Implementation of Artificial Intelligence in Operating System Scheduling

Loura Shiny M 19BCE1524
Prayasi Gopi 19BCE1719
Shreyaas K 19BCE1800

Vellore Institute of Technology, Chennai - 600127

ABSTRACT

Scheduling is a key process to be carried out by the Operating System. To begin with, we will be looking into the different types of basic scheduling algorithms, comparing them with each other and figuring their merits and demerits. Artificial intelligence is an ever-expanding domain. Through AI machines can learn from experience and will have the ability to deal with uncertainty making scheduling more optimal. Fuzzy Logic is one of the best ways to implement AI in OS scheduling. There are a few existing works on how Fuzzy Logic can be applied in OS Scheduling. We will discuss and analyze them. Then we will discuss about our proposed scheduling algorithm using Fuzzy Logic. Through this project we will try to enhance our knowledge and understanding of different concepts of OS.

KEYWORDS:

Operating System (OS), First Come First Serve (FCFS), Shortest Job First (SJF), Shortest Remaining Time First (SRTF), Round Robin (RR), Priority Scheduling (PS), Fuzzy Logic, Fuzzification, Defuzzification.

1. INTRODUCTION:

The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy. This is Scheduling in Operating System. The aim of this project is to understand the different types of scheduling, their working and compare their algorithms. [1] Scheduling uses ad hoc heuristics and algorithm which means created specifically for solving one particular problem at a particular point in time, rather than being a systematic approach to solving a more general set of problems. Essentially, it only does one thing, and cannot be used for anything else without modification. If the scheduling policy is once determined, it is unable to change without resetting the operating system which takes much time.

Now, to overcome these difficulties we'll try to use AI in OS scheduling. Artificial intelligence is the ability of the machine to learn from experience and deal with uncertainty which will in turn make it more optimal. From the research done we can see that using AI algorithms such as fuzzy logic and neural networks we can optimize scheduling techniques.

2. LITERATURE SUREVEY

Before diving into the topic, we went ahead and did some research on some existing works and here is the survey on a few of them

2 a. *Design and Implementation of Modified Fuzzy based CPU Scheduling Algorithm*

[2] This paper is about designing and implementing fuzzy based Scheduling Algorithm with comes under the domain of Artificial Intelligence.

CPU Scheduling is the base of multiprogramming. This paper discusses the existing works and then introduces a new of fuzzy rules which demonstrates scheduling

done with new priority improves average waiting time and average turnaround time.

In the first section the paper talks about types of scheduling techniques. Then the discussion continues to three scheduling algorithms - Shortest Job First, Priority Scheduling and Fuzzy based CPU scheduling. After this, Fuzzy based algorithm is discussed in detail with many examples and graphs. This algorithm takes input both job priority and execution time and decides new priority using some fuzzy rules. Fuzzy logic is a style of multi-valued logic. It deals with approximation rather than exactness. In contrast to classical sets (Classical set takes true or false values) fuzzy logic variables (also known as linguistic variable) can have a truth value that ranges in interval between 0 and 1.

The next section of this paper deals with the proposed algorithm. Here suitable linguistic variables are taken as input and output to compute a crisp value for new priority. Pre-Priority (PP) measured as Very Low, Low, Medium, High and Very High. Execution Time (ET) measured as Very Small, Small, Medium, Long and Very Long. New Priority (NP) measured as Very Low, Low, Medium, High and Very High. The proposed scheduling is a collection of linguistic fuzzy rules which describe the relationship between defined input variables (PP and ET) and output (NP).

For example, if PP is High and ET is Small THEN NP is High. All the various cases are defined and tabulated accordingly. Now the arrival time isn't taken into consideration (con), all the process of the same time are grouped together and then sorted. The proposed algorithm is simple. All the process has their new priority found, then grouped according to their arrival time and the sorted in their respective groups and executed accordingly. It is very clear that the proposed fuzzy based algorithm obtains better result rather than another scheduling algorithm. It reduces the average waiting time and average turnaround time (pro).

The paper concludes by talking about the results and performance of the modified fuzzy algorithm by taking up different examples. They give their result.

2 b. *Analysis of Process Scheduling Using Neural Network in Operating System*

[3] This paper is comparison between the different AI Algorithms for Scheduling. Process scheduling plays a vital role in multitasking for any operating system. There are many factors involved during process scheduling like priorities, free memory, user demand and processor which if not handled properly can be very complex and time consuming. Neural network has adaptive nature which can be used to handle the complex part easily. The main aim of this paper is to review different types of scheduling algorithms working on the principle of neural network and offer constructive criticism to improve their efficiency.

Efficient process scheduling and context switching is the most important step of multitasking. Optimization here means the system has to schedule the process according to the users need. This can be determined using and studying the user behaviour, this will tell the system which process is important and which isn't. For process scheduling, there are many algorithms used. All techniques have some merits and demerits to them which was discussed in this paper. Methods that will be discussed are job shop scheduling, pre-emptive scheduling, multithreading, genetic algorithm, decision tree learning, Bayesian system, rule-based system and neural network.

Job shop scheduling – Queues the job in linear manner. Pre-emptive scheduling – Assigns quantum time to process during which process takes the CPU. After which it is put back into queue. Multithreading – More than one thread works concurrently within a process. Genetic algorithm – Selection, crossover and mutation functions are used to select the most suitable process. Decision tree learning – non-leaf nodes are split, and the output is generated by the series of decisions. Bayesian system – It works on the bases of probabilities. Rule-based system – Uses set of IF-THEN rules to trigger the rule. Neural network – It works on layered structure and output is generated using the activation. The paper explains each of the above algorithms in detail. Then compares them in a neat tabular format and concludes.

2 c. *Proposed Fuzzy CPU Scheduling Algorithm (PFCS) for Real Time Operating Systems*

Fuzzy Logic Terminology

[5] Fuzzy logic is yet another topic under AI to be discussed under scheduling. Apart from the concept of discrete values 0 and 1, it allows intermediate values to be defined between conventional evaluations. A Fuzzy logic system is nonlinear mapping of an input data to output data. The process of fuzzy logic is to first collect the crisp set of inputs and convert it to the fuzzy sets using fuzzy linguistic variables, fuzzy linguistic terms

and membership functions. This is known as Fuzzification. Here fuzzy linguistic variables refer to the input or output variables of the system whose values are non-numeric whereas a membership function is used to quantify a linguistic term.

