Computers, Materials & Continua
DOI:10.32604/cmc.2021.014675
Article

Tech Science Press

An Adjustable Variant of Round Robin Algorithm Based on Clustering Technique

Samih M. Mostafa[1,*] **and Hirofumi Amano**[2]

[1]Faculty of Computers and Information, South Valley University, Qena, 83523, Egypt
[2]Research Institute for Information Technology, Kyushu University, Fukuoka, 819-0395, Japan
[*]Corresponding Author: Samih M. Mostafa. Email: samih_montser@sci.svu.edu.eg
Received: 08 October 2020; Accepted: 25 October 2020

Abstract: CPU scheduling is the basic task within any time-shared operating system. One of the main goals of the researchers interested in CPU scheduling is minimizing time cost. Comparing between CPU scheduling algorithms is subject to some scheduling criteria (e.g., turnaround time, waiting time and number of context switches (NCS)). Scheduling policy is divided into preemptive and non-preemptive. Round Robin (RR) algorithm is the most common preemptive scheduling algorithm used in the time-shared operating systems. In this paper, the authors proposed a modified version of the RR algorithm, called dynamic time slice (DTS), to combine the advantageous of the low scheduling overhead of the RR and favor short process for the sake of minimizing time cost. Each process has a weight proportional to the weights of all processes. The process's weight determines its time slice within the current period. The authors benefit from the clustering technique in grouping the processes that are similar in their attributes (e.g., CPU service time, weight, allowed time slice (ATS), proportional burst time (PBT) and NCS). Each process in a cluster is assigned the average of the processes' time slices in this cluster. A comparative study of six popular scheduling algorithms and the proposed approach on nine groups of processes vary in their attributes was performed and the evaluation was measured in terms of waiting and turnaround times, and NCS. The experiments showed that the proposed algorithm gives better results.

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

1 Introduction

This section is divided into two subsections: (i) CPU scheduling, and (ii) Clustering technique.

1.1 CPU Scheduling

Allocating and de-allocating the CPU to a specific process is known as process scheduling (also known as CPU scheduling) [1–3]; the piece of the operating system that performs these functions is called the scheduler. From the processes that are waiting in the memory to receive

service from the CPU, the scheduler chooses the next process to be assigned to the CPU. The scheduling scheme may be non-preemptive or preemptive, and many CPU scheduling algorithms have been suggested. First Come First Served (FCFS) algorithm executes processes upon their order of arrives. Shortest Job First (SJF) algorithm selects the process with the shortest burst time. Under Shortest Remaining Time First (SRTF) algorithm, the execution of the running process is paused when a process with shorter burst time arrives. Under priority scheduling, the execution of the running process is paused when a process with higher priority arrives [4].

The most common of the preemptive scheduling algorithms is the RR scheduling [5], known hereafter as Standard RR (SRR), used in timesharing and real-time operating systems [6]. In RR scheduling, the operating system is driven by a regular interrupt by the system timer after a short fixed interval called standard time slice (STS) [7–10]. After that interruption, the scheduler switches the context to the next process selected from the circular queue [11,12]. Thus, all processes are given a chance to receive CPU service time for a short fixed period. The fixed time slice influences the efficiency of RR algorithm; short time slice leads to high overheads, and long time slice may leads to starvation between processes. In addition, scheduling criteria (i.e., waiting time, turnaround time, and NCS) affect performance of the scheduling algorithm [13–15].

