	medium	medium	medium
6	medium	high	medium
7	high	low	low
8	high	medium	low

The proposed algorithm in [27] consists of, first, finding the average burst time of processes then take it as input to FIS as well as the ABT. The output of the system is considered as the suitable time quantum for a given CPU scheduling scenario.

In [28], the algorithm of quantum evolution is integrated into ant colony optimization algorithm. The mix of these two algorithms results on the proposed algorithm which is an Improved Quantum Ant Colony Optimization (IMAQACO) algorithm being able to solve complex function problems. In this algorithm, the pheromone is represented by the quantum state vectors, the control pheromone evaporation factor is realized by the adaptively dynamical updating strategy. The ant movement and the change of the quantum probability amplitude convergence tend is realized by the quantum rotation gate and finally the ant location variation is achieved by quantum non-gate. Experiments illustrated in [28] show that the proposed IMAQACO can avoid falling into the local optimum, improve the convergence speed and take on stronger global optimization ability and higher convergence speed.

In the proposed study of [29], an applied Genetic Approach based Round Robin algorithm for CPU scheduling is introduced. The fitness of the chromosome represents the average waiting time where the fittest chromosome is one with minimum average waiting time. At the beginning, the quantum is the median of all the processes burst time.

Three approaches to implement round robin are developed and compared which are Simple RR with static quantum, Simple RR with dynamic quantum and GA based RR with dynamic quantum. This work shows as well a comparison of these algorithms on the basis of average waiting time. Table 12 shows the sequences of five processes along with burst time and the respective average waiting time for algorithms mentioned above. The result highlights that the performance of Round Robin is improved with the dynamic quantum. The Genetic Approach based RR gives drastically enhanced performance over simple RR.

Table 12. Comparison of the three algorithms

SR.NO	Process burst time [P1, P2; P3, P4, P5]	RR Static	RR dynamic	GA RR dynamic
1	[15,12,5,18,16]	36.0	29.4	20.6
2	[12,14,10,12,11]	39.2	39	22.0
3	[80,70,20,15,75]	123	117	71
4	[40,45,50,30,35]	85	85	72
5	[19,40,7,25,18]	54.4	49	35
6	[47,37,17,15,8]	62.4	61	37.6
7	[120,118,55,84,77]	292.2	272.6	155
8	[70,72,46,67,58]	194.6	208.8	119.2
9	[32,26,17,45,38]	82.8	79.4	53.2
10	[78,57,18,28,33]	111	109.8	62.6

3. DRAWBACKS OF ROUND ROBIN SCHEDULING ALGORITHM

Round Robin scheduling algorithm has many disadvantages, which are as following [30]-[33]:

3.1. HighAverage Waiting Time

For round robin architecture, the process spends the time in the ready queue waiting his turn to own the processor. Due to the presence of QT, processes are pushed to leave the processor and return to the waiting state. This procedure produces a high average waiting time, which presents the main disadvantage.

3.2. Low Througput

The Throughput presents the number of process completed per time unit. Due to its time slice, Round Robin is characterized by a high number of context switches, which leads to overall degradation of the system performance (throughput). Cross-referencing and computer searching. An improperly titled paper may never reach the audience for which it was intended, so be specific.

3.3. Context Switch

Ones the time slice finished, the process is forced to leave the CPU. The scheduler stores the context of the current process in stack or a register and allots the CPU to the next process in the ready queue. This concept is known by context switching which leads to time wastage and scheduler overhead.

3.4. High Response Time

Response time is defined as the time between submission of a request and the first CPU response. In general, round robin made larger response time, which causes system performance degradation. To achieve high performance, the reduction of response time shall be indispensable.

3.5. Very High Turnaround Time

The TT is the time between submission of a process and its completion. Round Robin scheduling algorithm is characterized by a high turnaround time. So as a result, to improve the system performance this parameter should be reduced.

4. PROPOSED WORK

4.1. Problem Formulation

The average turnaround time in RR scheduling algorithm is the most important criteria to evaluate the algorithm performance. The lowest the average turnaround time is, the better scheduling performance is. For RR scheduling algorithm, the average turnaround time are directly affected by the time quantum. Figure 12 Shows the way in which turnaround time varies with the time quantum [34].

