Computers, Materials & Continua
DOI:10.32604/cmc.2022.021978
Article

Tech Science Press

ATS: A Novel Time-Sharing CPU Scheduling Algorithm Based on Features Similarities

Samih M. Mostafa[1,*], Sahar Ahmed Idris[2] and Manjit Kaur[3]

[1]Faculty of Computers and Information, South Valley University, Qena, 83523, Egypt
[2]College of Industrial Engineering, King Khalid University, Abha, Saudi Arabia
[3]School of Engineering and Applied Sciences, Bennett University, Greater Noida, India
[*]Corresponding Author: Samih M. Mostafa. Email: samih_montser@sci.svu.edu.eg
Received: 22 July 2021; Accepted: 23 August 2021

Abstract: Minimizing time cost in time-shared operating systems is considered basic and essential task, and it is the most significant goal for the researchers who interested in CPU scheduling algorithms. Waiting time, turnaround time, and number of context switches are the most time cost criteria used to compare between CPU scheduling algorithms. CPU scheduling algorithms are divided into non-preemptive and preemptive. Round Robin (RR) algorithm is the most famous as it is the basis for all the algorithms used in time-sharing. In this paper, the authors proposed a novel CPU scheduling algorithm based on RR. The proposed algorithm is called Adjustable Time Slice (ATS). It reduces the time cost by taking the advantage of the low overhead of RR algorithm. In addition, ATS favors short processes allowing them to run longer time than given to long processes. The specific characteristics of each process are; its CPU execution time, weight, time slice, and number of context switches. ATS clusters the processes in groups depending on these characteristics. The traditional RR assigns fixed time slice for each process. On the other hand, dynamic variants of RR assign time slice for each process differs from other processes. The essential difference between ATS and the other methods is that it gives a set of processes a specific time based on their similarities within the same cluster. The authors compared between ATS with five popular scheduling algorithms on nine datasets of processes. The datasets used in the comparison vary in their features. The evaluation was measured in term of time cost and the experiments showed that the proposed algorithm reduces the time cost.

Keywords: Clustering; CPU scheduling; round robin; average turnaround time; average waiting time

1 Introduction

1.1 CPU Scheduling

CPU scheduling is defined as allocating and de-allocating the CPU to a specific process/thread. CPU scheduling is the most important and most effective task in the performance of the operating system [1–3]. CPU scheduling should provide efficient and fair usage of the computing resource (i.e., CPU time). The main goal of CPU scheduling is managing CPU time for running and waiting processes to provide the users with efficient throughput. The scheduler is the part of the OS that responsible to perform this task by choosing the next process to be allocated or de-allocated. The scheduling technique is divided into non-preemptive or preemptive. The non-preemptive technique does not suspend the running process until the process releases the CPU voluntarily. In contrast, under predetermined conditions, the preemptive technique suspends the running process. The task of choosing a process for execution is defined as scheduling and the algorithm used for this choice is indicated as the scheduling algorithm. Scheduling process described in Fig. 1 [4].

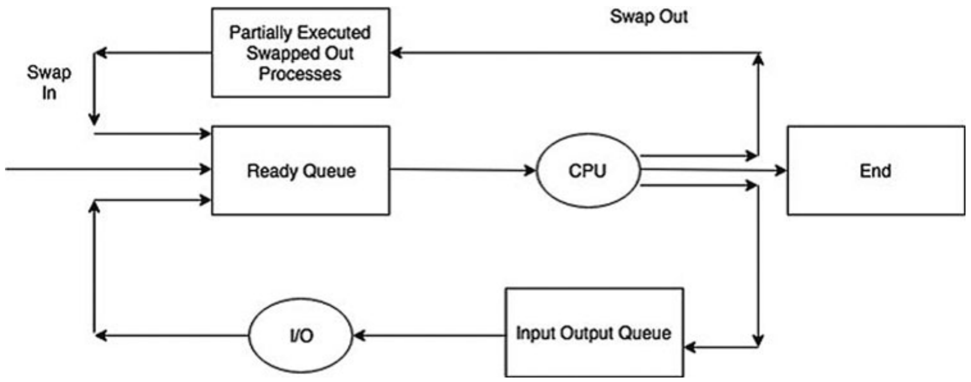


Figure 1: Schematic of scheduling

1.2 *Scheduling Criteria*

Different CPU scheduling algorithms have different characteristics and the choice of a specific algorithm influences the performance of the system, so the characteristics of the algorithms must be considered. For comparing between CPU scheduling algorithms, many scheduling criteria have been suggested (i.e., waiting time (WT), turnaround time (TT) and number of context switches (NCS)). WT is the sum of the periods that the processes spent waiting in the ready queue. TT is the interval from the submission time of a process to the completion time. NCS is the number of times the process is stopped, put at the tail of the queue to be resumed. The scheduler executes the process at the head of the queue. The scheduler is considered efficient if it minimizes WT, TT, and NCS.

1.3 *Basic Scheduling Algorithm*

1.3.1 First Come First Serve (Non-Preemptive Scheduler)

The easiest and the simplest non-preemptive CPU scheduling algorithm is the First come First Serve (FCFS) algorithm. The policy of FCFS implementation is managed with a FIFO queue in which the process that arrives to the ready queue first is assigned to the CPU first (see Fig. 2).

1.3.2 Shortest Job First (Non-Preemptive Scheduler)

In the Shortest Job First (SJF), the CPU is assigned to the process with the smallest burst time. SJF compares between the burst times of all processes residing in the ready queue and selects the process with the smallest burst time. If two processes have the same burst times, FCFS scheduling is used (see Fig. 3).