1.2 Clustering Technique

Clustering means dividing the data into groups that are useful and meaningful [16]; greater homogeneity (or similarity) within a cluster and greater difference between clusters bring about better clustering. Clustering is a type of classification in that it generates labels of the clusters [17, 18]. Classification of data can be completed using clustering without prior knowledge. A clustering algorithm is the process of dividing abstract or physical object into a collection of similar objects. Cluster is a collection of data points; points in the same cluster are like each other and different from points in other clusters. The type of the data determines the algorithm used in the clustering, for example, statistical algorithms are used for clustering numeric data, categorical data is clustered using conceptual algorithms, fuzzy clustering algorithms allow data point to be classified into all clusters with a degree of membership ranging from 0 to 1, this degree indicates the likeness of the data point to the mean of the cluster. Clustering algorithms are categorized into traditional and modern [19]. K-means is the most commonly and simplest clustering algorithm. Its simplicity comes from the use of the squared error as stopping criterion. In addition, the time complexity of the K-means algorithm is low $O(nkt)$, where n: the number of objects, k: The number of clusters, and t: The number of iterations. K-means algorithm divides the dataset into K clusters $(C_1, C_2, \ldots, C_K)$, represented by their centers or means to minimize some objective function that depends on the vicinity of the subjects to the cluster centroids. The function to be minimized in K-means is described in Eq. (1) [20].

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

where K is the number of the clusters set by the user, 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, and the function '*dist*' computes the distance between the object x and

the centroid m_k, $1 \leq k \leq K$. The K-means clustering method requires all data to be numerical. The pseudo-code of K-means algorithm is as follows:

Algorithm: K-Means

Input —Dataset
—number of clusters

Output —K clusters

Step-1: —Initialize K centers of the cluster

Step-2: —Repeat
—Calculate the mean of all the objects belonging to that cluster

$$\mu_k = \frac{1}{N_k} \sum_{q=1}^{N_k} x_q$$

where μ_k is the mean of cluster k and N_k is the number of points belonging to that cluster
—Assign objects to the closest cluster centroid
—Update cluster centroids based on the assignment
—**Until** centroids do not change

Determining the optimal number of clusters in a dataset is an issue in clustering. The most commonly clustering evaluation technique used is the Silhouette method which measures clustering quality by determining how well each data point lies within its cluster. The Silhouette method can be summarized as follows:

1. Apply the clustering algorithm for different values of k. For instance, by varying k from 1 to n clusters.
2. For each k, the total Within-cluster Sum of Square (WSS) is calculated.
3. Plot the curve of WSS according to the value of k.
4. The location of a knee in the curve indicates the appropriate number of clusters.

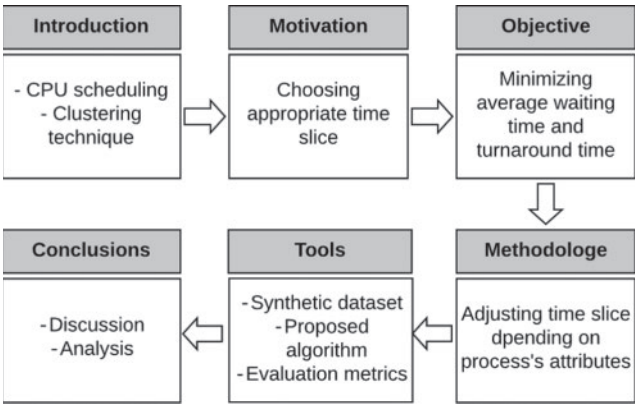

Figure 1: Organization of the paper

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 the ith data point and all data points in different clusters; a_i, is the average distance between the ith data point and all other data points in the same cluster [16,21].

Motivation: The time slice used in RR scheduling algorithm influences the performance of the timesharing systems. The time slice used should be chosen to avoid starvation (resulted from choosing long time slice) and overheads of more context switches (resulted from choosing short time slice).

Organization: The rest of this paper is divided as follows: Section 2 discusses the related work. The proposed algorithm is presented in Section 3. Section 4 discusses the experimental implementation. Section 5 concludes this research work (see Fig. 1).

