



Performance Evaluation of a New Proposed Variant of Fittest Job First Dynamic Round Robin (VFJFDRR) Scheduling Algorithm with Intelligent Time Slice

Rakesh Mohanty*, Suchitra Panigrahi

Department of Computer Science and Engineering

Veer Surendra Sai University of Technology

Burla, Odisha, India

rakesh.iitmphd@gmail.com, panigrahisuchitra@gmail.com

Abstract—Round Robin (RR) is one of the most suitable and efficient scheduling algorithms for time-sharing systems. Recently many variants of RR scheduling algorithm have been developed and studied in the literature with the use of dynamic time quantum. In this paper, a new variant of Fittest Job First Dynamic Round Robin (FJFDRR) algorithm is proposed which we call as VFJFDRR in short. In this algorithm, the processes are arranged in the ready queue according to our calculated fit factor and are assigned dynamic time quantum using the intelligent time slice method. We have evaluated the performance and compared the results of our proposed VFJFDRR algorithm with the FJFDRR algorithm using three different cases of the input data set. Experimental results show that VFJFDRR performs better than FJFDRR in terms of average waiting time and average turnaround time when the processes are arranged in decreasing and random order of burst time. However VFJFDRR provides greater average waiting time and average turnaround time than FJFDRR when the processes are arranged in increasing order of burst time. In all the three cases, the number of context switches is increased for VFJFDRR making the algorithm more suitable for fair sharing.

Keywords: Operating System, Scheduling, Round Robin, Dynamic Time Quantum, Intelligent Time Slice.

I. INTRODUCTION

Operating system (OS) is an interface between an end user and system software. Recently emerging OS use the concepts of *multi-tasking*, *multi-processing* and *multi-programming* in order to achieve efficiency and improved system performance. *Multi-tasking* allows a user to perform more than one task at a time. *Multiprocessing* is the coordinated execution of programs by more than one processor. *Multi-programming* is the technique of executing several programs simultaneously in a single processor system. *Scheduling* refers to the way the processes are assigned to the processor. A *scheduling algorithm* is used for determining the sequence in which the processes will be dispatched to the processor so as to keep it busy. Design of efficient scheduling algorithms for OS with multi-tasking, multi-programming and multiprocessing environments is a major challenging research issue. Though many sophisticated scheduling algorithms have been designed, there is a scope for improvement for existing algorithms in the literature with a new strategy. In this paper our objective is to develop a new improved variant of a recently developed scheduling algorithm. In the next section we introduce some basic terminologies related to scheduling and operating system.

A. Basic Terminologies:

Prior to allocation to a processor the processes are put in a queue called *ready queue*. The time at which a process enters the ready queue is its *arrival time*. The amount of time a process spends waiting in the ready queue is called the *waiting time*. The time interval between the submission of a request by a process and the first response received is the *response time*. The processor time required to complete execution of a process is known as *burst time*. *Turnaround time* is the time duration from submission the submission of a process to its completion. *Throughput* is the number of processes that are completed per unit time. *Processor*

Utilization is a term used to describe how much the processor is busy or utilised. The number of times the processor switches from one process to another is called as the number of *context switches*.

The main goal of scheduling is to maximize processor utilization, throughput and minimization of response time, waiting time and turnaround time.

B. Well-Known Scheduling Algorithms:

The scheduling algorithms can be classified into two types such as non-preemptive and pre-emptive. A scheduling algorithm is non-preemptive, if once a process has been given the processor; the processor can not be taken away from that process. In case of pre-emptive algorithms, the processor can be taken away from the currently running process on the arrival of a higher priority process in the ready queue. First Come First Serve (FCFS), Shortest Job First (SJF) and Priority Scheduling algorithms are non-pre-emptive, where as Round Robin (RR) and Shortest Remaining Time Next (SRTN) are few widely used pre-emptive algorithms. We present below these algorithms.