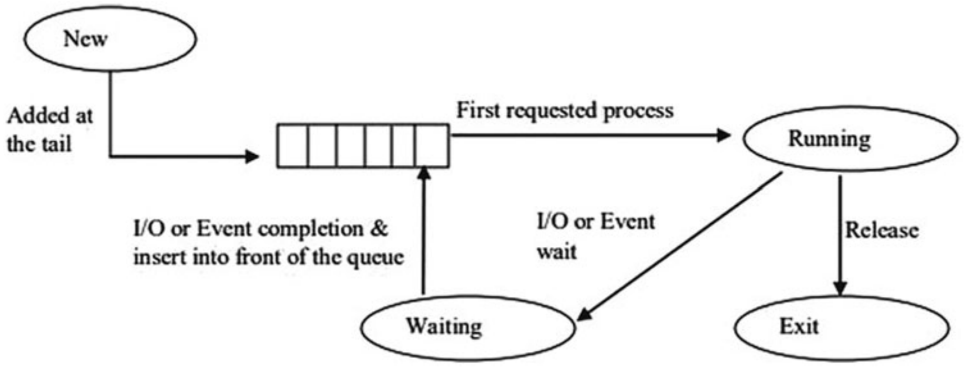


Figure 2: FCFS CPU scheduling

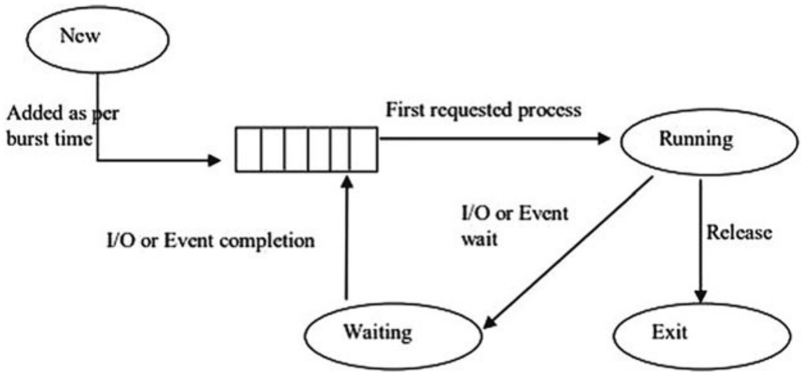


Figure 3: SJF CPU scheduling

1.3.3 Round Robin Scheduling

Round Robin scheduling (Fig. 4) allocates each process an equal portion of the CPU time. The policy of RR implementation is managed with a FIFO queue. Processes are in a circular queue; the process is put to the tail of the queue and the selected process for execution is taken from the front [5].

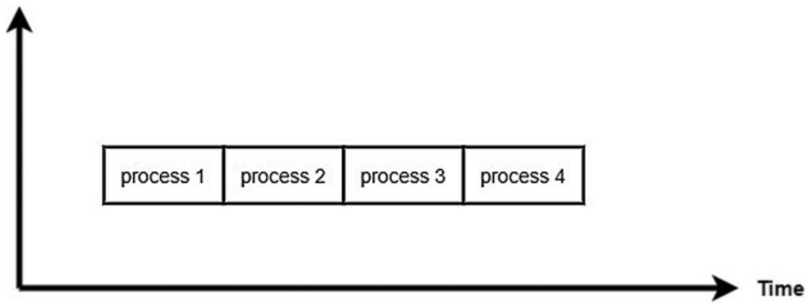


Figure 4: Round robin scheduling

The OS is driven by an interrupt (i.e., clock tick). Processes are chosen in a fixed sequence for execution. On each clock tick, the process running is paused and the next process starts execution. All processes wait in the queue for the slot of CPU time where all of them are treated as of equal importance. Process is not permitted to run to completion, but it is preempted. The implications of the preemptive process switching and the overhead are significant and must be taken into consideration. There is an inescapable time overhead when the process and context are switched (represented by the black bars in Fig. 5).

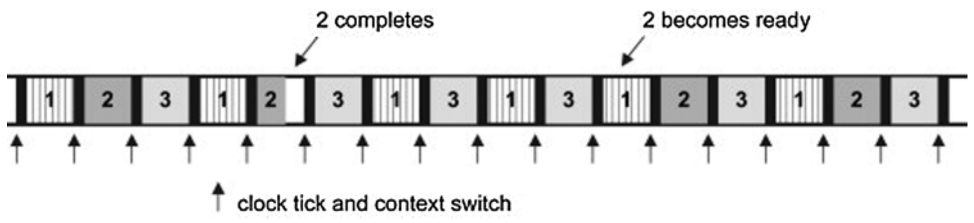
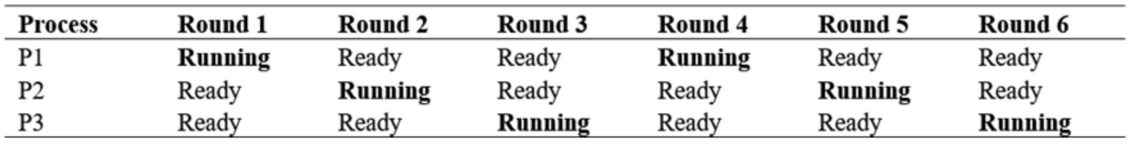


Figure 5: Context switches overhead

For example, assume that the scheduler runs three processes {A, B, C} in the sequence A, B, C, A, B, C, A, …., until they are all completed. This sequence for these processes is shown in Fig. 6. It is noticed that the processor is busy all the time because there is a process is running. The pseudocode of the RR algorithm is shown in Algorithm 1.

Algorithm 1: RR Algorithm Pseudocode

```
1 Implement the queue as a FIFO queue.
2 New processes are added to the end of the queue.
3 Assign a slot of time to the processes.
4 If the burst time of a process A less than or equal the assigned time in step 3
5       Process A runs and then leaves the queue.
6       The scheduler will proceed to next process B (if found).
7 Else
8    If the queue contains other processes (e.g., B)
9          Process A is paused after the assigned time in step 3.
10         Process A is put at the tail of the queue.
11         The scheduler will proceed to next process B.
12         Process A is resumed in the next round.
13    Else
14         Process A runs until completion without interruption.
15    End if
16 End if
17 If queue is not empty
18    Go to step 4
```



Process	Round 1	Round 2	Round 3	Round 4	Round 5	Round 6
P1	**Running**	Ready	Ready	**Running**	Ready	Ready
P2	Ready	**Running**	Ready	Ready	**Running**	Ready
P3	Ready	Ready	**Running**	Ready	Ready	**Running**

Figure 6: A sequence of process state for RR scheduling with three processes

1.4 Clustering Technique

Dividing the data into useful, meaningful, or both is known as clustering [6]. The greater the difference between the clusters and the greater the similarity between the elements within the same cluster, the better the clustering [7]. Both clustering and classification are fundamental tasks in machine learning. Clustering is used mostly as an unsupervised learning, and classification for unsupervised learning. The goal of classification is predictive, and that of clustering is descriptive which means that the target of clustering is to find out a new set of groups, the assessment of these groups is intrinsic and they are of interest in themselves. Clustering gathers data elements into subsets that similar elements are grouped together, while different elements belong to different groups [8,9]. The categorization process is determined by the selected algorithm [7]. Features' types determine the algorithm selected in the clustering (e.g., statistical algorithms for numeric data, conceptual algorithms for categorical data, or fuzzy clustering algorithms that allow data element to be joined to all clusters with a membership degree from 0 to 1). Most commonly used clustering algorithms are divided into traditional and modern clustering algorithms [10].

1.4.1 K-means Clustering Algorithm

K-means is the most common of clustering algorithms, the steps of K-means are shown in algorithm 2. The simplicity of K-means comes from the use of the stopping criterion (i.e., squared error). Suppose that D be the number of dimensions, N the number of elements, and K the number of centers, and K-means runs I iterations, hence K-means time complexity is $O(NKI)$. The goal of K-means is to minimize some objective function which is described in Eq. (1) [11].

$$\underset{\{m_k\},1 \le k \le K}{min} \sum_{k=1}^{K} \sum_{x \in C_k} \pi_x dist(x, m_k) \tag{1}$$

where K is the number of the clusters, π_x is the weight of x, $m_k = \sum_{x \in C_k} \frac{\pi_x x}{n_k}$ is the centroid of cluster C_k, The distance between centroid and the object x is computed by the function '*dist*', $1 \le k \le K$. Determining the number of clusters k is discussed in the next subsection.

