Hindawi Publishing Corporation
Advances in Fuzzy Systems
Volume 2014, Article ID 841976, 10 pages
http://dx.doi.org/10.1155/2014/841976

Research Article

Designing of 2-Stage CPU Scheduler Using Vague Logic

Supriya Raheja,[1] Reena Dhadich,[2] and Smita Rajpal[3]

[1] *Department of Computer Science & Engineering, ITM University, Gurgaon, India*
[2] *Department of Computer Science, University of Kota, Rajasthan, India*
[3] *Alpha Global IT, Toronto, ON, Canada*

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

Received 15 January 2014; Revised 29 April 2014; Accepted 25 June 2014; Published 22 July 2014

Academic Editor: Adel M. Alimi

In operating system the CPU scheduler is designed in such a way that all the resources are fully utilized. With static priority scheduling the scheduler ensures that CPU time will be assigned according to the highest priority but ignores other factors; hence it affects the performance. To improve the performance, we propose a new 2-stage vague logic based scheduler. In first stage, scheduler handles the uncertainty of tasks using the proposed vague inference system (VIS). In second stage, scheduler uses a vague oriented priority scheduling (VOPS) algorithm for selection of next process. The goal of this work is to handle the uncertainty as well as to optimize both the average and the amount of variation with respect to performance matrices average waiting time, average turnaround time, and average normalized turnaround time. A simulation using MATLAB is also conducted to evaluate the performance. Simulation results show that the proposed scheduler using VOPS algorithm is better than the scheduler with traditional priority scheduling algorithm. Results are based on the dual concept of fuzzy theory and its generalization, vague theory. Additionally, this work comprises the evaluation of VOPS and shortest job first algorithm. The outcome of proposed VOPS algorithm is much closer to the result obtained by traditional shortest job first.

1. Introduction

CPU Scheduler is the main module of any operating system, as it selects the next task to be run. It runs a scheduling algorithm for selection of task. When more than one task is ready to execute, scheduling algorithms will decide which one is to run first. Hence, the designer of scheduler must consider the scheduling algorithm which provides effective throughput.

Operating system is not capable to know the exact attributes values of the task, namely, execution time and so forth. Recent developments in priority scheduling have made significantly enhancements to the models considering uncertainty factors. These models accommodate the attributes of tasks with Fuzzy logic.

This paper is written with the aim of focusing on the uncertainty and impreciseness of attributes with vague logic and generalized form of fuzzy logic. The vague set theory improves the modeling of real world, becoming a committing tool to deal with imprecise knowledge. In addition to vague implementation, second aim is to improve the performance of priority scheduling algorithm. In this paper we are proposing a new 2-stage scheduler with VOPS (vague oriented priority scheduling) algorithm. Proposed CPU Scheduler works in two stages; for the first stage we are introducing a vague inference system to generate dynamic priorities for tasks by considering the impreciseness of attributes and for second stage of scheduler we are projecting a new vague oriented priority scheduling algorithm to schedule the next process to CPU.

The paper is organized as follows. Section 2 outlines the related work to the priority scheduling. Section 3 considers the necessary characteristics of vague set theory and the relationship between vague member function. Section 4 includes the proposed scheduler. In Section 5, VOPS is compared with the other algorithms. Finally, Section 6 summarizes the conclusion.

2. Related Work

Priority scheduling algorithm is used by scheduler when all tasks are not equally important [1]. In the traditional priority scheduling algorithm, a fixed priority is assigned to each task by system. We call it as a static priority since it is assigned at once and does not change. At each scheduling event, the ready queue is sorted according to priority. It prescribes an environment in which the highest priority is scheduled to the CPU for execution. Tasks of equal priority are scheduled as first come first serve (FCFS). When the task with higher priority comes, the scheduled task is preempted and the new arrive task is scheduled to the CPU [2, 3]. The concept of static priority comprises uncertainty and impreciseness as it does not consider the current state of tasks in the ready queue.

To consider the current state of the ready queue, the dynamic priority concept over static priority has introduced that proved the better performance [4, 5]. Dynamic priority means the priority of the tasks varies based on different factors, namely, CPU burst time, waiting time, and response ratio at different levels. So far, the parameters used are crisp. These algorithms did not consider uncertainties and impreciseness, like the number of tasks in the ready queue, task with highest priority having maximum burst time which can starve the tasks with lower burst time, and so forth.

Fuzzy logic was introduced by Professor Zadeh in his seminal paper and provides the basis for further development [6, 7]. This imprecise information can be encountered in the scheduling algorithms. To handle the impreciseness or uncertainty in scheduling algorithms fuzzy based priority scheduling algorithms has been proposed [8, 9]. These algorithms have used the concept of fuzzy logic to solve the shortcoming of traditional Priority scheduling algorithm. Undoubtedly, the fuzzy based priority scheduling algorithms improves the performance of static priority scheduling algorithm [9]. Our present work is based on vague set theory which is the further generalization of fuzzy set theory. It is undoubtedly true that the dynamic priority with vague set theory based scheduling is more complicated than the static priority scheduling, although the difference is not as much as it appears.

The important aspect of this paper is to generate dynamic priorities for the tasks as it is necessary for the system to consider all the attributes to facilitate dynamic priority. Here system considers two attributes: burst time of the tasks and the static priority. For this we are proposing a 2-stage scheduler using "vague oriented priority scheduling (VOPS) algorithm." We claim that the proposed algorithm improves the average waiting time, average turnaround time, and average normalized turnaround time.

In next section we will discuss preliminaries of vague set theory which is essential to define the proposed work.

