

A Novel CPU Scheduling Algorithm–Preemptive & Non-Preemptive

Sukumar Babu Bandarupalli¹, Neelima Priyanka Nutulapati²,
Prof. Dr. P.Suresh Varma³

¹(Computer Science & Engg, Vijaya Institute of Technology for Women, India)

²(MCA & CSE Dept, SRK Institute of Technology, India)

³(Computer Science & Dean, Adikavi Nannaya University, India)

ABSTRACT: The main objective of this paper is to introduce a new CPU algorithm called A Novel CPU Scheduling Algorithm which acts as both preemptive and non-preemptive based on the arrival time. The proposed algorithm helps to improve the CPU efficiency in real time uni-processor-multi programming operating system. CPU Scheduling is the basis of multi-programmed operating system. The scheduler is responsible for multiplexing processes on the CPU. There are many scheduling algorithms available for a multi-programmed operating system like FCFS, SJF, Priority, Round Robin etc. In this paper, the results of the existing algorithms (FCFS, SJF, Priority and Round Robin) are compared with the proposed algorithm.

Keyword: Operating System, uni-processor, uni programming, multi-programming, Resource utilization, Scheduling, FCFS, SJF, Priority, Round Robin, ERR etc.

I. INTRODUCTION

Operating system performs variety of tasks in which scheduling is one of the basic task. Scheduling is heart of any computer system since it contains decision of giving resources between possible processes. Sharing of computer resources between multiple processes is also called scheduling [1]. Process is a smallest work unit of a program which requires a set of resources for its execution that are allocated to it by the CPU. These processes are many in number and keep coming in a particular fashion, different scheduling techniques are employed that enable faster and efficient process execution thereby reducing the waiting time faced by each process and increasing CPU utilization. A process has five basic states namely NEW, Ready, Running, Waiting and Terminate [1] [2].

Throughout its lifetime a process migrates between various scheduling queues by different schedulers until it gets terminated. These queues mainly contain the ready queue which contains set of processes ready for CPU response. The second queue is the device or the I/O queue which contains all the processes that are waiting for I/O response [1]. The operating system must select processes for scheduling from these queues in a specific manner. This selection process using a particular scheduling technique is carried out by schedulers. Schedulers in general try to maximize the average performance of a system according to the given criterion [3]. Scheduling algorithms are broadly classified into preemptive and non-preemptive scheduling disciplines.

The algorithm proposed in this article is both preemptive and non-preemptive in nature and attempts to give fair CPU execution time by focusing on average waiting time and turnaround time of a process.

This article comprises of the following sections: Section 2 presents scheduling parameters, which will decide against which parameters the new CPU algorithm will be tested. Section 3 introduces existing scheduling algorithms. Section 4 explains the proposed Algorithm – A Novel CPU Scheduling algorithm. Section 5 contains pseudo code of the proposed algorithm. Section 6 explains the two basic elements that make up the simulation and provide an interactive user interface. Section 7 presents a graphical comparison of the new algorithm with existing algorithms. Section 8 will provide conclusion of the work.

II. SCHEDULING PARAMETERS

THERE ARE DIFFERENT SCHEDULING ALGORITHMS WITH DIFFERENT CHARACTERISTICS WHICH DECIDE SELECTION OF PROCESS USING DIFFERENT CRITERIA FOR EXECUTION BY CPU. THE CRITERIA FOR A GOOD SCHEDULING ALGORITHM DEPEND, AMONG OTHERS, ON THE FOLLOWING MEASURES [1].

- A. **CPU Utilization:** It is the average fraction of time, during which the processor is busy [2, 4].
- B. **Throughput:** It refers to the amount of work completed in a unit of time. The number of processes the system can execute in a period of time. The higher the number, the more work is done by the system [4].
- C. **Waiting Time:** The average period of time a process spends waiting. Waiting time may be expressed as turnaround time less the actual execution time [4].
- D. **Turnaround time:** The interval from the time of submission of a process to the time of completion is the turnaround time [4].
- E. **Response time:** Response time is the time from submission of a request until the first response is produced [4].
- F. **Priority:** give preferential treatment to processes with higher priorities [4].
- G. **Fairness:** Avoid the process from starvation. All the processes must be given equal opportunity to execute [4].