2 d. *Implementing a Process Scheduler Using Neural Network Technology*

[6] ***Decision tree learning***: This approach is used as a priority assignment function with discrete values. A balanced decision tree is built such that predicates (functions) are present at the nodes and final decisions are present as leaf nodes. This approach recursively divides or classifies the set into branches as equally as possible. Based on the final decisions of the branches, the processes are executed. This approach can quickly classify problems to different branches. Discrete values are the main disadvantage for this approach as process data mostly deals with continuous data. This method was also sensitive to noise and hence it was avoided.

Rule-based knowledge system: This technique deals with a simple set of "if-then" rules where the scheduler is directed to act according to the conditions mentioned. Though it might work for the user, this approach is very tedious for the programmers and the system would become quite static.

Bayesian networks: They are very similar to neural networks. They represent the connection between two nodes a and b by assigning a weight to that connection by P(b|a). Every node's probability can be calculated by doing a summation of the incoming connection weights multiplied by the certainty of the nodes they are connected to. The disadvantage of this approach is that the nodes must have a meaningful interpretation. Such a structure is not possible in real structure.

Artificial neural networks: These are simplified models of biological networks like those found in the brains of humans. It is the ideal AI technique to use because they can be trained to different situations. This way, the existing operating system kernel's scheduler will be delegated by the decisions taken by the neural network.

2 e. *AI in operating system: An expert scheduler*

[1] The possibility that betters decisions could be made by an Intelligent Agent employing expert system techniques is investigated in this report. The ability to learn from experience and to deal with uncertainty are two characteristics of expert systems that would be essential aspects of such an Intelligent Agent. CPU scheduling is of two forms which is short-term and long-term scheduling. A scheduler which has working upon an expert system is the approach or technique which is proposed in this article. This report enlightens the implementation of a learning mechanism for process scheduling in a multiprocessor environment. The processor remains under-utilized as the I/O subsystem becomes a bottleneck and reduces the processing rate. Here the intelligent agent tries to learn its own set of heuristics for optimally scheduling a set of co-operating process. By simulating a relatively simple multiprocessor system we examine here. But in this there are few limitations. Some criteria were not taken into account such as inter-process communication, context-switching and so on. Therefore, AI techniques to prune search tree could be implemented. Another limitation was that the heuristics which were used in this process were fairly simple. This same heuristic was applied whenever a scheduling decision has to be made. In order to overcome this changing load conditions, multiple heuristics can be generated. Here an assumption was made that process's CPU time requirements is known exactly. But this assumption is a deficiency to this approach. To overcome this deficiency, the simulation could be easily enhanced to accept CPU time requirements that are given as estimates with some non-zero probability that they will return out to be good estimate. This can be enhanced by making accept the CPU time requirements as a Poisson distribution and use the mean as the estimate. This time requirement deficiency is great drawback for this technique and hence this solved with the help of Poisson distribution as mentioned earlier. This was later used in the fuzzy logic implementation. Here if we keenly analyse two drawbacks one which is that context switching is not allowed and another is the deficiency in time requirement, fuzzy logic would be the best solution for these limitations.

2 f. Utilization of Fuzzy Logic in CPU Scheduling in Various Computing Environments

[7] Fuzzy logic comprises of 4 steps which is fuzzification, creating a rule base, inference based on rules and defuzzification. The simulation part is the creating rule base and the inferring the rule base. Here they have implemented this for two scheduling process in operating system namely pre-emptive priority and round robin. In the pre-emptive priority scheduling, the scheduler uses quantum of 1 to check the arrival of new process. If a process with higher priority is arrived then the current process is pre-empted. Possibility of starvation was not taken into account here. The round robin scheduler was implemented with quantum 4- and 8-time units. To this the fuzzy logic is implemented

and for the input given for static priority, the dynamic priority is found.

Here the first scenario is simulated with 75%, I/O 25% CPU bound. The pre-emptive priority scheduler with fuzzy logic integration worked better than standard priority scheduler. The fuzzy priority scheduler had an average wait time that was 51.4% less than pre-emptive priority scheduler, 55.9% less than the round robin scheduler with time quantum of 4 and 57.5% less than round robin scheduler with a time quantum of 8. This clearly shows that the fuzzy scheduler outperforms other schedulers when the simulations have 50% I/O and CPU bound.

The second scenario is simulated with 50%, I/O 50% CPU bound. Here the fuzzy priority scheduler had an average wait time that was 39.8%less than the pre-emptive priority scheduler, 59.9% less than the round robin scheduler with a quantum of 4 and 50.9% less than the quantum of 8.

The third scenario is simulating with 25% I/O and 75% CPU. Here the fuzzy priority scheduler had an average wait time that was 25.6%less than the pre-emptive priority scheduler, 55.3% less than the round robin scheduler with a quantum of 4 and 43.8% less than the quantum of 8.

This shows that with the technique they have achieved there is great response time in comparison to the pre-emptive priority scheduler. Also, the round robin had a greater average response time in comparison to fuzzy priority and pre-emptive priority. Therefore, this proved that in terms of average waiting and response time, the fuzzy priority scheduler works more better than the standard priority scheduler. But there is an uncertainty when it comes for implementation in homogeneous and heterogeneous environments. Also, Starvation is not taken into account in this implementation. And this is proved only for average waiting time and response time an average arrival time is not proved.

3. DIFFERENT SCHEDULING ALGORITHMS

Operating System uses different scheduling algorithms according to its need to make sure that processors are executed properly. A Process Scheduler assigns different processes to the CPU according to these algorithms. We considered five of these algorithms:

- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-Next (SJN) Scheduling
- Priority Scheduling
- Shortest Remaining Time
- Round Robin (RR) Scheduling

These algorithms are either non-preemptive or preemptive. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

4. COMPARISON OF SCHEDULING ALGORITHMS

Initially, we considered four different processes with different arrival times, execution time and priority. We calculated the average waiting time and average turnaround time for each of the mentioned algorithms.