2 Related Works

For better CPU performance in most of the operating systems, the RR scheduling algorithm is widely implemented. Many versions of the RR algorithm have been proposed to minimize turnaround and average waiting times. This section discusses the most common variants of the RR. Tab. 1 shows a comparison between the known variants of SRR. Variable Time Round-Robin scheduling (VTRR) is a dynamic version of the SRR algorithm proposed by Aaron et al. [22]. VTRR assigned a time slice to a process taking into consideration the time needed to all processes. A weighting version of SRR is proposed by Tarek et al. [23]. The authors groups the processes into five categories based on the burst times. The weight of a process is inversely proportional to its burst time; process with low weight receives less time and vice versa. If the burst time of a process less than or equal to 10 tu, it receives 100% of the time defined by SRR. If the burst time of a process less than or equal to 25 tu, it receives 80% of the time defined by SRR, and so on. Changeable Time Quantum (CTQ) is a dynamic version of SRR proposed by Samih et al. [14]. In every round, CTQ finds the time quantum that gives the smallest average waiting time, and the processes executes for this time. Modifying the time slice at the beginning of each round is another dynamic version of SRR proposed by Lipika [24], in which the time slice is calculated with respect to the residual burst times (RBT) in the successive rounds. Lipika benefited from SJF in which the processes are ordered increasingly based on their burst times (i.e., the process with the highest burst time will be at the tail of the ready queue and the process with the lowest burst time will be at the head of the queue) [25–27]. Adaptive80 RR is a dynamic variant of SRR proposed by Christoph and Jeonghw [6]. The time quantum equals process's burst time at 80th percentile. The authors imitated Lipika's [24] in sorting the processes in increasing order, and the time slice in each is calculated depending on the burst times of the processes in the queue. A hybrid scheduling technique based on SJF and SRR named SJF and RR with dynamic quantum (SRDQ) is proposed by Samir et al. [28]. SRDQ divided the queue into Q1 and Q2 based on the median; Q2 for long the processes (longer than the median) and Q1 for the short processes (shorter than the median). Like Lipika's [24] and Adaptive80 RR [6] algorithms, SRDQ sorts the processes in the queue in ascending order in each subqueue. Proportional Weighted Round Robin (PWRR) is a modified version of SRR proposed by Samih [15]. PWRR assigns the time slice to a process proportional to its weigh which is calculated by dividing its burst time by the summation of all processes' burst times in the queue. Adjustable Round Robin (ARR)

is another dynamic version of the SRR proposed by Samih et al. [13]. Under a predefined condition, ARR gives short process a chance of execution until termination without interruption. Amended Dynamic Round Robin (ADRR) is a dynamic version of SRR proposed by Uferah et al. [12]. In every cycle, the time slice is adjusted based on the burst time of the process. Like Lipika's [24], SRDQ [28] and Adaptive80 RR [6] algorithms, the processes are sorted in ascending order. Dynamic Round Robin (DRR) CPU scheduling algorithm is a dynamic version of the SRR algorithm based on clustering technique proposed by Samih et al. [29]. DRR starts by grouping similar processes in a cluster; the number of clusters is obtained from Silhouette method. Each cluster has a weight and is assigned a time slice, and all processes in a cluster are assigned the same time slice. They benefit from the clustering technique in grouping processes that resemble each other in their features.

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

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

3 The Proposed Algorithm

The processes' weights (PW), ATS, PBT, and NCS depend on the process's burst times (BT), and are calculated in the next subsections. The proposed work starts by grouping similar processes in clusters. The similarity between processes depends on BT, ATS, NCS, PW and PBT. K-means is the most commonly algorithm used in clustering. The proposed work consists of three phases: Data preparation, data clustering, and dynamic implementation.

3.1 Data Preparation

In the data preparation phase, PW and NCS are calculated. The weight of the ith process (PW_i) is calculated from Eq. (3):

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

where N is the number of the processes, and BT_i is the burst time of the ith process. NCS is calculated from Eq. (4):

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

STS is determined by SRR, and $\lfloor X \rfloor$ indicates the largest integer smaller than or equal to X. If the burst time of a process is greater than the STS, the time slice of this process equals STS. The ATS assigned to a process in a round is calculated from Eq. (5).

$$ATS_i = \begin{cases} STS & if \;\; BT_i > STS \\ BT_i & if \;\; BT_i \leq STS \end{cases} \tag{5}$$

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

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

The proportional time slice (PTS) of a process in a round is calculated from Eq. (7).