3. Vague Set Theory

Professor Zadeh had changed the approach in the field of logics by proposing a novel fuzzy logic in 1965 where each element in fuzzy set has a single membership value in between 0 and 1 [6].

Definition 1 (fuzzy set). Let $X = \{x_1, x_2 \ldots x_n\}$ be the universe of discourse. A fuzzy set A in X is defined as the set of ordered pairs $A = \{x, \mu_A(x) : x \in X\}$, where $\mu_A(x)$ is the grade of membership of element x in the set A [6, 10, 11]. The greater $\mu_A(x)$, the greater "element x belongs to the set A."

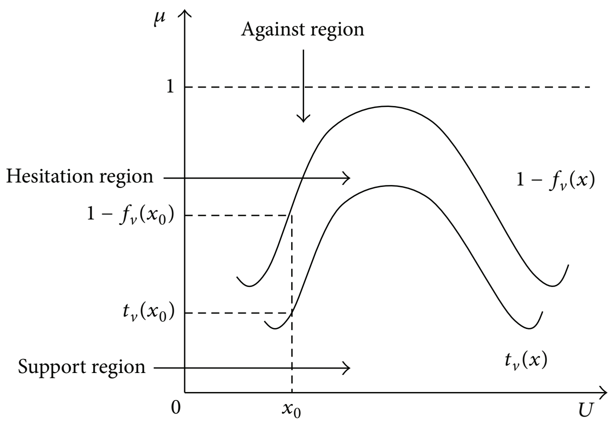

Figure 1: Vague membership function.

Gau and Buehrer [12] had pointed this single membership value $\mu_A(x)$ as it combines the evidences for both favor and against value of x. Fuzzy set theory cannot address the two evidences individually, even cannot address the two evidences at the same duration. Gau and Buehrer had introduced the concept of vague set theory [12] over fuzzy set theory.

Definition 2 (vague set). A vague set V in the universe of discourse X is characterized by two membership functions:

(1) a truth membership function t_v:$X \in [0, 1]$,

(2) 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" [12]. The total value of these two independent functions cannot exceed 1, that is, $t_v(x) + f_v(x) \leq 1$. The grade of membership of x in the vague set V is bounded by a subinterval $[t_v(x), 1 - f_v(x)]$ of $[0, 1]$.

Definition 3 (vague value). The vague set V is written as $V = \langle x, [t_v(x), f_v(x)] \rangle : x \in X$, where the interval $[t_v(x), 1 - f_v(x)]$ is called the "vague value" of x in V [12].

For example, consider a universe X for burst time of the tasks. If the vague value for the burst time of a task is $[0.7, 0.8]$, means support value (t_v) is 0.7, against value (f_v) is 0.2 and 0.1 comes under the hesitated value means not in support and not in the against as shown in Figure 1.

Definition 4 (median membership function). Median membership represents the overall evidence contained in a vague value. It is defined as $M_M = (t_v + 1 - f_v)/2$ having constraint $0 \leq (t_v + 1 - f_v)/2 \leq 1$ as shown in Figure 2.

The vague value $[1, 1]$ has the highest M_M, which means the corresponding element wholly belongs to the vague set. While the vague value $[0, 0]$ has the lowest M_M means that

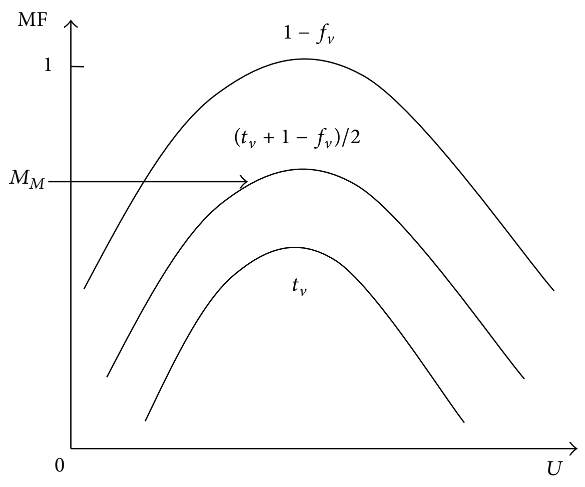

Figure 2: Median membership function.

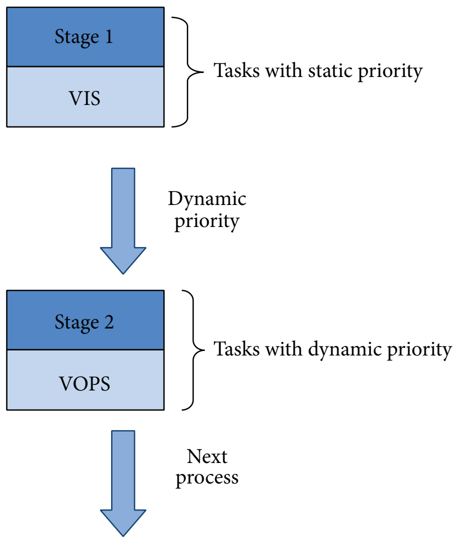

Figure 3: 2-Stage scheduler.

the corresponding element does not belong to the vague set [13–16].

3.1. How Vague Logic Is Different from Fuzzy Logic? Let X be a universe of discourse; say the collection of burst time of tasks in the ready queue of the operating system. Let V be a vague set of all "high burst time task" of the universe X, and let F be a fuzzy set of all "high burst time task" of X. Suppose an intelligent agent I_1 suggests the membership value $\mu_F(x)$ for the element x in the fuzzy set F by his expert knowledge. On the adverse, another intelligent agent I_2 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 knowledge. 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 I_1 and I_2 have their limitation of perception, assessment, working ability 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 I_2 suggests independently the membership values $t_v(x)$ and $f_v(x)$, but makes a further check by keeping the constraint, $t_v(x) + f_v(x) \leq 1$. If it is not satisfied, the agent can change his assessment [16].