Process	Arrival Time	Burst Time	Priority
1	0	8	2
2	1	4	3
3	2	7	1
4	3	5	4

Figure 1: *Four processes with the arrival time, execution time and priority.*

SCHEDULING ALGORITHM	AVERAGE TURN AROUND TIME	AVERAGE WAITING TIME
FCFS	14	9
SJF	13	7
SRTF	12	6
RR	18	12
PS	15	9

Figure 2: *Average waiting time and turnaround time for all five above mentioned algorithms.*

Here is the summary of all the five scheduling algorithms-
(Table at the end of the report)

a. FCFS:

Definition: Processes are executed on first come, first serve basis.
Pre-emptive/ non-pre-emptive: Non-pre-emptive
Merits:
- Easy to understand.
- Easy to implement.

Demerits:
-Poor in performance as average wait time is high.

b. SJF:

Definition: The process is executed on which process is closest to its completion.
Pre-emptive/ non-pre-emptive: Non-pre-emptive
Merits:
- Best approach to minimize waiting time.

Demerits:
-The processer should know in advance how much time process will take.
-It may cause starvation if shorter processes keep coming.

c. SRTF:

Definition: Shortest remaining time is the pre-emptive version of the SJF algorithm.
Pre-emptive/ non-pre-emptive: Pre-emptive
Merits:
- Makes the processing of the jobs faster than SJF algorithm, given its overhead charges are not counted.

Demerits:
- The context switch is done a lot more times in SRTF than in SJR, and consumes CPU's valuable time for processing.

d. PS:

Definition: Each process is assigned a priority. Process with highest priority is to be executed first and so on
Pre-emptive/ non-pre-emptive: Non-pre-emptive

Merits:
- Easy to implement.
-The ones with higher priority get executed first.

Demerits:
-Poor in performance as average wait time is high.

e. RR:

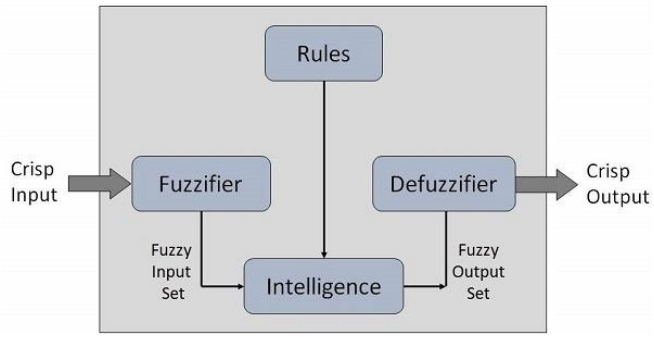
Definition: -The CPU is shifted to the next process after fixed interval time, which is called time quantum/time slice.
-The process that is pre-empted is added to the end of the queue.
Pre-emptive/ non-pre-emptive: Pre-emptive
Merits:
- This method helps for starvation free execution of processes.

Demerits:
- This method spends more time on context switching
- Heavily dependent on quantum time

5. FUZZY LOGIC

Fuzzy logic in simple terms is a vague value. In life not everything is black and white, there is a lot of grey, i.e, not everything is true (1) or false (0), there is a lot of maybes, most likely, unlikely, might be, etc. Fuzzy logic is a logic that thinks like human. Instead of yes or no, we have yes, maybe, most likely, likely, no, etc.

Fuzzy architecture has four components, the rules, fuzzifier, intelligences and defuzzifier. A crisp input is given to the fuzzifier it creates a fuzzy input set which is given to the intelligence. Using the established rules, the fuzzy input set is processed and a fuzzy output set is given to the defuzzifier where defuzzification happens to get a crisp output.
Membership functions work on fuzzy sets of variables. Membership functions quantifies linguistic term and represent a fuzzy set graphically. There is no right or die way to approach a problem with fuzzy logic which its advantage and disadvantage. Everyone will have their own approach to a given problem, thus it's a favourable method to researchers. It's also a favourable as the construction of Fuzzy Logic System is very easy and understandable. Also, easy to represent and reason mathematically



Fuzzy Inference System (FIS):

Fuzzy Inference System is the key unit of a fuzzy logic system having decision making as its primary work. It uses the "IF…THEN" rules along with connectors "OR" or "AND" for drawing essential decision rules.
FIS produces a fuzzy set irrespective of its input.
So, to get a crisp output we need to do defuzzification.
There are two methods in FIS:

- Mamdani Fuzzy Inference System
- Takagi-Sugeno Fuzzy Model (TS Method)

Steps involved

1. Define fuzzy inputs and output.
2. Define the membership functions for the inputs and outputs.
3. Determine set of fuzzy rules.
4. Using membership functions to give fuzzy inputs.
5. Get fuzzy output.
6. Defuzzification of the output distribution.

6. DISCUSSION AND ANALYSIS ON FUZZY APPROACHES

To begin our research, we read and analyzed some of the existing papers on Implementation of Artificial Intelligence in Operating System Scheduling. Our individual analysis is as mentioned in the Literature Survey. Then a general discussion was done on these papers. Out them all three of the proposed algorithms were tried and analyzed.

After ample discussion on the different proposed methods to introduce fuzzy logic in OS scheduling, we decided to try to define our own algorithms where a new priority, dynamic priority is assigned to each process and then they are executed accordingly using priority scheduling.

We will be looking into three different approaches from the research done.

6 a. FUZZY APPROACH – FUZZY_P

[5] A new approach of scheduling which involves fuzzy logic is discussed here. The proposed method primarily involves burst time and priority as inputs and membership functions are calculated for each of them. Since, our approach is fuzzy based, there are no hard and fast rules on selecting the membership functions. They are regarded as universal approximators and hence they find their place in engineering and non-engineering applications. Hence, they can be quantified on a random basis with an expectation of getting better results than the existing ones.

In this approach, membership functions are calculated with burst time, priority and arrival time and the new priority is then assigned after comparing the priority with these parameters. Once the new priority has been found, the processes get scheduled in the way priority scheduling takes place and the turn-around time and waiting time have been calculated. The motive of researching on all possible ways for finding a new priority to make scheduling more efficient is the primary idea.

