Hindawi Publishing Corporation
Advances in Fuzzy Systems
Volume 2016, Article ID 2784067, 11 pages
http://dx.doi.org/10.1155/2016/2784067

Research Article

Designing of Vague Logic Based 2-Layered Framework for CPU Scheduler

Supriya Raheja

School of Engineering & Technology, Department of CSE & IT, Northcap University (Formerly ITM University), Gurgaon 122017, India

Correspondence should be addressed to Supriya Raheja; supriya.raheja@gmail.com

Received 30 November 2015; Accepted 20 March 2016

Academic Editor: Mehmet Onder Efe

Fuzzy based CPU scheduler has become of great interest by operating system because of its ability to handle imprecise information associated with task. This paper introduces an extension to the fuzzy based round robin scheduler to a Vague Logic Based Round Robin (VBRR) scheduler. VBRR scheduler works on 2-layered framework. At the first layer, scheduler has a vague inference system which has the ability to handle the impreciseness of task using vague logic. At the second layer, Vague Logic Based Round Robin (VBRR) scheduling algorithm works to schedule the tasks. VBRR scheduler has the learning capability based on which scheduler adapts intelligently an optimum length for time quantum. An optimum time quantum reduces the overhead on scheduler by reducing the unnecessary context switches which lead to improve the overall performance of system. The work is simulated using MATLAB and compared with the conventional round robin scheduler and the other two fuzzy based approaches to CPU scheduler. Given simulation analysis and results prove the effectiveness and efficiency of VBRR scheduler.

1. Introduction

Multitasking environment of a computer system defines the role of CPU scheduler. The scheduler uses a scheduling algorithm to decide when to schedule the task and for how long. The goal of system designer is to design the CPU scheduler in such a way that it gives users more effective and efficient throughput [1–3]. In this paper, the author discusses the round robin (RR) CPU scheduler.

Traditional RR CPU scheduler is not able enough to know the exact attributes of task like burst time, length of time quantum, arrival time, and so forth which affect the performance of system. Recent research works handle the uncertainty of attributes using fuzzy logic [4]. These developments undoubtedly improve the performance of system. The proposed work extends the fuzzy based RR scheduler to vague logic based RR scheduler. The author calls it VBRR CPU scheduler. Vague set theory over fuzzy set theory improves the modelling of real world and becomes a promising tool to handle the impreciseness [5].

VBRR scheduler functions are fourfold. First, it addresses the impreciseness and uncertainty using vague logic. Second, it dynamically provides an optimum length of time quantum. Third, it reduces the unnecessary context switches which further reduce the scheduler overhead. Fourth, and the last, it improves the overall performance of system in terms of average response time, average waiting time, average turnaround time, and average normalized turnaround time.

VBRR scheduler works on 2-layered framework. First layer is of vague inference system which itself contains four units: Vague Logic Unit, Grade Function Unit, Data Base Unit, and *D*-Vague Logic Unit. First two functions of VBRR are performed at this layer only. Second layer runs scheduling algorithm to schedule the tasks. The latter two functions are performed at this particular layer.

The rest of paper is organized as follows. Section 2 outlines the related work with RR scheduler. Section 3 briefly describes the vague set theory and how it is a better tool over fuzzy set theory. Section 4 introduces the 2-layered framework for RR CPU scheduler. Section 5 presents the simulation and results to analyze the performance of VBRR scheduler with the traditional RR scheduler and two other fuzzy based CPU schedulers. Finally, Section 6 concludes the proposed work.

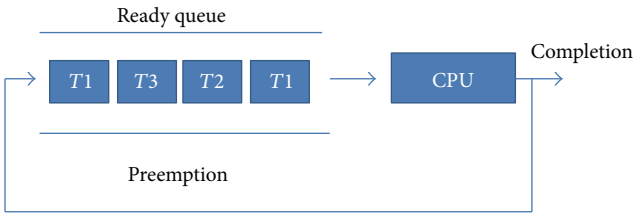

Figure 1: Round robin scheduling.

2. State of the Art

Round robin (RR) scheduling is specially designed for multitasking systems where users get an impression that multiple tasks are simultaneously running. It is the simplest, fairest, and most widely used scheduling algorithm by CPU scheduler [6]. A fixed size time quantum is assigned to each task in a ready queue and each task is treated equally as shown in Figure 1. Fixed size time quantum is a small portion of CPU time. Round robin scheduling is based on the First Come First Served (FCFS) scheduling. But FCFS scheduling is nonpreemptive type of scheduling whereas round robin is a preemptive scheduling. Therefore, the tasks are interrupted by the system after the completion of assigned time quantum and the system forces the task to temporarily suspend the execution. Later, task resumes its execution when it gets a time quantum.

RR scheduling algorithm improves the response time of tasks as compared to other scheduling algorithms. Racu et al. have analyzed the RR scheduling algorithm with respect to response time in 2007. They have derived a best case response time algorithm from the worst case response time algorithm [7].

RR scheduling algorithm provides an effective response time but with increased waiting time and turnaround time due to the fixed size time quantum. In 2008, Park et al. gave an approach of quantum based fixed priority scheduling [8]. They combined the concept of quantum with priority of tasks [9] in 2009; they further integrated the concept of preemption threshold with quantum based approach. Every time a task is dispatched to the CPU, a context switch occurs, which adds overhead to the task execution and increases the execution time. However, context switching is an essential feature of any multitasking system; it is the process of loading the states of one task in main memory and storing the states of another task. In RR scheduling, the context switch is the important concern as each task is preempted after the time quantum. It is intensive in computation as well as having the ability to increase the monetary value of the system in terms of processor time. Moreover, it can be costlier operation on an operating system [10]. Hence, there should be a focus on the designing of scheduling algorithm to avoid unnecessary context switches to the possible extent.