Algorithm 2: *K*-means clustering algorithm

```
Input:   D, a data set of N points; K, number of clusters.
Output:  A set of K clusters.
    1  Initialization.
    2  Repeat
    3     For each point p in D do
    4        Find the nearest center and assign p to the corresponding cluster.
    5     End for
    6     Update clusters by calculating new centers using mean of the members.
    7  Until stop-iteration criteria satisfied.
    8  Return clustering result.
```

1.4.2 Silhouette Method

Silhouette method is one of the most popular clustering evaluation techniques. It determines how well each element lies within its cluster, so it measures the clustering quality. It can be summarized as follows:

(1) Applying the selected clustering algorithm for different values of k.
(2) Calculating the Within-cluster Sum of Square (WSS) for each k.
(3) Plotting WSS curve.
(4) The knee's position in the curve points out the suitable number of clusters.

Eq. (2) defines the Silhouette coefficient (S_i) of the ith data point.

$$S_i = \frac{b_i a_i}{max(b_i, a_i)} \tag{2}$$

where b_i is the average distance between all elements in different clusters and the *ith* element; a_i is the average distance between all elements in the same cluster and the *ith* element [12,13]. Fig. 7 shows the organization of the paper.

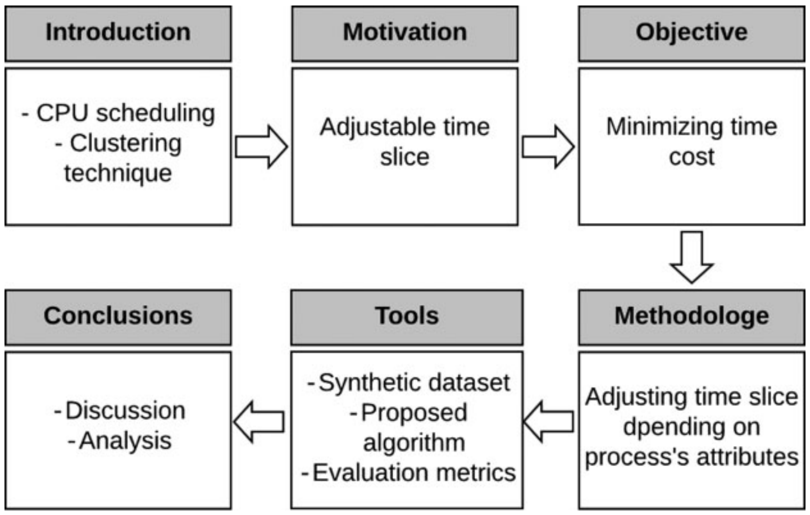


Figure 7: Organization of the paper

2 Related Works

Various forms of the RR algorithm have been proposed to minimize time cost. This section shows the most common of these forms. Tab. 1 compares between these versions of RR. Harwood et al. [14] proposed VTRR (Variable Time Round-Robin scheduling) algorithm which is a dynamic form of RR algorithm. VTRR takes into consideration the time needed to all processes when assigning the time slice to a process. Tarek et al. proposed BRR (Burst RR) a weighting form of RR by grouping five groups of processes and each process belongs to a group depending on its burst time. The weight of each process is inversely proportional to its burst time [15]. Mostafa et al. [16] proposed CTQ (Changeable Time Quantum), CTQ finds the time slice that gives smallest waiting time and turnaround time and every process runs for that time. Mishra et al. [17] proposed IRRVQ (improved Round Robin with varying time quantum) in which the processes are sorted in ascending order and the queue is divided into heavy and light. The time slice in each round is equal to the median processes' burst time. On a similar approach, Lipika proposed a dynamic form of RR, the time slice is calculated at beginning of each round depending on the residual burst times of the processes in the successive rounds, the processes are sorted in ascending order [18]. McGuire et al. [19] proposed Adaptive80 RR, the time slice equals the burst time of the process at 80th percentile. In the same way that Lipika took, Adaptive80 RR sorts the processes in ascending order. Samir et al. proposed SRDQ (SJF and RR with dynamic quantum), SRDQ divided the queue into Q1 (for short processes) and Q2 (for long processes; the process with burst time long than the median is considered long and the process with burst time small than the median is considered short. Like Lipika's and Adaptive80 RR, SRDQ sorts the processes in ascending order [20]. Mostafa [21] proposed PWRR (Proportional Weighted Round Robin) in which the burst time of a process is divided by the summation of all burst times and the time slice is assigned to a process based on its weight. Mostafa et al. [22] proposed ARR (Adjustable Round Robin) in which the short process is given a chance for completion without pausing, this is done under a predefined condition. In the same way, Uferah et al. proposed ADRR (Amended Dynamic Round Robin) in which the time slice is assigned to the process based on its burst time. Like some of its predecessors, ADRR sorts the processes in ascending order [23]. Samih et al. proposed DRR (Dynamic Round Robin) which uses clustering technique in grouping similar processes in a cluster, it differs from its predecessors in that it allocates time for the cluster and all processes get the same time within the same cluster [7]. Mostafa et al. [24] proposed DTS (Dynamic Time Slice), DTS takes the same approach as DRR in clustering the processes using K-means clustering technique, the only difference between them is the method of calculating the time slice.

Table 1: Comparison of common forms of RR (TT denotes turnaround time, and WT denotes waiting time)

Authors	Year	Technique name	Technique type	Based on	Performance Metrics		
					WT	TT	NCS
Aaron et al.	2001	VTRR	Dynamic	RR	√	√	√
Tarek et al.	2007	BRR	Dynamic	RR	√	√	√
Samih et al.	2010	CTQ	Dynamic	SRR	√	√	√
Mishra et al.	2014	IRRVQ	Dynamic	RR and SJF	√	√	–
Lipika Datta	2015	—	Dynamic	RR and SJF	√	√	√
Christoph et al.	2015	Adaptive80 RR	Dynamic	RR and SJF	√	√	√
Samir et al.	2017	SRDQ	Dynamic	RR and SJF	√	√	√
Samih	2018	PWRR	Dynamic	RR	√	√	√
Samih et al.	2019	ARR	Dynamic based on threshold	RR	√	√	√
Uferah et al.	2020	ADRR	Dynamic	RR and SJF	√	√	√
Samih et al.	2020	DRR	Dynamic based on clustering	RR and K-means	√	√	√
Samih et al.	2020	DTS	Dynamic based on clustering	RR and K-means	√	√	√

3 Proposed Algorithm

Before starting the proposed algorithm, the meanings of abbreviations used should be clarified as shown in Tab. 2.