Algorithm:

(i) Input arrival time, burst time and priority for n processes.

(ii) Calculate maximum priority, maximum arrival time and maximum burst time.

(iii) Calculate membership functions as follows:

```
memb_funpri[i]=priority[i]/max_pti+1;
//Membership function for priority
memb_funbur[i]=1-(burst_time[i]/max_bti+1);
//Membership function for burst time
memb_funarr[i]=1-(arr_time[i]/max_arr+1);
//membership function for arrival time
```

(iv) Calculation of new priority based on membership functions:

```
if(priority[i]<memb_funpri[i]&&priority[i]<memb_fun
bur[i]&& priority[i]<memb_funarr[i])
        dpi_p[i]=memb_funpri[i]+
memb_funbur[i]+memb_funarr[i];
else
if(memb_funpri[i]>memb_funbur[i]&&memb_funpri[i
]>memb_funarr[i])
        dpi_p[i]=memb_funpri[i];
else if(memb_funbur[i]>memb_funarr[i])
        dpi_p[i]=memb_funbur[i];
else
        dpi_p[i]=memb_funarr[i];
```

(v) Applying priority scheduling with new priority

(vi) Calculate wait time and turn around time.

6 b. FUZZY APPROACH – FUZZY_S

[8] The scheduling of CPU is done by 2 stage VOPS (vague oriented priority scheduling) algorithm. In the traditional priority scheduling algorithm, a fixed priority is assigned to each task by system called as a static priority since it is assigned at once and does not change. At each and every scheduling event, the ready queue is sorted according to priority. The working is based on the highest priority that is scheduled to the CPU for execution. This task is not optimal as it does not consider the present state of the tasks. For this, they considered a new priority called as dynamic priority. 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. This can reduce the average waiting time and average turn-around time. But this approach

does not use tasks in ready queue, task with highest priority having maximum burst time which can starve the tasks with lower burst time, and so forth. This Proposed CPU Scheduler works in two stages; for the first stage they introduced a vague inference system to generate dynamic priorities for tasks by considering the impreciseness of attributes and for second stage of scheduler they projected a new vague oriented priority scheduling algorithm to schedule the next process to CPU. The crisp priority value is converted to vague value in this process. They have simulated the obtained values and the results obtained were better than the other scheduling algorithms.

Algorithm:

Step 1: For process i:=1,….N loop
Assign B:= burst time;
A:= arrival time;
P:= static priority;
Compute the dynamic priority PDi using VIS.

Step 2: For process i:=1,….N loop Find Dynamic priority
Find the maximum burst time
Find the minimum burst time
Find the maximum Static priority
Find the minimum Static priority

Compute tb and fb for burst time

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

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

Compute t_p and f_p for static priority

$$t_P = \frac{P_{\max} - P}{P_{\max} + 1},$$

$$f_P = \frac{P - P_{\min}}{P_{\max}}, \qquad t_P + f_P \le 1.$$

Step 3: Compute membership function for burst time and static priority

$$M_B = \frac{1 - t_B + f_B}{2}, \quad M_B \le 1,$$

$$M_P = \frac{1 - t_P + f_P}{2}, \quad M_P \le 1.$$

Find the maximum of M_B and M_P to get the Dynamic priority

$$P_D = \max(M_B, M_P), \quad P_D \le 1.$$

Step 4: finally average waiting time and turn around time is computed

6 c. FUZZY APPROACH – FUZZY_L

[2] The proposed work is to consider the arrival time, execution time and priority to get a **New Priority** using Mamdani type inference. All the three inputs are grouped into three sets as follows:

- **Priority (0 - 10)** – Low (0 - 5), Medium (2.5 – 7.5), High (5 - 10)
- **Execution Time (0 - 12)** – Small (0 - 6), Medium (3 - 9), Long (6 - 12)
- **Arrival Time (0 - 8)** – Set 1 (0 - 4), Set 2 (2 - 6), Set 3(4 - 8)
- **New Priority (0 - 10)** – Low (0 - 5), Medium (2.5 – 7.5), High (5 - 10)

Using Simulink, a MATLAB Fuzzy Inference Stimulator we stimulated the proposed New Priority

Following the steps mentioned,

1. Defining fuzzy inputs and output:

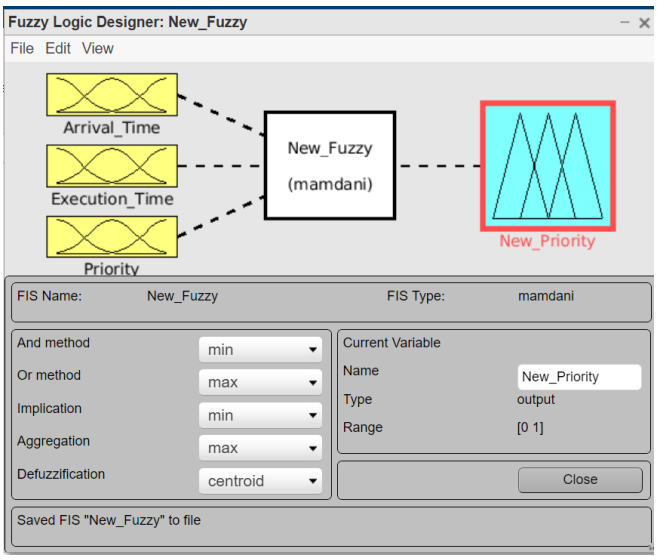


Figure 3: *Defining the input and output variables in Simulink.*

2. *Defining the membership functions for the inputs and outputs:*

Input:

Arrival Time:

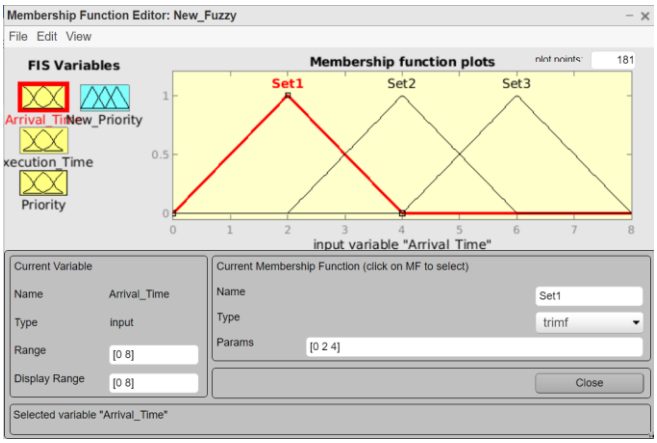


Figure 4: *Defining member functions – Set1, Set2, Set3 for Arrival Time. Setting range, name, type and parama for each function.*

Execution Time:

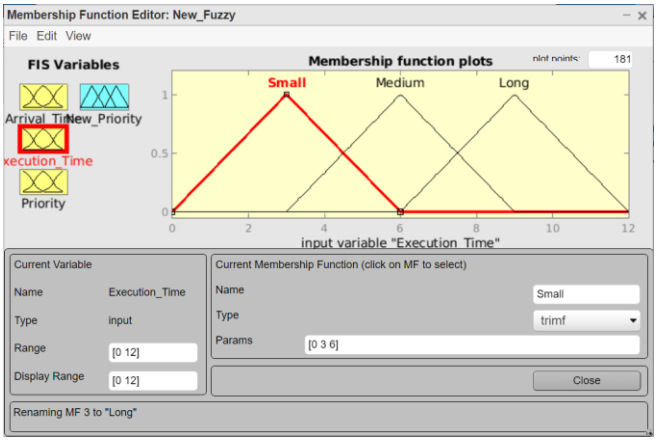


Figure 5: *Defining member functions – Small, Medium, Long for Execution Time. Setting range, name, type and parama for each function.*

Priority:

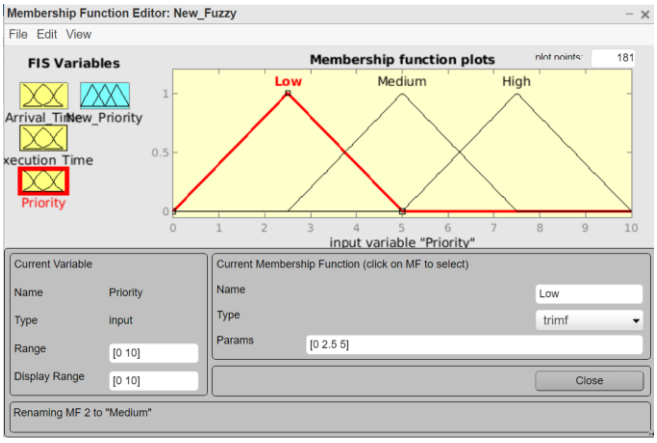


Figure 6: *Defining member functions – Low, Medium, High for Arrival time. Setting range, name, type and parama for each function.*

Output:

New Priority:

Figure 7: *Defining member functions – Low, Medium, High for New Priority. Setting range, name, type and parama for each function.*

3. *Determining set of fuzzy rules:*

Now, three sets under three features gives us 27 combinations of rules.

ARRIVAL TIME	EXECUTION TIME	PRIORITY	NEW PRIORITY
Set1	Small	Low	High
Set1	Small	Medium	High
Set1	Small	High	High
Set1	Medium	Low	Medium
Set1	Medium	Medium	Medium
Set1	Medium	High	High
Set1	Long	Low	Low
Set1	Long	Medium	Medium
Set1	Long	High	High
Set2	Small	Low	High
Set2	Small	Medium	Medium
Set2	Small	High	High
Set2	Medium	Low	Medium
Set2	Medium	Medium	Medium
Set2	Medium	High	High
Set2	Long	Low	Low
Set2	Long	Medium	Medium
Set2	Long	High	Medium
Set3	Small	Low	Low
Set3	Small	Medium	Medium
Set3	Small	High	High
Set3	Medium	Low	Low
Set3	Medium	Medium	Low
Set3	Medium	High	Medium
Set3	Long	Low	Low
Set3	Long	Medium	Low
Set3	Long	High	Medium

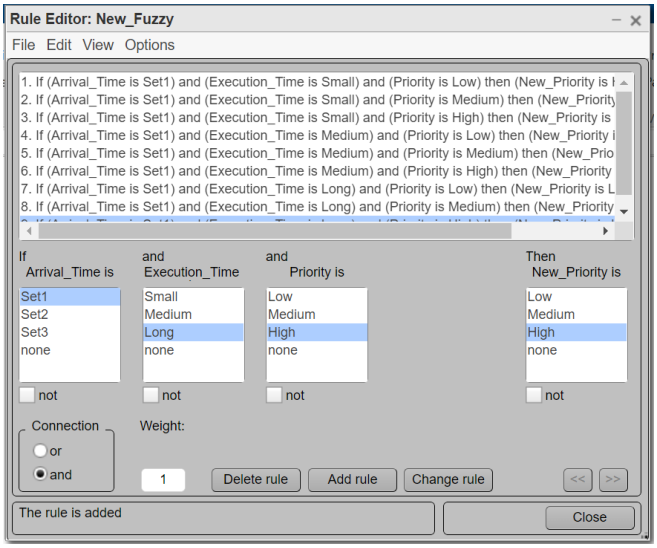


Figure 8: *Setting the fuzzy rules.*

4. *Using membership functions to give fuzzy inputs and get fuzzy output.*

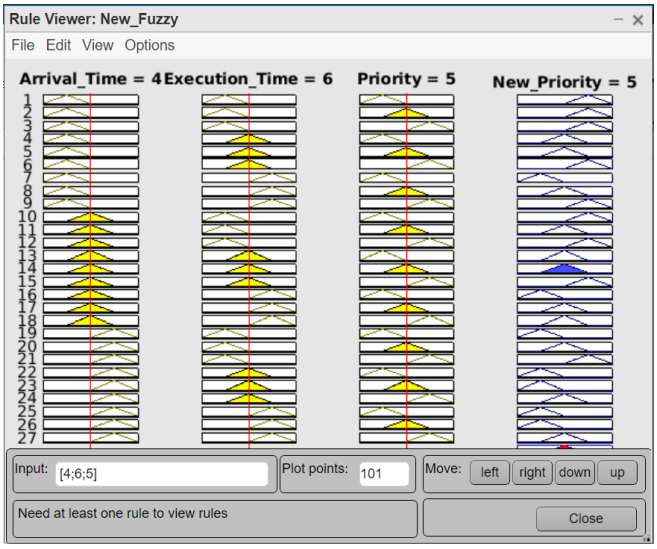


Figure 9: *Rules view. Here the Arrival Time = 4, Execution Time = 6, Priority = 5 which makes New Priority = 5 according to the rules set.*