FCFS is a simple and basic algorithm which arranges processes in order of their arrival time in the ready queue. In this algorithm, if a process with shorter burst time arrives after a process with longer burst time, then waiting time for the shorter process will increase. So to favor the processes with shorter burst time, SJF algorithm is used. Here the process having shortest burst time is executed first. The Priority algorithm schedules the processes in the increasing order of priority number assigned to each of the processes. In SRTN, the scheduler always dispatches that process in the ready queue which has the shortest expected remaining time to completion. In RR algorithm, a time slice is assigned to each process and each process is assigned the processor for that amount of time slice in each round till its completion in a circular manner. RR is most suitable for time sharing systems as it improves the responsiveness of the system.

Existing non-preemptive algorithms use only one of the basic parameters such as arrival time or user priority or burst time. Preemptive RR algorithm uses static time slice. These limitations of existing RR algorithm motivate us to design a new algorithm which uses more than one basic parameters and dynamic time quantum concept to improve the performance metrics of scheduling algorithms.

C. Related Work:

Various scheduling algorithms have been proposed in the literature along with comprehensive studies in [1] and [2]. Recently, many real time scheduling algorithms have been developed with dynamic time quantum. Best Job First (BJF) proposed in [3] combined the basic parameters of scheduling algorithms which includes burst time, user priority and arrival time of processes. Yaashuwanth. and et al.[4] have proposed an algorithm for real time systems to overcome the limitations of both Rate monotonic and Deadline monotonic algorithm by adding a Priority component. The static time quantum which is a limitation of RR was replaced with dynamic time quantum by Matarneh [5]. In FJFDRR [6] algorithm, dynamic time quantum has been calculated taking the median of the remaining burst time. The processes are scheduled giving importance to user priority and shortest burst time priority rather than using single parameter. Work done in paper [7], [8] and [9] have contributed a great deal towards soft real time systems. Dynamic time quantum is computed in the similar manner in [10] as in [6] but the processes are scheduled according to increasing order of shortest burst time only.

D. Our Contribution:

In this paper, the processes are arranged in the ready queue in a similar manner as in [6], but we are calculating dynamic time quantum using intelligent time slice to develop a variant of the FJFDRR algorithm. Intelligent time slice is calculated in the similar manner as in [7]. In our work, we have made a study of scheduling algorithms with a special focus on RR algorithm with dynamic time slice. We have proposed a new variant of FJFDRR algorithm, which we call as VFJFDRR. The pseudo code, flowchart and illustration of VFJFDRR have been presented. Using the performance metrics such as average turnaround time, average waiting time and number of context switches, we have evaluated the performance of FJFDRR and VFJFDRR. Our experiments involve three cases of data sets and computation of waiting time, turnaround time and number of context switches using Gantt chart.

E. Organization of the Paper:

In section II, the pseudo code, flowchart and illustration of our proposed algorithm is presented. Section III shows the results of experimental analysis of our algorithm and its comparison with FJFDRR. Conclusions and directions for future work are given in section IV.

II. OUR PROPOSED VFJFDRR ALGORITHM

In our proposed algorithm the processes are arranged in the ready queue according to the newly calculated Fit factor f and dynamic time slice is assigned to processes using intelligent time slice method.

A. Uniqueness of our Approach:

Generally with every process three factors are associated which are user priority, burst time and arrival time. Above factors play an important role in deciding the sequence in which the processes will be executed. Out of all these factors user priority plays the most significant role than burst time. Assigning different weights to these factors according to their importance we calculate the Fit factor f . Generally in RR algorithm, processes are taken from the ready queue in FCFS manner for execution. But in our algorithm, f is calculated for each process. The process having the lowest f value will be scheduled first. The criteria which will decide the early execution of the processes are higher user priority and shorter burst time.

As user priority has higher importance than burst time, so it is given a weight age of 60% and burst time is given 40%, weight age. We assume all the processes have same arrival time i.e. arrival time=0. Let User Priority, User Priority Weight, Shorter Burst time Priority, Burst time Priority Weight of the processes be denoted as P_U , W_{UP} , P_{SBT} and W_{BTP} respectively. Then Fit Factor f can be calculated as

$$f = P_U * W_{UP} + P_{SBT} * W_{BTP} \text{ ----- (1)}$$

The performance of RR solely depends on the choice of time quantum. Our proposed algorithm makes use of dynamic intelligent time slice which allocates time quantum to each process independently based on the shortness component (SC), priority component (PC) and the context switch avoidance component (CSC). We formulate some relationships among the above parameters, which are being used in our proposed algorithm. PC for a process is given as 1 whose priority number is 1 and for the rest of the processes it is given as 0. Let P_i be the process id of i^{th} process, where $i = 1, 2, 3 \dots n$. Let SC_{pi} be the shortness component and T_{Bi} be the burst time of process P_i .

