Abstract

Round Robin (RR) algorithm is widely used in modern operating systems (OS) as it has a better responsiveness as periodic quantum (occurring at regular intervals) in addition to have a good feature such as low scheduling overhead of n processes in a ready queue which takes a constant time O(1). But, RR algorithms have the worse features such as having low throughput, long average turnaround and waiting time, in addition to the number of context switches for (n) processes is (n) switches.

Shortest Job First (SJF) however, itself is not practical in time sharing Oss due to its low response. More over the scheduling overhead of n processes in a ready queue is O(n), But the good features of SJF algorithm are the best average turnaround time and waiting time.

By considering a static set of n processes, desirable features of CPU scheduler to maximize CPU utilization, response time and minimize waiting time and turnaround time are obtained by combining the kernel properties of SJF algorithm with the best features of RR algorithm to produce a new algorithm as an original and novel algorithm called; " *Hybrid CPU Scheduling algorithm SJF-RR in Static Set of Processes* " which, proposed in this research.

The proposed algorithm is implemented through an innovative *optimal* equation to adapt time quantum factor for each process in each round as a periodic quantum (occurred at irregular intervals). That is while applying proposed algorithm, mathematical calculations take place to have particular time quantum for each process.

Once, a criterion has been selected for comparison, deterministic modeling with the same numbers for input is proven that proposed algorithm is the best.

1. *Introduction*

Operating system (OS) always need to be more complex than ever before. As we know in multitasking environment the Scheduling algorithms are the mechanism by which the resource CPU is allocated to a process, which requires careful attention to assure fairness and avoid process starvation. Scheduling decision try to minimize the following: turnaround time, response time, and average waiting time for processes and the number of context switches because CPU remains idle during context switches, which reduces CPU utilization [1, 6–B]. There are many schedulers to select a new process quit often. A process may execute only a few milliseconds before waiting for an I/O request. Because of the short duration between executions, the schedulers must be very fast. There are two types of schedulers; Preemptive and Non-Preemptive algorithms. Preemptive algorithms include: Shortest-Job-First algorithm (SJF), Priority algorithm and Round-Robin algorithm.

2. *Objective*

"*Hybrid CPU Scheduling algorithm SJF-RR in Static Set of Processes*" has the goal for achieving a good result of significant advantage to reduce the number of processes in the ready queue on the basis of finishing short jobs relatively faster than other. This strategy is in a hope to increase the throughput and response and to reduce the turnaround time and the average waiting time.

3. *Background*

3-1 *Schedulers types*

An OS has many schedulers. There are two main CPU schedulers as in figure (1):

1. The LONG –TERM scheduler: It is a job scheduler, determines which jobs are admitted to the system for processing. In a batch system, there are often more jobs submitted than can be executed immediately. These jobs are spooled to mass storage device, where they are kept for latter execution the LONG-TERM scheduler selects jobs from this job pool and loads them into memory for execution as in figure (1). The

LTS executes much less frequently. It may be minutes between the arrivals of new jobs in the system. The LTS controls the degree of multiprogramming, i.e., the number of processes in memory [1, 2].

2. The SHORT-TERM scheduler: It is a CPU scheduler, selects from among these jobs in memory, which are ready to execute, allocates the CPU to one of them as in figure (1).
The STS, on the other hand, must select a new process quit often. A process may execute only a few milliseconds before waiting for an I/O request. Because of the short duration between executions, the STS must be very fast [1].

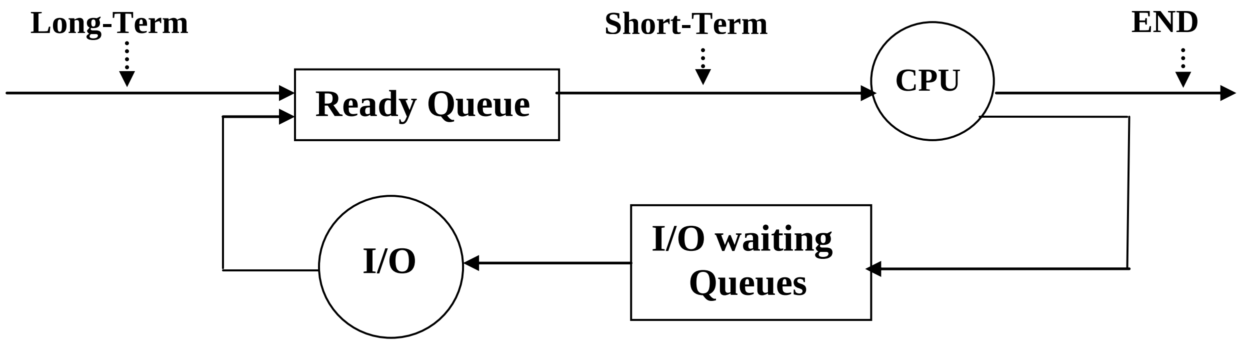


Figure (1) Scheduler and Queuing diagram

There are two types for STS; Preemptive and Non-Preemptive algorithms. Preemptive algorithms include: Shortest-Job-First Algorithm (SJF), Priority Algorithm and Round-Robin Algorithm.

3-2 Performance Criteria

CPU scheduling deals with the problem of deciding which of the processes in the ready queue is to be allocated the CPU. There are many different CPU scheduling algorithms. Many criteria have been suggested for comparing CPU scheduling algorithms. These criteria include: [3]

1. CPU utilization: It is very serious to keep CPU as busy as possible. CPU utilization may range from 0 percent to 100% percent. In a real system, it should range from 40 to 90% percent.
2. Throughput: Measure of work being done is the number of jobs completed per time unit.
3. Turnaround time: The interval from the time the process being submission to the time of completion is the turnaround time.
4. Waiting time: The CPU scheduling algorithm is not really affected by the amount of time that a job being executed but the amount of time that each job spends waiting for I/O in ready-queue.
5. Response time: The length of time between when the job arrives in the system and when it starts to produce output. For interactive jobs not like in static set of job, the response time might be more important than turnaround [4].