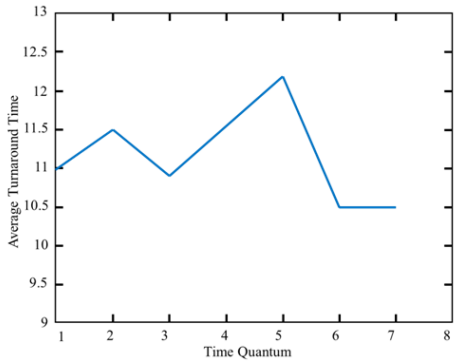

Figure 12. The variation of the turnaround time with the time quantum

$$T_t = \sum_{I=0}^{N} \frac{S_t^i + W_t^i}{N} \quad (1)$$

where *Tt* is the average turnaround time,
S_t^i is the service time of job *i* in the ready queue,
W_t^i is the waiting time of job *i* in the ready queue,
N is the number of jobs in the ready queue.

Our aim is to minimize the average turnaround time by managing the optimal quantum time length *q*. In the other hand, the average waiting time depends essentially on the time quantum and can be described as the function below:

$$W_t^i = N(q + o) \quad (2)$$

where o is the context time switch.

Assuming that the context time switch is negligible compared to the quantum time length. Therefore, (1) becomes (3).

$$T_t = \sum_{I=0}^{N} \frac{S_t^i + Nq}{N} \quad (3)$$

Consequently, the average turnaround time highly depends on the value of the selected quantum length as well as the tasks service time in the ready queue. Then the compromise is to find the time quantum length q in such way that the turnaround time is minimized.

4.2. Artifical Neural Network Model

Artifical Neural Network (ANN) is one of the most effective computer modeling procedures based on factual approach, as of now being utilized in numerous fields of building for modeling complex connections which are exceptionally troublesome to portray with physical models [35], [36]. Thefascination of neural systems comes from their exceptional data, handling characteristics related primarily to nonlinearity, high parallelism, fault and noise tolerance and learning and generalized capability.

4.2.1 Multi Layer Feed Forward Neural Network

The multi-layer network architecture consists of one input layer, one output layer and has one or more intermediary layers, called hidden layers. The hidden layers are based on neurons which present the computation units. Before directing the input to the output layer, intermediary computations are carried out by the hidden layers [37]. The input layer neurons and the hidden layer neurons are linked via a corresponding weight. Furthermore, the hidden layer neurons are as well linked to the output layer neurons via a corresponding weight.

4.2.2 Back Propagation Network

Back propagation network is a systematic technique for training multilayer artificial neural networks. Gradient descent algorithms are applied [38]. For non-linear networks, the way the gradient is computed reveals the term of back propagation. In multilayer networks, this algorithm allows experimental acquisition of the input/output mapping knowledge and is carried out in three stages: first the feed forward of the input training pattern in which each input neuron receives an input signal and broadcasts it to each hidden neuron. The latter computes the activation and send it to each output unit which again computes the activation to obtain the net output.

Second, back propagation of the associated error in which, when training, the final output is compared with the target value and appropriate error is calculated. And finally, the weight adjustment, during which, weights are updated correspondingly. In the other hand, the error factor is computed for all units. Once the error factors are obtained, weights are updated simultaneously.

4.2.3. ANN Model for Performance Characteristics

Neural network model development follows a specific procedure as shown in Figure 13. This procedure follows four principle steps which are:

i) Collection of input/output data set;
ii) Preprocessing of input/output data set;
iii) Neural network designing and training;
iv) Testing of the network;

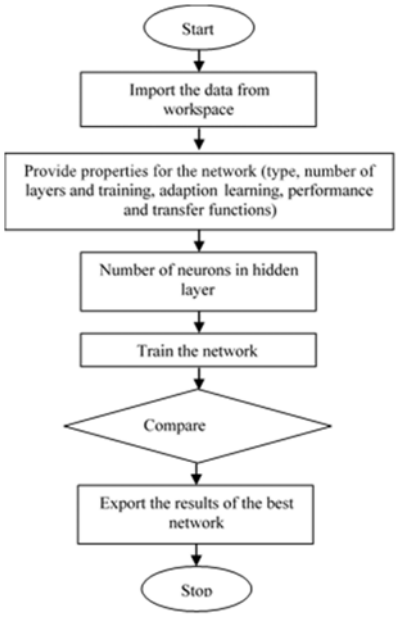

Figure 13. Flowchart of the development of neural network model

To insist, the suggestion of neural network model to be used in this study in order to estimate the length of the quantum time that will be used in the RR scheduling algorithm. The inputs for the neuron network model are tasks burst times in the ready queue Figure 14. The model output is the optimum length of the time quantum that gives minimum turnaround time.

