Common part for RMS & EDF algorithms

We are having structure for process which will store its information

Some important elements in structure are

Available time: it will tell us time at when the process is ready to execute ,for example if process is ready to execute at 50 secs
Then available time will be equal to 50secs

Deadline time: this will tell the deadline time for a process,for Example if process has deadline at 200secs,then deadline time
Will be equal to 200secs

Occur: this will keep track of number of times a process has executed

Remaining time: this will keep track of remaining time that process has to execute,for example: if process has processing time of 50 secs and it has executed 20 secs already then remaining time will be 30 secs

Behaviour of functions

Setup_process(y): this function will take process of index y to its next round by updating available time,deadline time,remaining time and occur

isdeadlinemissed(y):for suppose if we run remaining time of process of index y this function will return true if its deadline
Is going to miss else it will return false
And if deadline is going to miss then we will take that process to its next round

Design of RMS algorithm

NOTE :
1)we will have array of process
2) we will sort all process based on their priority in our main function
3)we will have timer variable which will tell time at current situation

Continuation of remaining functions

can_prempt_and_deadline(x,y):

Process of index x should have more priority than process of index y because at time of calling this function only we will take care Of this.
Let's have two variables a,b
a stores the sum of timer and remaining time of execution of process of index y
b stores the available time of process of index x
If a is greater than b then we can say that process x is available before time at when process y will completes its execution and process of index x will have higher priority than process of index y so process of index x can preempt process of index y and if we run process of index x after preempting process of index y then process of index x deadline should not miss so will also check whether process of index x will be going to miss its deadline or not.
And finally this function will return true if process of index x can preempt process of index y and process of index x should not miss its deadline when it Gets chance to execute
,in all other cases this function will return false

preempt(x):

Question : what this function will do?

Answer: if any other process can preempt process of index x then
This function will return index of that process which can preempt process Of index x,if no process can preempt process of index x then this function will return -1
NOTE:process of index 0 will have higher priority than process of index 1, because the process array is in sorted order based on priority

Implementation:

Let's take set S,this set S contains all the processes indexes that are having index less than x, and for every element z that belongs to S **can_prempt_and_deadline(z,x)** function should return true
And now in set S we are having all indexes of processes which are having higher Priority than process of index x, And all processes which are having indexes in set S can be able to preempt Process of index x
And all processes which are having indexes in set S are not going to miss deadline when it takes CPU after preempting process of index x

If our set S is empty means no one can preempt process of index x,so Process x will finish its execution and timer is updated to time at when process of index x finishes its execution and this function will return -1

If our set S is not empty ,then we will look for process which is having minimum **Available time** in set S,let's say that index is **k** ,
So here k is index of process which is having minimum **Available time** and this function will return **k**
we will update timer to minimum Available time of process of index **k**

process_selector():

Let's take set S,this will contain indexes all processes which are ready to execute
If timer>=available time of process of index p,then we can say that Process of index p is available or ready to execute
If set S is not empty, then we will return smallest index in set S

If set S is empty ,means no processes are ready to execute then we will return index of process which is having minimum **Available time**

If all processes has reached their occurrence limit then this function will return -1

Main function:

Let x,pid be variable

Step1: we will sort array of process based on their priority

Step2: initialize x to **process_selector()**

Step3: assign pid to **preempt(x)**

Step4: check whether any process are missing their deadlines are not if they are missing then take that process to it's next round

```
Step5:
 1)if pid is equal to -1 then
         1)take process of index x to its next period
         2) assign x to  process_selector()
         3) if x is equal to -1 then
               1)  exit
         4)If x is not equal to -1  then
            1)we need to check whether process of index x is ready
               To execute or not ,
            2)if it is not ready then
               1) we should assign timer to available time of process
                  Of index x
            3)Go to Step3                                    ,
 2)if pid is not equal to -1 then
              1) assign x to pid and go to Step3
```

Design of EDF algorithm

NOTE :
1)we will have array of process
2)we will have timer variable which will tell time at current situation