3-3 Traditional Algorithms in literature

As mentioned before Preemptive algorithms include: Shortest-Job-First algorithm (SJF), Priority algorithm and Round-Robin algorithm. Deterministic modeling is simple and fast measurement to determine that the minimal algorithm; which has the minimum value, is the best. It gives exact numbers allowing the algorithms to be compared. However, it requires exact numbers for input and its answers apply only to those cases [1, 6-A].

<u>First:</u> Shortest-Job-First Algorithm (SJF)

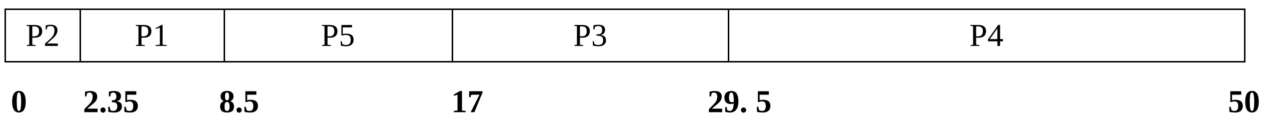
SJF is provably *optimal*, in that it gives the minimum average waiting time for a given set of jobs. With a nonpreemptive algorithm for static set of processes that assumes the run times are known in advance. In a company, for example, people can predict quite accurately how long it will take to run these processes, since similar work is done every day. When several equally important jobs are sitting in the input queue waiting to be started, the scheduler picks the shortest job first [5]. The proof shows that moving a short job before a long one decrease the waiting time of the short job more than it increases the waiting time of the long job.

Consequently, average waiting time is decreased. The real difficulty with SJF is how to know the length of the next CPU request yet SJF is frequently used in job scheduling. When the CPU is available, it is assigned to that job with the smallest next CPU burst. If two jobs have the same next CPU burst, First Come First Served (FCFS) is used. But although SJF is optimal, it cannot be implemented as Short-Term CPU scheduling level in time sharing.

Example

Consider the following set of jobs to implement SJF scheduler using Gantt-chart:

Job	Burst Time
1	6.15
2	2.35
3	12.5
4	20.5
5	8.5
Total Burst Time = 50	



The average waiting time = [0 + 2.35 + 8.5 + 17 + 29.5] / 5
= 57.35 / 5
= 11.47 t . . . . (1)

The average turnaround time = [2.35 + 8.5 + 17 + 29.5 + 50] / 5
= 107.35 / 5
=21.47 t . . . . (2)

<u>Second</u>: Priority Algorithm

SJF is a special (which is the best) case of the general *priority-scheduling* algorithm. A priority is associated with each process, and the CPU is allocated to the job with the highest priority. Equal priority jobs are scheduling FCFS. A SJF algorithm is simply a priority algorithm where the priority is the inverse of the predicated next CPU burst. The larger the CPU burst, the lower the priority, and vice versa.

<u>Priority can be defined either internally or externally</u>: [1]

1- Internally derived priorities use some measurable quantities to define the priority of a process. For example, time limits, memory requirements, the number of open files, etc,…
2- External priorities are set by criteria which are external to the O.S, such as the type and amount of funds being paid for computer use, the department sponsoring the work, and other external, often political factors.

A major problem with priority scheduling algorithm is *indefinite blocking* or *starvation*. A process which is ready to run but lacking the CPU can be considered blocked, waiting for the CPU.

A priority scheduling algorithm can leave some low-priority processes waiting indefinitely for the CPU. Generally, one of these things will happen:

1- Either the job will eventually be run.
2- The computer system will crash and lose all unfinished low-priority jobs.

3- Another solution to the problem of indefinite blockage of low-priority jobs is *aging which is* a technique of gradually increasing the priority of job that stay in the system for a long time.

Third: Round-Robin Algorithm (RR)

Another scheme for preventing long bursts from getting too much priority is a preemptive strategy called *round-robin* (RR). RR keeps all the bursts in a queue and runs the first one, like FCFS. But after a length of time *q* (called a *quantum*), if the current burst hasn't completed, it is moved to the tail of the queue and the next burst is started [5].

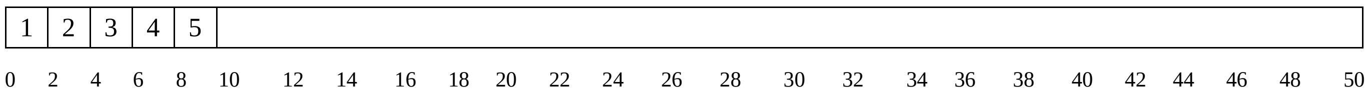
A time quantum is generally from 10 to 100 ms so no process allocates the CPU for more than one time quantum in each round. If its CPU burst exceeds a time quantum, it will be preempted and put back in the ready queue and treated as a circular queue. So Round-robin is a preemptive scheduling algorithm.