Table 2: List of abbreviations used

Abbreviations	Meaning
PW	Process weight
PTQ	Permitted time quantum
PTS	Proportional time slice
BT	Burst time
RBT	Residual burst time
PBT	Proportional burst time
NCS	Number of context switches
TRR	Traditional Round Robin
FTS	Fixed time slice
CW	Cluster weight
CTS	Cluster time slice
C_{avg}	Average of burst times in a cluster
ATS	Adjustable time slice

The process features PW, PTQ, PBT, and NCS basically depend on BT. Firstly, the proposed approach rounds up similar processes in clusters and the resemblance between processes depends on these features. ATS algorithm uses k-means in the clustering process. Preparation of the data, clustering the data, and the dynamic implementation are the three stages of the proposed work, which are described in the following subsections.

3.1 Data Preparation

PW and NCS are calculated in this stage. Eq. (3) calculates PW, and Eq. (4) calculates NCS.

$$PW_i = \frac{BT_i}{\sum_{j=1}^{N} BT_j} \tag{3}$$

where BT_i is burst time of the *ith* process, N is the number of the processes.

$$NCS_i = \begin{cases} \left\lfloor \frac{BT_i}{FTS} \right\rfloor \; if\; BT_i \neq h \times FTS \;\; h = 1, 2, 3, \ldots \\ \frac{BT_i}{FTS} - 1 \; if\; BT_i = h \times FTS \;\; h = 1, 2, 3, \ldots \end{cases} \tag{4}$$

FTS is determined by TRR algorithm (i.e., FTS that will be used to calculate the dynamic time slice is determined by the RR in the OS). $\lfloor X \rfloor$ means the largest integer smaller than or equal to X. PTS changes from one process to another within the same round, and it changes from one round to another. The PTQ assigned to a process in a round is calculated from Eq. (5).

$$PTQ_i = \begin{cases} FTS & \text{if } BT_i > FTS \\ BT_i & \text{if } BT_i \le FTS \end{cases} \tag{5}$$

PBT of a process in a round is calculated from Eq. (6).

$$PBT_i = \frac{BT_i}{\sum_{z=1}^{n} PTQ_z} \tag{6}$$

PTS is calculated from Eq. (7).

$$PTS_i = (1 - PBT_i) \times FTS \tag{7}$$

3.2 Data Clustering

The mean reason of choosing K-means clustering algorithm in this work is that K-means works properly only with the numerical features [9]. The parameter k is determined using Silhouette method, and k-means creates k clusters of data points. BT, PW, PTQ, PBT and NCS are the features used in the clustering. The cluster is represented by the centroid within this cluster. K-means is a gradient-decent procedure starts with an initial set of K cluster-centers and consecutively updates this set. Fig. 8 displays how K-means cluster a dataset (e.g., d31 dataset) [25].

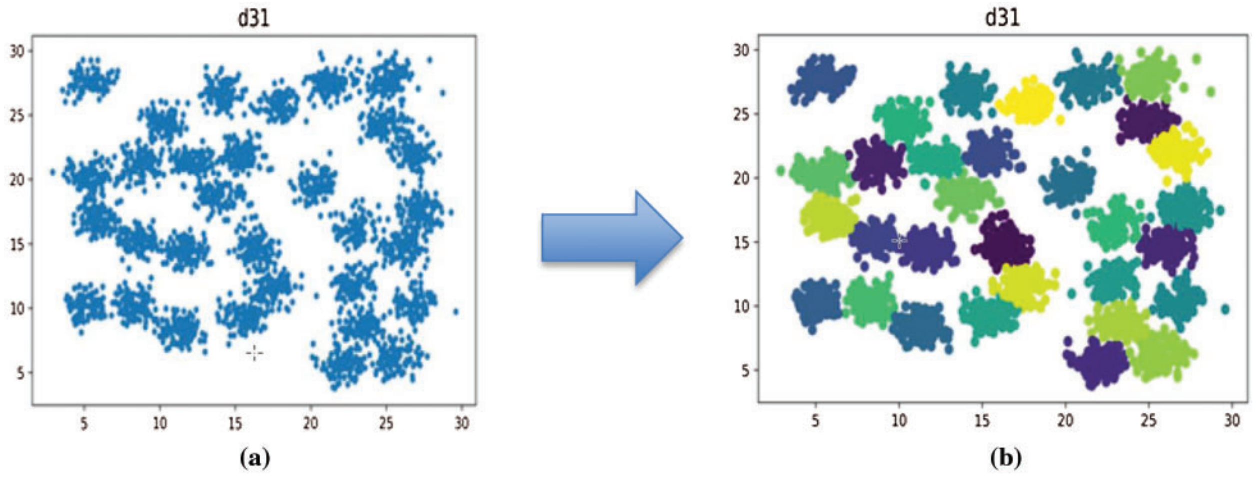


Figure 8: Clustering dataset using K-means A) d31 dataset before clustering B) d31 dataset after clustering

3.3 Dynamic Implementation

Process with long burst time causes more overhead resulted from numerous NCS. A threshold is predetermined to avoid this overhead. The proposed algorithm allows the process that close to finish to be completed and leave the queue. CW_l (i.e., weight of the lth cluster) is computed from Eq. (8) and CTS_l is computed from Eq. (9).

$$CW_l = \frac{Cavg_l}{\sum_{m=1}^{k} Cavg_m} \tag{8}$$

$$CTS_l = \left(1 - \frac{CW_l}{\sum_{l=1}^{k} CW_l}\right) \times FTS \tag{9}$$

The proposed algorithm behaved similar to the DTS algorithm [24] which takes into consideration not only the burst time of the process in the current round, but also the in the successive rounds. In addition, the proposed algorithm splits the queue into Q1 for short processes (shorter than median) and Q2 for long processes (longer than median) [20]. The proposed algorithm assigns each process with a time slice computed from Eq. (10).

$$ATS_{r,l} = \begin{cases} CTS_l + threshold & \begin{cases} if\ \left(threshold \times \left(\frac{BT_r}{FTS} + 1\right) \geq mod(BT_r, CTS)\right) \\ and\ \ if\ \ (BT_r < Median) where\ \ mod(BT_r, CTS) > 0 \end{cases} \\ CTS_l & \begin{cases} if\ \ mod(BT_r, CTS) = 0 \\ and\ \ if (BT_r \geq Median) \end{cases} \end{cases} \tag{10}$$

where $ATS_{r,l}$ is the adjustable time slice assigned to the process P_r in the cluster l. In consecutive rounds, RBT is updated according to Eq. (11). The proposed algorithm is described in Fig. 9.

$$RBT_i = BT_i - CTS_i \tag{11}$$

4 Experimental Implementation

The computer's specifications used in the experiments are shown in Tab. 3:

4.1 Benchmark Datasets

Nine artificial datasets were generated to be used in the experiments. Each dataset has a number of processes varying in the BTs which have been randomly generated. To prove that the proposed algorithm is valid for diverse data, each dataset contains different number of processes varying in their burst times and accordingly the PW, NCS, PTQ, PBT, and PTS are different. Tab. 4 presents the characteristics of the datasets used.