Number of context switches depends on the size of time quantum used. The smaller size quantum increases the context switches which will degrade the performance of system [6]. Matarneh in 2009 proposed a solution for the fixed size of time quantum. The proposed solution (SARR) adjusts the size of time quantum, based on the CPU burst value of each task [11]. The operating system is not able to adapt this solution immediately as system takes some time to observe the user behaviour, as the needed burst time for user is analyzed initially by operating system. Behera et al. have discussed the problem of high waiting time and turnaround time with RR scheduling algorithm due to the usage of static time quantum [12]. In 2010, they proposed a new version of RR scheduling and had given it the name Dynamic Time Quantum with Readjusted RR (DQRRR) scheduling algorithm. Mostafa et al. have also looked inside the problem of size of time quantum. In the same year, they introduced a method based on integer programming to decide the length of time quantum so that each task gets fair response time and also increases the throughput of the system by reducing unnecessarily context switches [13]. In 2011, Noon et al. provided a solution for fixed time quantum and called this new algorithm AN. The idea of AN algorithm was the same; the operating system itself adjusts the size of time quantum depending on the burst time of all tasks waiting in ready queue [14].

Thus, the system designer has an option to decide the size of time quantum by making it not too long and not too short. For example, consider a process whose CPU burst time is 6.2 ms and the fixed time quantum is 2 ms. After execution of 3 cycles, this task will be preempted and assigned to the ready queue. Task needs 0.2 ms more and if before preemption 0.2 ms is given to the process, it will complete its execution. Consequently, its waiting time may be reduced and a chance for another process to be in the queue [15]. The size of time quantum varies with respect to different parameters, namely, burst time, number of tasks present in ready queue, and so forth. These parameters can have imprecise or uncertain values. In order for intellect in a real world where things are not absolutely true, we need a different logic. To keep in line, Professor Zadeh has given fuzzy logic [16] during second half of the last century. Raheja et al. in 2012 have explored the fuzzy approach with RR scheduling. The authors have provided a fuzzy based solution to generate an optimum value for time quantum. The author calls this approach FBRR scheduling algorithm [15]. Alam in 2013 has also explored the option of using fuzzy logic with round robin scheduling algorithm to deal with imprecise values. He designed a Fuzzy Inference System (FIS) that decides the length of time quantum which is optimal for the system, so that each task receives a fair response time and throughput of the system. Otherwise, with fixed time quantum, the throughput of the system decreases due to the unnecessary context switches. The author calls this approach BFRR scheduling algorithm [10].

Fuzzy logic uses a concept of single membership value which cannot consider both the evidences of user: evidence in favor and evidence in against. Vague logic effectively handles the impreciseness using both the evidences which fuzzy logic cannot handle [5, 17]. This paper presents a new RR scheduler using the vague set theory. I have called this scheduler "Vague Logic Based Round Robin Scheduler" or VBRR scheduler. VBRR scheduler adapts dynamically the size of time quantum based on the current state of ready queue; thus it is a

kind of an intelligent scheduler which further improves the larger waiting time, turnaround time, and normalized turnaround time for tasks. VBRR scheduler also reduces the scheduling overhead by reducing the number of context switches. Finally, this work compares the performance of VBRR with traditional RR, FBRR, and BFRR approaches.

3. Vague Set Theory

A number of mathematicians and researchers have been providing a solution to deal with vagueness and impreciseness of knowledge. Professor Zadeh [16] had introduced a novel approach, that is, fuzzy logic. In fuzzy logic, each element has a single grade of membership in interval $[0, 1]$ [18]. There are a number of generalized forms of fuzzy set theory discussed in the literature by different authors. Atanassov [19] has introduced intuitionistic fuzzy set theory. He extended the fuzzy membership into two functions: membership and nonmembership in the same interval [19, 20]. Pawlak [21] had described another approach to impreciseness called rough set. In this approach, impreciseness is not expressed by single membership but expressed as a pair of sets which provides the lower and upper approximation of the set. Lower and upper approximations give the information about the elements undoubtedly belonging to set and the elements which possibly belong to set, respectively. Professor Gau and Buehrer [5] had introduced another approach over fuzzy set theory and called it vague set theory. They partitions the concept of single membership value $\mu_F(x)$ into two values: one represents favor value and another represents against value. They pointed out that fuzzy set theory cannot consider both against and favor values individually and even cannot use them at the same time. To differentiate the fuzzy set theory from vague set theory, let us consider $X = \{x_1, x_2, \ldots, x_n\}$ is the universe of discourse.

Definition 1 (fuzzy set). A fuzzy set F in X is defined as a pair where $\{x, \mu_F(x)\}$ where $\mu_F(x) \to [0, 1]$. For each object $x \in F$, the value $\mu_F(x)$ represents the degree of membership of object x in F as shown in Figure 2. If the value of $\mu_F(x)$ is more towards 1, the object x belongs more to the set [16].

Definition 2 (vague set). A vague set V in X is characterized by two membership functions: a truth membership function $t_v : X \in [0, 1]$ and a false-membership function $f_v : X \in [0, 1]$, where $t_v(x)$ is a lower bound of the grade of membership of x derived from the "evidence for x" and $f_v(x)$ is a lower bound on the opposition of x derived from the "evidence against x" as shown in Figure 3, and the total of these two independent membership values cannot exceed 1; that is, $t_v(x) + f_v(x) \leq 1$.

Definition 3 (vague value). The grade of membership of x in the vague set V is bounded by a subinterval $[t_v(x), 1 - f_v(x)]$ of $[0, 1]$, where the interval $[t_v(x), 1 - f_v(x)]$ represents the "vague value" of x in V [5].

3.1. How Are Both Set Theories Different from Each Other? Let X be the universe of discourse, say, the collection of burst time of active tasks in operating system. Let V be a vague set of all "high burst time tasks" of the universe X, and let F be a fuzzy set of "high burst time tasks" of X. Suppose an intelligent agent $I1$ suggests the membership value $\mu_F(x)$ for the element x in the fuzzy set F by his expert intellectual potentiality. On the contrary, another intelligent agent $I2$ suggests independently two membership values $t_v(x)$ and $f_v(x)$ for the same element x in the vague set V by his own expert intellectual potentiality. The $t_v(x)$ is degree of the true-membership value of x and $f_v(x)$ is the false-membership value of x in the vague set V. Both human agents $I1$ and $I2$ have their limitation of perception, assessment, and working power with real life situations. In the case of fuzzy set F, there is no further check for membership value $\mu_F(x)$. In the second case, the agent $I2$ suggests independently the membership values $t_v(x)$ and $f_v(x)$ and makes a further check by keeping the constraint, $t_v(x) + f_v(x) \leq 1$. If it is not satisfied, the agent can rethink and reshuffle his assessment.