Rather than conventional methods of RR scheduling algorithm which use fixed quantum length, the proposed RR scheduling algorithm is based on an intelligent quantum giving superior results and minimizing the turnaround time in almost all sets of jobs in the ready queue.

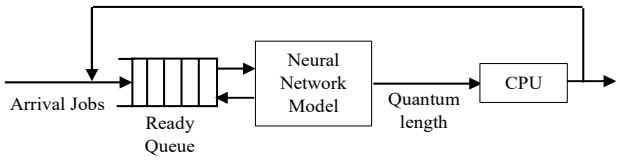

Figure 14. The architecture of the proposed scheduling algorithm.

This research is based on researches revealed by [34] which used multi layer perceptron neural network with one hidden layer. The number of neurons in the input layer are determined by the number of jobs in the ready queue which is Ten (10). Here the output of the neural network is the quantum length that results into minimum turnaround time.

The service time of the jobs are generated randomly to have values between 1.0 and 10.0 units. In [34], ten thousand samples of the ten jobs are simulated. Here, for each sample (consisting of 10 jobs) the average turnaround time is calculated for fifty different values of quantum time length starting from 0.1 and stepping up to 5.0 at a step of 0.1.

In [34], obtaining the optimum results was by the use of 31 neurons in one hidden layer. By the way, for our design, we keep the same inputs Table 13 and modifying each time the number of neurons in the hidden layer. Various network architectures were investigated in order to determine the optimal quantum length (i.e. the highest coefficient of determination, the lowest root mean square error and the lowest mean bias error) for each combination of input variables. The number of hidden layers is fixed for two Figure 15. First, the neural network model is trained for 10 neurons for each hidden layer. Then after evaluating the performance, we add ten other neurons for each hidden layer. Scaled conjugate gradient method was implemented to obtain the optimal parameters of the neural network model.

In addition, the tangent sigmoid in the hidden layer is also investigated.

Table 13. Several sets of Tasks Burst Times

Inputs (Tasks Burst Times)									
S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
4.1	5.8	3,9	9,6	4,5	8,3	8,1	4,2	1,9	5
3.8	2.2	4,6	9,3	6,4	1	1	4	2,5	3,5
2.9	2	10	9,7	1,4	8,1	9,6	0,1	5,1	4,7
9.7	2,8	0,8	9,3	3,3	1,7	6,9	4,8	7,7	8,5
6	2,9	7,2	8,6	3,1	7,8	6,3	2,8	9	7,5
8	7,8	5,9	2,8	5,3	8	1,8	8,9	9	3,9
5.3	7,1	10	6,3	4,8	1,1	3,4	2,6	8,6	3,7
5.9	5,8	8,5	7,7	5,5	5,8	5,7	7,9	2,7	3,1
8.6	6,7	6,2	5,8	4,1	1,6	8,8	7,9	0,8	5,6
7.7	1	4,3	9,3	6,3	8,2	6,1	6,7	8,7	0,3
4	2,2	6,7	5,2	5,6	1,4	9,4	8,6	3,9	3,5
5	8,5	5,6	6,1	6	1,4	2,1	3,6	5,3	1,1
2,6	6,5	6,6	0,2	9,6	4,7	5,6	4	8	2,7
7,1	4,6	1,9	0,2	6	0,9	6,7	6,9	5,1	8,7
0,3	7,3	3	0,4	1,4	8,4	9	6,3	4,3	0,4
2,6	1,9	0,3	0,2	2	1,4	0,8	4,2	5,7	5,8
1,9	6	8,2	7,1	5,2	1,1	0,3	1,3	3,8	4,5
5,1	4,2	5,4	0,9	4,2	4,3	5,3	2,9	5,3	1,6
2,2	4,5	4,9	6,2	2	4,8	7,5	0,4	5,5	8,6
3,2	4,6	3,6	1	2,3	8	3,6	2,7	4,6	0,6
7,1	9,8	6,7	5,8	5,5	4,8	3,2	3,7	2,1	6,7
4,4	7,9	1,8	5,5	1,5	1,4	0,1	3,5	2,1	9
1,6	8	2,9	5,3	9,1	9,8	8,1	1	0,4	2,7
4,9	7,4	7,2	5,6	8,7	9,1	5,1	4,8	5	1,4
1,8	6,1	8,1	6,7	9,2	1	4,1	8,7	5,1	9,6
1,8	7,6	4,8	4,3	9,6	1,5	4,4	4,2	5,7	8,5
7,1	4,2	8,2	4,3	8,8	5,7	6	7	4,9	7,4