Then SC_{pi} can be calculated as follows:

$$SC_{pi} = \begin{cases} 1, & T_{Bi} - T_{B(i-1)} < 0 \\ 0, & \text{otherwise} \end{cases}$$

Let PC_{pi} be the priority component of process P_i and T_{q0} be the original time slice for all processes. Let T be a temporary variable such that $T = SC_{pi} + PC_{pi} + T_{q0}$. Then Context Switch Avoidance Component for process P_i can be computed as

$$CSC_{pi} = \begin{cases} T_{Bi} - T, & T_{Bi} - (T) < T \\ 0, & \text{otherwise} \end{cases}$$

Let $T_{IS}(P_i)$ be the intelligent time slice for process P_i , then

$$T_{IS}(P_i) = T + CSC_{pi}$$

In the first round processes having SC value 1 are assigned full ITS as the time slice and for others, half the value of ITS is assigned. For successive rounds, the processes having SC value 1 are assigned a time quantum equal to double the value of time quantum in the previous round. For other processes, time slice is computed as the sum of time quantum and half the time quantum of previous round which can be used for current cycle.

B. Our Proposed Algorithm:

In our algorithm, all the processes are scheduled using the newly calculated Fit factor f . The process having the least f value will be scheduled first. Here intelligent time slice method for dynamically assigning time slices to processes is used. We have used the following notations in our pseudo code for the VFJFDRR algorithm.

Let $n \rightarrow$ number of processes in the ready queue.

$f(P_i) \rightarrow$ fit factor for P_i

$T_{Bi} \rightarrow$ burst time of P_i

$T_q(P_i) \rightarrow$ time quantum of P_i

$T_{RB}(P_i) \rightarrow$ remaining burst time of P_i

The pseudo code and flow chart of our proposed VFJFDRR algorithm are presented in Fig. 1 and Fig. 2 respectively.

1. For $i = 1, 2, 3 \dots n$, Calculate $f(P_i)$.
2. Sort P_i for $i = 1, 2, 3, \dots n$ in ascending order
such that $P_i < P_j$ iff $f(P_i) < f(P_j)$ for $i \neq j$
3. while(ready queue $\neq$ null)