4.2 Performance Evaluation

The proposed algorithm has been compared with five common algorithms VTRR, DTS, ADRR PWRR and RR. The submitted processes may be arrived in the same time or different times. The experiments were conducted taking into account the two cases.

Case 1 (same arrival time): The average turnaround times and waiting times comparisons are shown in Fig. 10, the NCS comparisons are shown in Fig. 11. The improvement of ATS over the compared algorithms is shown in Fig. 12. Time cost comparisons are shown in Tab. 5 and the improvement is shown in Tab. 6.

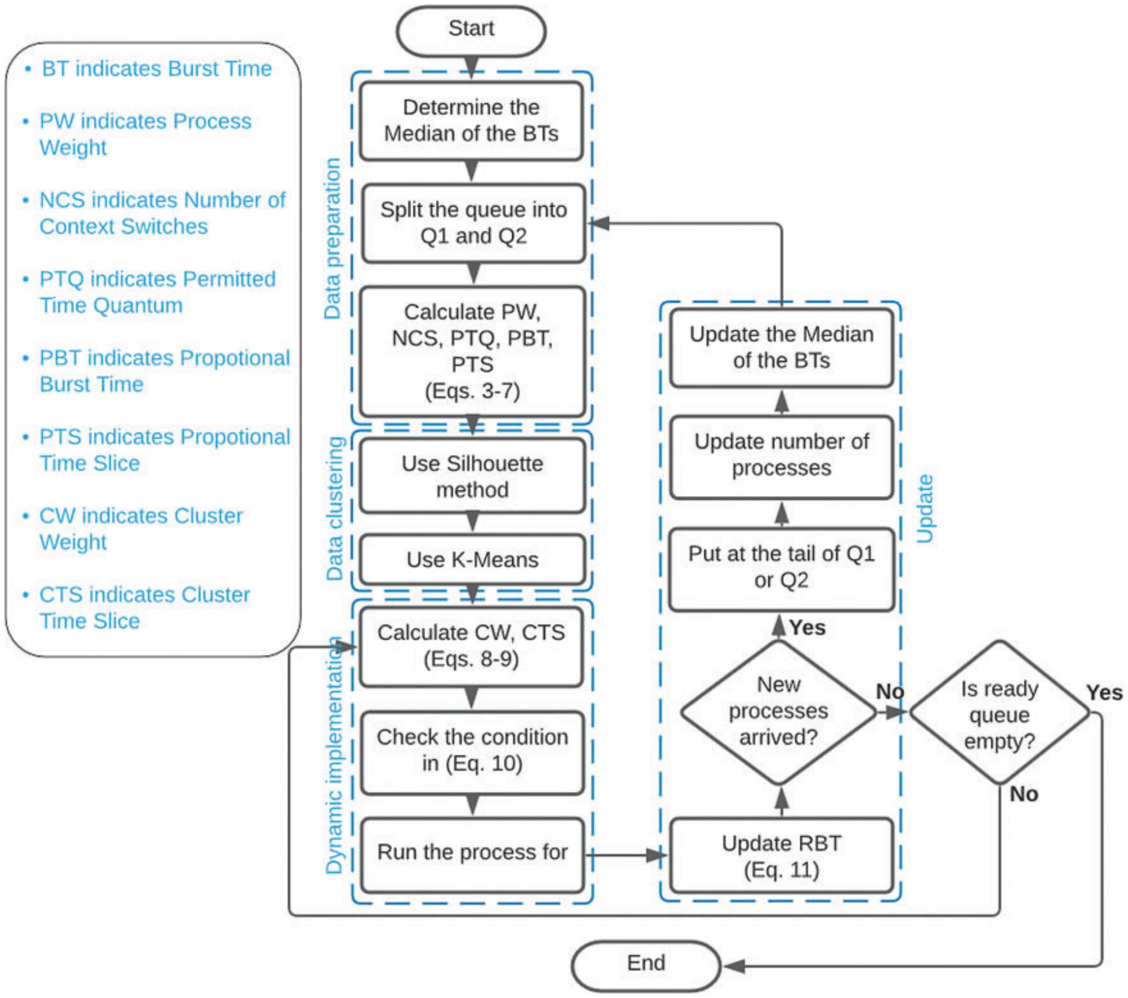

BT indicates Burst Time
PW indicates Process Weight
NCS indicates Number of Context Switches
PTQ indicates Permitted Time Quantum
PBT indicates Propotional Burst Time
PTS indicates Propotional Time Slice
CW indicates Cluster Weight
CTS indicates Cluster Time Slice
Start
Data preparation
Determine the Median of the BTs
Split the queue into Q1 and Q2
Calculate PW, NCS, PTQ, PBT, PTS (Eqs. 3-7)
Data clustering
Use Silhouette method
Use K-Means
Dynamic implementation
Calculate CW, CTS (Eqs. 8-9)
Check the condition in (Eq. 10)
Run the process for
Update
Update the Median of the BTs
Update number of processes
Put at the tail of Q1 or Q2
Yes
New processes arrived?
No
Is ready queue empty?
Yes
No
Update RBT (Eq. 11)
End

Figure 9: Algorithm flowchart

Table 3: Test bed specifications

Hardware specifications	Software specifications
CPU - Intel core i5-2400 (3.10 GHz) 1 TB HDD 16GB RAM	Gnu/Linux Fedora 28 OS (Python 3.7.6 (default, Jan 8 2020, 20:23:39)

Case 2 (different arrival time): Process arrival was modeled as Poisson random process. The arrival times are exponentially distributed [16]. The average turnaround times and waiting times comparison are shown in Fig. 13, the NCS comparisons are shown in Fig. 14. The improvement of ATS over the compared algorithms is shown in Fig. 15. Time cost comparisons are shown in Tab. 7 and the improvement is shown in Tab. 8.

Table 4: Datasets specifications

Dataset ID	#Processes	#Features	Features				
			PW	NCS	PTQ	PBT	PTS
1	10	5	√	√	√	√	√
2	15	5	√	√	√	√	√
3	20	5	√	√	√	√	√
4	25	5	√	√	√	√	√
5	30	5	√	√	√	√	√
6	35	5	√	√	√	√	√
7	40	5	√	√	√	√	√
8	45	5	√	√	√	√	√
9	50	5	√	√	√	√	√

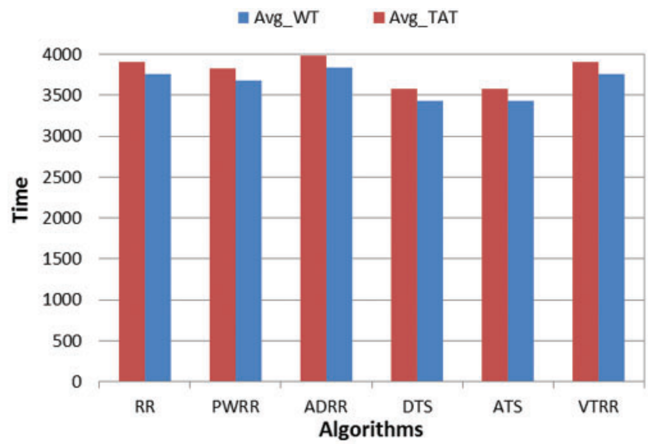


Figure 10: Avg_WT and Avg_TAT comparison (case 1)