4. Vague Logic Based 2-Stage CPU Scheduler

The proposed vague logic based scheduler has the ability of learning and keeping track of tasks. Based on this capability scheduler adjusts their priorities periodically. For example, if a new task T2 with higher static priority and maximum burst time arrives in the RQ and task T1 with slightly lower priority but minimum burst time is already in the RQ. Then the traditional scheduler will schedule the task T2 based on highest priority but our scheduler will recomputed the priority based on the current values of burst time and static priority and schedules accordingly.

Proposed scheduler works in two stages as shown in Figure 3; in first stage vague Inference system (VIS) calculates the dynamic priority for each task and in second stage VOPS algorithm selects the highest priority task to be executed.

4.1. Vague Inference System (VIS). Our considerations are limited to a single processor system that supports a fixed number of tasks N. Each of the N tasks requires B burst time (execution time) and system assign P static priority to these tasks. Each task is independent of each other.

Let N tasks be $\{T_i \mid i = 1, \ldots, N\}$, then

$$\begin{aligned}
&B_i = \text{burst time of ith task}\\
&B_{\max} = \max\{B_1, B_2, \ldots, B_N\}\\
&B_{\min} = \min\{B_1, B_2, \ldots, B_N\}\\
&\text{Likewise } P_i = \text{static priority of ith task}\\
&P_{\max} = \max\{P_1, P_2, \ldots, P_N\}\\
&P_{\min} = \min\{P_1, P_2, \ldots, P_N\},
\end{aligned}$$

where $B_{\max}$ is the maximum burst time; $B_{\min}$ is the minimum burst time; RQ is the ready queue; $P_{\max}$ is the maximum priority; $P_{\min}$ is the minimum priority available in ready queue.

Proposed VIS has two units, vague logic unit (VLU) and the median membership function unit (MFU) as shown in Figure 4.

4.1.1. Vague Logic Unit (VLU). The idea of pairing the membership (true-membership) and nonmembership (false-membership) values in VLU from vague set theory are to represent the imprecise or uncertain information of tasks discussed in Section 2. VLU takes the input as crisp data and converts the crisp data into the vague data $[t_v, 1 - f_v]$, t_v true-membership function, and f_v false-membership function. These two functions are used to describe the boundaries of membership value. It takes two crisp inputs burst time and static priority as shown in Figure 4. As scheduler tracks the tasks, the unit VLU considers the current state of tasks in RQ

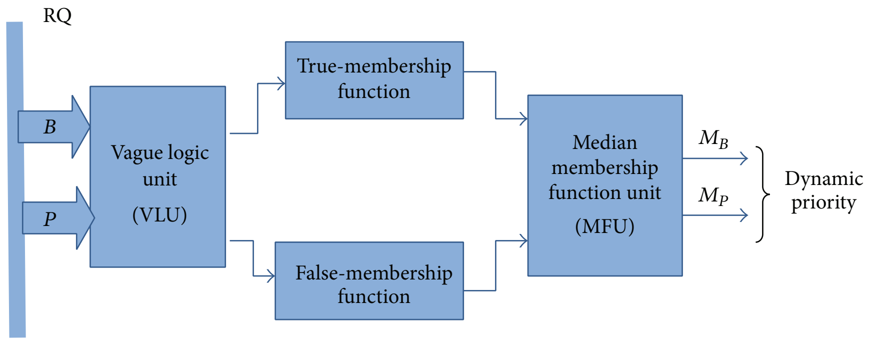

FIGURE 4: Vague inference system.

like the membership degree with respect to maximum and minimum burst time. True and false membership functions t_B and f_B for burst time are computed using (1) and (2), respectively. Consider

$$t_B = \frac{B_{\max} - B + 1}{B_{\max} + 1}, \tag{1}$$

$$f_B = \frac{B - B_{\min}}{B_{\max} + 1}, \qquad t_B + f_B \leq 1. \tag{2}$$

By combining the two equations, (1) and (2), into (3), we get the vague data for burst time of task. Consider

$$[t_B, 1 - f_B] = \left[\frac{B_{\max} - B + 1}{B_{\max} + 1}, 1 - \frac{B - B_{\min}}{B_{\max} + 1}\right]. \tag{3}$$

Computation for member functions for static priority t_P and f_P is given in (4). Consider

$$\begin{aligned} t_P &= \frac{P_{\max} - P}{P_{\max} + 1}, \\ f_P &= \frac{P - P_{\min}}{P_{\max}}, \qquad t_P + f_P \leq 1. \end{aligned} \tag{4}$$

Similarly, by combining (4), we get the vague data for static priority of task as in (5). Consider

$$[t_P, 1 - f_P] = \left[\frac{P_{\max} - P}{P_{\max} + 1}, 1 - \frac{P - P_{\min}}{P_{\max}}\right]. \tag{5}$$

4.1.2. Median-Membership Function Unit (MFU). The output from the VLU is passed as the input to the second unit MFU. MFU computes the aggregate truth value enclosed in the vague value by using the median membership function. Equation (6) could be used to calculate M_B for burst time and M_P for static priority. Consider

$$\begin{aligned} M_B &= \frac{1 - t_B + f_B}{2}, \qquad M_B \leq 1, \\ M_P &= \frac{1 - t_P + f_P}{2}, \qquad M_P \leq 1. \end{aligned} \tag{6}$$