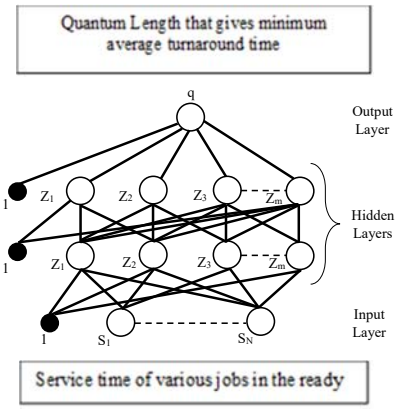

Figure 15. A multilayer feedforward neural network.

5. SIMULATION RESULTS

As mentioned above, Neural Network Model is used to predict the optimal time quantum which gives the lowest turnaround time. A set of 10 tasks with burst times are generated randomly to have values between 0.1 and 10 units. Twenty-seven samples of ten tasks each are simulated. Optimal quantum time obtained for each architecture is compared with estimated one. In this paper, MATLAB (R2014) is used to achieve simulations and tests.

Results obtained after simulation are resumed in table 11. Actually, Table 14 resumed the estimated time quantum as well as the optimal time quantum obtained with each architecture (e10_10: is the error associated to the architecture having two hidden layers containing ten neurons each, Q10_10: is the optimal quantum time given by the architecture having two hidden layers containing ten neurons each).

Results mentioned in Table14 reveals that the best results are obtained when using the architecture with two hidden layers having thirty (30) neurons each. This architecture is taken to be the model with the minimum error value in the validation stage Figure 16. Moreover, focusing on this architecture, Figure 16 shows that the mean square error decrease considerably during training phase attempting almost 10^-12 when best results are validated.

Table 14. Estimated Quantum, Optimal Quantum and Error obtained with each architecture