Continuation of remaining functions

resolve(x,y):if process y is having higher priority than x this will return true else it will return false

can_prempt_and_deadline(x,y):the difference between this function in rms and edf is in rms it will consider priority based on periods,but here priority will be considered based on Deadline time
And remaining all things are same for both rms and edf

preempt(x):
Let’s take set S,this set S contains all the processes indexes that are having index less than n, and for every element z that belongs to S
can_prempt_and_deadline(z,x) function should return true
And now in set S we are having all indexes of processes which are having higher Priority than process of index x, And all processes which are having indexes in set S can be able to preempt Process of index x
And all processes which are having indexes in set S are not going to miss deadline when it takes CPU after preempting process of index x

If our set S is empty means no one can preempt process of index x,so Process x will finish its execution and timer is updated to time at when process of index x finishes its execution and this function will return -1

If our set S is not empty ,then we will look for process which is having minimum **Available time** in set S,let’s say that index is **k** ,
So here k is index of process which is having minimum **Available time** and this function will return **k**
we will update timer to minimum Available time of process of index **k**

process_selector():

If no processes are ready to execute then this function in rms and edf will behave same

But if some processes are ready to execute then we need to select

The process which is having minimum Deadline time

And we need to return that index for which process is having minimum Deadline time

Main function:

Let x,pid be variable

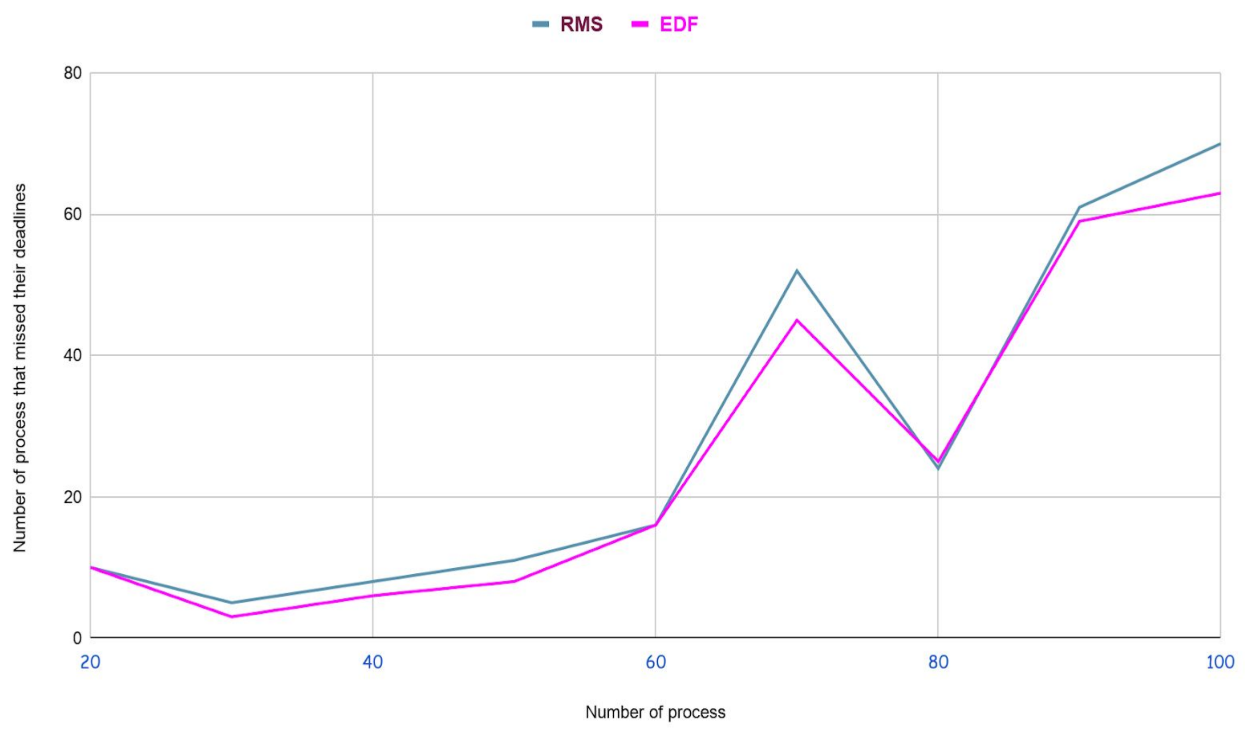
Step1: initialize x to **process_selector()**