for $i = 1$ to n do

if ($i == 1$) then

if($SC_{P_i} == 0$) then

$$T_q(P_i) = 0.5 * T_{IS}(P_i);$$

Else

$$T_q(P_i) = T_{IS}(P_i);$$

End if

Else

If($SC_{P_i} \neq 0$) then

$$T_q(P_i) = T_q(P_{i-1}) + 0.5 * T_q(P_{i-1});$$

else

$$T_q(P_i) = 2 * T_q(P_{i-1});$$

End if

$$T_{RB}(P_i) = T_B(P_i) - T_q(P_i);$$

If ($T_{RB}(P_i) \leq 0$) then

$$T_q(P_i) = T_{RB}(P_i);$$

End if

End of for

End of while

4. Average waiting time, average turnaround time and number of context switches are calculated.

Figure. 1 Pseudo code for VFJFDRR

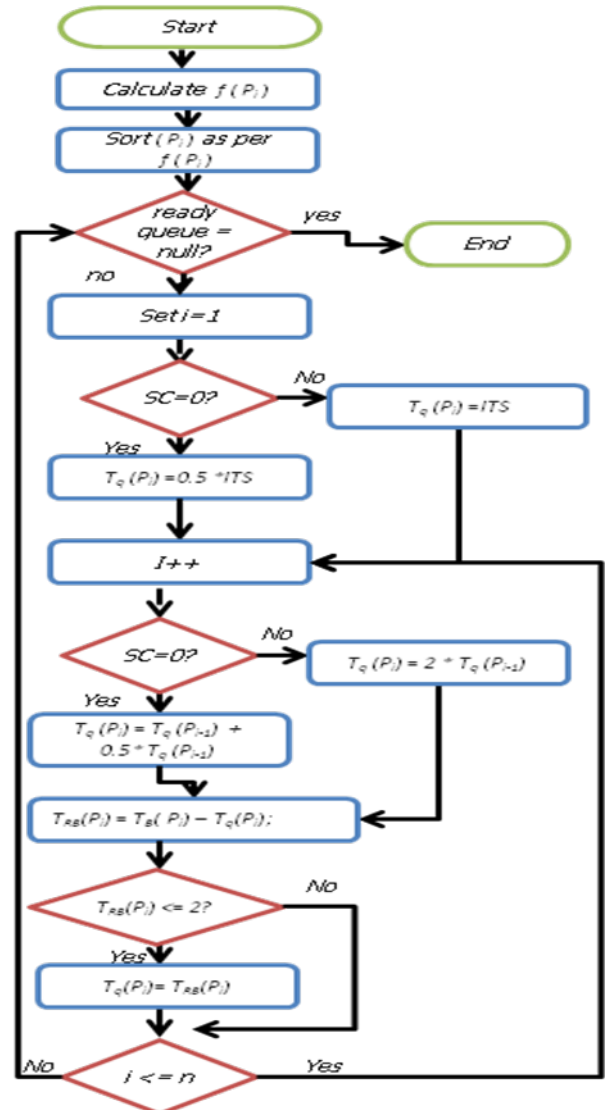


Figure. 2 Flow Chart for VFJFDRR

C. Illustration:

Given the burst time of five processes as 1, 35, 12, 9 and 98 with user priority 5, 2, 4, 3 and 1 respectively. Then the fit factor f for each process is calculated as 3.4, 2.8, 3.6, 2.6 and 2.6 for P1 through P5 respectively using equation 1. After computing f , we arrange the processes in the ready queue according to increasing order of fit factor. Next, Intelligent time slice (ITS) component is calculated for each individual process depending on the shortness component (SC), priority component (PC) and context switch avoidance component (CSC). SC was calculated to be 0, 0, 1, 1, and 0. The PC was calculated to be 0, 0, 0, 0, and 1. Original time slice (OTS) was arbitrarily chosen as 12. Next OTS, PC and SC were added and their result subtracted from burst time. If the value comes to be less than OTS then it is assigned as the CSC component. Here the values are -11, 0, -1, -4, and 0. To calculate the ITS, all OTS, PC, SC and CSC are added together. In the First round, processes whose SC value is 1, i.e., for P3 and P4, are given the ITS value as their time quantum and for processes with SC value as zero, i.e., P1, P2, P5 are given half the value of ITS as their time quantum. In successive rounds processes P3 and P4 are given double their time quantum in the previous cycle and processes P1, P2 and P5 are assigned time

quantum equal to the sum of their previous time quantum and half the value of the previous time quantum. Remaining burst time for each processes is calculated and the time quantum is dynamically assigned to the remaining burst time based on some condition. Gantt Chart is prepared to determine the waiting time, turn around time and number of context switching for each process. Average waiting time, average turn around time is computed from the Gantt Chart.

III. PERFORMANCE EVALUATION & RESULTS

A. Assumptions:

In a uni-processor environment, all the experiments are performed and all the processes are independent. Time slice is assumed to be not more than the maximum burst time. The attributes like burst time, number of processes and the user-priorities of all the processes are known before submitting the processes to the processor. All processes are processor bound. No processes are I/O bound.

B. Experimental Frame work:

Our experiment consists of several input and output parameters. The input parameters consists of the number of processes, burst time, Original time slice (OTS), and user-priority. The output parameters consist of average waiting time, average turnaround time and number of context switches.

C. Data Set:

We have performed three experiments for evaluating performance of VFJFDRR and FJFDRR. For the experiments, we have considered three cases of data set. They are processes with burst time in decreasing, random order and increasing order respectively.