Further, these median membership values are used by VIS to assign the dynamic priority P_D to each task as given in (7). Our VIS has the restriction that each task's dynamic priority must be less than or equal to 1. Consider

$$P_D = \max(M_B, M_P), \qquad P_D \leq 1. \tag{7}$$

4.2. Vague Oriented Priority Scheduling Algorithm. In VOPS algorithm, the tasks are sorted in descending order according to their dynamic priority and the highest priority task is scheduled first to the CPU. Whenever a new task arrives, the dynamic priority is recalculated for all the ready tasks and ready queue is updated accordingly. For Example, if there are 3 tasks T1, T2, and T3 in the RQ. Then the dynamic priority of all these tasks will be calculated. Assuming task T2 is having highest priority, and then T2 will be assigned to CPU. Now during the execution of T2, another task T4 has arrived. After the completion of T2, the dynamic priorities for each task will be recalculated and task with highest priority would be scheduled to CPU.

4.2.1. Performance Metrics

Definition 5 (waiting time). Waiting time is the total time tasks waits in the ready queue for CPU. The average waiting time is calculated as given in (8). Consider

$$W_s = \frac{\sum_{i=1}^{N} W_i}{N}, \tag{8}$$

where W_i is the waiting time for each task i.

Definition 6 (turnaround time). Turnaround time is the total time between the submission of a task and its complete execution. For better performance, the average turnaround time should be minimized and it can be calculated as in (9). Consider

$$T_s = \frac{\sum_{i=1}^{N} T_i}{N}, \tag{9}$$

where T_i is the turnaround time for each task i. Equation (9) can be further defined and given in (10). Consider

$$T_s = \frac{\left(\sum_{i=1}^{N} W_i + B_i\right)}{N}. \tag{10}$$

```
Step 1. For process i:= 1, ..., N loop
            (i) Assign B:= burst time;
                    A:= arrival time;
                    P:= static Priority;
            (ii) Compute the dynamic priority $P_{Di}$ using VIS.
            End loop
Step 2. For process i:= 1, ..., N loop
            RQ:= Sort the processes in decreasing order of $P_{Di}$
    End loop
Step 3. For process i:= 1, ..., N loop
            (i) Schedule and dispatch process i ∈RQ with CPU.
            (ii) Calculate waiting time, turnaround time and normalized turnaround time for process i.
            (iii) After completion of each process i, check the RQ. If any new process arrives in the
                ready queue then go to Step 1.
        End loop
Step 4. Calculate average waiting time (Ws), average turnaround time (Ts) and average
        normalized turnaround time (NTs) using (8), (10) and (11) respectively.
```

Algorithm 1

```
$b_{max}$ = max(b);
$b_{min}$= min(b);
for i = 1 : n
    $t_B = (b_{max} - b(i) + 1)/(b_{max} + 1)$;
    $f_B = (b(i) - b_{min})/(b_{max} + 1)$;
end
```

Algorithm 2

```
[y, z] = Priority(P, n);
[a, c] = burst_time(B, n);
for i = 1 : n
$M_B(i) = (a(i) + 1 - c(i))/2$;
$M_P(i) = (y(i) + 1 - z(i))/2$;
$D_P(i) = \max(M_B(i), M_P(i))$;
End
```

Algorithm 3

Definition 7 (normalized turnaround time). Normalized turnaround time is the ratio of turnaround time to burst time. It represents the relative delay of the task and can be calculated as in (11). Consider

$$NT_s = \frac{\sum_{i=1}^{N} (T_i/B_i)}{N}. \tag{11}$$

4.2.2. VOPS Algorithm Sequence. Algorithm 1 is illustrated step by step.

5. Simulation and Results

MATLAB is used to simulate our effort in designing the algorithm. To implement the VIS of the two functions which are defined inside the VLU, *burst_time* and *priority*.

Algorithm 2 is the snippet for function *burst_time* that shows how the burst values are extracting from the RQ.

Algorithm 3 gives the idea about the generation of dynamic priorities which is further used by scheduler to schedule the task.

For the working of algorithm, consider a simple example involving 5 different tasks T1 to T5. Table 1 contains the given burst time and the static priority for these tasks. This simple example shows how the dynamic priorities are calculated using VIS.

The VIS extracts the static priority and burst time for each task T1–T5. After applying the *priority* and *burst_time* functions on this task set, we get the values as given below:

$$\begin{aligned} y = t_P &= 0 \quad 0.1429 \quad 0.7143 \quad 0.2857 \quad 0.5714, \\ z = f_P &= 0.8333 \quad 0.6667 \quad 0 \quad 0.5000 \quad 0.1667, \\ a = t_B &= 0.8800 \quad 0.0400 \quad 0.7600 \quad 0.6400 \quad 0.6800, \\ c = f_B &= 0 \quad 0.8400 \quad 0.1200 \quad 0.2400 \quad 0.2000. \end{aligned} \tag{12}$$

After applying (6), the values of M_B and M_P are

$$\begin{aligned} M_B &= 0.9400 \quad 0.1000 \quad 0.8200 \quad 0.7000 \quad 0.7400, \\ M_P &= 0.0833 \quad 0.2381 \quad 0.8571 \quad 0.3929 \quad 0.7024. \end{aligned} \tag{13}$$

Finally by considering the maximum value between M_B and M_P, we can calculate the value of dynamic priority using (7). Consider

$$P_D = 0.9400 \quad 0.2381 \quad 0.8571 \quad 0.7000 \quad 0.7400. \tag{14}$$

The updated task set with dynamic priority is given in Table 2.