Step2: assign pid to **preempt(x)**

Step3: check whether any process are missing their deadlines are not if they are missing then take that process to it's next round

```
Step4:
 1)if pid is equal to -1 then
        1)take process of index x to its next period
        2) assign x to  process_selector()
        3) if x is equal to -1 then
              1)  exit
        4)If x is not equal to -1  then
           1)we need to check whether process of index x is ready
              To execute or not ,
           2)if it is not ready then
              1) we should assign timer to available time of process
                 Of index x
           3)Go to Step2                                        ,
 2)if pid is not equal to -1 then
             1) assign x to pid and go to Step2
```

Analysis of Graph1:

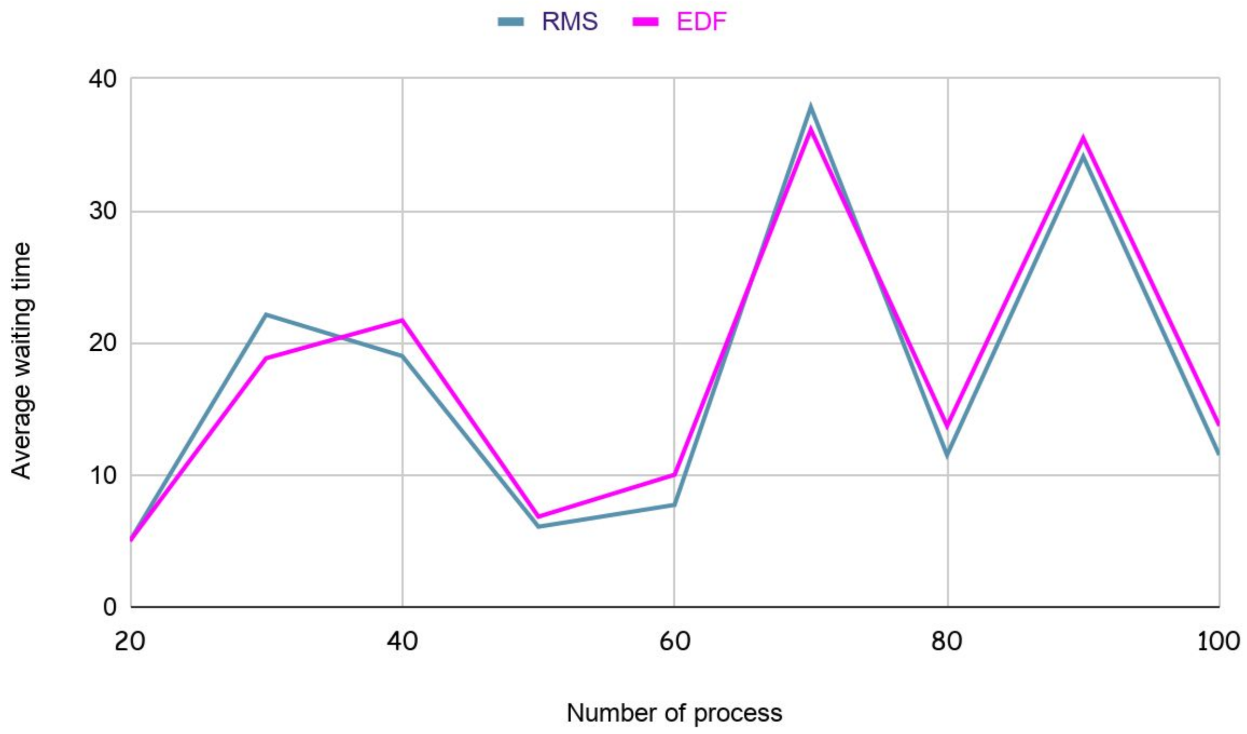


Let n be Number of process
Usually number of deadlines missed should be more or equal for RMS compared to EDF,because in EDF we are giving chance to process whose deadline is early so that there will high chance for every process to meet it’s deadline
Here for all values of **n from 20 to 100** except at n=80,the deadlines missed for RMS is greater than or equal to EDF,because more Chance in EDF will be given for a process to meet its deadline
At n=80
Let’s consider below case
consider process “p” with high processing time and its period is also near to it’s processing time.
In RMS ,process “p” may get less CPU time compared to other processes because when the process with higher priority is available,it will take CPU
In EDF,process “p” may get more CPU time compared to RMS,because Process p may have deadline before all other process ,so no one can Preempt

so in that case there is possibility that the process with more processing time gets CPU more time in EDF compared to RMS
In that time which was spent on executing high processing time
RMS could spent that time on executing some 2 or more other number of process Which are having low processing time
,so in that case EDF will have Less number of process which had successfully completed than RMS,
So EDF will have more number of process that miss their deadlines than RMS