D. Performance Metrics:

The significance of our performance metrics are as follows:

- Turnaround time (TAT): For the better performance of the algorithm, average turn-around time should be less.
- Waiting time (WT): For the better performance of the algorithm, average waiting time should be less.
- Number of Context switches (CS): For reducing the overheads of the operating system CS should be less while to attain fair sharing among the processes, CS should be high.

E. Experiments Performed:

In order to evaluate the performance of our proposed VFJFDRR algorithm and FJFDRR algorithm, we have taken a set of five processes in three different cases. Although for simplicity we have taken five processes, the algorithm works effectively for large number of processes. In each case, we have compared the experimental results of VFJFDRR algorithm with the FJFDRR algorithm presented in [3].

Case 1: We assume five processes P1, P2, P3, P4 and P5 are arriving at same with decreasing burst time 82, 43, 27, 15 and 9 respectively. The priorities assigned are 3, 1, 4, 2 and 5. The table-I, table-II and table-III show the output using algorithm FJFDRR and VFJFDRR. Table-IV shows the comparison between the two algorithms. Figure-3 and

Figure-4 shows Gantt chart for algorithms FJFDRR and VFJFDRR respectively.

Table-1 Computation of Fit Factor-Case 1

PROCESS	BT	S.P	P	F
P1	82	5	3	3.8
P2	43	4	1	2.2
P3	27	3	4	3.6
P4	15	2	2	2.0
P5	9	1	5	3.4

Table-2 Computation of ITS-Case 1

PROCESS	BT	P	OTS	PC	SC	CSC	ITS
P1	82	3	15	0	0	0	15
P2	43	1	15	1	1	0	17
P3	27	4	15	0	1	11	27
P4	15	2	15	0	1	-1	15
P5	9	5	15	0	1	-7	9

Table-3 VFJFDRR-Case 1

PROCESS	SC	ITS	ROUNDS				
			1 ST	2 ND	3 RD	4 TH	5 TH
P1	0	15	8	12	18	27	17
P2	1	17	17	26	0	0	0
P3	1	27	27	0	0	0	0
P4	1	15	15	0	0	0	0
P5	1	9	9	0	0	0	0

Table-4 Comparison between FJFDRR and VFJFDRR

ALGORITHM	WT _{AVG}	TAT _{AVG}	CS
FJFDRR	53.0	88.2	8
VFJFDRR	45.2	80.4	10

P4	P2	P5	P3	P1	P2	P1	P1	
0	15	42	51	78	105	121	156	176

Figure. 3: Gantt chart for FJFDRR (Case-1)

P4	P2	P5	P3	P1	P2	P1	P1	P1	P1	
0	15	32	41	68	76	102	114	132	159	176

Figure. 4: Gantt chart for VFJFDRR (Case-1)

Case 2: We assume five processes P1, P2, P3, P4 and P5 are arriving at same with random burst time 1, 35, 12, 9 and 8 respectively. The priorities assigned are 5, 2, 4, 3 and 1. The table-V, table-VI and table-VII show the output using algorithm FJFDRR and VFJFDRR. Table-VIII shows the comparison between the two algorithms. Figure-5 and Figure-6 shows Gantt chart for algorithms FJFDRR and VFJFDRR respectively.

Table-5 Computation of Fit Factor -Case 2

PROCESS	BT	S.P	P	f
P1	1	1	5	3.4
P2	35	4	2	2.8
P3	12	3	4	3.6
P4	9	2	3	2.6
P5	98	5	1	2.6

Table-6 Computation of ITS-Case 2

PROCESS	BT	P	OTS	PC	SC	CSC	ITS
P1	1	5	12	0	0	-11	1
P2	35	2	12	0	0	0	12
P3	12	4	12	0	1	-1	12
P4	9	3	12	0	1	-4	9
P5	98	1	12	1	0	0	13

Table-7 VFJFDRR-Case 2

PROCESS	SC	ITS	ROUNDS				
			1 ST	2 ND	3 RD	4 TH	5 TH
P1	0	1	1	0	0	0	0
P2	0	12	6	9	20	0	0
P3	1	12	12	0	0	0	0
P4	1	9	9	0	0	0	0
P5	0	13	7	11	17	26	37