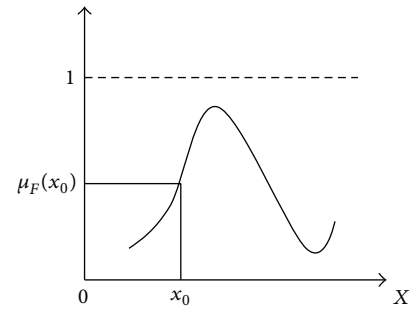

Figure 2: Fuzzy membership function.

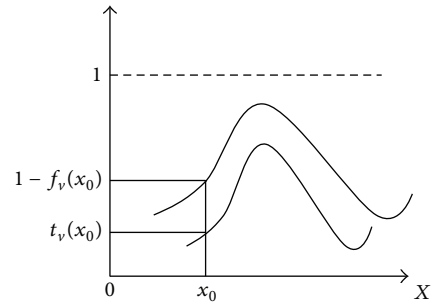

Figure 3: Vague membership functions.

4. VBRR CPU Scheduler

VBRR CPU scheduler works with 2-layered framework as shown in Figure 4. At the first layer, scheduler has vague inference system [22–25] which has further four modules. This layer basically handles the impreciseness and uncertainty of active tasks in a system. Moreover, it provides an optimum value for time quantum to the second layer of scheduler where scheduling algorithm works. VBRR scheduling algorithm schedules the tasks for a given time quantum and uses this time quantum to improve the performance of system.

Next sections describe both the layers in detail.

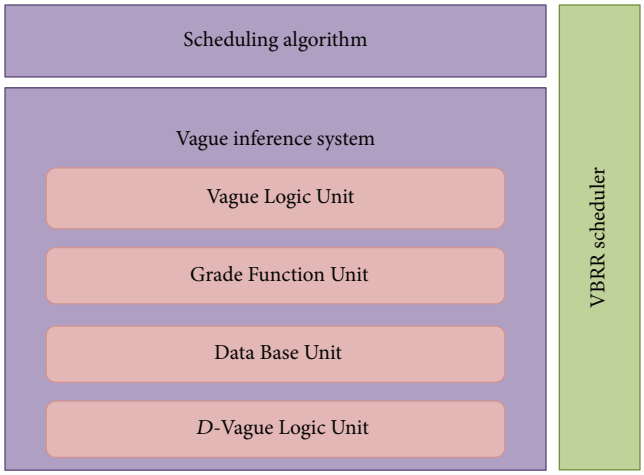

Figure 4: 2-layered framework for VBRR scheduler.

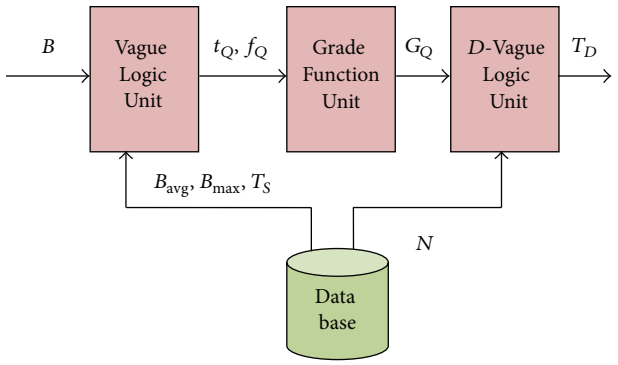

Figure 5: Vague inference system for optimum time quantum.

4.1. Vague Inference System (VIS). Our VIS contains four units as shown in Figure 5 as follows:

(i) Vague Logic Unit.

(ii) Grade Function Unit.

(iii) Data Base Unit.

(iv) *D*-Vague Logic Unit.

4.1.1. Vague Logic Unit (VLU). VLU takes the input burst time (B) from the database and performs the vaguification process [22]. Vaguification process maps the input parameters to the vague value: true-membership (t_Q) and false-membership (f_Q). It considers the present state of ready queue to deal with the uncertainty and impreciseness of tasks. When task arrives in the system, system is not able to exactly know the actual attributes of task; therefore, the VLU considers the maximum B_{max} and minimum B_{min} values of burst time present in ready queue to handle the exact deviation in current values. Moreover, VLU is also considering the static time quantum (t_S) which is initially assigned by the system. Based on all these values, our Vague Logic Unit computes t_Q using (1) and f_Q using (2):

$$t_Q = \frac{B_{avg}}{B_{avg} + T_S + B_{max}}, \tag{1}$$

$$f_Q = \frac{B_{avg}}{B_{avg} + T_S + B_{min}}. \tag{2}$$

Both t_Q and f_Q must satisfy the following axioms:

Axiom 1: $t_Q \leq 1$.

Axiom 2: $f_Q \leq 1$.

Axiom 3: $t_Q + f_Q \leq 1$.

Axiom 4: $0 \leq t_Q \leq 1 - f_Q \leq 1$.

4.1.2. Grade Function Unit (GFU). GFU evaluates the degree of accuracy of vague value. The Grade Function is applied to the membership values t_Q and f_Q which are received from the VLU, as given in

$$G_Q = t_Q + f_Q. \tag{3}$$

Here G_Q represents the grade value. Since, in vague set theory, the sum of these two functions cannot be greater than 1, grade value should also satisfy the following axiom as shown in Figure 6:

Axiom 5: $G_Q \leq 1$.

4.1.3. Data Base Unit. It stores all the related information about the tasks like burst time, arrival time, number of currently active tasks, maximum and minimum value of burst time, average burst time, and so forth. All other units extract the current required information from the database.

4.1.4. D-Vague Logic Unit. *D*-Vague Logic Unit finally maps the grade value to the output value, that is, optimum value of time quantum. The size of time quantum can be calculated using

$$T_D = (G_Q) \times \frac{(N + T_s)}{2}. \tag{4}$$

This dynamic time quantum depends on the grade value, static time quantum, and number of active tasks present in ready queue. It extracts the data from the database. In the next section, the author discusses how the 2nd layer uses this dynamic time quantum.

4.2. VBRR Scheduling Algorithm. In our proposed VBRR scheduling algorithm, the tasks are sorted according to their burst time, so that task with shortest burst time will be removed earlier from the ready queue. Optimum time quantum is calculated from the VIS defined in previous section and is used by all active tasks until the complete execution of them. The author discusses the algorithm in Algorithm 4 and the respective pseudocode for the algorithm in Pseudocode 1 and Boxes 1, 2, and 3.