Example: consider the following set of jobs, which were previously used to illustrate SJF where the time quantum equal 2:

Job	Burst Time
1	6.15
2	2.35
3	12.5
4	20.5
5	8.5
Total Burst Time =	50

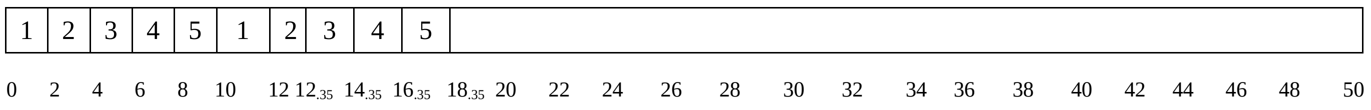
Round1

job	Burst Time (B_{ij})	Burst Time ($B_{i(j+1)}$) = B_{ij}-q
1	6.15	6.15 – 2 = 4.15
2	2.35	2.35 – 2 = 0.35
3	12.5	12.5 – 2 = 10.5
4	20.5	20.5 – 2 = 18.5
5	8.5	8.5 – 2 = 6.5
$\sum(B_{ij})$ = 50		$\sum(B_{i(j+1)})$= 40



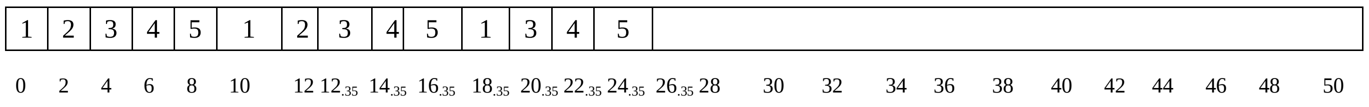
Round2

job	Burst Time (B_{ij})	Burst Time ($B_{i(j+1)}$) = B_{ij}-q
1	4.15	4.15 – 2 = 2.15
2	0.35	0.35 – 2 = 0
3	10.5	10.5 – 2 = 8.5
4	18.5	18.5 – 2 = 16.5
5	6.5	6.5 – 2 = 4.5
$\sum(B_{ij})$ = 40		$\sum(B_{i(j+1)})$= 31.65



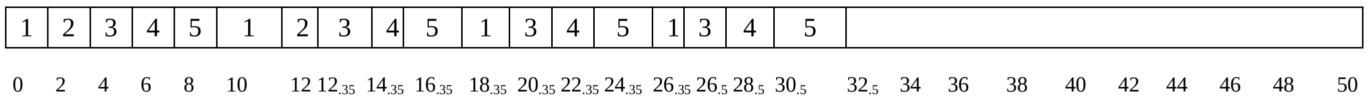
<u>Round3</u>

job	Burst Time (B_{ij})	Burst Time ($B_{i(j+1)}$) = B_{ij}-q
1	2.15	2.15 – 2 = 0.15
2	- - -	- - - - - - -
3	8.5	8.5 – 2 = 6.5
4	16.5	16.5 – 2 = 14.5
5	4.5	4.5 – 2 = 2.5
$\sum(B_{ij})$ = 31.65		$\sum(B_{i(j+1)})$= 23.65



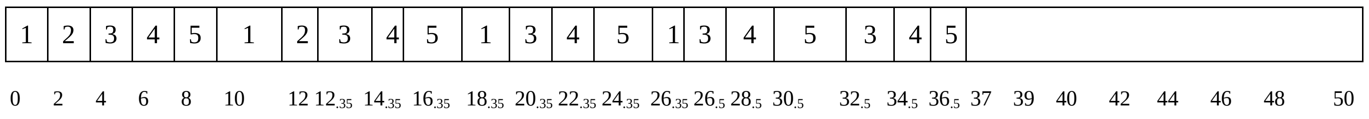
<u>Round4</u>

job	Burst Time (B_{ij})	Burst Time ($B_{i(j+1)}$) = B_{ij}-q
1	0.15	0.15 – 2 = 0
2	- - -	- - - - - - -
3	6.5	6.5 – 2 = 4.5
4	14.5	14.5 – 2 = 12.5
5	2.5	2.5 – 2 = 0.5
$\sum(B_{ij})$ = 23.65		$\sum(B_{i(j+1)})$= 17.65



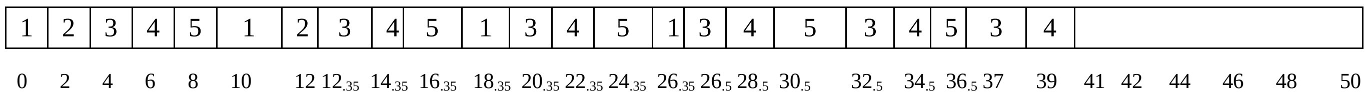
<u>Round5</u>

job	Burst Time (B_{ij})	Burst Time ($B_{i(j+1)}$) = B_{ij}-q
1	- - -	- - - - - - -
2	- - -	- - - - - - -
3	4.5	4.5 – 2 = 2.5
4	12.5	12.5 – 2 = 10.5
5	0.5	0.5 – 2 = 0
$\sum(B_{ij})$ = 17.65		$\sum(B_{i(j+1)})$= 13



<u>Round6</u>

job	Burst Time (B_{ij})	Burst Time ($B_{i(j+1)}$) = B_{ij}-q
1	- - -	- - - - - - -
2	- - -	- - - - - - -
3	2.5	2.5 – 2 = 0.5
4	10.5	10.5 – 2 = 8.5
5	- - -	- - - - - - -
$\sum(B_{ij})$ = 13		$\sum(B_{i(j+1)})$= 9



Round 7

job	Burst Time (B_{ij})	Burst Time ($B_{i(j+1)}$) = B_{ij}-q
1	- - -	- - - - - - -
2	- - -	- - - - - - -
3	0.5	0.5 – 2 = 0
4	8.5	8.5 – 2 = 6.5
5	- - -	- - - - - - -
	∑(B_{ij}) = 9	∑ ($B_{i(j+1)}$)= 6.5