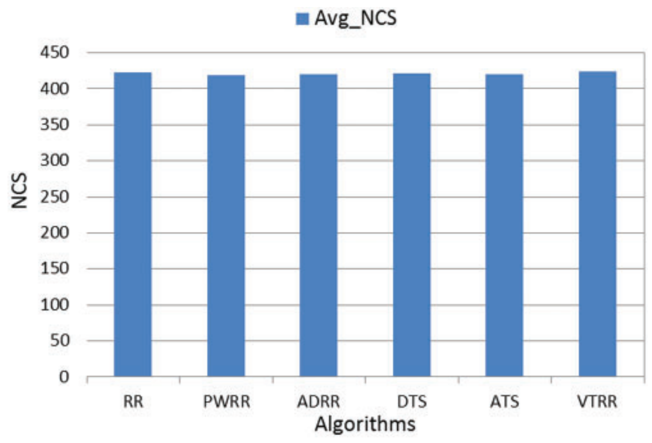


Figure 11: NCS comparison (case 1)

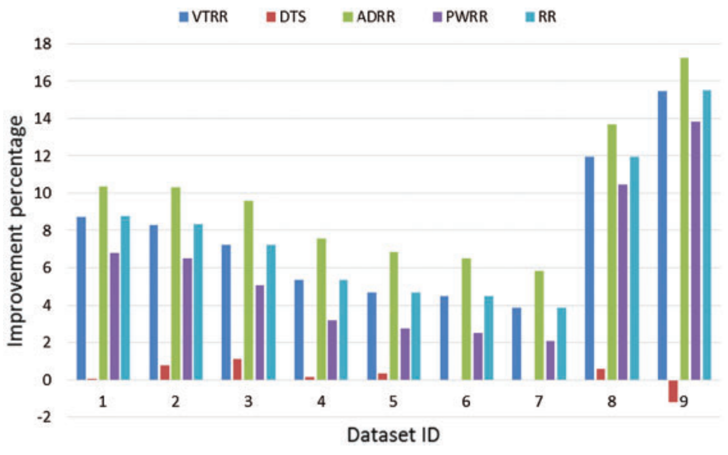


Figure 12: Improvement comparison (case 1)

Table 5: Average waiting time and turnaround time comparison between the proposed algorithm and five scheduling algorithms (case 1)

Dataset	ATS			DTS			RR		
	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS
1	1055.81	1202.81	141	1055.83	1202.83	142	1137.68	1284.68	148
2	1632.86	1775.46	205	1646.13	1788.73	206	1751.48	1894.08	210
3	2244.61	2388.21	278	2270.28	2413.88	279	2368.01	2511.61	277
4	2900.94	3045.9	346	2904.73	3049.69	347	2998.81	3143.77	346
5	3556.94	3702.61	415	3570.15	3715.82	416	3658.62	3804.29	415
6	4163.22	4307.91	487	4162.25	4306.93	488	4272.39	4417.08	481
7	4874.44	5021.64	568	4873.75	5020.95	569	4980.18	5127.38	560
8	5064.15	5211.59	641	5094.15	5241.6	642	5664.03	5811.47	631
9	5409.32	5557.64	705	5343.27	5491.59	706	6289.8	6438.12	704
Average	3433.588	3579.308		3435.615556	3581.336		3680.111	3825.831	
Improvement%				0.194947607	0.180004		6.05453	5.762972	
Dataset	VTRR			RR			ADRR		
	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS
1	1163.1	1310.1	142	1163.9	1310.9	142	1186	1333	140
2	1786.8	1929.4	207	1787.33	1929.93	207	1828	1970.6	205
3	2424.78	2568.38	279	2425.15	2568.75	277	2490	2633.6	277
4	3069.03	3213.99	349	3069.24	3214.2	349	3144	3288.96	345
5	3734.82	3880.49	420	3735	3880.67	420	3824	3969.67	415
6	4361.7	4506.38	491	4361.8	4506.49	488	4457.71	4602.4	486
7	5073.72	5220.92	573	5073.8	5221	568	5180.5	5327.7	569
8	5760.97	5908.41	645	5761.04	5908.49	640	5877.78	6025.22	641
9	6414.06	6562.38	716	6414.64	6562.96	716	6553.2	6701.52	705
Average	3754.331	3900.05		3754.656	3900.377		3837.91	3983.63	
Improvement%	7.972041	7.591376		7.986159	7.604464		10.00378	9.535625	

Table 6: Improvement percentages of the proposed algorithm over five scheduling algorithms (case 1)

	DTS			PWRR			VTRR			RR			ADRR		
	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS
1	0.002	0.002	1.399	7.196	6.373	4.730	9.224	8.189	0.704	9.287	8.245	0.704	10.977	9.767	−0.714
2	0.806	0.742	1.442	6.773	6.263	2.381	8.615	7.979	0.966	8.643	8.004	0.966	10.675	9.903	0.000
3	1.131	1.063	0.714	5.211	4.913	−0.361	7.430	7.015	0.358	7.444	7.028	−0.361	9.855	9.318	−0.361
4	0.130	0.124	1.143	3.264	3.113	0.000	5.477	5.230	0.860	5.483	5.236	0.860	7.731	7.390	−0.290
5	0.370	0.356	1.425	2.779	2.673	0.000	4.763	4.584	1.190	4.767	4.588	1.190	6.984	6.728	0.000
6	−0.023	−0.023	1.016	2.555	2.472	−1.247	4.551	4.404	0.815	4.553	4.407	0.205	6.606	6.399	−0.206
7	−0.014	−0.014	1.045	2.123	2.062	−1.429	3.928	3.817	0.873	3.929	3.818	0.000	5.908	5.745	0.176
8	0.589	0.573	0.774	10.591	10.322	−1.585	12.096	11.794	0.620	12.097	11.795	−0.156	13.842	13.504	0.000
9	−1.236	−1.203	1.674	13.999	13.676	−0.142	15.665	15.311	1.536	15.672	15.318	1.536	17.455	17.069	0.000

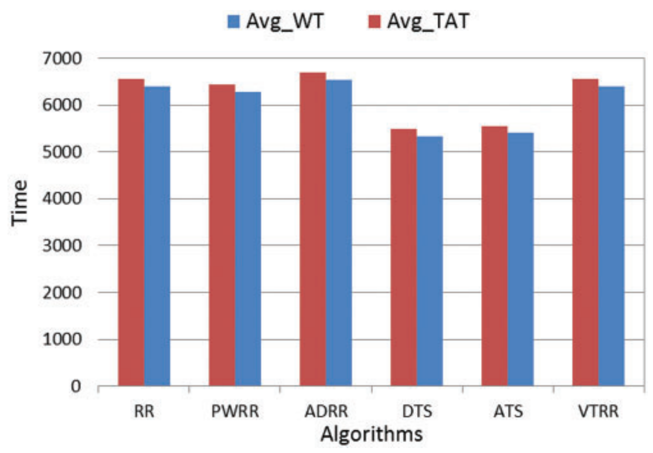


Figure 13: Avg_WT and Avg_TAT comparison (case 2)