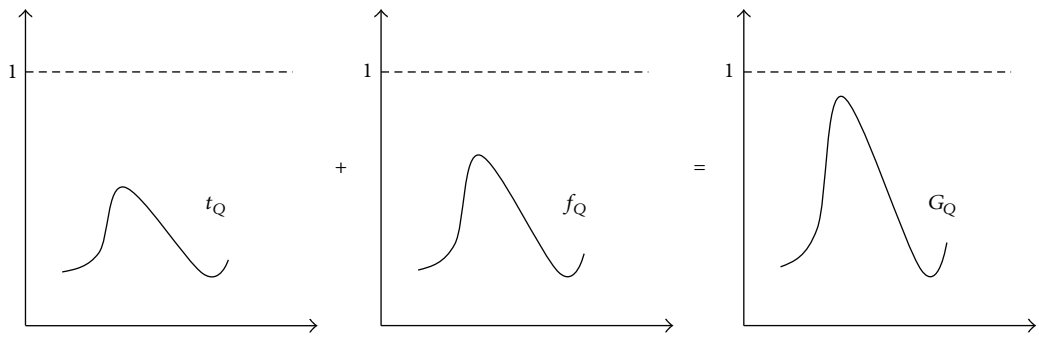

Figure 6: Grade value.

```
Begin
    Initialize the variables
        N = number of ready tasks
      T_S = Static Time Quantum
    Do Loop
        Initialize the variables
            B[N] = Burst Time of task
            A[N] = Arrival Time of task
    Until (N == 0)
    Initialize sum = 0
    Do Loop
        sum = sum + B[N]
    Until (N == 0)
    Calculate B_avg
      B_avg = sum/N
    for (i = 1 to N)
        B_max = max(B[i])
        B_min = min(B[i])
    endfor
    Calculate G_Q using true-membership and false-membership function
        t_Q = True_membership (B_avg, T_S, B_max)
        f_Q = False_membership (B_avg, B_min, T_S)
        G_Q = t_Q + f_Q
    Calculate T_D
        T_D = (G_Q) × (N + T_S)/2
    Do Loop
        Schedule the task with CPU for T_D time
        if a new task has arrived
            Recalculate the T_D
        else
            continue with execution
        endif
    Until (N == 0)
    Calculate waiting time W_time, response time R_time, turnaround time T_time and normalized turnaround time N_time.
    Avg_Calc( )
End
```

Pseudocode 1

Algorithm 4. (1) Initialize the variables B_{avg}, B_{max}, and B_{min} with the average burst time, maximum burst time, and minimum burst time present in ready queue.

(2) Initialize the variables N and T_S.

(3) Sort the ready queue in increasing order of burst time.

(4) Compute the time quantum T_D from the VIS (Section 4.1).

(5) Dispatch and execute the tasks according to the round robin scheduling algorithm using the computed T_D until the ready queue is empty.

True_membership (B_{avg}, T_S, B_{max})
Calculate true-membership value t_Q
$$t_Q = \frac{B_{avg}}{(B_{avg} + T_S + B_{max})}$$
return (t_Q)
endfunction

Box 1

False_membership (B_{avg}, B_{min}, T_S)
Calculate false-membership value f_Q
$$f_Q = \frac{B_{avg}}{(B_{avg} + T_S + B_{min})}$$
return (f_Q)
endfunction

Box 2

(6) If a new task arrived, repeat steps from (1) to (5).

(7) Calculate the waiting time, response time, turnaround time, and normalized turnaround time for each task.

(8) Finally calculate the average waiting time (AWT), average response time (ART), average turnaround time (ATT), average normalized turnaround time (ATT), and the total context switches (NCS).

5. Simulation and Results

The author has analyzed the performance of proposed work with the RR, FBRR, and BFRR scheduling algorithms. For each comparison, selected sets of tasks were considered and for each case the performance metrics are computed. Results prove that VBRR has better performance as compared to other three algorithms. She presents a Sample Task Set to show the detailed performance analyses of all scheduling algorithms with the help of Gantt chart.

5.1. Sample Task Set. Assume a set of five tasks with burst time ($T1 = 20$, $T2 = 20$, $T3 = 5$, $T4 = 3$, $T5 = 1$). For simplicity, the arrival time of all tasks is considered as 0 ms.

5.1.1. Using RR Scheduling. Assume the static time quantum T_S assigned by system as 2 ms. The scheduling order of tasks using $T_S = 2$ ms is shown in Figure 7(a).

5.1.2. Using FBRR Scheduling Algorithm. Similarly, the size of time quantum for the Sample Task Set after applying the FBRR scheduling approach is TQ = 2.6 ms. The scheduling order of tasks with FBRR algorithm is shown in Figure 7(b).

5.1.3. Using BFRR Scheduling Algorithm. After applying BFRR scheduling algorithm, the size of time quantum for the Sample Task Set is TQ = 2.5 ms. The scheduling order of tasks with BFRR algorithm is shown in Figure 7(c).

5.1.4. Using VBRR Scheduling Algorithm. VBRR algorithm firstly initializes the following variables:

$$\begin{aligned} B_{max} &= 20, \\ B_{min} &= 1, \\ B_{avg} &= 9.8. \end{aligned} \tag{5}$$

Then, VIS returns the true- and false-membership functions as given below:

$$\begin{aligned} t_Q &= \frac{9.8}{31.8} = 0.3 \\ f_Q &= \frac{9.8}{12.8} = 0.7. \end{aligned} \tag{6}$$

After that, it computes G_Q as shown in Figure 8 by using t_Q and f_Q:

$$G_Q = t_Q + f_Q = 1. \tag{7}$$

Then, the time quantum t_D is calculated as

$$t_D = 1 * \frac{(5 + 2)}{2} = 3.5 \text{ ms}. \tag{8}$$

The tasks are scheduled with $t_D = 3.5$ ms. Scheduling order of tasks is shown in Figure 9.

5.2. Performance Metrics. To evaluate the performance of VBRR scheduler, the author has chosen all the major performance metrics as discussed below.