1	2	3	4	5	1	2	3	4	5	1	3	4	5	1	3	4	5	3	4	5	3	4	3	4	

0 2 4 6 8 10 12 $12_{.35}$ $14_{.35}$ $16_{.35}$ $18_{.35}$ $20_{.35}$ $22_{.35}$ $24_{.35}$ $26_{.35}$ $26_{.5}$ $28_{.5}$ $30_{.5}$ $32_{.5}$ $34_{.5}$ $36_{.5}$ 37 39 41 $41_{.5}$ $43_{.5}$ 44 46 48 50

Round 8

job	Burst Time (B_{ij})	Burst Time ($B_{i(j+1)}$) = B_{ij}-q
1	- - -	- - - - - - -
2	- - -	- - - - - - -
3	- - -	- - - - - - -
4	6.5	6.5 – 2 = 4.5
5	- - -	- - - - - - -
	∑(B_{ij}) = 6.5	∑ ($B_{i(j+1)}$)= 4.5

1	2	3	4	5	1	2	3	4	5	1	3	4	5	1	3	4	5	3	4	5	3	4	3	4	4	

0 2 4 6 8 10 12 $12_{.35}$ $14_{.35}$ $16_{.35}$ $18_{.35}$ $20_{.35}$ $22_{.35}$ $24_{.35}$ $26_{.35}$ $26_{.5}$ $28_{.5}$ $30_{.5}$ $32_{.5}$ $34_{.5}$ $36_{.5}$ 37 39 41 $41_{.5}$ $43_{.5}$ $45_{.5}$ 46 48 50

Round 9

job	Burst Time (B_{ij})	Burst Time ($B_{i(j+1)}$) = B_{ij}-q
1	- - -	- - - - - - -
2	- - -	- - - - - - -
3	- - -	- - - - - - -
4	4.5	4.5 – 2 = 2.5
5	- - -	- - - - - - -
	∑(B_{ij}) = 4.5	∑ ($B_{i(j+1)}$)= 2.5

1	2	3	4	5	1	2	3	4	5	1	3	4	5	1	3	4	5	3	4	5	3	4	3	4	4	4	

0 2 4 6 8 10 12 $12_{.35}$ $14_{.35}$ $16_{.35}$ $18_{.35}$ $20_{.35}$ $22_{.35}$ $24_{.35}$ $26_{.35}$ $26_{.5}$ $28_{.5}$ $30_{.5}$ $32_{.5}$ $34_{.5}$ $36_{.5}$ 37 39 41 $41_{.5}$ $43_{.5}$ $45_{.5}$ $47_{.5}$ 48 50

Round 10

job	Burst Time (B_{ij})	Burst Time ($B_{i(j+1)}$) = B_{ij}-q
1	- - -	- - - - - - -
2	- - -	- - - - - - -
3	- - -	- - - - - - -
4	2.5	2.5 – 2 = 0.5
5	- - -	- - - - - - -
	∑(B_{ij}) = 2.5	∑ ($B_{i(j+1)}$)= 0.5

1	2	3	4	5	1	2	3	4	5	1	3	4	5	1	3	4	5	3	4	5	3	4	3	4	4	4	4	

0 2 4 6 8 10 12 $12_{.35}$ $14_{.35}$ $16_{.35}$ $18_{.35}$ $20_{.35}$ $22_{.35}$ $24_{.35}$ $26_{.35}$ $26_{.5}$ $28_{.5}$ $30_{.5}$ $32_{.5}$ $34_{.5}$ $36_{.5}$ 37 39 41 $41_{.5}$ $43_{.5}$ $45_{.5}$ $47_{.5}$ $49_{.5}$50

Round 11

job	Burst Time (B_{ij})	Burst Time ($B_{i(j+1)}$) = B_{ij}-q
1	- - -	- - - - - - -
2	- - -	- - - - - - -
3	- - -	- - - - - - -
4	0.5	0.5 – 2 = 0
5	- - -	- - - - - - -
$\sum(B_{ij})$ = 0.5		$\sum(B_{i(j+1)})$= 0

1	2	3	4	5	1	2	3	4	5	1	3	4	5	1	3	4	5	3	4	5	3	4	3	4	4	4	4	4

0 2 4 6 8 10 12 12.35 14.35 16.35 18.35 20.35 22.35 24.35 26.35 26.5 28.5 30.5 32.5 34.5 36.5 37 39 41 41.5 43.5 45.5 47.5 49.5 50

Average waiting time= [W.P1+W.P2+W.P3+W.P4+W.P5] / 5
P1= 26.35-[(20.35–18.35)+2+2] = 26.35 – 6 = 20.35
P2 =12 – 2 =10
P3 = 41-12 = 29
P4 = 41.5-12= 29.5
P5 =36.5-8 = 28.5

Average waiting time= [20.35+10+29+29.5+28.5]/5 = 117.35 / 5 = 23.47 t . . . . . . (3)

Average Turnaround time = [T.P1+T.P2+T.P3+T.P4+T.P5] / 5

The average turnaround time = 26.5+12.35+41.5+50+37=167.35 / 5 = 33 t . . . . . . (4)

Since the Shortest-Job-First has less than half the average waiting time of FCFS as we know, the round-robin is an intermediate algorithm.

3-4 *Round Robin Algorithm Features*

Round Robin algorithm is designed especially for time-sharing systems. In RR algorithm the scheduling decision refers to the concept of selecting the next process for execution. During each scheduling decision, a context switch occurs, meaning that the current process will stop its execution and put back to the ready queue and another process will be dispatched. Context switches are overheads, because CPU remains idle during context switches, which reduces CPU utilization.