III. OVERVIEW OF EXISTING CPU SCHEDULING ALGORITHMS.

A. *FIRST COME FIRST SERVED (FCFS) Scheduling.*

It is the simplest CPU Scheduling algorithm. The criteria of this algorithm is 'the process that requests first, holds the CPU first' or which process enter the ready queue first is served first [3]. The workload is processed in the order of arrival time, with no preemption [2]. Once a process has been submitted to the CPU, it runs into completion without being interrupted. Such a technique is fair in the case of smaller processes but is quite unfair for long unimportant job [5]. Since FCFS does not involve context switching therefore it has minimal overhead. It has low throughput since long processes can keep processor occupied for a long time making small processes suffer. As a result waiting time, turnaround time and response time can be low [6].

B. *Shortest Job First (SJF) Scheduling.*

The criteria of this algorithm are which process having the smallest CPU burst, CPU is assigned to that process next. If two process having the same CPU burst time FCFS is used to break up the tie [3]. SJF can be worked as preemptive and non-preemptive in nature based on the arrival time and burst time of the processes. SJF reduces average waiting time of the processes as compared to FCFS. SJF favors shorter processes over longer ones which is an overhead as compared to FCFS. It selects the job with the smallest burst time ensuing CPU availability for other processes as soon as the current process reaches its completion. This prevents smaller processes from suffering behind larger processes in the ready queue for a longer time [5] [7].

C. *Priority Based Scheduling.*

In this algorithm, priority is associated with each process and on the basis of that priority CPU is allocated to the processes. Higher priority processes are executed first and lower priority processes are executed at the end. If multiple processes having the same priorities are ready to execute, control of CPU is assigned to these processes on the basis of FCFS [1]. Priority Scheduling can be preemptive and non-preemptive in nature.

D. *Round Robin (RR) Scheduling.*

It is a preemptive scheduling algorithm. It is designed especially for time sharing systems. In this algorithm, a small unit of time called time quantum or time slice is assigned to each process [2]. When the time quantum expired, the CPU is switched to another process. Performance of Round Robin totally depends on the size of the time quantum.

IV. PROPOSED WORK: NOVEL CPU SCHEDULING ALGORITHM

The proposed algorithm A Novel CPU Scheduling algorithm is both preemptive and non-preemptive in nature. In this algorithm a new factor called condition factor (F) is calculated by the addition of burst time and arrival time ie., $F = \text{Burst Time} + \text{Arrival Time}$.

This factor f is assigned to each process and on the basis of this factor process are arranged in ascending order in the ready queue. Process having shortest condition factor (F) are executed first and process with next shortest factor (F) value is executed next. By considering the arrival time the new algorithms acts as preemptive or non-preemptive.

Proposed CPU scheduling algorithm reduces waiting time, turnaround time and response time and also increases CPU utilization and throughput.

The working procedure of A Novel Preemptive Scheduling of Preemptive and Non Preemptive algorithm is as given below:

- ✓ Take the list of processes, their burst time and arrival time.
- ✓ Find the condition factor F by adding arrival time and burst time of processes.
- ✓ First arrange the processes, burst time, condition factor based on arrival time ascending order.
- ✓ Iterate step a, b until burst time becomes zero.
 - a. If arrival time of first and second process are equal the arrange them based on their condition factor f.
 - b. Decrement the burst time and increment arrival time by 1
- ✓ When burst time becomes find the waiting time and turnaround time of that process.
- ✓ Average waiting time is calculated by dividing total waiting time with total number of processes.
- ✓ Average turnaround time is calculated by dividing total turnaround time by total number of processes.