Definition 5 (average waiting time). Waiting time is the total time a task waits in ready queue for its turn of execution. Let us consider the time, when task has arrived in the system as arrival time and when task has got the CPU as start time. Waiting time can be calculated as

$$W_{time} = \text{start time} - \text{arrival time}. \tag{9}$$

For RR scheduling, the task has preempted and resumed the execution multiple times until the task has completed its execution. Then, for further turns, start time acts as next starting time and arrival time as the last time when the task has been executed. The total waiting time is calculated by adding all the waiting times of a task using (9):

$$\begin{aligned} W_{time} = {} & (\text{start time} - \text{arrival time}) \\ & + (\text{start time} - \text{last execution time}) + \cdots . \end{aligned} \tag{10}$$

Average waiting time is calculated as the waiting time of all the active tasks in ready queue using (10). It is calculated as

$$\text{AWT} = \frac{\left(\sum_{i=1}^{N} W_{time}\right)}{N}. \tag{11}$$

Here, N is the number of active tasks. The performance of CPU scheduler depends on the average waiting time. The

Avg_Calc ( )

Calculate average waiting time (AWT)

$$\text{AWT} = \sum_{i=1}^{N} \frac{W_{\text{time}}}{N}$$

Calculate average response time (ART)

$$\text{ART} = \sum_{i=1}^{N} \frac{R_{\text{time}}}{N}$$

Calculate average turnaround time (ATT)

$$\text{ATT} = \sum_{i=1}^{N} \frac{T_{\text{time}}}{N}$$

Calculate average normalized turnaround time (ANT)

$$\text{ANT} = \sum_{i=1}^{N} \frac{N_{\text{time}}}{N}$$

endfunction

Box 3

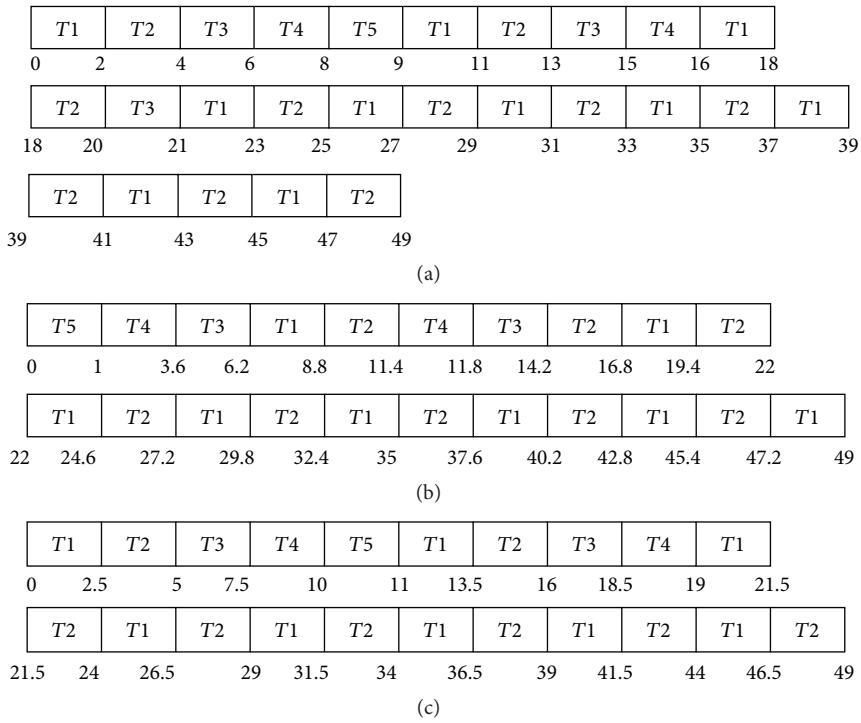

Figure 7: (a) Gantt chart using RR approach. (b) Gantt chart using FBRR scheduling algorithm. (c) Gantt chart using BFRR scheduling algorithm.

reduction in AWT is considered as performance improvement of scheduler.

Definition 6 (average response time). Response time is the first start time of task from its arrival to the system. It represents the time when first time task has got the CPU. It is calculated as

$$R_{\text{time}} = \text{start time} - \text{arrival time}. \tag{12}$$

For the nonpreemptive scheduling algorithms, both the waiting time and response time return the same results. Average response time is calculated as the response time of all the active tasks using (12). It is calculated as

$$\text{ART} = \frac{\left(\sum_{i=1}^{N} R_{\text{time}}\right)}{N}. \tag{13}$$

Definition 7 (average turnaround time). Turnaround time is the total life time of a task which means the total time a task takes to finish its execution. It can be calculated as

$$T_{\text{time}} = \text{burst time} + \text{waiting time}. \tag{14}$$

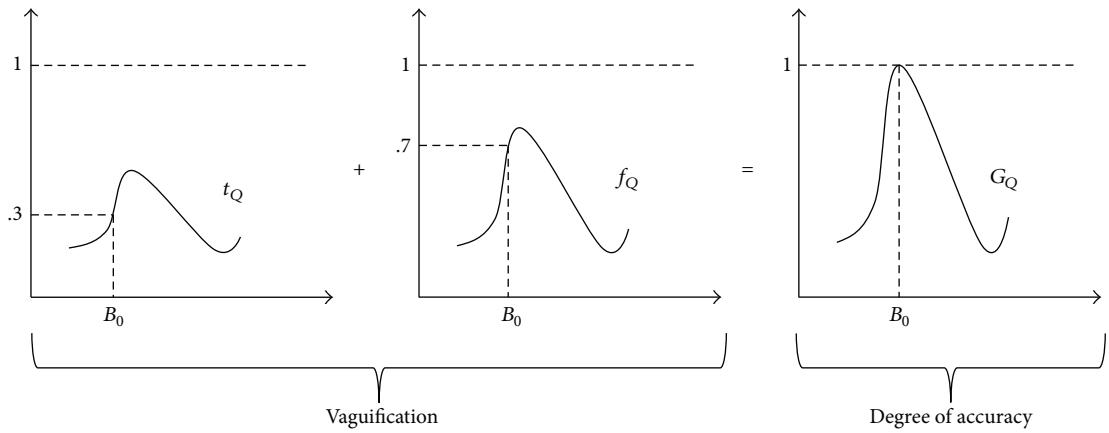

Figure 8: Vaguification to degree of accuracy (Sample Task Set).