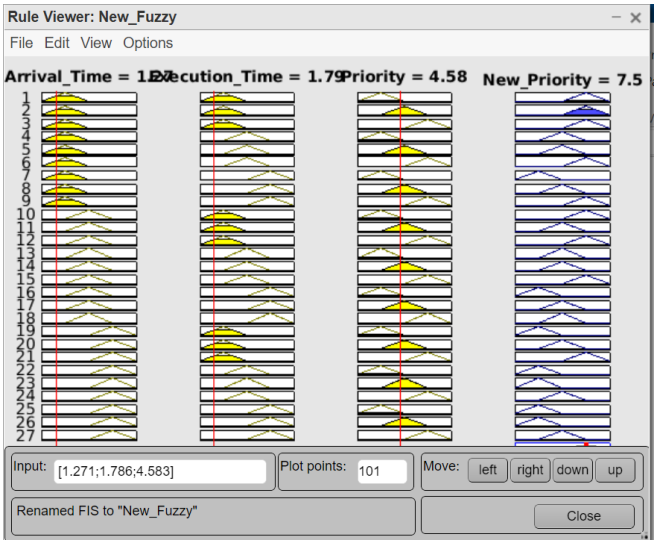


Figure 10: *Rules view. Here the Arrival Time = 1.2, Execution Time = 1.9, Priority = 4.58 which makes New Priority = 7.5 according to the rules set.*

Surface View

(Execution, Priority):

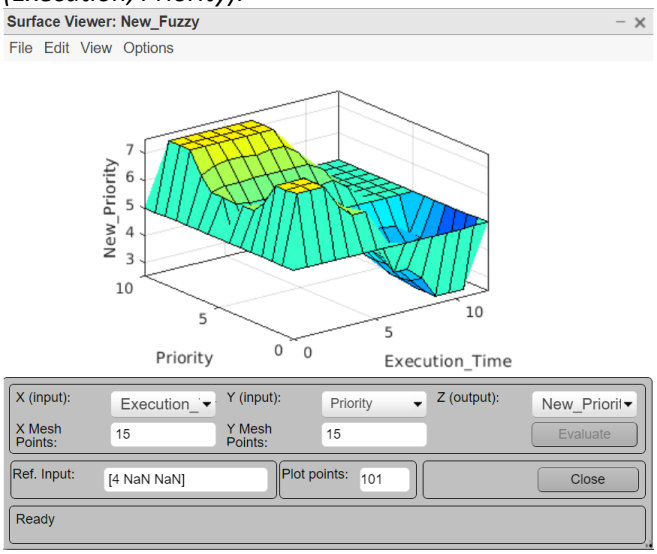


Figure 11: *Surface view of the rule, with Execution Time on x-axis and Priority on y-axis.*

(Arrival Time, Execution time):

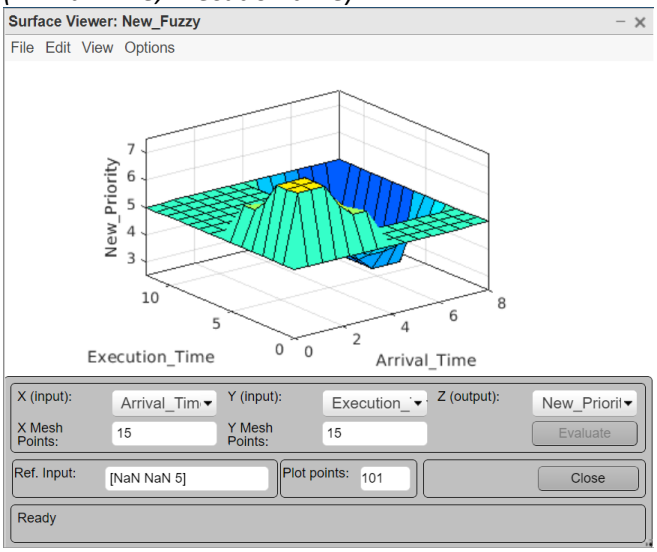


Figure 12: *Surface view of the rule, with Arrival Time on x-axis and Execution Time on y-axis.*

(Arrival Time, Priority):

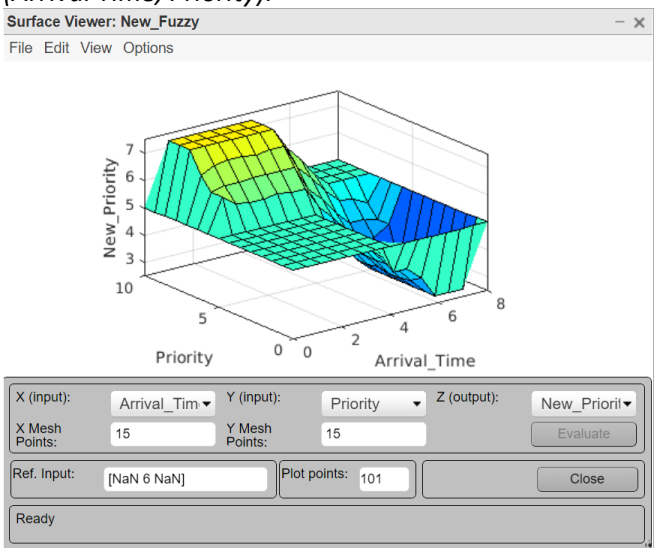


Figure 9: *Surface view of the rule, with Arrival Time on x-axis and Priority on y-axis.*

5. Defuzzification of the output distribution.

From many defuzzification method, centroid was chosen to get a the defuzzied, crisp output. Centroid defuzzification returns the centre of gravity of the fuzzy set along the x-axis. If you think of the area as a plate with uniform thickness and density, the centroid is the point along the x-axis about which the fuzzy set would balance.