Now the VOPS algorithm is applied over the updated Task set 1. The highest priority task is scheduled first to the CPU. Scheduling is represented in the form of Gantt chart as given in Figure 5(a).

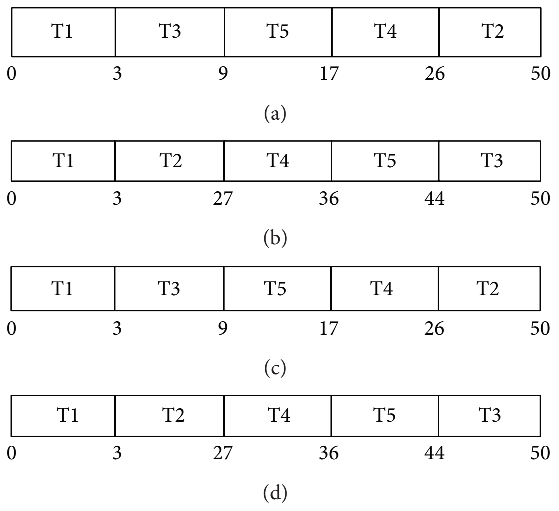

Figure 5: (a) Gantt chart for VOPS. (b) Gantt chart for PS. (c) Gantt chart for SJF. (d) Gantt chart for FS.

Table 1: Example Task Set 1.

	Task				
	T1	T2	T3	T4	T5
Priority	6	5	1	4	2
Burst time	3	24	6	9	8

Table 2: Updated Task Set 1.

	Task				
	T1	T2	T3	T4	T5
Dynamic priority	0.94	0.23	0.85	0.7	0.74
Burst time	3	24	6	9	8

Table 3: Outcomes of Task Set 1.

S. no.	Scheduling technique	W_s (ns)	T_s (ns)	N_s (ns)
1	PS	22	32	3.99
2	SJF	11	21	1.92
3	FS	12	22	3.99
4	VOPS	11	21	1.92

When we schedule the task set in Table 1 using PS, SJF, and FS algorithm, the Gantt chart is shown in Figures 5(b), 5(c), and 5(d), respectively.

Based on the Gantt chart waiting time, turnaround time and normalized turnaround time are computed as shown in Figures 6(a), 6(b), and 6(c), respectively.

Further we have calculated the average waiting time (W_s), average turnaround time (T_s), and average normalized turnaround time using (8), (10), and (11) for each algorithm, respectively. Outcomes of all algorithms are shown in Figure 6(d) and also summarized in Table 3.

From the outcomes of Task Set 1 given in Table 3, one can analyze that the performance of VOPS is better than traditional approach PS and Fuzzy approach FS. Also the results of VOPS algorithm is similar to SJF algorithm which is known as an optimum scheduling algorithm. Now consider the second task set with 4 tasks as given in Table 4.

As mentioned in Task Set 1, VIS extracts the priority and burst time of each task and generates the dynamic priority as given in Table 5.

Once again after applying VOPS algorithm to the updated Task Set 2 and it is scheduled as shown in Figure 7(a).

Then all of the remaining three algorithms PS, SJF, and FS are applied. Gantt chart for all is shown in Figures 7(b), 7(c), and 7(d), respectively.

Figures 8(a), 8(b), and 8(c) represent the comparison of the waiting time, turnaround time, and the normalized turnaround time of each task.

The overall computation of Sample Task Set 2 in terms of average waiting time, average turnaround time, and average normalized turnaround time is shown in Figure 8(d).

Overall results of each scheduling techniques are analyzed in Table 6.

Similarly, one can judge that the performance of proposed VOPS algorithm, given in Table 6, is better than the PS and FS and more close to SJF.

Table 4: Example Task Set 2.

	Task			
	T1	T2	T3	T4
Priority	7	1	15	4
Burst Time	10	5	4	8

Table 5: Updated Task Set 2.

	Task			
	T1	T2	T3	T4
Dynamic priority	0.55	0.94	0.82	0.74
Burst time	10	5	4	8

Table 6: Outcomes of Task Set 2.

S. no.	Scheduling technique	W_s (ns)	T_s (ns)	N_s (ns)
1	PS	10	16.75	2.64
2	SJF	7.5	14.25	1.91
3	FS	8	14.75	2.02
4	VOPS	7.75	14.5	2.01

VOPS algorithm are simulated on different task sets having the random values of burst time and static priority and the average waiting time, average turnaround time, and average normalized turnaround time for each task set are calculated. For accuracy VOPS algorithm is compared for all task sets with other algorithms such as priority scheduling (PS), shortest job first (SJF), and fuzzy based priority scheduling (FS).

Figure 9 shows the performance in terms of average waiting time. Figures 10 and 11 show the performance in terms of average turnaround time and average normalized turnaround time. Form the graph we can evaluate the performance of our proposed work. From the graphs we can see that our