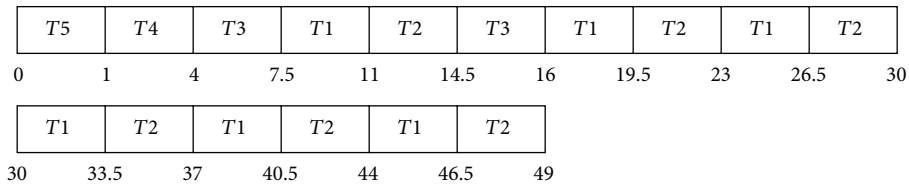

Figure 9: Gantt chart using VBRR scheduling algorithm.

Here, burst time represents the time task needs CPU and waiting time is calculated using (9).

Average turnaround time is calculated as the turnaround time of all the active tasks in ready queue using (14). It is calculated as

$$\text{ATT} = \frac{\left(\sum_{i=1}^{N} T_{\text{time}}\right)}{N}. \tag{15}$$

Definition 8 (average normalized turnaround time). Normalized turnaround time represents the relative delay of a task. It is simply a ratio of turnaround time to burst time. It is calculated as

$$N_{\text{time}} = \frac{T_{\text{time}}}{\text{burst time}}. \tag{16}$$

Average normalized turnaround time is calculated as the normalized turnaround time of all the active tasks in ready queue using (16). It is calculated as

$$\text{ANT} = \frac{\left(\sum_{i=1}^{N} N_{\text{time}}\right)}{N}. \tag{17}$$

Definition 9 (context switch). Context switch is a mechanism that is followed by the CPU to switch from one task to another task in a multitasking system. It saves and loads the computer registers. It allows multiple tasks to run; basically it shares a CPU among multiple tasks. Context switching is computationally intensive and does not perform any useful task, so the focus of designers is to optimize the number of context switches.

From the Gantt charts shown in Figures 7 and 9, the performance metrics of each scheduling algorithms are calculated using (11), (13), (15), and (17).

All the graphs for Sample Task Set shown in Figure 10 illustrate the reduction in the values of performance metrics from the RR to VBRR scheduling. In Figure 10(f), on the x axis of graph, "1" represents RR scheduler, "2" represents BFRR scheduler, "3" represents FBRR scheduler, and "4" represents VBRR scheduler. From the graph, we can compare the performance of these four algorithms. There is a reduction in performance values at "4" which proves the improvement in performance of VBRR as compared to other three algorithms. Hence, VBRR scheduler has better performance.

This paper considers multiple task sets to better evaluate the performance of VBRR scheduler over other three approaches. In this work, the author is illustrating the performance with the help of results of 15 task sets.

Figures 11, 12, 13, 14, and 15 show average waiting time, average response time, average turnaround time, average normalized turnaround time, and number of context switches, respectively. VBRR algorihm is having lowest values in case of all performance metrices. From these graphs, we can conclude that the VBRR algorithm has better performance over the other three algorithms: RR, FBRR, and BFRR algorithms.

VBRR scheduler improves the performance of CPU scheduler mainly by three reasons. Firstly, it deals with the associated impreciseness and uncertainity with task. Secondly, it provides a dynamic environment of execution for tasks by assigning a dynamic time quantum. Thirdly, it improves the performance of system by reducing the average