Qest	e10_10	Q10_10	eQ20_20	Q20_20	eQ30_30	Q30_30	eQ40_40	Q40_40	eQ50_50	Q50_50	e60_60	Q60_60
5	0,915	4,085	1,46E-12	5	7,05E-06	5,000	0,506	4,494	1,036	3,964	0,012	4,988
4,7	0,350	4,350	-0,3	5	3,11E-09	4,700	0,399	4,301	1,481	3,219	0,000	4,700
2,9	2,055	0,845	-2,1	5	1,58E-08	2,900	-1,735	4,635	-1,875	4,775	-0,005	2,905
4,8	-0,172	4,972	-0,200	5,000	2,09E-09	4,800	0,180	4,620	0,071	4,729	-0,002	4,802
3,2	1,148	2,052	-1,8	5	0,124	3,076	-0,081	3,281	0,862	2,338	-0,002	3,202
4	2,764	1,236	-1	5	7,98E-09	4,000	0,114	3,886	0,279	3,721	0,000	4,000
3,7	0,276	3,424	-1,3	5	1,09E-08	3,700	0,944	2,756	0,286	3,414	0,001	3,699
3,1	0,849	2,251	-1,9	5	1,27E-08	3,100	-0,128	3,228	-0,010	3,110	0,000	3,100
2,1	-0,445	2,545	-2,9	5	-0,170	2,270	-0,222	2,322	-1,657	3,757	-0,001	2,101
4,3	0,150	4,150	-0,7	5	6,98E-09	4,300	-0,116	4,416	1,777	2,523	-0,641	4,941
4	-0,238	4,238	-1	5	1,18E-08	4,000	-0,088	4,088	0,285	3,715	0,011	3,989
2,1	-1,387	3,487	-2,9	5	0,563	1,537	-0,008	2,108	0,362	1,738	-0,002	2,102
3,3	-0,959	4,259	-1,7	5	8,69E-09	3,300	0,046	3,254	0,337	2,963	0,252	3,048
4,6	0,661	3,939	-0,4	5	3,49E-09	4,600	0,030	4,570	0,225	4,375	0,611	3,989
0,5	-0,695	1,195	-4,5	5	-0,001	0,501	-0,199	0,699	-0,386	0,886	-0,012	0,512
4,2	-0,732	4,932	-0,8	5	-0,384	4,584	0,068	4,132	1,087	3,113	-0,002	4,202
1,3	-1,769	3,069	-3,7	5	3,28E-09	1,300	-0,598	1,898	-1,967	3,267	-3,383	4,683
4,3	0,053	4,247	-0,7	5	5,30E-09	4,300	0,009	4,291	-0,293	4,593	0,005	4,295
4,9	0,393	4,507	-0,1	5	-0,093	4,993	0,467	4,433	-0,098	4,998	0,000	4,900
4,6	0,056	4,544	-0,4	5	6,61E-09	4,600	0,449	4,151	0,172	4,428	1,803	2,797
3,7	0,167	3,533	-1,3	5	9,89E-09	3,700	0,033	3,667	-0,020	3,720	-0,447	4,147
2,3	-1,330	3,630	-2,7	5	5,55E-10	2,300	-0,339	2,639	-0,320	2,620	-0,006	2,306
2,9	-1,897	4,797	-2,1	5	1,40E-08	2,900	-0,798	3,698	-1,437	4,337	-0,004	2,904
2,8	-1,565	4,365	-2,2	5	1,42E-08	2,800	0,495	2,305	0,223	2,577	-0,403	3,203
4,1	-0,100	4,200	-0,9	5	-0,286	4,386	0,014	4,086	-0,899	4,999	0,004	4,096
4,9	0,123	4,777	-0,1	5	-0,094	4,994	1,425	3,475	-0,095	4,995	0,001	4,899
4,9	0,131	4,769	-0,1	5	1,95E-09	4,900	0,116	4,784	0,400	4,500	0,007	4,893

The error is actually the difference between estimated time quantum and optimal time quantum. Figure 17 shows estimated time quantum and optimal time quantum behaviors. The two graph shapes are approximately the same. Consequently, this neural network architecture was able to predict to a very good accuracy of the optimal quantum length that minimizes the average turnaround time in almost most of the cases.