So the input which was given may face above situation so that's why
At n=80 number of process that missed their deadlines will be more in EDF as Compared to RMS

Analysis of Graph 2



waiting time for a process for a particular round if it executed successfully=(time at which it finishes its execution - time at which process is ready to execute - execution time of process)

Here “time at which process is ready ,execution time of process” is same for both RMS and EDF
So waiting time for process is directly proportional to the time at when process finishes its execution
waiting time for a process for a particular round if it is not executed successfully=period of that process
NOTE: input is given same for both Graphs
Consider two cases:
Case 1)number of deadlines missed in EDF>=number of deadlines missed in RMS
Assuming EDF may miss the processes with high period Because of given input
RMS may miss processes of high period because processes of high period will have lower priority
If process have very less processing time then only it can execute successfully else it will miss its deadline
In this case,number of processes that successfully completed its execution in RMS >= EDF,
Here RMS will execute it’s high priority processes before others ,in RMS whenever higher priority process is ready to execute CPU will be given to that process so the process can finish it’s execution before it’s deadline and this process in RMS will complete it’s Execution before EDF could complete that means if RMS finish it’s execution at time X ,EDF will finish it’s execution at time “>=X” ,EDF will finish execution at higher time than RMS because In EDF ,For a particular process the CPU will be given to it whenever It is nearing it’s deadline,so EDF will finish its execution just few seconds before it’s deadline, there is chance that in RMS more processes can miss its deadline but this case is for When number of deadlines missed in EDF> =number of deadlines missed in EDF
So waiting time for processes in RMS should be less in this case compared to EDF because in RMS more processes will complete

its execution early compared to EDF so RMS will wait less time compared to EDF

Conclusion: in these case we will have average waiting time greater Or equal for EDF compared to RMS in most of subcases

Observation:at n=20,n=60,n=80 in **Graph1** we are having deadline missed greater or equal for EDF compared to RMS,
So by above argument the average waiting time for EDF should be greater or equal compared to RMS
At n=20 average waiting time for both are equal
At n=60,n=80 average waiting time for EDF is greater than RMS
So given input may leads EDF to miss processes which
Are having higher period,so because of that we got better performance for RMS compared to EDF
If EDF miss processes which are having lower period ,then EDF may give better performance than RMS

Case2) number of deadlines missed in RMS > number of deadlines missed in EDF

Possible case for where RMS can have more average waiting time than EDF is:

If the more process which are skipped by RMS have low processing time
And the more process which are skipped by EDF have high processing time,Then RMS will execute processes which are having high processing time So because of that some process may need to wait more time to get CPU,whereas in EDF we are executing low processing time Processes ,so other processes will wait less time in EDF compared to RMS,and RMS will skip processes which are having higher period
so more number will be added to waiting time

At n=30,n=70 our input may leads to above case ,so we are getting more average time for RMS compared to EDF

Possible case for where EDF can have more average waiting time than RMS is:

If number of deadlines missed in RMS is more greater than number of deadlines missed in EDF then RMS may have more Average waiting time compared to EDF
BUT below explanation **is for when number of deadlines missed in RMS is greater but not so much greater than number of deadlines missed in EDF**
If EDF skips more processes which are having low processing time,then
This means it will execute more processes with high processing time,so
Other processes may need to wait so much time this is one disadvantage for EDF
And another disadvantage for EDF, is that EDF will finish execution when it is nearing its deadline whereas in RMS the process will finish it's execution before EDF could finish,as waiting time is proportional to finish time RMS will finish in less time so it should have less average waiting time Compared to EDF
And advantage for EDF is number of deadlines missed in RMS is greater than EDF but it's not much greater ,so this advantage cannot compensate the two disadvantages,so considering this cases
We can say EDF may have more average waiting time compared to RMS
So our given input at n=40,50,90,100 may leads to above cases,so we are getting more average waiting time for EDF compared to RMS