V. PSEUDO CODE

Initialization variables

Burst time[n] \leftarrow 0
Arrival time[n] \leftarrow 0
Numprocess[n] \leftarrow 0
Factor[i] \leftarrow 0
Turn[n] \leftarrow 0
Wait[n] \leftarrow 0
Temp \leftarrow 0
Current time \leftarrow 0
Wait time=0

```
Turn time=0
Avgwaitime=0.0
Avgturntime=0.0
Read burst time[n] and arrival time[n]
Compute factor[i] <- bursstime[i] + arrival time[i]
```

a. Pseudo code for Non Preemptive

```
Arrange the elements in ascending order based on condition factor
For i ← 0 to n-1
  For j ← I to n
    If factor[i] > factor[j]
      Temp ← burst time[i]
      Burst time[i] ← burst time[j]
      Burst time[j] ← temp
      Temp ← arrival time[i]
      Arrival time[i] ← arrival time[j]
      Arrival time[j] ← temp
      Temp ← factor[i]
      Factor[i] ← factor[j]
      Factor[j] ← temp
  End for
For i ← 0 to n
  Begin
    Wait[i]=wait[i]+burst time[i]
    Turn[i]=wait[i]+btime[i]
    Wait time = wartime + wait[i]
    turn time = turn time + turn[i]
  End
Avgwaittime = wait time/n
Avgturntime = turn time/n
```

b. Pseudo code for Preemptive

```
Arrange the elements in ascending order based on arrival time
For i ← 0 to n-1
  For j ← I to n
    If factor[i] > factor[j]
      Temp ← burst time[i]
      Burst time[i] ← burst time[j]
      Burst time[j] ← temp
      Temp ← arrival time[i]
      Arrival time[i] ← arrival time[j]
      Arrival time[j] ← temp
      Temp ← factor[i]
      Factor[i] ← factor[j]
      Factor[j] ← temp
  End for
For i ← 0 to n-1
  For j ← I to n
    If atime[i] == atime[j] then sort elements in ascending order based on factor[i] and factor[j]
    If burs time! =0 && atime==current time
      Begin
        Atime[i] =atime[i]+1
        Btime[i] =btime[i]-1
        Current time++
        If btime[i]==0
          turn time=current time
      End
  End
For i ← 0 to n
  Begin
    Wait [i]=turn[i]-btime[i]-atime[i]
    turn [i]=turn[i]+atime[i]
    Wait time = wartime + wait[i]
    Turn time = turn time + turn[i]
```