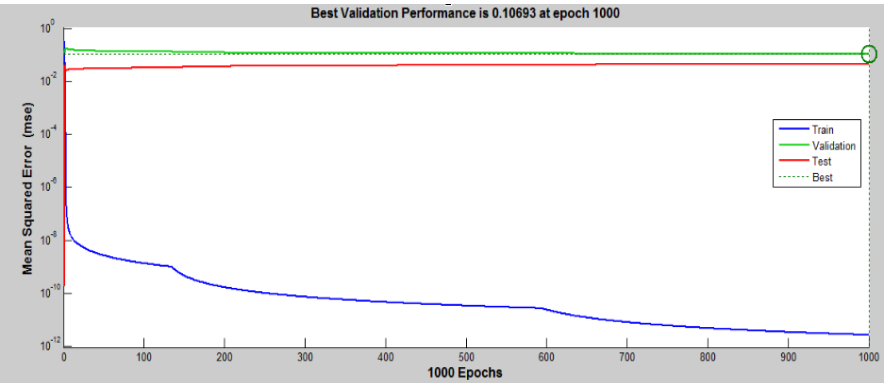

Figure 16. Best performance validation

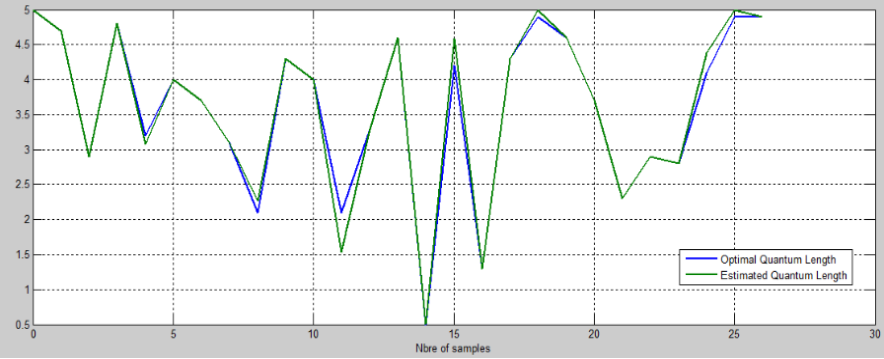

Figure 17. The estimated time quantum and optimal time quantum behaviors

6. CONCLUSION

In this paper, we apply the neural network models to predict the optimal quantum length in the RR scheduling algorithm. The dynamic change of the quantum length to be used in the RR scheduling algorithm present the biggest advantage of using neural network models. The proposed method is validated using twenty-seven samples including ten tasks each (in the ready queue). After simulating and testing, the neural network model shows the capability to predict the optimal quantum length value that yields minimum average turnaround time. The neural network-based RR scheduling algorithm has the ability to change dynamically the quantum length value based on tasks service time in the ready queue. Model training and validation are achieved off-line, for this reason, CPU computational time is kept the same.

COMPETING INTERESTS

The authors declare that they have no competing interests.

REFERENCES

[1] Rammig, F., Ditze, M., Janacik, P., Heimfarth, T., Kerstan, T., Oberthuer, S., & Stahl, K., "Basic concepts of real time operating systems," in *Hardware-dependent Software*, Springer, Dordrecht, pp. 15-45, 2009.
[2] Tanenbaum, A. S., "Multimedia Operating Systems," *Modern Operating Systems, 2nd Edition, Upper Saddle River, New Jersey: Prentice-Hall, Inc*, pp. 531-578, 2001.