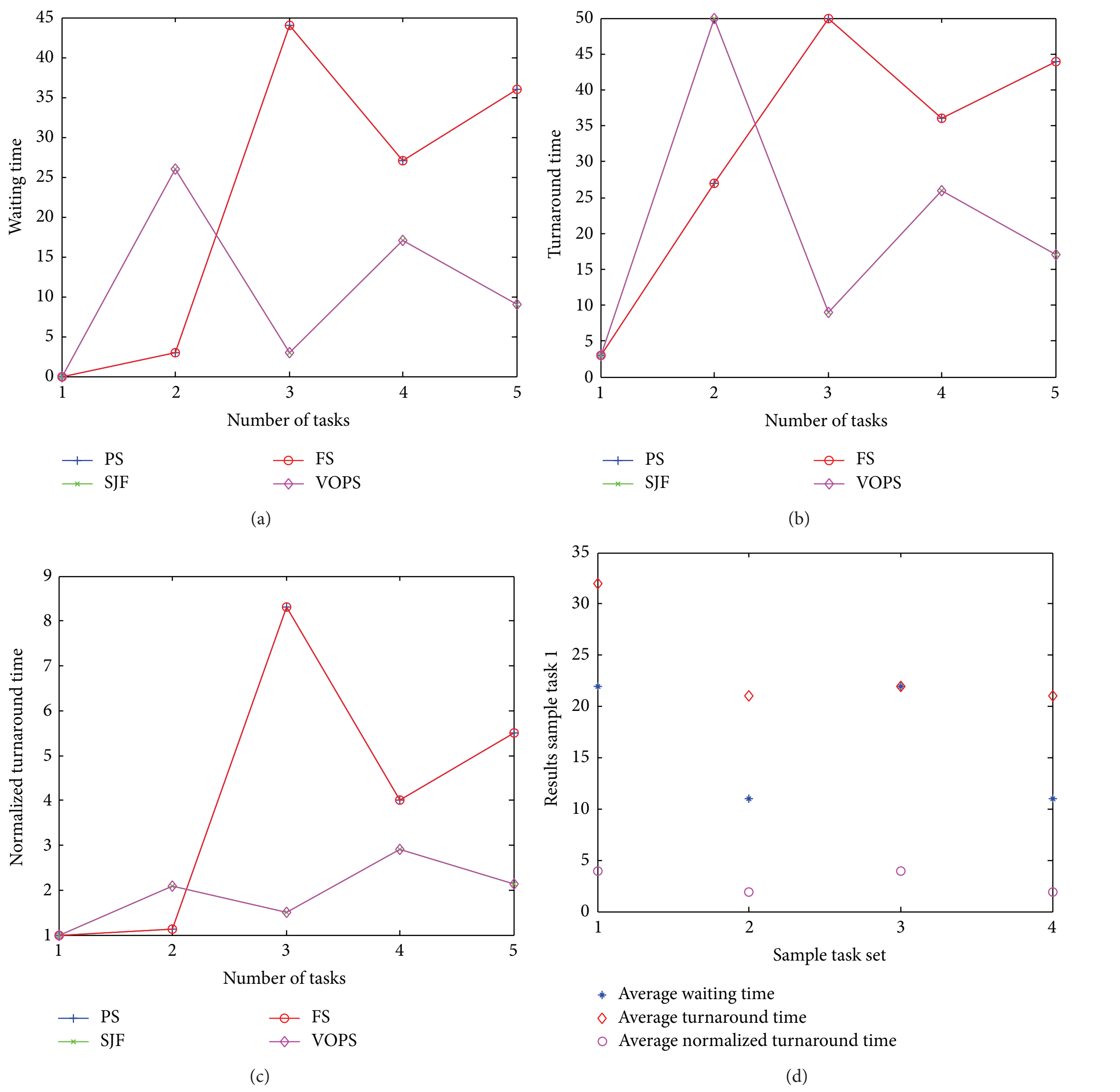

Figure 6: (a) Waiting time for sample task set 1. (b) Turnaround time for sample task set 1. (c) Normalized turnaround time for sample task set 1. (d) Results of sample task 1.

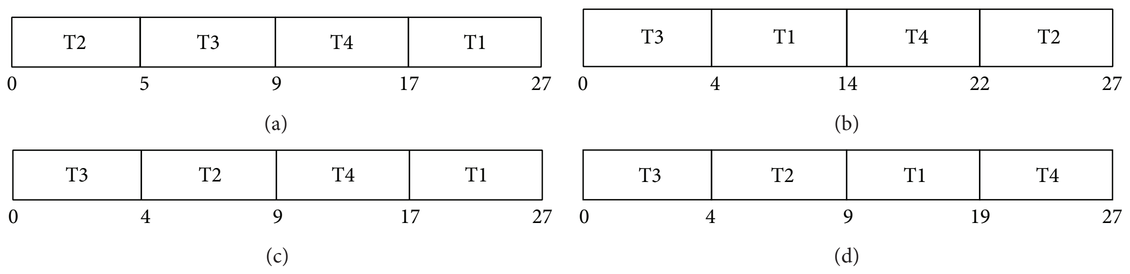

Figure 7: (a) Gantt chart for VOPS. (b) Gantt chart for PS. (c) Gantt chart for SJF. (d) Gantt chart for FS.