End

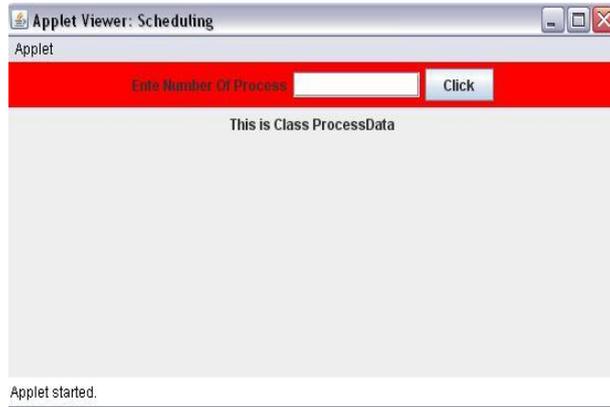
Avgwaittime = wait time/n

Avgturntime = turn time/n

VI. SIMULATION DESIGN

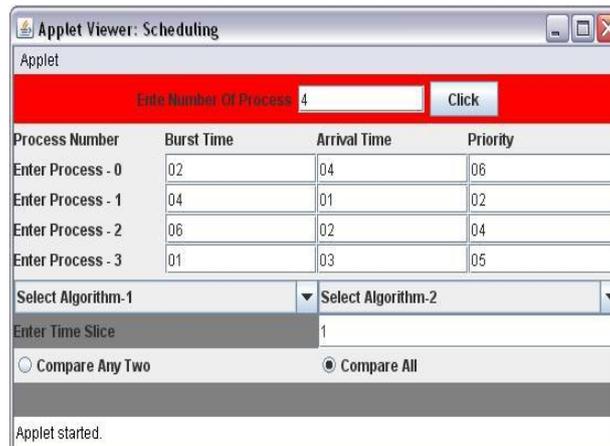
The simulation provides an interactive GUI interface to the user through which a user can input data related to different processes then applying different algorithms based on the algorithm choices given in the simulator. The simulator employs JAVA swings.

The front end user interfaces are designed using java awt's and swings. The parent screen i.e., screen1 allows the user to enter number of processes to be in execution.



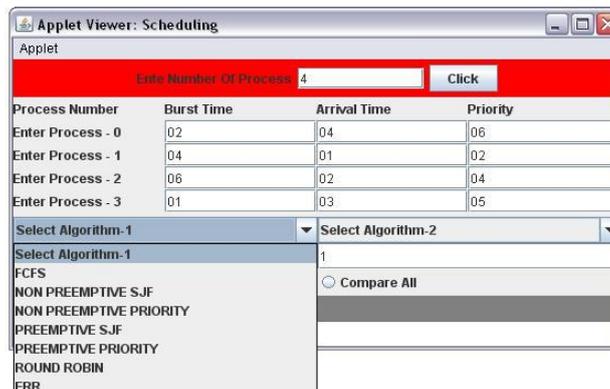
Screen1 : Enter Number of Process

Screen 2 allows the user to enter the details of the processes burst time, arrival time and priority.



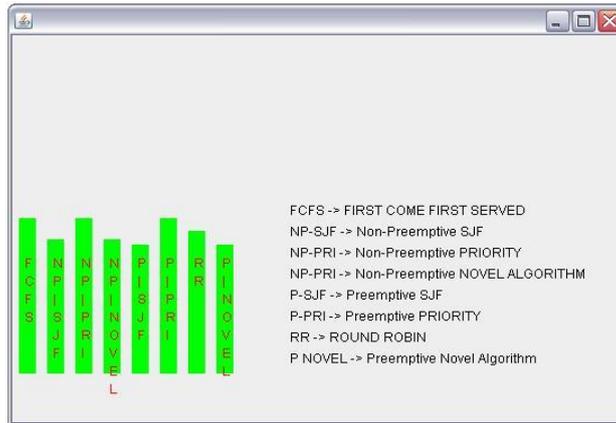
Screen 2: Enter Burst , Anival and Priority of processes

Screen 3 shows that a user can compare any two algorithms of his choice or he can compare all the scheduling algorithms.



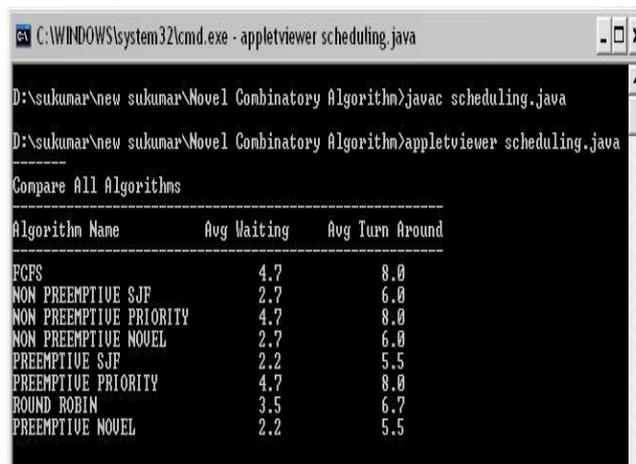
Screen 3: Selection of Algorithm1 and Algorithm2