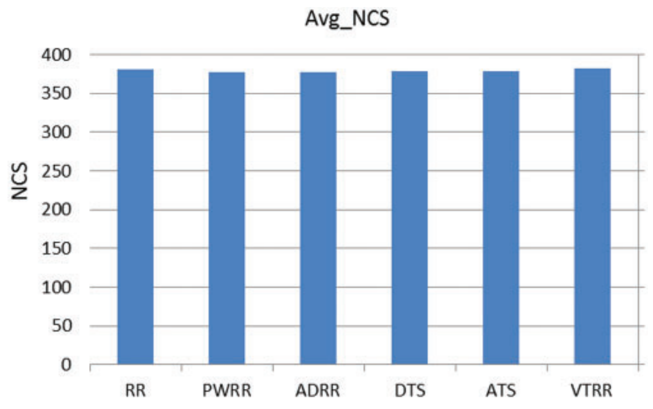


Figure 14: NCS comparison (case 2)

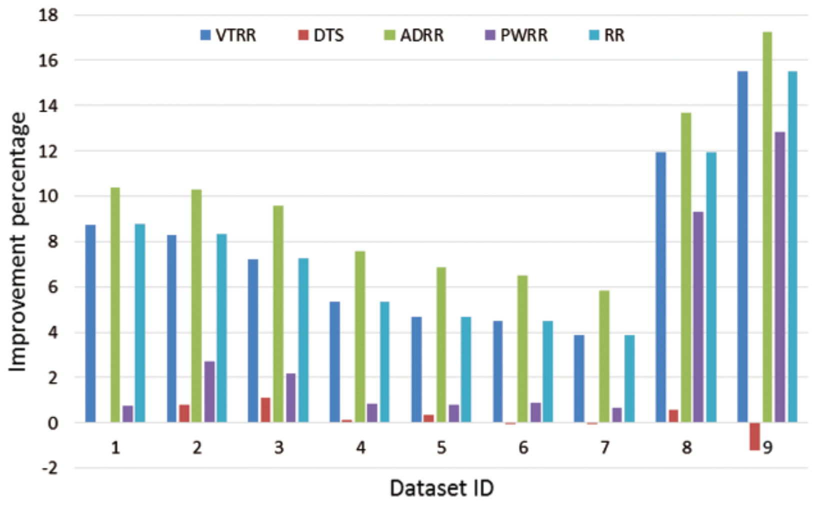


Figure 15: Improvement comparison (case 2)

Table 7: Average waiting time and turnaround time comparison between the proposed algorithm and five scheduling algorithms (case 2)

Dataset	ATS			DTS			PWRR		
	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS
1	1054.75	1201.61	126	1054.77	1201.63	127	1136.54	1136.54	133
2	1631.23	1773.69	184	1644.48	1786.94	185	1749.73	1749.73	189
3	2242.37	2385.82	250	2268.01	2411.47	251	2365.64	2365.64	249
4	2898.04	3042.85	311	2901.83	3046.64	312	2995.81	2995.81	311
5	3553.38	3698.91	373	3566.58	3712.1	374	3654.96	3654.96	373
6	4159.06	4303.6	438	4158.09	4302.62	439	4268.12	4268.12	432
7	4869.57	5016.62	511	4868.88	5015.93	512	4975.2	4975.2	504
8	5059.09	5206.38	576	5089.06	5236.36	577	5658.37	5658.37	567
9	5403.91	5552.08	634	5337.93	5486.1	635	6283.51	6283.51	633
Average	3430.15	3575.73		3432.18	3577.75		3676.43	3676.43	
Improvement%				0.19	0.18		6.05	5.76	

(Continued)

Table 7: Continued

Dataset	VTRR			RR			ADRR		
	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS
1	1161.94	1308.79	127	1162.74	1309.59	127	1184.81	1331.67	126
2	1785.01	1927.47	186	1785.54	1928	186	1826.17	1968.63	184
3	2422.36	2565.81	251	2422.73	2566.18	249	2487.51	2630.97	249
4	3065.96	3210.78	314	3066.17	3210.99	314	3140.86	3285.67	310
5	3731.09	3876.61	378	3731.27	3876.79	378	3820.18	3965.7	373
6	4357.34	4501.87	441	4357.44	4501.98	439	4453.25	4597.8	437
7	5068.65	5215.7	515	5068.73	5215.78	511	5175.32	5322.37	512
8	5755.21	5902.5	580	5755.28	5902.58	576	5871.9	6019.2	576
9	6407.65	6555.82	644	6408.23	6556.4	644	6546.65	6694.82	634
Average	3750.58	3896.15		3750.90	3896.48		3834.07	3979.65	
Improvement%	7.97	7.59		7.99	7.60		10.00	9.54	

Table 8: Improvement percentages of the proposed algorithm over five scheduling algorithms (case 2)

	DTS			PWRR			VTRR			RR			ADRR		
	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS
1	0.0019	0.0017	1.3986	7.1962	6.3728	4.7297	9.2245	8.19	0.7	9.2869	8.2455	0.704	10.9772	9.7667	0.0000
2	0.8061	0.7419	1.4423	6.7726	6.2627	2.3810	8.6154	7.98	0.97	8.6425	8.0039	0.966	10.6751	9.9026	0.0000
3	1.1307	1.0634	0.7143	5.2111	4.9132	−0.3610	7.4304	7.02	0.36	7.4445	7.0283	−0.361	9.8550	9.3177	−0.4016
4	0.1305	0.1243	1.1429	3.2636	3.1131	0.0000	5.4770	5.23	0.86	5.4834	5.2361	0.86	7.7309	7.3902	−0.3226
5	0.3700	0.3555	1.4252	2.7792	2.6728	0.0000	4.7627	4.584	1.19	4.7673	4.5884	1.191	6.9838	6.7275	0.0000
6	−0.0233	−0.0228	1.0163	2.5552	2.4715	−1.2474	4.5505	4.404	0.82	4.5527	4.4065	0.21	6.6063	6.3986	−0.2288
7	−0.01	−0.014	1.04	2.12	2.06	−1.43	3.93	3.817	0.87	3.9292	3.8184	0.00	5.9079	5.7447	0.1953
8	0.58	0.57	0.77	10.6	10.32	−1.59	12.1	11.79	0.62	12.097	11.7949	−0.156	13.8425	13.5037	0.0000
9	−1.24	−1.2	1.67	13.1	13.68	−0.14	15.67	15.31	1.54	15.672	15.3181	1.536	17.4553	17.0690	0.0000