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

3.2 Data Clustering

In the data clustering phase, Silhouette method is used to find the optimum number, k, of clusters, then K-means algorithm clusters the data into k number. BT, PW, ATS, PBT and NCS are the clustering metrics used in the proposed work. Each data point is assigned to the nearest centroid, each group of points assigned to the same centroid results cluster. The assignment and updating steps are repeated until all centroids do not change. The Euclidean ($L2$) distance is used to quantify the notion of 'closest'.

3.3 Dynamic Implementation

In the dynamic implementation, the process with long burst time results small PTS, which causes many NCS. To avoid overhead resulted from more NCS, a threshold which is an implementation choice is determined. The weight of the lth cluster, CW_l, is calculated from Eq. (8):

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

where $Cavg_l$ is the average of burst times in the lth cluster. Eq. (9) calculates the time slice assigned to the lth cluster (CTS_l).

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

Each process $P_r; r = 1, 2, 3, \ldots, c$ executes for CTS_l. Awarding more time to the process that is close to its completion enables it to complete its execution and leave the queue, which in turn decreases the number of processes in the queue.

Unlike Previous works (e.g., Samih et al. [29]) which give short process in the current round an opportunity to run until termination under predefined condition, the proposed algorithm takes into account not only the current round, but also successive rounds. Depending on the process's BT, the proposed algorithm gives it more time in the current and successive rounds according to Eq. (10):

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

where $DTS_{r,l}$ is the dynamic time slice assigned to process P_r in cluster l. In successive rounds, RBT will be updated according to Eq. (11). The proposed algorithm is described in Fig. 2.

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

3.4 Illustrative Example

The following example illustrates the proposed technique. Tab. 2 contains the first dataset used in the experiments. The optimal number of clusters is indicated from the location of a knee in the curve in Fig. 3.

Cluster 1 contains the 7th and 8th processes. Cluster 0 contains the others (Tab. 3).

From Eq. (8), CW_0 equals 0.981702176 tu, and CW_1 equals 1.754574456 tu. From Eq. (9), CTS_0 equals 6.41227 tu, and CTS_1 equals 3.58773 tu. The 7th and 8th processes will be assigned 3.58773 tu, the others will be assigned 6.41227 tu.

4 Experimental Implementation

The specifications of the computer used in the experiments are: Intel core i5-2400 (3.10 GHz) processor, 1 TB HDD, 16 GB memory, Python 3.7.6, and Gnu/Linux Fedora 28 OS.

4.1 Benchmark Datasets

The performance of the compared algorithms was tested using nine synthetic datasets [29]. The burst times in each dataset are randomly generated. The number of process, BTs, $NCSs$, PBT, and $ATSs$ in each dataset differ from the other. Detailed information on datasets is presented in Tab. 4.