Here, the factors arrival time, execution time and priority are considered together in defining a new priority. For example, in the beginning, when the arrival time is withing set 1, only the processes with high priority and less execution time are given a high priority but as soon as the time mark crosses and set 2 arrival time starts, the next set's processes are also considered and accordingly the new priority of the processes change.

Algorithm:

1. Get arrival time, execution time and priority for n-processes.
2. Group them into three sets according to the arrival times.
3. Assign new priorities accordingly.
4. Sort them in descending order according to their new priority.
5. Start executing with the process that has the highest priority.
6. Further, average waiting time and average turnaround time can be calculated.

7. RESULTS

Firstly, the proposed algorithm was run for a simple value that were used for the basic OS scheduling algorithms.

Process	Arrival Time	Burst Time	Priority
1	0	8	2
2	1	4	3
3	2	7	1
4	3	5	4

Figure 14: *Four processes with the arrival time, execution time and priority as input for the proposed fuzzy approaches.*

FUZZY APPROACHES	AVERAGE TURN AROUND TIME	AVERAGE WAITING TIME
FUZZY P	15	9
FUZZY S	14	8
FUZZY L	16	10

Figure 15: *Average turnaround time and waiting time for the proposed fuzzy approaches.*

FUZZY_P

```
Enter number of processes:4
Enter Arrival Time for process 1: 0
Enter Arrival Time for process 2: 1
Enter Arrival Time for process 3: 2
Enter Arrival Time for process 4: 3
Enter Burst Time for process 1: 8
Enter Burst Time for process 2: 4
Enter Burst Time for process 3: 7
Enter Burst Time for process 4: 5
Enter Priority for process 1: 2
Enter Priority for process 2: 3
Enter Priority for process 3: 1
Enter Priority for process 4: 4

Process NO      Arrival Time    Burst Time      Static Prioirity        Dynamic Priority
P[0]            0               8               2                       1.500000
P[1]            1               4               3                       1.750000
P[2]            2               7               1                       1.250000
P[3]            3               5               4                       2.000000

Average waiting time: 9
Average turnaround time: 15


...Program finished with exit code 0
Press ENTER to exit console.
```

Figure 17: *Output for FUZZY_P, dynamic priority method.*

FUZZY_S

```
Enter the 2 static priority:
3
Enter the 3 burst time:
7
Enter the 3 arrival time:
2
Enter the 3 static priority:
1
Enter the 4 burst time:
5
Enter the 4 arrival time:
3
Enter the 4 static priority:
4
The maximum burst time is: 8.000000
The minimum burst time is: 4.000000
The maximum static priority is: 4.000000
The minimum static priority is: 1.000000
computing tb and fb
0.111111        0.555556        0.222222        0.444444
0.444444        0.000000        0.333333        0.111111
computing tp and fp
0.400000        0.200000        0.600000        0.000000
0.250000        0.500000        0.000000        0.750000
computing Mb and Mp
0.666667        0.222222        0.555556        0.333333
0.425000        0.650000        0.200000        0.875000
computing Dynamic priority:
0.666667        0.650000        0.555556        0.875000
The sorted dynamic priority is:
0.875000        0.666667        0.650000        0.555556
Average waiting time: 8.250000
Average turnaround time: 14.250000


...Program finished with exit code 0
Press ENTER to exit console.
```

Figure 17: *Output for FUZZY_S, dynamic priority method.*

FUZZY_L

```
Enter number of processes:4
Enter Arrival Time for process 1: 0
Enter Arrival Time for process 2: 1
Enter Arrival Time for process 3: 2
Enter Arrival Time for process 4: 3
Enter execution time for process 1: 8
Enter execution time for process 2: 4
Enter execution time for process 3: 7
Enter execution time for process 4: 5
Enter priority 1: 2
Enter priority 2: 3
Enter priority 3: 1
Enter priority 4: 4
Assigned new priority 1: 5
Assigned new priority 2: 6.47
Assigned new priority 3: 3.86
Assigned new priority 4: 6.03

Average waiting time: 10
Average turnaround time: 16


...Program finished with exit code 0
Press ENTER to exit console.
```

Figure 18: *Output for FUZZY_L, new priority method.*

SCHEDULING ALGORITHM	AVERAGE WAITING TIME	AVERAGE TURNAROUND TIME
FCFS	14	9
SJF	13	7
SRTF	12	6
RR	18	12
PS	15	9
FUZZY_S	14	8
FUZZY_P	15	9
FUZZY_L	16	10

Figure 19: *Overall comparison of Average Waiting Time and Average Turnaround Time*

GRAPHS

- OS Basic Scheduling Algorithm:

Average Waiting Time:

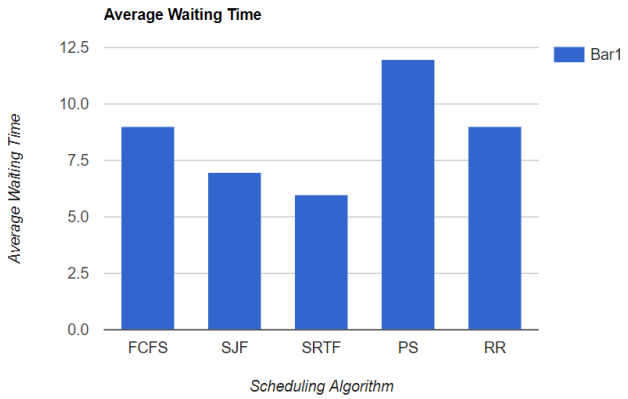


Figure 20: *Bar graph of the average waiting time for the scheduling algorithm.*

Average Turnaround Time:

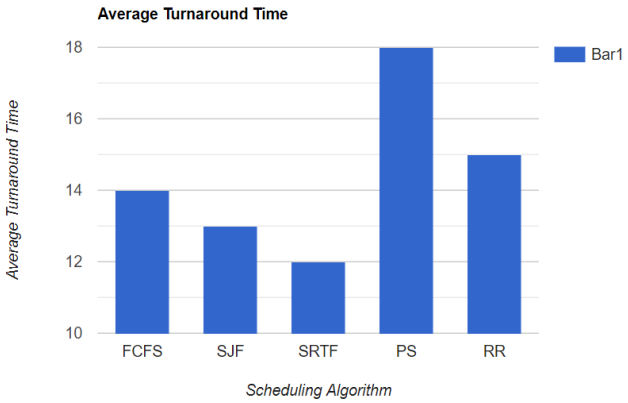


Figure 21: *Bar graph of the average turnaround time for the scheduling algorithm.*

- Proposed Scheduling Algorithm

Average Waiting Time:

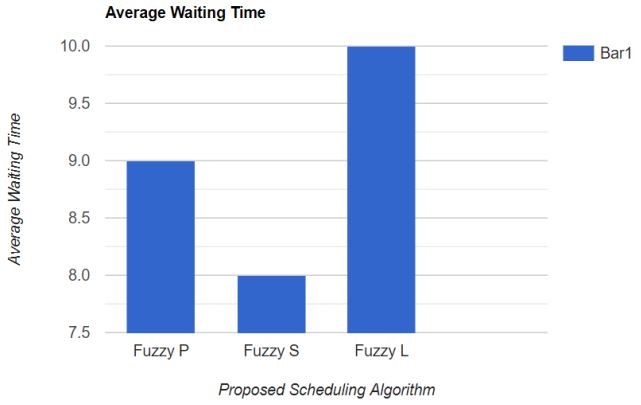


Figure 22: *Bar graph of the average waiting time for the proposed scheduling algorithm.*

Average Turnaround Time:

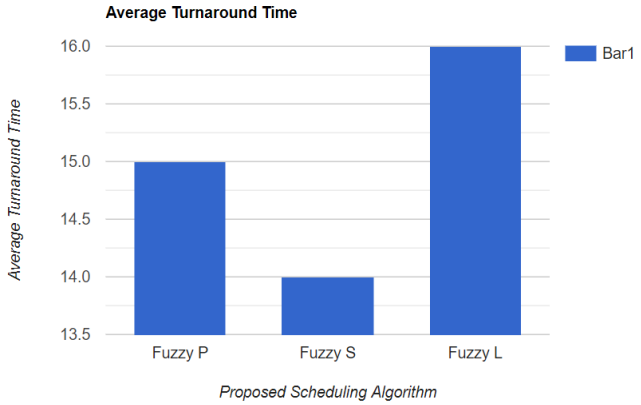


Figure 23: *Bar graph of the average turnaround time for the proposed scheduling algorithm.*

6. CONCLUSION

Scheduling is an intense process that is influenced by many factors. The existing basic operating system scheduling algorithms though good aren't practical to be applied in everyday situations. A system's need keeps on changing on a day-to-day base and the operating system of any system must be smart enough to adapt the system's need and use.

Artificial Intelligence has proven to revolutionize the usage of technology by i's ability to make the machine adapt to situations and learn from experience. Fuzzy Logic has opened the chances to change the way one approaches operating system's scheduling.

From the three proposed fuzzy approaches **Fuzzy_P, Fuzzy_S and Fuzzy_L** we can see how the same concept of changing the assigned priority according to different algorithms can change the output. From the discussion and example scenarios considered above we can say that **Fuzzy_S has the best values as it has the lowest Average Waiting Time and Average Turnaround Time** when computed. Thus, from this we can tell that all three approaches are unique in their own ways.

7. REFERENCE

[1]Tonogai, Dale, AI in Operating Systems: An Expert Scheduler, EECS Department, University of California, Berkeley, 1988, UCB/CSD-88-487. http://www2.eecs.berkeley.edu/Pubs/TechRpts/1988/6149.html, Tonogai:CSD-88-487

[2] Design and Implementation of Modified Fuzzy based CPU Scheduling Algorithm. Comments: 6 Pages Subjects: Operating Systems (cs.OS); Artificial Intelligence (cs.AI). Authors: Rajani Kumari, Vivek Kumar Sharma, Sandeep Kumar. Journal reference: International Journal of Computer Applications, Volume 77, No 17, September 2013. DOI: 10.5120/13612-1323 .Cite as: arXiv:1706.02621 [cs.OS] (or arXiv:1706.02621v1 [cs.OS] for this version)

[3] Analysis of Process Scheduling Using Neural Network in Operating System. Comments: 12 Pages Authors: Harshit Agarwal, Gaurav Jariwala. First Online: 30 January 2020. DOI: https://doi.org/10.1007/978-981-15-0146-3_97. Print ISBN: 978-981-15-0145-6. Online ISBN: 978-981-15-0146-3

[4] Lim S., Cho SB. (2007) Intelligent OS Process Scheduling Using Fuzzy Inference with User Models. In: Okuno H.G., Ali M. (eds) New Trends in Applied Artificial Intelligence. IEA/AIE 2007. Lecture Notes in Computer Science, vol 4570. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-73325-6_72

[5]Proposed Fuzzy CPU Scheduling Algorithm (PFCS) for Real Time Operating Systems.
Source: BVICAM's International Journal of Information Technology . Jul-Dec2013, Vol. 5 Issue 2, p583-588. 6p. Author(s): Ajmani, Prerna; Sethi, Manoj

[6]Implementing a Process Scheduler Using Neural Network Technology, Bex, P.J.G.I.

[7]Utilization of Fuzzy Logic in CPU Scheduling in Various Computing Environments Bailey Granam Computer Science Stetson University DeLand, Florida,

USA bgranam@stetson.edu Hala ElAarag Computer Science Stetson University DeLand, Florida, USA helaarag@stetson.edu

[8]Supriya Raheja, Reena Dhadich, Smita Rajpal, "Designing of 2-Stage CPU Scheduler Using Vague Logic", Advances in Fuzzy Systems, vol. 2014, Article ID 841976, 10 pages, 2014. https://doi.org/10.1155/2014/841976