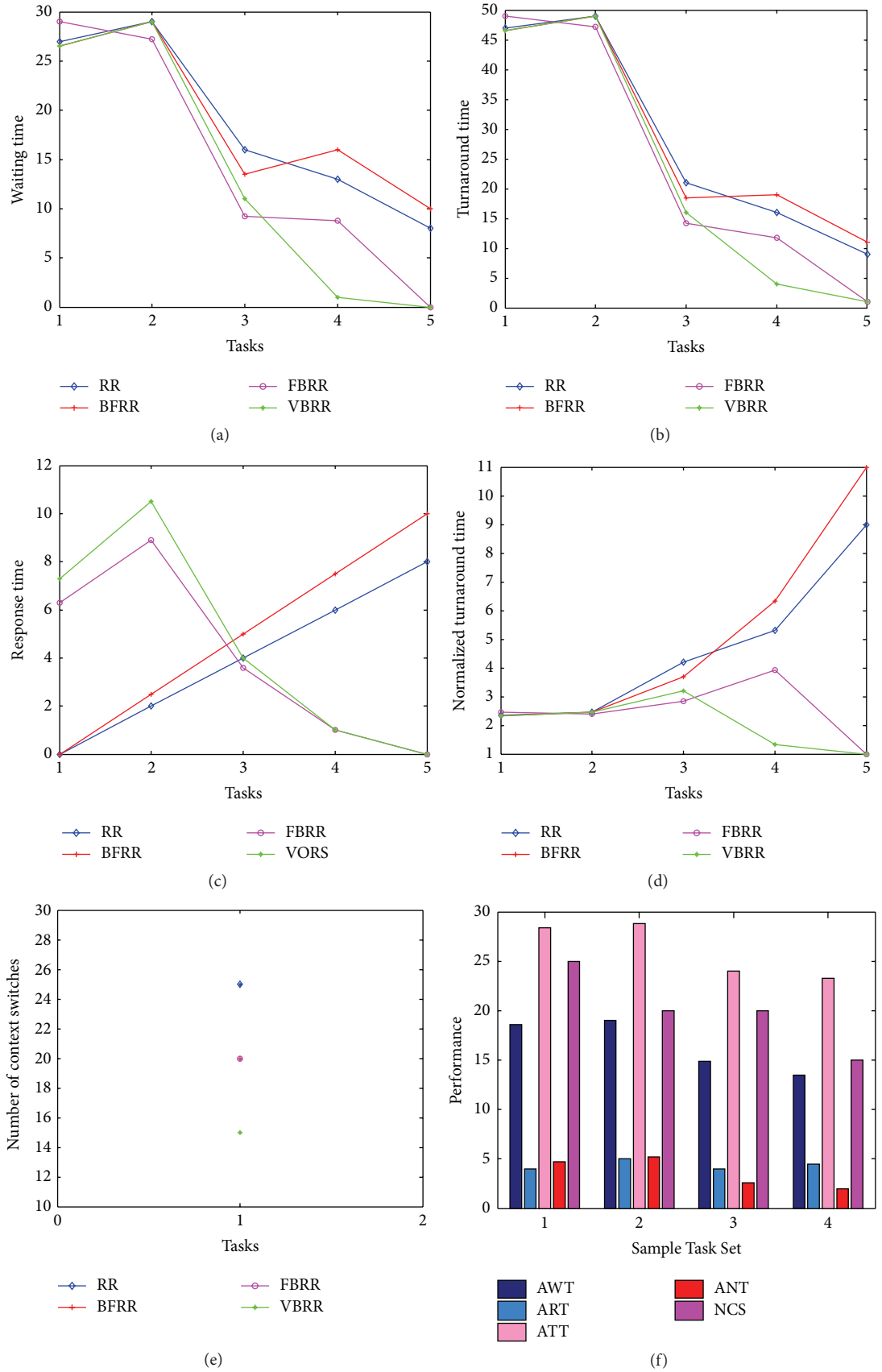

FIGURE 10: (a) Waiting time (Sample Task Set). (b) Turnaround time (Sample Task Set). (c) Response time (Sample Task Set). (d) Normalized turnaround time (Sample Task Set). (e) Context switches (Sample Task Set). (f) Overall performance (Sample Task Set).

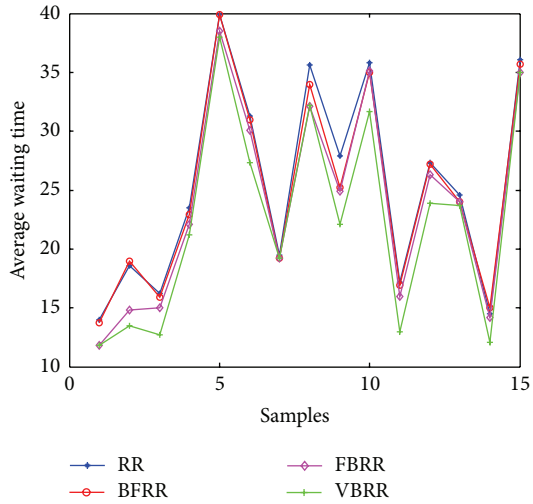

Figure 11: Average waiting time.

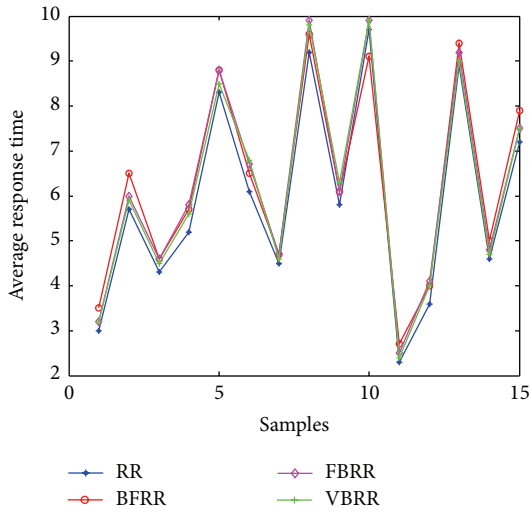

Figure 12: Average response time.

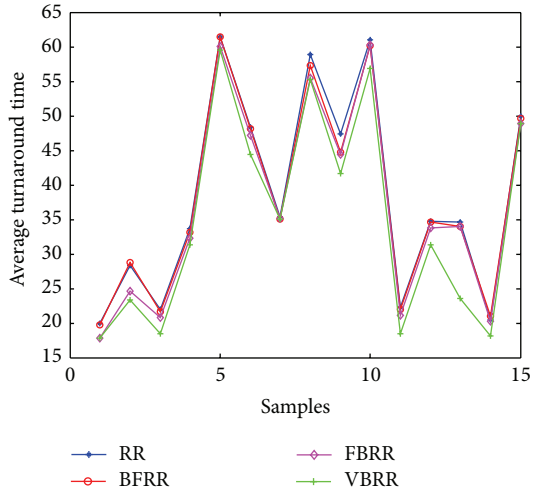

Figure 13: Average turnaround time.

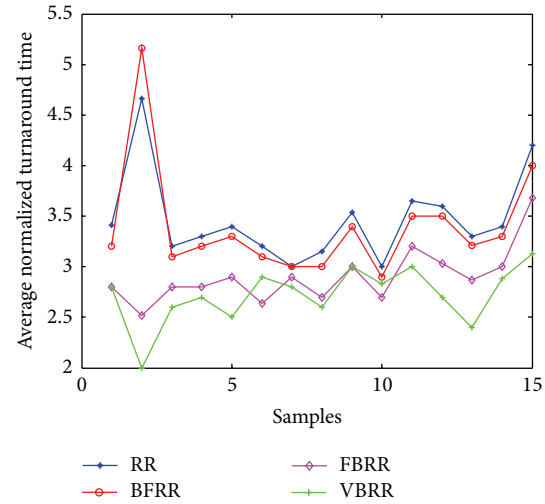

Figure 14: Average normalized turnaround time.

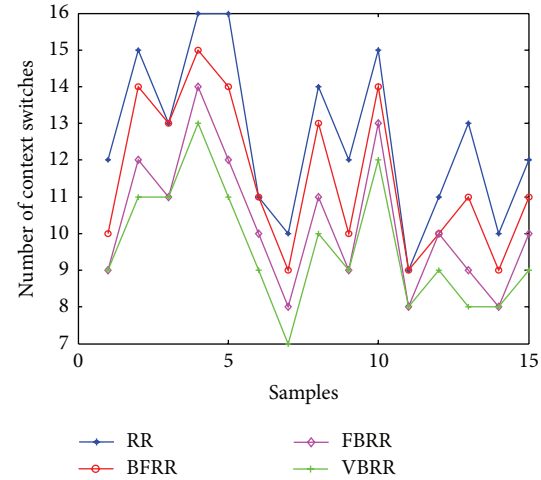

Figure 15: Number of context switches.

waiting time, average turnaround time, average normalized turn aroound time, and number of context switches.