Round Robin algorithm, as briefly mentioned in the abstract , has the worse average turnaround time and waiting time.

In addition, by considering a static set of *n* processes the number of context switches for only one round is *n* switches. While (E) the expected number of rounds = B/q, where B is the average CPU burst time for processes and q is a fixed time quantum. So, the total number of context switches over all rounds is *(En)* switches. RR algorithms have low throughput as well.

Setting the quantum too short causes too many context switches and lower the CPU efficiency. On the other hand, setting the quantum too long may cause poor response time and approximates First Come First Served (FCFS) algorithm.

4. *Proposal Algorithm*

4.1 *Introduction*

As we have seen there are many scheduling algorithms each with own parameters. If anyone likes to answer the question of; how do we select a CPU Scheduling algorithm for a practice system? As a result, selecting an algorithm can be quite difficult. The first problem is defining the criteria to be used in selecting an algorithm. The criteria are often defined in terms of CPU Utilization, response time, throughput, turnaround time and waiting time [2].

Round Robin (RR) algorithm gives better responsiveness in addition to have a good feature such as low scheduling overhead but RR algorithms have low throughput and the worse average turnaround time and waiting time. Also the number *n* of context switches for each rounds having *n* processes.

SJF, however, itself is not practical in time sharing OSs due to its low response. More over the high scheduling overhead but the good features of SJF algorithm are the best average turnaround time and waiting time. By considering a static set of *n* processes, desirable features of CPU scheduler to maximize CPU utilization and response time are obtained by combining the best features of RR algorithm (through implementing the innovative *optimal* equation to adapt time quantum factor for each process in each round) associated with the kernel properties of SJF algorithm as a novel algorithm; Hybrid CPU Scheduler algorithm SJF-RR which is proposed in this research [1, 3].

4.2 *Theoretical work*

For implementing "*Hybrid CPU Scheduling algorithm SJF-RR in Static Set of Processes*" proposed in this research, mathematical calculations take place to have variant time quantum for specific process in each round by applicable innovative equation (Eq 1).

This equation is the optimal equation to adapt time quantum factor for each process in each round on the base of proportionally evolution the time quantum to be partial to treat the process of a short burst in contrast with the process of larger burst and vice versa:

$$q_{ij} = q^{\sim} + \frac{q^{\sim}}{(\mathrm{B}_{ij} - \mathrm{q}_{\mathrm{i(j-1)}})^2} \qquad . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ \text{Eq 1}$$

Where (q_{ij}) is a new adaptive time quantum for process (i) in the particular round (j). ($q^{\sim}$) is a fixed time quantum given by the designer for the operating system. (B_{ij}) is the burst CPU time for process (i) in round (j) which must be decreased for the specific process on the subsequent round, as shown in the equation below. This result exploited by the former equation as the quantum ($\mathrm{q_{ij}}$) for the particular process to be larger in subsequent round.

So magnifying denominator of (Eq 1) implies to have a larger value of ($\mathrm{q_{ij}}$) which is important to achieve the same duty in a good performance in regard to waiting time and turnaround time criteria.
In the other hand, (Eq 2) is to calculate (B_{ij}) for particular process in each round.

$$\mathrm{B_{ij}} = \mathrm{B_{i(j-1)}} - \mathrm{q_{i(j-1)}} \qquad . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ . \ \text{Eq 2}$$

Where (B_{ij}) is the burst CPU time for process (i) in round (j) which will be minimized periodically in successive rounds by the upgraded value of time quantum. While ($B_{i(j-1)}$) is the burst CPU time for process (i) in round before (*j-1*). Also ($q_{i(j-1)}$) is the particular quantum for the process (i) in the round before (*j-1*).

So, this is the core of the goal for this research because it implies the processor won't be allocated fairly (by equal quantum) for all processes but to achieve those processes with shorter burst time (by relatively large time quantum) prior than others with long burst time. This concept is meant by having the best features of RR algorithm associated with the kernel properties of SJF algorithm that has the least process turnaround time and process average waiting time.

Briefly, Hybrid CPU Scheduling algorithm SJF-RR achieves a good result of significant advantage to reduce the number of processes in the ready queue through finishing short jobs relatively faster than other in a hope to increase the throughput, response time and to reduce the average turnaround time and waiting time.

4.3 Experimental Work

As mentioned before, in practical work time quantum or time slice of traditional RR is generally from 10 to 100 ms so no process allocates the CPU for more than one time. But the deference with Hybrid CPU Scheduler algorithm (SJF-RR) is that the time quantum isn't equal for all processes also it isn't equal for the particular process in deferent rounds.
If the CPU burst exceeds a time quantum, it will be preempted and put back in the ready queue and treated as a circular queue.
So Hybrid CPU Scheduler algorithm (SJF-RR) is a preemptive scheduling algorithm.

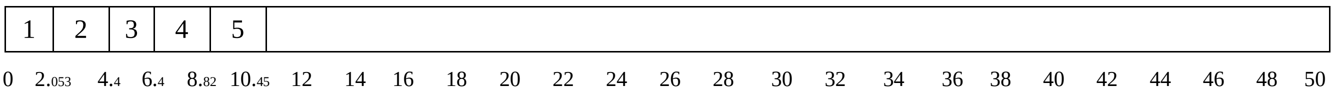
Example

Consider the following set of jobs, which were previously used to illustrate RR where the time quantum equal 2 :