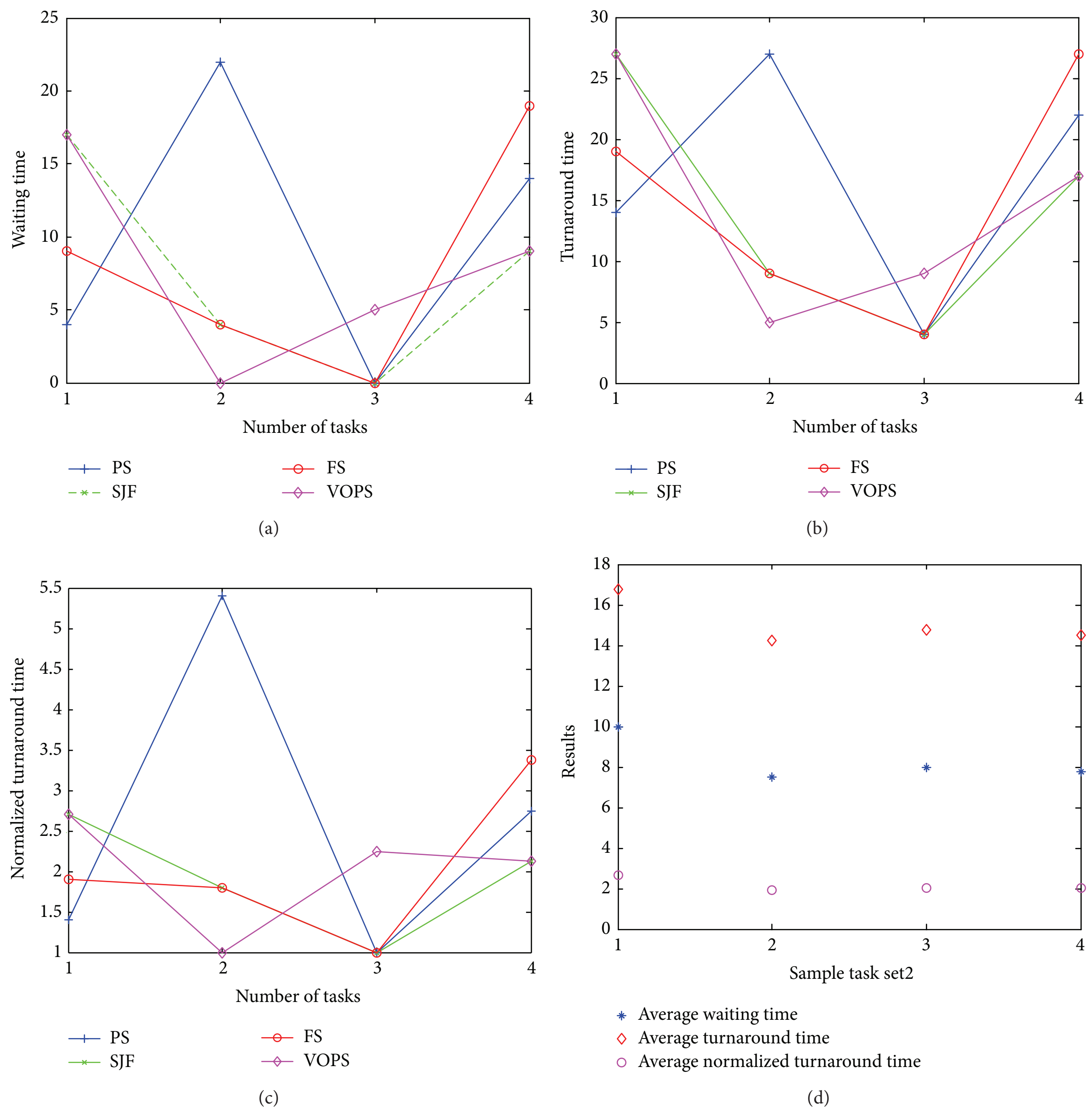

FIGURE 8: (a) Waiting time for sample task set 2. (b) Turnaround time for sample task set 2. (c) Normalized turnaround time for sample task set 1. (d) Results of sample task 2.

algorithm results are in between the FS and SJF, better than FS and nearby SJF but in some cases equal to SJF. As we know, SJF is the optimum scheduling algorithm; it clearly signifies the performance of our algorithm. However, we agree there is no so much difference between FS and VOPS, but VOPS is considering the impreciseness in more general way by perceiving the evidence in favour and evidence in against.

VOPS scheduling algorithm has better performance mainly by two reasons.

The proposed VOPS algorithm is considering the current state of tasks in ready queue and address the impreciseness of data. As a result, VOPS is effective for dynamic environment. Secondly, the performance of scheduling algorithms mainly depends on multiple factors, namely, average waiting time, average turnaround time, and average normalized turnaround time. The reduction in the value of factors improves the performance. As a result, VOPS improves the performance of the system.

6. Conclusion

As discussed above some applications and research show that SJF algorithm is better than the priority scheduling algorithm since it provides the less waiting time and less turnaround time. In dynamic applications, priority algorithm must be required by scheduler to assign the different priority order

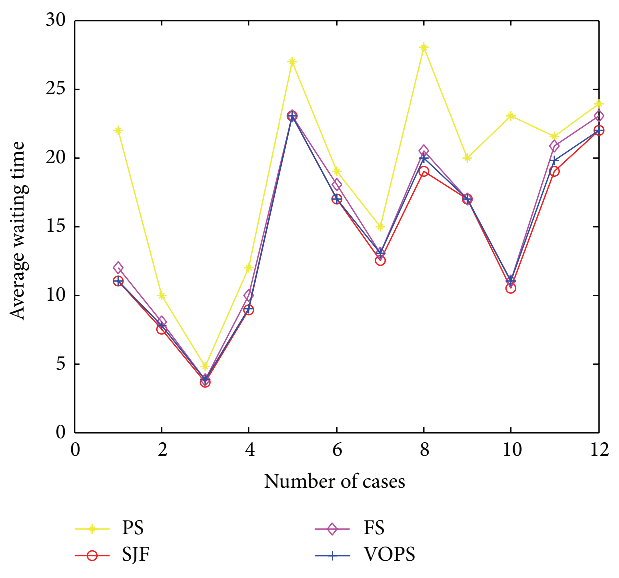

FIGURE 9: Average waiting time.