6. Conclusion

As discussed above, round robin scheduling algorithm improves the response time of the tasks, but with increased waiting time and turnaround time. Moreover, the size of time quantum plays the key role in the performance of RR algorithm. Smaller size can increase the number of context switches that makes the performance of scheduler even worse. There must be a technique inside CPU scheduler, which allows it to dynamically calculate the size of time quantum rather than a fixed size time quantum. This work provides a solution to all the above issues with RR scheduling. The author has introduced a Vague Logic Based Round Robin CPU scheduler. At first layer of VBRR, a vague inference system is designed which dealt with the imprecise parameters of task like burst time, static time quantum by considering the current state of active tasks. Based on the current

situation, this vague inference system finally generated an optimum value for time quantum which makes the scheduler work in a dynamic environment. At the second layer, a Vague Logic Based Round Robin scheduling algorithm is introduced to schedule the tasks. The author has evaluated the performance of VBRR algorithm over multiple task sets and compared the performance with RR, BFRR, and FBRR scheduling algorithms. Simulation results illustrate that the VBRR scheduler improves the performance in terms of average waiting time, average turnaround time, and average normalized turnaround time by maintaining the average response time. It consequently improves the performance of the system by reducing the number of context switches.

Competing Interests

The author declares that she has no competing interests.

References

[1] A. S. Tanenbaum and A. S. Woodfhull, *Operating Systems Design and Implementation*, PHI, 3rd edition, 2006.

[2] J. Blazewicz, K. H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz, *Scheduling Computer and Manufacturing Processes*, Springer, Berlin, Germany, 2000.

[3] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating Systems Concepts*, John Wiley & Sons, 8th edition, 2009.

[4] A. A. Aburas and V. Miho, "Fuzzy logic based algorithm for uniprocessor scheduling," in *Proceedings of the IEEE International Conference on Computer and Communication Engineering*, pp. 499–504, Kuala Lumpur, Malaysia, May 2008.

[5] W.-L. Gau and D. J. Buehrer, "Vague sets," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no. 2, pp. 610–614, 1993.

[6] B. Caprita, W. C. Chan, J. Nieth, C. Stein, and H. Zheng, "Group Ratio Round-Robin: O(1) proportional 18 share scheduling for uniprocessor and multiprocessor systems," in *Proceedings of the USENIX Annual Technical Conference (ATEC '05)*, p. 36, 2005.

[7] R. Racu, L. Li, R. Henia, A. Hamann, and R. Ernst, "Improved response time analysis of tasks scheduled under preemptive round-robin," in *Proceedings of the 5th International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS '07)*, pp. 179–184, ACM, October 2007.

[8] M. Park, H. J. Yoo, J. Chae, and C.-K. Kim, "Quantum-based fixed priority scheduling," in *Proceedings of the International Conference on Advanced Computer Theory and Engineering (ICACTE '08)*, pp. 64–68, IEEE, Phuket, Thailand, December 2008.

[9] M. Park, H. J. Yoo, and J. Chae, "Integration of preemption threshold and quantum-based scheduling for schedulability enhancement of fixed priority tasks," in *Proceedings of the 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA '09)*, pp. 503–510, Beijing, China, August 2009.

[10] B. Alam, "Fuzzy round robin CPU scheduling algorithm," *Journal of Computer Science*, vol. 9, no. 8, pp. 1079–1085, 2013.

[11] R. J. Matarneh, "Self-adjustment time quantum in round robin algorithm depending on burst time of the now running processes," *American Journal of Applied Sciences*, vol. 6, no. 10, pp. 1831–1837, 2009.

[12] H. S. Behera, R. Mohanty, and D. Nayak, "A new proposed dynamic quantum with re-adjusted round robin scheduling algorithm and its performance analysis," *International Journal of Computer Applications*, vol. 5, no. 5, pp. 10–15, 2010.

[13] S. M. Mostafa, S. Z. Rida, and S. H. Hamad, "Finding time quantum of round robin CPU scheduling algorithm in general computing systems using integer programming," *International Journal of Research and Reviews in Applied Sciences*, vol. 5, no. 1, pp. 64–71, 2010.

[14] A. Noon, A. Kalakech, and S. Kadry, "A new round robin based scheduling algorithm for operating systems: dynamic quantum using the mean average," *IJCSI International Journal of Computer Science Issues*, vol. 8, no. 3, pp. 224–229, 2011.

[15] S. Raheja, R. Dadhich, and S. Rajpal, "An optimum time quantum using linguistic synthesis for round robin cpu scheduling algorithm," *International Journal on Soft Computing*, vol. 3, no. 1, pp. 57–66, 2012.

[16] L. A. Zadeh, "Fuzzy sets," *Information and Computation*, vol. 8, pp. 338–353, 1965.

[17] H. Bustince and P. Burillo, "Vague sets are intuitionistic fuzzy sets," *Fuzzy Sets and Systems*, vol. 79, no. 3, pp. 403–405, 1996.

[18] H.-J. Zimmermann, *Fuzzy Set Theory And Its Applications*, Kluwer Academic, Norwell, Mass, USA, 2001.

[19] K. T. Atanassov, "Intuitionistic fuzzy sets," *Fuzzy Sets and Systems*, vol. 20, no. 1, pp. 87–96, 1986.

[20] K. Atanassov, *Intuitionistic Fuzzy Sets: Theory and Applications*, Physica-Springer, New York, NY, USA, 2000.

[21] Z. Pawlak, "Rough sets," *International Journal of Computer & Information Sciences*, vol. 11, no. 5, pp. 341–356, 1982.

[22] S. Raheja, R. Dhadich, and S. Rajpal, "Many valued logics for modeling vagueness," *International Journal of Computer Applications*, vol. 61, no. 7, pp. 35–39, 2013.

[23] S. Raheja, R. Dhadich, and S. Rajpal, "Designing of 2-stage CPU scheduler using vague logic," *Advances in Fuzzy Systems*, vol. 2014, Article ID 841976, 10 pages, 2014.

[24] S. Raheja, R. Dhadich, and S. Rajpal, "2-Layered architecture of vague logic based multilevel queue scheduler," *Applied Computational Intelligence and Soft Computing*, vol. 2014, Article ID 341957, 12 pages, 2014.

[25] S. Raheja, R. Dadhich, and S. Rajpal, "Designing of vague logic based fair-share CPU scheduler: VFS CPU scheduler," *International Journal of Fuzzy System Applications*, vol. 4, no. 3, pp. 25–49, 2015.