5 Conclusion

This paper presented a dynamic version of RR. The proposed goal is to reduce the time cost (i.e., waiting time and turnaround time). The traditional RR uses fixed slot of time for each process in all rounds regardless the BT of the process, however, the proposed algorithm (ATS) assigns particular time slice for each process. This time slice is different from the other processes in the same cluster. If the BT of a process is longer than the assigned time slice, the process will be put at the end of the queue and assigned a new time slice in the new round. In addition, the proposed algorithm starts by grouping similar processes in a group depending on the similarity between the features. Each cluster is a signed a slot of time and every process in this cluster is assigned this time slice. In a round, some processes may finish their BT and leave the queue, and may new processes arrive. In both cases the number of the clusters and clusters' weights will be updated and therefore the time slice assigned to a cluster and the number of the processes in the cluster will be updated. Furthermore, ATS takes into account the remaining times of the survived processes and allow short process to complete (under predetermined conditions) without interruption even if its BT longer than the assigned Time Slice, in other words, ATS grants more time to the process that is close to complete in the current and consecutive rounds. Comparison has been done between ATS and five common versions of RR from the point of view of time cost. The results showed that the proposed algorithm achieves noticeable improvements.

Funding Statement: The authors extend their appreciation to Deanship of Scientific Research at King Khalid University for funding this work through the Research Groups Project under Grant Number RGP.1/95/42.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] K. Chandiramani, R. Verma and M. Sivagami, "A modified priority preemptive algorithm for CPU scheduling," *Procedia Computer Science*, vol. 165, pp. 363–369, 2019.
[2] I. S. Rajput and D. Gupta, "A priority based round robin CPU scheduling algorithm for real time systems," *International Journal of Innovations in Engineering and Technology*, vol. 1, no. 3, pp. 1–11, 2012.
[3] M. R. Reddy, V. V. D. S. S. Ganesh, S. Lakshmi and Y. Sireesha, "Comparative analysis of CPU scheduling algorithms and their optimal solutions," in *Int. Conf. on Computing Methodologies and Communication (ICCMC)*, Erode, India, pp. 255–260, 2019.
[4] R. Somula, S. Nalluri, M. K. NallaKaruppan, S. Ashok and G. Kannayaram, "Analysis of CPU scheduling algorithms for cloud computing," in *Smart Intelligent Computing and Applications. Smart Innovation, Systems and Technologies*, vol. 105, pp. 375–382, 2019.
[5] A. Silberschatz, P. B. Galvin and G. Gagne, *Operating System Concepts-10th*. Hoboken, New Jersey, USA: John Wiley & Sons, Inc., 10th ed., 2018.
[6] P. Lasek and J. Gryz, "Density-based clustering with constraints," *Computer Science and Information Systems*, vol. 16, no. 2, pp. 469–489, 2019.
[7] S. M. Mostafa and H. Amano, "Dynamic round robin CPU scheduling algorithm based on K-means clustering technique," *Applied Sciences*, vol. 10, no. 15, pp. 5134, 2020.
[8] L. Rokach and O. Maimon, "Clustering methods," in *The Data Mining and Knowledge Discovery Handbook*, Springer, Boston, pp. 321–349, 2005.
[9] P. Berkhin, "A Survey of clustering data mining techniques," in *Grouping Multidimensional Data. Springer*, Berlin, Heidelberg, pp. 321–349, 2006.
[10] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," *Annals of Data Science*, vol. 2, no. 2, pp. 165–193, 2015.
[11] J. Wu, "Cluster analysis and K-means clustering: An introduction," in *Advances in K-Means Clustering*, Springer, Berlin Heidelberg, pp. 1–16, 2012.
[12] U. G. Inyang, O. O. Obot, M. E. Ekpenyong and A. M. Bolanle, "Unsupervised learning framework for customer requisition and behavioral pattern classification," *Modern Applied Science*, vol. 11, no. 9, pp. 151–164, 2017.
[13] Y. Liu, Z. Li, H. Xiong, X. Gao and J. Wu, "Understanding of internal clustering validation measures," in *IEEE Int. Conf. on Data Mining*, Sydney, NSW, Australia, pp. 911–916, 2010.
[14] A. Harwood and H. Shen, "Using fundamental electrical theory for varying time quantum uni-processor scheduling," *Journal of Systems Architecture*, vol. 47, no. 2, pp. 181–192, 2001.
[15] T. Helmy and A. Dekdouk, "Burst round robin as a proportional-share scheduling algorithm," in *4th IEEE GCC Conference on Towards Techno-Industrial Innovations*, Bahrain, pp. 424–428, 2007.
[16] S. M. Mostafa, S. Z. Rida and S. H. Hamad, "Finding time quantum of round robin CPU scheduling algorithm in general computing systems using integer programming," *International Journal of Research and Reviews in Applied Sciences (IJRRAS)*, vol. 5, no. 1, pp. 64–71, 2010.

[17] M. Mishra and F. Rashid, “An improved round robin CPU scheduling algorithm with varying time quantum,” *International Journal of Computer Science, Engineering and Applications*, vol. 4, no. 4, pp. 1–8, 2014.

[18] L. Datta, “Efficient round robin scheduling algorithm with dynamic time slice,” *International Journal of Education and Management Engineering*, vol. 5, no. 2, pp. 10–19, 2015.

[19] C. McGuire and J. Lee, “The adaptive80 round robin scheduling algorithm,” in *Transactions on Engineering Technologies*, Dordrecht, Netherlands, Springer, pp. 243–258, 2015. https://doi.org/10.1007/978-94-017-7236-5_17.

[20] S. Elmougy, S. Sarhan and M. Joundy, “A novel hybrid of shortest job first and round robin with dynamic variable quantum time task scheduling technique,” *Journal of Cloud Computing*, vol. 6, no. 1, pp. 1–12, 2017.

[21] S. M. Mostafa, “Proportional weighted round robin: A proportional share CPU scheduler in time sharing systems,” *International Journal of New Computer Architectures and Their Applications*, vol. 8, no. 3, pp. 142–147, 2018.

[22] S. M. Mostafa and H. Amano, “An adjustable round robin scheduling algorithm in interactive systems,” *Information Engineering Express (IEE)*, vol. 5, no. 1, pp. 11–18, 2019.

[23] U. Shafi, M. Shah, A. Wahid, K. Abbasi, Q. Javaid *et al.*, “A novel amended dynamic round robin scheduling algorithm for timeshared systems,” *International Arab Journal of Information Technology*, vol. 17, no. 1, pp. 90–98, 2020.

[24] S. M. Mostafa and H. Amano, “An adjustable variant of round robin algorithm based on clustering technique,” *Computers, Materials & Continua*, vol. 66, no. 3, pp. 3253–3270, 2020.

[25] P. Fränti and S. Sieranoja, “Clustering basic benchmark-D31,” 2020. [Online]. Available: http://cs.joensuu.fi/sipu/datasets/D31.txt.