Screen 4 shows the Graphical Comparison of all the algorithms



Screen 4 : GUI Comparison of All Algorithms

Screen 5 shows the non-graphical comparison of all the algorithms

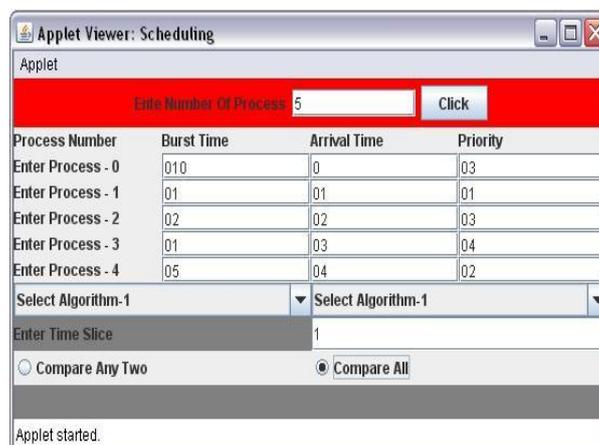


Screen 5 : CUI Comparison of All Algorithms

VII. COMPARISON OF PROPOSED ALGORITHM WITH OTHER CPU SCHEDULING ALGORITHMS.

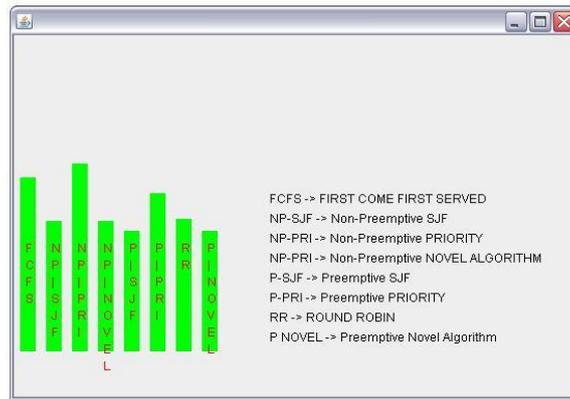
To compare the performance of the proposed scheduling algorithm it was implemented and compared with the existing scheduling algorithm. GUI and CUI figures, shows the comparison between the proposed algorithm with existing algorithms.

Example 2: Screen 6 takes different processes burst time and arrival time and priority values



Screen 6: Example 2 for different set of Processes

Screen 7 shows the graphical representation of example 2



Screen 7: Example 2 GUI Comparison

Screen 8 shows the non-graphical representation of example 2

```
C:\WINDOWS\system32\CMD.exe - appletviewer Scheduling.java
D:\sukumar\new sukumar\Novel Combinatory Algorithm>javac Scheduling.java
D:\sukumar\new sukumar\Novel Combinatory Algorithm>appletviewer Scheduling.java
-----
Compare All Algorithms
-----
Algorithm Name      Avg Waiting      Avg Turn Around
-----
FCFS                7.6              11.4
NON PREEMPTIVE SJF 3.2              7.0
NON PREEMPTIVE PRIORITY 9.0              12.8
NON PREEMPTIVE NOVEL 3.2              7.0
PREEMPTIVE SJF     2.2              6.0
PREEMPTIVE PRIORITY 6.0              9.8
ROUND ROBIN        3.4              7.2
PREEMPTIVE NOVEL  2.2              6.0
```

Screen 8: Example 2 CUI Comparison

Example 3: Screen 9 takes different processes burst time and arrival time and priority values

Applet Viewer: Scheduling

Enter Number Of Process: 6 Click

Process Number	Burst Time	Arrival Time	Priority
Enter Process - 0	04	01	04
Enter Process - 1	06	02	01
Enter Process - 2	010	03	08
Enter Process - 3	05	04	02
Enter Process - 4	020	05	03
Enter Process - 5	01	06	06