Table-8 Comparison between FJFDRR and VFJFDRR

ALGORITHM	WT _{AVG}	TAT _{AVG}	CS
FJFDRR	47.2	78.2	8
VFJFDRR	33.2	64.2	11

P5	P4	P2	P1	P3	P5	P2	P5	
0	12	21	33	34	46	100	123	155

Figure. 5: Gantt chart for FJFDRR (Case-2)

P5	P4	P2	P1	P3	P5	P2	P5	P2	P5	P5	
0	7	16	22	23	35	46	55	72	92	118	155

Figure. 6: Gantt chart for VFJFDRR (Case-2)

Case 3: We assume five processes P1, P2, P3, P4 and P5 are arriving at same with increasing burst time 9, 15, 27, 43 and 82 respectively. The priorities assigned are 5, 2, 4, 3 and 1. The table-IX, table-X and table-XI show the output using algorithm FJFDRR and VFJFDRR. Table-XII shows the comparison between the two algorithms. Figure-7 and Figure-8 shows Gantt chart for algorithms FJFDRR and VFJFDRR respectively.

Figure-9, figure-10 and figure-11 show the graph of average waiting time, average turnaround time and context switch respectively for FJFDRR and VFJFDRR.

Table-9 FJFDRR-Case 3

PROCESS	BT	S.P	P	f
P1	9	1	5	3.4
P2	15	2	2	2.0
P3	27	3	4	3.6
P4	43	4	3	2.2
P5	82	5	1	3.8

Table-10 Computation of ITS-Case 3

PROCESS	BT	P	OTS	PC	SC	CSC	ITS
P1	9	5	15	0	0	-6	9
P2	15	2	15	0	0	0	15
P3	27	4	15	0	0	12	27
P4	43	1	15	1	0	0	16
P5	82	3	15	0	0	0	15

Table-11 VFJFDRR-Case 3

PROCESS	SC	ITS	ROUNDS				
			1 ST	2 ND	3 RD	4 TH	5 TH
P1	0	9	5	4	0	0	0
P2	0	15	8	7	0	0	0
P3	0	27	14	13	0	0	0
P4	0	16	8	12	18	5	0
P5	0	15	8	12	18	27	17

Table-12 Comparison between FJFDRR and VFJFDRR

ALGORITHM	WT _{AVG}	TAT _{AVG}	CS
FJFDRR	53.0	88.2	8
VFJFDRR	65.4	100.6	15

P2	P4	P1	P3	P5	P4	P5	P5	
0	15	42	51	78	105	121	156	176

Figure. 7: Gantt chart for FJFDRR (Case-3)

P2	P4	P1	P3	P5	P2	P4	P1	P3	P5	P4	P5	P4	P5	P5	
0	8	16	21	35	43	50	62	66	79	91	109	127	132	159	176

Figure. 8: Gantt chart for VFJFDRR (Case-3)