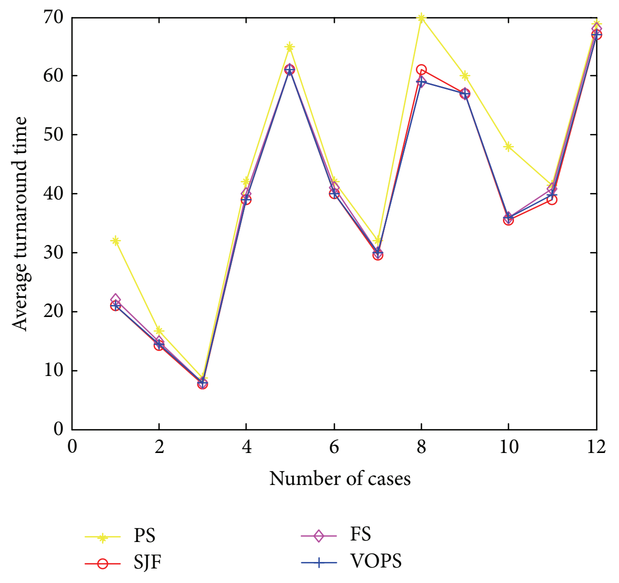

FIGURE 10: Average turnaround time.

to different tasks. In this paper we have introduced a 2-stage CPU scheduler based on vague set theory. In the first stage, scheduler used the VIS to handle the impreciseness and uncertainty. For this purpose, VIS applied the vague set theory and converted crisp data in to vague data. To determine the total evidence in vague values, the median membership function is used in the VIS. Finally VIS generated the dynamic priority for all tasks in ready queue by considering the factors burst time and the user priority. In the second stage, scheduler has selected the next task with vague oriented priority scheduling (VOPS) algorithm. The proposed 2-stage scheduler improves the performance of the system as compared to the system using conventional priority scheduling algorithm and fuzzy scheduling algorithm.

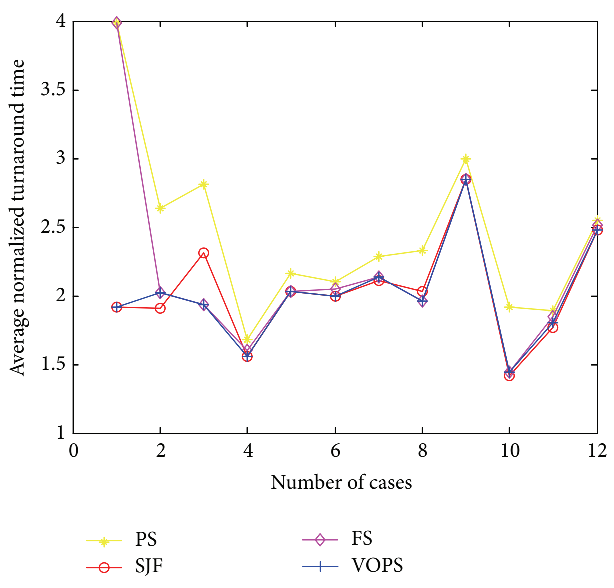

FIGURE 11: Average normalized turnaround time.

To validate the performance we have simulated different task sets which verifies that our propose algorithm significantly reduces the average waiting time, average turnaround time, and average normalized turnaround time and consequently improves the performance.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] C. Gomathy and S. Shanmugavel, "An efficient fuzzy based priority scheduler for mobile ad hoc networks and performance analysis for various mobility models," in *Proceedings of the Wireless Communications and Networking Conference (WCNC '04)*, vol. 2, IEEE, 2004.

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

[3] T. S. Andrew and S. W. Albert, *Operating Systems Design and Implementation*, 2nd edition, 2008.

[4] 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, Phuket, Thailand, December 2008.

[5] H. S. Behera, S. Sahu, and S. K. Bhoi, "Weighted mean priority based scheduling for interactive systems," *Journal of Global Research in Computer Science*, vol. 2, pp. 1–6, 2011.

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

[7] J. Zimmerman, *Fuzzy Set Theory And Its Application*, Kluwer Academic Publishers, Norwell, Mass, USA, 2001.

[8] A. A. Aburas and V. Miho, "Fuzzy logic based algorithm for uniprocessor scheduling," in *Proceedings of the International Conference on Computer and Communication Engineering:*

Global Links for Human Development (ICCCE '08), pp. 499–504, Kuala Lumpur, Malaysia, May 2008.

[9] S. J. Kadhim and K. M. Al-Aubidy, "Design and evaluation of a fuzzy-based CPU scheduling algorithm," *Communications in Computer and Information Science*, vol. 70, pp. 45–52, 2010.

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

[11] K. Atanassov, *Intuitionistic Fuzzy Sets: Theory and Applications*, vol. 35 of *Studies in Fuzziness and Soft Computing*, Physica, New York, NY, USA, 2000.

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

[13] A. Lu and W. Ng, "Vague sets or intuitionistic fuzzy sets for handling vague data: which one is better?" in *Conceptual Modeling—ER 2005*, L. Delcambre, C. Kop, H. C. Mayr, J. Mylopoulos, and O. Pastor, Eds., vol. 3716 of *Lecture Notes in Computer Science*, pp. 401–416, Springer, Berlin, Germany, 2005.

[14] D. Hun Hong and C. Chang-Hwan, "Multicriteria fuzzy decision-making problems based on vague set theory," *Fuzzy Sets and Systems*, vol. 114, no. 1, pp. 103–113, 2004.

[15] L. Yong, W. Guoyin, and F. Lin, "A general model for transforming vague sets into fuzzy sets," in *Transactions on Computational Science II*, vol. 5150 of *Lecture Notes in Computer Science*, pp. 133–144, 2008.

[16] J. Wang, W. Xu, J. Ma, and S. Wang, "A vague set based decision support approach for evaluating research funding programs," *European Journal of Operational Research*, vol. 230, no. 3, pp. 656–665, 2013.