Job	Burst Time
1	6.15
2	2.35
3	12.5
4	20.5
5	8.5
Total Burst Time = 50	

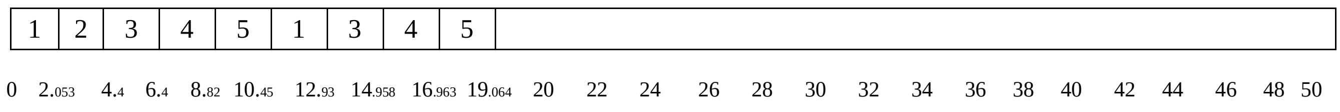
Round (1)

P	$(B_{i\,(j-1)} - q_{i\,(j-1)}) = B_{ij}$	$(B_{i\,j} - q_{i\,(j-1)})$	$(B_{i\,j} - q_{i\,(j-1)})^2$	$q_{ij} = q^- + \{ q^-/(B_{ij} - q_{i\,(j-1)})^2 \}$
1	6.15-0=6.15	6.15-0=6.15	37.82	2.053
2	2.35-0=2.35	2.35-0=2.35	5.52	2.35
3	12.5-0=12.5	12.5-0=12.5	156.25	2.013
4	20.5-0=20.5	20.5-0=20.5	420.25	2.005
5	8.5-0=8.5	8.5-0 =8.5	72.25	2.027
	$\sum B_{ij} = 50$			$\sum q_{ij} = 10.448$



Round (2)

P	$(B_{i\,(j-1)} - q_{i\,(j-1)}) = B_{ij}$	$(B_{i\,j} - q_{i\,(j-1)})$	$(B_{i\,j} - q_{i\,(j-1)})^2$	$q_{ij} = q^- + \{ q^-/(B_{ij} - q_{i\,(j-1)})^2 \}$
1	6.15-2.053=4.097	4.097-2.053=2.044	4.178	2.48
2	2.35-2.35=0	- - - - -	- - - - -	- - - - -
3	12.5-2.013=10.487	10.487-2.013=8.447	71.35	2.028
4	20.5-2.005=18.495	18.495-2.005=18.49	341.88	2.005
5	8.5-2.027=6.473	6.473-2.027 =4.446	19.767	2.101
	$\sum B_{ij} = 39.552$			$\sum q_{ij} = 8.614$



Round (3)

P	$(B_{i\,(j-1)} - q_{i\,(j-1)}) = B_{ij}$	$(B_{i\,j} - q_{i\,(j-1)})$	$(B_{i\,j} - q_{i\,(j-1)})^2$	$q_{ij} = q^- + \{ q^- / (B_{ij} - q_{i\,(j-1)})^2 \}$
1	4.097-2.48=1.617	1.617-2.48 =<0	- - - - -	1.617
2	2.35-2.35=0	- - - - -	- - - - -	- - - - -
3	10.487-2.028=8.459	8.459-2.028=6.431	41.357	2.048
4	18.495-2.005=16.49	16.49-2.005=14.485	209.81	2.0095
5	6.473-2.101=4.372	4.372-2.101=2.271	5.1574	2.388
	$\sum B_{ij} = 30.938$			$\sum q_{ij} = 8.062$

1	2	3	4	5	1	3	4	5	1	3	4	5												

0 2.053 4.4 6.4 8.82 10.45 12.93 14.958 16.963 19.064 20.68 22.73 24.74 27.124 28 30 32 34 36 38 40 42 44 46 48 50

Round (4)

P	$(B_{i\,(j-1)} - q_{i\,(j-1)}) = B_{ij}$	$(B_{i\,j} - q_{i\,(j-1)})$	$(B_{i\,j} - q_{i\,(j-1)})^2$	$q_{ij} = q^- + \{ q^- / (B_{ij} - q_{i\,(j-1)})^2 \}$
1	1.617-1.617=0	- - - - -	- - - - -	- - - - -
2	2.35-2.362=0	- - - - -	- - - - -	- - - - -
3	8.459-2.048=6.411	6.411-2.048=4.363	19.0357	2.105
4	16.49-2.009=14.481	14.481-2.009=12.472	155.55	2.013
5	4.372-2.388=1.984	1.984-2.388 =<0	- - - - -	1.984
	$\sum B_{ij} = 22.876$			$\sum q_{ij} = 6.102$

1	2	3	4	5	1	3	4	5	1	3	4	5	3	4	5	

0 2.053 4.4 6.4 8.82 10.45 12.93 14.958 16.963 19.064 20.68 22.73 24.74 27.124 29.23 31.24 33.225 34 36 38 40 42 44 46 48 50

Round (5)

P	$(B_{i\,(j-1)} - q_{i\,(j-1)}) = B_{ij}$	$(B_{i\,j} - q_{i\,(j-1)})$	$(B_{i\,j} - q_{i\,(j-1)})^2$	$q_{ij} = q^- + \{ q^- / (B_{ij} - q_{i\,(j-1)})^2 \}$
1	1.617-1.617=0	- - - - -	- - - - -	- - - - -
2	2.35-2.362=0	- - - - -	- - - - -	- - - - -
3	6.411-2.105=4.307	4.307-2.105=2.202	4.849	2.412
4	14.481-2.013=12.468	12.468-2.013=10.455	109.307	2.018
5	1.984-1.984=0	- - - - -	- - - - -	- - - - -
	$\sum B_{ij} = 16.775$			$\sum q_{ij} = 4.43$

1	2	3	4	5	1	3	4	5	1	3	4	5	3	4	5	3	4	

0 2.053 4.4 6.4 8.82 10.45 12.93 14.958 16.963 19.064 20.68 22.73 24.74 27.124 29.23 31.24 33.225 35.637 37.655 39 40 42 44 46 48 50

Round (6)