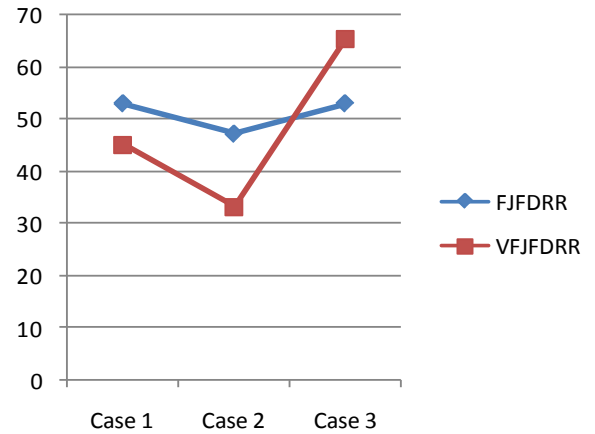


Figure. 9: Graph for Avg. WT

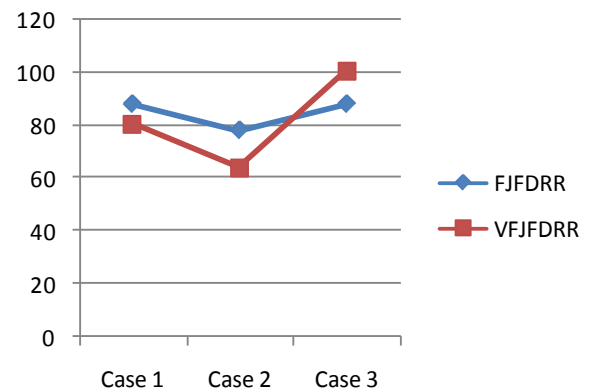


Figure. 10: Graph for Avg. TAT

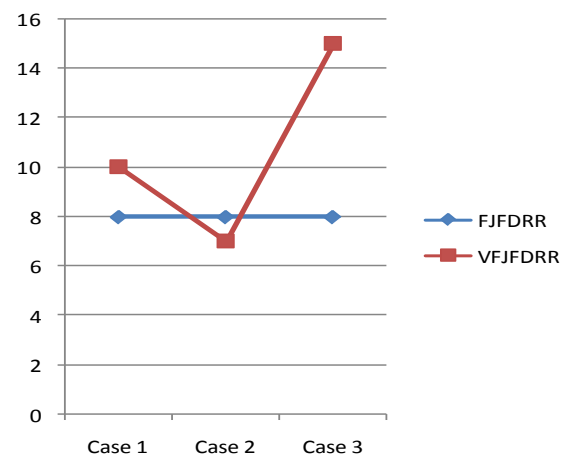


Figure. 11: Graph for context switch (CS)

IV. CONCLUSIONS

From our experimental tabular and graphical results, we have observed that VFJFDRR algorithm is performing better than FJFDRR proposed in paper [6] in terms of average waiting time and average turnaround time when the processes are arranged in decreasing and random order of their burst time. However, when the processes are arranged according to increasing order of burst time FJFDRR gives better results as compared to VFJFDRR algorithm. We also observe that in VFJFDRR, the number of context switches increases thereby increasing the fair sharing among the processes. Developing a new and realistic variant of FJFDRR algorithm using arrival time and deadline can be a challenging work for future.

REFERENCES

- [1] E.O. Oyetunji and et al. "Performance Assessment of some Processor Scheduling Algorithms" Research Journal of Information and Technology, 1(1):22-26, 2009.
- [2] A Pant "A Comparison between FCFS and Mixed Scheduling" International Journal of Computer Science and Technology (IJCT), 2(2):076-079, 2011.
- [3] Mohammed A. F. Al-Husain, "Best-Job-First Processor Scheduling Algorithm", Information Technology Journal 6 (2):288-293, 2007.
- [4] Yaashuwanth and et al."A New Scheduling Algorithm for Real Time System". International Journal of Computer Science and Information Security (IJCSIS), 6(2): 061-066, 2009.
- [5] Rami J. Matarneh, "Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the Now Running Processes", American Journal of Applied Sciences, 6 (10):1831-1837, 2009.
- [6] Rakesh Mohanty and et al."Design and Performance Evaluation of a New Proposed Fittest Job First Dynamic Round Robin Scheduling Algorithm" International Journal of Computer Information Systems, 2(2): 054-060, 2011.
- [7] Rakesh Mohanty and et al."Priority Based Dynamic Round Robin Algorithm with Intelligent Time Slice for Soft Real Time Systems". International Journal of Advanced Computer Science and Applications (IJACSA), 2(2): 46-50, 2011.
- [8] Yaashuwanth and et al." Intelligent Time Slice for Round Robin in Real Time Operating Systems" International Journal of Research and Reviews in Applied Sciences, 2(2): 126-131, 2010.
- [9] H.S Behera and et al." A New Dynamic Round Robin and SRTN Algorithm with Variable Original Time Slice and Intelligent Time Slice for Soft Real Time Systems" International Journal of Computer Applications, 16(9), 2011.
- [10] Rakesh Mohanty and et al."Design and Performance Evaluation of a New Proposed Shortest Remaining Burst Round Robin (SRBRR) Scheduling Algorithm". Proceedings of International Symposium on Computer Engineering & Technology (ISCET), 2010.