[3] Emilio, M. D. P. "Embedded system design for high-speed data acquisition and control," *Springer*, 2016.
[4] Ganssle, J., "The art of designing embedded systems," *Newnes*, 2008.
[5] Jackson, L. E., & Rouskas, G. N., "Deterministic preemptive scheduling of real-time tasks," *Computer*, vol. 35(5), pp. 72-79, 2002.
[6] Sagar, P. M., "Embedded operating systems for real-time applications," in *M. Tech. credit seminar report, Electronic Systems Group, EE Dept, IIT Bombay*, 2002.
[7] William S., "Operating Systems Internal and Design Principles, 5thEdition," ISBN-10: 0-13-230998, 2006.
[8] Singla, M. K., & Scholar, M. E., "Task Scheduling Algorithms for Grid Computing with Static Jobs: A Review," *International Journal of Computer Science Engineering (IJCSE)*, vol. 2(5), pp. 218, 2013.
[9] Silberschatz, A., Galvin, P. B., & Peterson, J. L., "Operating system concepts," *Addison-Wesley*, 1991.
[10] Oyetunji, E. O., & Oluleye, A. E., "Performance assessment of some CPU scheduling algorithms," *Research Journal of Information Technology*, vol. 1(1), pp. 22-26, 2009.
[11] Siahaan, A. P. U., "Comparison Analysis of CPU Scheduling FCFS," *SJF and Round Robin*, 2017.
[12] Kishor, L., & Goyal, D., "Comparative Analysis of Various Scheduling Algorithms," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 2(4), pp.1488, 2013.
[13] Englander, I., & Englander, A., "The architecture of computer hardware and systems software: an information technology approach," *Wiley*, 2003
[14] Yavatkar, R., & Lakshman, K., "A CPU scheduling algorithm for continuous media applications," in *International Workshop on Network and Operating Systems Support for Digital Audio and Video*, Springer, pp. 210-213, 1995.
[15] Rajguru, A. A., & Apte, S. S., "A Performance Analysis of Task Scheduling Algorithms using Qualitative Parameters," *International Journal of Computer Applications*, vol. 74(19), 2013.
[16] Saxena, A., "An Effeceint Multi Parameteric Cpu Scheduling Algorithm for Single Processor Systems".
[17] Hiranwal, S., & Roy, K. C., "Adaptive round robin scheduling using shortest burst approach based on smart time slice," *International Journal of Computer Science and Communication*, vol. 2(2), pp. 319-323, 2011.
[18] Mohanty, R., Behera, H. S., Patwari, K., Dash, M., & Prasanna, M. L., "Priority based dynamic round robin (PBDRR) algorithm with intelligent time slice for soft real time systems," *arXiv preprint arXiv*:1105.1736, 2011.
[19] Kriesel, D. "A Brief Introduction to Neural Networks," 2007. URL http://www. dkriesel. com.
[20] Rojas, R., "Neural Networks-A Systematic Introduction Springer-Verlag," New York, 1996.
[21] Kröse, B., Krose, B., van der Smagt, P., & Smagt, P., "An introduction to neural networks," 1993.
[22] Mishra, M. K., "An improved round robin cpu scheduling algorithm," *International Journal of Global Research in Computer Science (UGC Approved Journal)*, vol. 3(6), pp. 64-69, 2012.
[23] Chhugani, B., & Silvester, M., "Improving Round Robin Process Scheduling Algorithm," *International Journal of Computer Applications*, vol. 166(6), 2017.
[24] Rajput, I. S., & Gupta, D. "A priority based round robin CPU scheduling algorithm for real time systems," *International Journal of Innovations in Engineering and Technology*, vol. 1(3), pp. 1-11, 2012.
[25] Abdulrahim, A., Abdullahi, S. E., & Sahalu, J. B. "A new improved round robin (NIRR) CPU scheduling algorithm," *International Journal of Computer Applications*, vol. 90(4), 2014.
[26] Emilio, M. D. P. "Embedded systems design for high-speed data acquisition and control," *Springer*, 2016.
[27] Colnarič, M., Verber, D., & Halang, W. A. "Real-time Characteristics and Safety of Embedded Systems," *Distributed Embedded Control Systems: Improving Dependability with Coherent Design*, pp. 3-28, 2008.
[28] Liu, C. L., & Layland, J. W., "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM (JACM)*, vol. 20(1), pp. 46-61, 1973.
[29] Dhumall, R. A., Maktum, T. A., & Ragha, L., "Dynamic Quantum based Genetic Round Robin Algorith," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 3(3), 2014.
[30] Tajwar, M. M., Pathan, M., Hussaini, L., & Abubakar, A., "CPU Scheduling with a Round Robin Algorithm Based on an Effective Time Slice," *Journal of Information Processing Systems*, 13(4), 2017.
[31] Kishor, L., & Goyal, D., "Comparative Analysis of Various Scheduling Algorithms," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 2(4), pp-1488, 2013.
[32] Khan, R., & Kakhani, G., "Analysis of priority scheduling algorithm on the basis of fcfs and sjf for similar priority jobs," *Int J Comput Sci Mob Comput*, vol. 4, pp. 324-331.
[33] Singh, A., Goyal, P., & Batra, S., "An optimized round robin scheduling algorithm for CPU scheduling," *International Journal on Computer Science and Engineering*, vol. 2(7), pp. 2383-2385, 2010.
[34] AlHeyasat, O., & Herzallah, R., "Estimation of Quantum Time Length for Round-robin Scheduling Algorithm using Neural Networks,". in *IJCCI (ICFC-ICNC)*, pp. 253-257, 2010.
[35] Kruse, R., Borgelt, C., Braune, C., Mostaghim, S., & Steinbrecher, M., "Computational intelligence: a methodological introduction," *Springer*, 2016.
[36] Tibshirani, R., "A comparison of some error estimates for neural network models," *Neural Computation*, vol. 8(1), pp. 152-163, 1996.
[37] Krenker, A., Bester, J., & Kos, A. "Introduction to the artificial neural networks," in *Artificial neural networks-methodological advances and biomedical applications. InTech*, 2011.
[38] Zupan, J., "Introduction to artificial neural network (ANN) methods: what they are and how to use them," *Acta Chimica Slovenica*, vol. 41, pp. 327-327, 1994.