P	$(B_{i\,(j-1)} - q_{i\,(j-1)}) = B_{ij}$	$(B_{i\,j} - q_{i\,(j-1)})$	$(B_{i\,j} - q_{i\,(j-1)})^2$	$q_{ij} = q^- + \{ q^-/(B_{ij} - q_{i\,(j-1)})^2 \}$
1	1.617-1.617=0	- - - - -	- - - - -	- - - - -
2	2.35-2.362=0	- - - - -	- - - - -	- - - - -
3	4.307-2.412=1.895	1.895-2.412 =<0	- - - - -	1.895
4	12.468-2.018=10.45	10.45-2.018=8.432	71.1	2.028
5	1.984-1.984=0	- - - - -	- - - - -	- - - - -
	$\sum B_{ij} = 12.345$			$\sum q_{ij} = 3.923$

1	2	3	4	5	1	3	4	5	1	3	4	5	3	4	5	3	4	3	4	

0 2.053 4.4 6.4 8.82 10.45 12.93 14.958 16.963 19.064 20.68 22.73 24.74 27.124 29.23 31.24 33.225 35.637 37.655 39.54 41.57 43 45 48 50

Round (7)

P	$(B_{i\,(j-1)} - q_{i\,(j-1)}) = B_{ij}$	$(B_{i\,j} - q_{i\,(j-1)})$	$(B_{i\,j} - q_{i\,(j-1)})^2$	$q_{ij} = q^- + \{ q^-/(B_{ij} - q_{i\,(j-1)})^2 \}$
1	1.617-1.617=0	- - - - -	- - - - -	- - - - -
2	2.35-2.362=0	- - - - -	- - - - -	- - - - -
3	1.895-1.895=0	- - - - -	- - - - -	- - - - -
4	10.45-2.028=8.422	8.422-2.028=6.394	40.88	2.049
5	1.984-1.984=0	- - - - -	- - - - -	- - - - -
	$\sum B_{ij} = 8.422$			$\sum q_{ij} = 2.049$

1	2	3	4	5	1	3	4	5	1	3	4	5	3	4	5	3	4	3	4	4	

0 2.053 4.4 6.4 8.82 10.45 12.93 14.958 16.963 19.064 20.68 22.73 24.74 27.124 29.23 31.24 33.225 35.637 37.655 39.54 41.57 43.62 45 48 50

Round (8)

P	$(B_{i\,(j-1)} - q_{i\,(j-1)}) = B_{ij}$	$(B_{i\,j} - q_{i\,(j-1)})$	$(B_{i\,j} - q_{i\,(j-1)})^2$	$q_{ij} = q^- + \{ q^-/(B_{ij} - q_{i\,(j-1)})^2 \}$
1	1.617-1.617=0	- - - - -	- - - - -	- - - - -
2	2.35-2.362=0	- - - - -	- - - - -	- - - - -
3	1.895-1.895=0	- - - - -	- - - - -	- - - - -
4	8.422-2.049=6.373	6.373-2.049=4.324	18.67	2.107
5	1.984-1.984=0	- - - - -	- - - - -	- - - - -
	$\sum B_{ij} = 6.373$			$\sum q_{ij} = 2.107$

1	2	3	4	5	1	3	4	5	1	3	4	5	3	4	5	3	4	3	4	4	4	

0 2.053 4.4 6.4 8.82 10.45 12.93 14.958 16.963 19.064 20.68 22.73 24.74 27.124 29.23 31.24 33.225 35.637 37.655 39.54 41.57 43.62 45.734 48 50

<u>Round (9)</u>

P	$(B_{i\,(j-1)} - q_{i\,(j-1)}) = B_{ij}$	$(B_{i\,j} - q_{i\,(j-1)})$	$(B_{i\,j} - q_{i\,(j-1)})^2$	$q_{ij} = q^{-} + \{ q^{-}/(B_{ij} - q_{i\,(j-1)})^2\}$
1	1.617-1.617=0	- - - - -	- - - - -	- - - - -
2	2.35-2.362=0	- - - - -	- - - - -	- - - - -
3	1.895-1.895=0	- - - - -	- - - - -	- - - - -
4	6.373-2.107=4.266	4.266-2.107=2.159	4.66	2.43
5	1.984-1.984=0	- - - - -	- - - - -	- - - - -
	$\sum B_{ij} = 4.266$			$\sum q_{ij} = 2.43$

1	2	3	4	5	1	3	4	5	1	3	4	5	3	4	5	3	4	3	4	4	4	4	

0 2.053 4.4 6.4 8.82 10.45 12.93 14.958 16.963 19.064 20.68 22.73 24.74 27.124 29.23 31.24 33.225 35.637 37.655 39.54 41.57 43.62 45.734 48.164 50

<u>Round (10)</u>

P	$(B_{i\,(j-1)} - q_{i\,(j-1)}) = B_{ij}$	$(B_{i\,j} - q_{i\,(j-1)})$	$(B_{i\,j} - q_{i\,(j-1)})^2$	$q_{ij} = q^{-} + \{ q^{-}/(B_{ij} - q_{i\,(j-1)})^2\}$
1	1.617-1.617=0	- - - - -	- - - - -	- - - - -
2	2.35-2.362=0	- - - - -	- - - - -	- - - - -
3	1.895-1.895=0	- - - - -	- - - - -	- - - - -
4	4.266-2.43=1.836	1.836-2.43 =< 0	- - - - -	1.836
5	1.984-1.984=0	- - - - -	- - - - -	- - - - -
	$\sum B_{ij} = 1.836$			$\sum q_{ij} = 1.836$

1	2	3	4	5	1	3	4	5	1	3	4	5	3	4	5	3	4	3	4	4	4	4	4

0 2.053 4.4 6.4 8.82 10.45 12.93 14.958 16.963 19.064 20.68 22.73 24.74 27.124 29.23 31.24 33.225 35.637 37.655 39.54 41.57 43.62 45.734 48.164 50