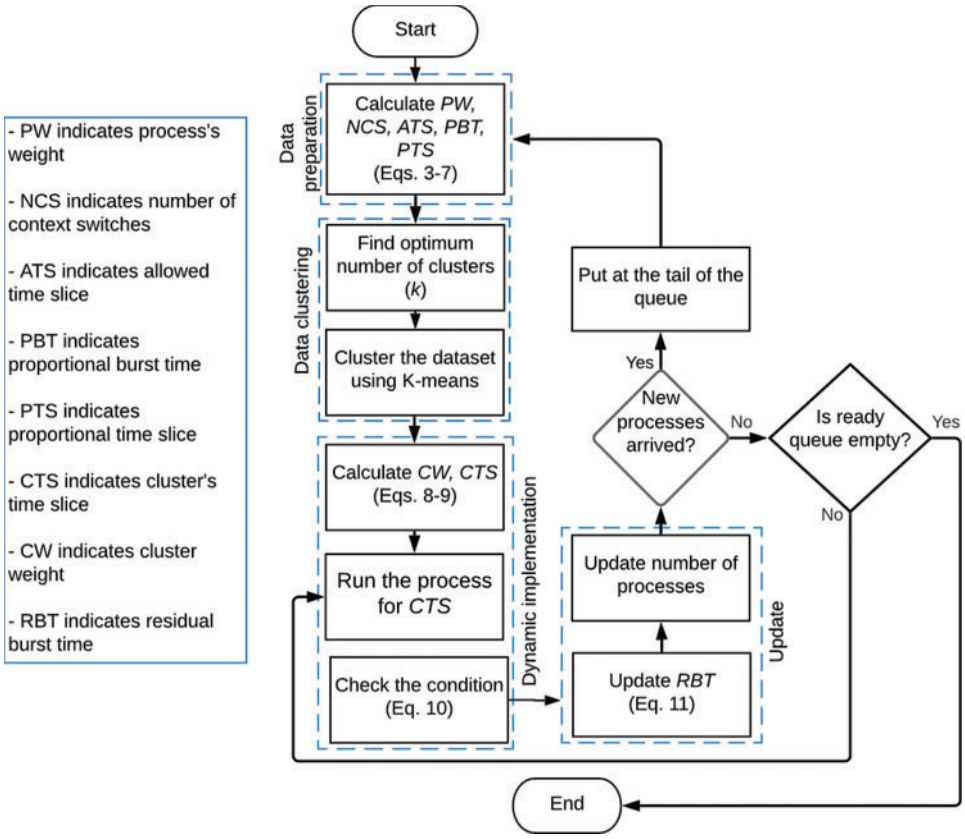

Figure 2: Algorithm flowchart

Table 2: Dataset_1

	BT	ATS	PBT	PW	NCS
0	109	10	−1.72043	0.077746	10
1	150	10	−6.12903	0.10699	14
2	3	3	9.677419	0.00214	0
3	50	10	4.623656	0.035663	4
4	49	10	4.731183	0.03495	4
5	49	10	4.731183	0.03495	4
6	409	10	−33.9785	0.291726	40
7	490	10	−42.6882	0.349501	48
8	47	10	4.946237	0.033524	4
9	46	10	5.053763	0.03281	4

4.2 Performance Evaluation

Six common algorithms; VTRR, SRR, ADRR PWRR, BRR and DRR were compared with the proposed algorithm on different nine collections of number of processes with different attributes. The experiments were performed in two cases; when the processes arrive at the same time, and when the processes arrive in different times.

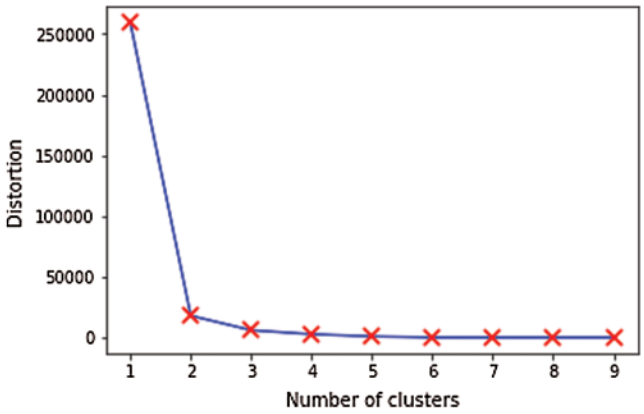

Figure 3: Optimal number of clusters ($k = 2$)

Table 3: Clustered Dataset_1

	BT	ATS	PBT	PW	NCS	y
0	109	10	−1.72043	0.077746	10	0
1	150	10	−6.12903	0.10699	14	0
2	3	3	9.677419	0.00214	0	0
3	50	10	4.623656	0.035663	4	0
4	49	10	4.731183	0.03495	4	0
5	49	10	4.731183	0.03495	4	0
6	409	10	−33.9785	0.291726	40	1
7	490	10	−42.6882	0.349501	48	1
8	47	10	4.946237	0.033524	4	0
9	46	10	5.053763	0.03281	4	0

Table 4: Datasets specifications. The first column presents the dataset ID, the second column presents the number of processes, and the third column presents the number of attributes (i.e., ATS, BT, PBT, PW, and NCS)

Dataset ID	#Processes	#Attributes
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

—In the first case, the time consumed in the clustering is trivial and can be ignored. Fig. 4 shows the average waiting times and turnaround times comparison. Fig. 5 shows the NCS comparison. Fig. 6 shows the improvement percentage of the proposed algorithm

over the compared algorithms. Tab. A1 shows a comparison of the time cost between the compared algorithms in terms of average waiting and turnaround times and NCS. Tab. A2 shows the improvement percentages of the proposed algorithm over the six scheduling algorithms.