Select Algorithm-1: Select Algorithm-1

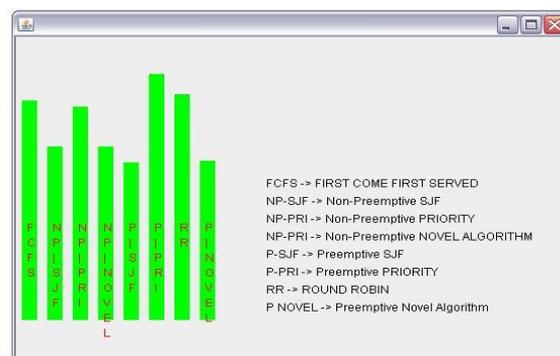
Enter Time Slice: 1

Compare Any Two Compare All

Applet started.

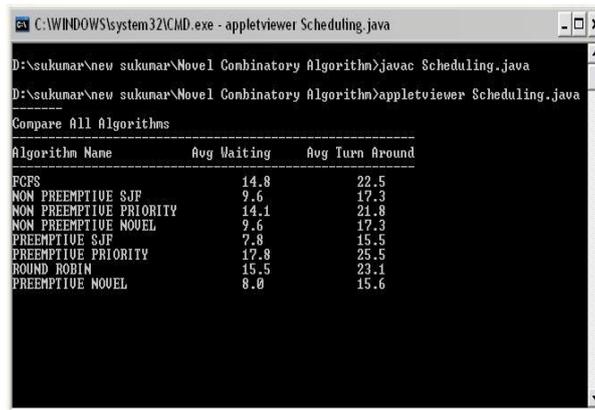
Screen 9: Example 2 for different set of processes

Screen 10 shows the graphical representation of example3



Screen 10: Example 3 GUI Comparison

Screen 11 shows the non-graphical representation of example3



```
C:\WINDOWS\system32\CMD.exe - appletviewer Scheduling.java
D:\sukunar\new sukunar\Novel Combinatory Algorithm>javac Scheduling.java
D:\sukunar\new sukunar\Novel Combinatory Algorithm>appletviewer Scheduling.java
Compare All Algorithms
-----
Algorithm Name      Avg Waiting      Avg Turn Around
-----
FCFS                 14.8             22.5
NON PREEMPTIVE SJF  9.6              17.3
NON PREEMPTIVE PRIORITY 14.1            21.8
NON PREEMPTIVE NOVEL 9.6              17.3
PREEMPTIVE SJF      7.8              15.5
PREEMPTIVE PRIORITY 17.8             25.5
ROUND ROBIN         15.5             23.1
PREEMPTIVE NOVEL   8.0              15.6
```

Screen 11: Example 3 CUI Comparison

VIII. CONCLUSION

The paper presents a new CPU scheduling algorithm called A Novel Pre-emptive and Non Pre-emptive CPU Scheduling Algorithm. Paper also contains simulation interface and its working, which interactively takes input from the user and compares the process set against different algorithm pairs. The result of the simulation for different process sets using different scheduling algorithms has been presented graphically in this piece of work. The last half of the paper provides analytical result with each set of graph. From the above graphs and results from example 1, 2 and 3, it is clear that proposed algorithm is more efficient than FCFS, Pre-emptive Priority and Non Pre-emptive Priority, Round Robin. And also observed that proposed algorithm gives almost equal performance like a SJF Pre-emptive and Non Pre-emptive algorithm

REFERENCES

- [1] Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, "Operating System Concepts", Sixth Edition.
- [2] Milan Milenkovic, "Operating Systems Concepts and Design", McGRAM-HILL, Computer Science Series, second edition.
- [3] P. Balakrishna Prasad, "Operating Systems" Second Edition.
- [4] A. Dhore "Opeating Systems", Technical Publications.
- [5] M. Dietel, "Operating Systems", Pearson Education, Second Edition.
- [6] <http://en.wikipedia.org/wiki/Scheduling>
- [7] M Gary Nutt, "Operating systems – A Modern Perspective, Second Edition, Pearson Education, 2000.