<u>Round (11)</u>

P	$(B_{i\,(j-1)} - q_{i\,(j-1)}) = B_{ij}$	$(B_{i\,j} - q_{i\,(j-1)})$	$(B_{i\,j} - q_{i\,(j-1)})^2$	$q_{ij} = q^{-} + \{ q^{-}/(B_{ij} - q_{i\,(j-1)})^2\}$
1	1.617-1.617=0	- - - - -	- - - - -	- - - - -
2	2.35-2.362=0	- - - - -	- - - - -	- - - - -
3	1.895-1.895=0	- - - - -	- - - - -	- - - - -
4	1.836-1.836=0	- - - - -	- - - - -	- - - - -
5	1.984-1.984=0	- - - - -	- - - - -	- - - - -
	$\sum B_{ij} = 0$			$\sum q_{ij} = 0$

1	2	3	4	5	1	3	4	5	1	3	4	5	3	4	5	3	4	3	4	4	4	4	4

0 2.053 4.4 6.4 8.82 10.45 12.93 14.958 16.963 19.064 20.68 22.73 24.74 27.124 29.23 31.24 33.225 35.637 37.655 39.54 41.57 43.62 45.734 48.164 50

4.4. Proposal Evaluation

As we have seen there are many scheduling algorithms each with own parameters but it's necessary to answer the question of; how does the proposed algorithm "*Hybrid CPU Scheduler algorithm SJF-RR"* effectively practical for practice system.

Deterministic modeling as waiting time and turnaround time used to give exact numbers allowing the algorithms to be compared.

Average waiting time= [W.P1+W.P2+W.P3+W.P4+W.P5] / 5

P1 = 20.68-[(12.93-10.45)+(2.053)]
= 20.68-[2.48+2.053]
= 20.68-4.533
= 16.147

P2 = 2.347

P3 = 37.655-[(29.23-27.124)+(22.73-20.68)+(14.958-12.93)+(6.4-4.4)]
= 37.655-[(2.412)+(2.106)+(2.05)+(2.028)+(2)]
= 37.655-10.596
= 27.059

P4 = 39.54-[(37.655-35.637)+(31.24-29.23)+(24.74-22.73)+
(16.963-14.958)+(8.82-6.4)]
= 39.54-[(2.18+2.01+2.01+2.005+2.42)
= 39.54-10.625
= 28.915

P5 = 31.24-[(27.124-24.74)+(19.064-16.963)+(10.45-8.82)]
= 31.24-[2.384+2.101+1.63]
= 31.24-6.115
= 25.125

Average waiting time= [16.147+2.347+27.059+28.915+25.125] / 5
= 99.593 / 5
= 19.918

Average Turnaround time = [T.P1+T.P2+T.P3+T.P4+T.P5] / 5
= [20.68+2.4+39.54+50+33.225] / 5
= 146.025 / 5
= 29.205

So after the criteria have been mentioned for selecting the optimal algorithm in terms of CPU Utilization, Response time, or throughput, it is explicit that the deterministic modeling used previously gives exact numbers allowing the algorithms to be compared.

Through illustrating various algorithm for particular system workload (set of jobs) to produce a formula or number which evaluates the performance of the algorithm for that workload refers that the proposed algorithm (Hybrid CPU Scheduler algorithm SJF-RR) is the optimal algorithm for given static set of jobs as shown below:

	SJF	Traditional RR Quantum = 2	Hybrid SJF-RR Quantum = 2
Waiting time	11.47	23.47	19.918
Turnaround time	21.47	33	29.205
Throughput	high	low	intermediate
Response time	low	high	high
Context Switching	(i)processes= *5*	(P_i*R) = 19	(P_i*R) = 14
Scheduling Overhead	$O(i)$	$O(1)$	$O(1)$
Scheduler Usage	*Long term*	*Short term*	*Short term*

Therefore, proposed algorithm achieves maximal CPU Utilization, Response time, or throughput in contrast with traditional RR such that average waiting time and turnaround time are linearly proportional to total execution time.

5. *Conclusion*

Optimistically, this research fulfilled the core idea behind the project's objectives. Achieving a good result of significant advantage to improve the performance and reduce the number of processes in the ready queue based on finishing short jobs relatively faster than other jobs. This strategy is in a hope to increase the throughput and response and to reduce the turnaround time and the average waiting time.

References

1. James L. Peterson & Abraham Silberschatz, "Operating System Concepts", Second Edition, University of Texas at Austin, Addison-Wesley Publisher company, Inc. Copyright © 1985.

2. A. M. Lister, "Fundamentals Of Operating Systems", Third Edition, University of Queensland, Published by Higher and Further Education Division, Copyright © 1984.

3. David Barron, Professor of Computer Studies, "Computer Operating Systems For micros, minis and mainframes", Second Edition, University of Southampton, Published in USA by Chapman and Hall, Copyright © 1984.

4. Marvin Solomon, " Introduction to Operating Systems", Computer Sciences Department, University of Wisconsin, Madison, Publication Date : 2006, updated January 2007.

5. Andrew S. Tanenbaum, " **Operating Systems Design and Implementation**", **Third Edition**, Vrije Universiteit Amsterdam, The Netherlands, Albert S. Woodhull - Amherst, Massachusetts , **Publisher: Prentice Hall, Pub Date: January 04, 2006**

6. Internet Website:

A ▪ http://en.wikipedia.org/wiki/Operating_system

B ▪ http:// scribd.com/doc/20609237/Operating-system-concepts-slides-ch01