—In the second case, new processes arrive to the queue; therefore, the clustering is repeated in every round. Fig. 7 shows the average waiting times and turnaround times comparison. Fig. 8 shows the NCS comparison. Fig. 9 shows the improvement percentage of the proposed algorithm over the six scheduling algorithms. Tab. B1 shows the running times comparison between the compared algorithms. Tab. B2 shows the average waiting time and turnaround time comparison between the compared algorithms. Tab. B3 shows the improvement percentages of the proposed algorithm over the six scheduling algorithms in terms of average waiting and turnaround times, and NCS.

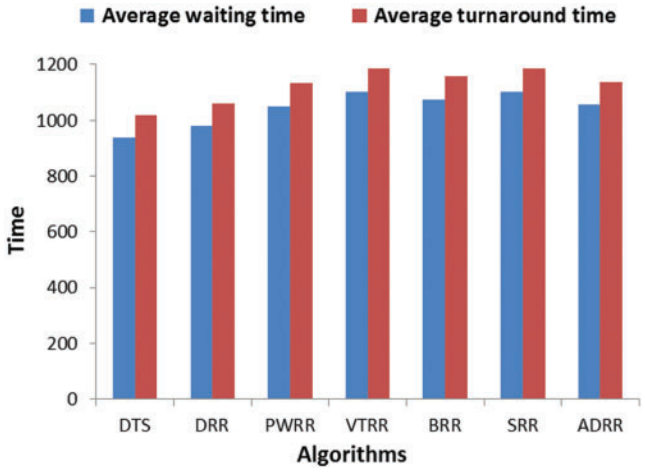

Figure 4: Comparing algorithms' time cost (case 1)

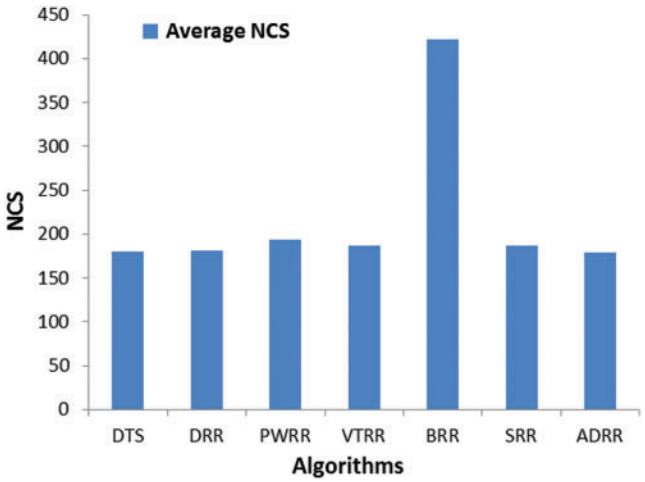

Figure 5: Comparing algorithms' NCS (case 1)

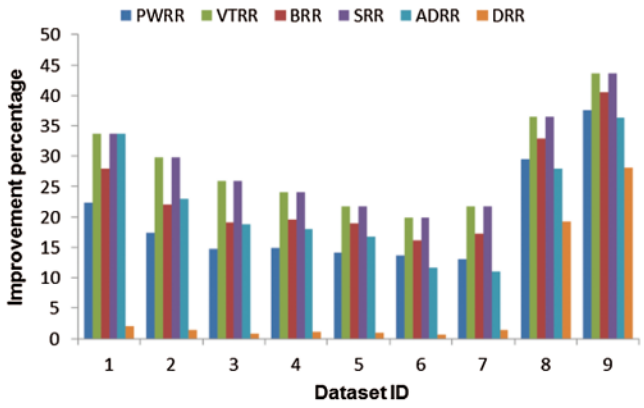

Figure 6: Improvement percentage of the proposed algorithm over the compared algorithms (case 1)

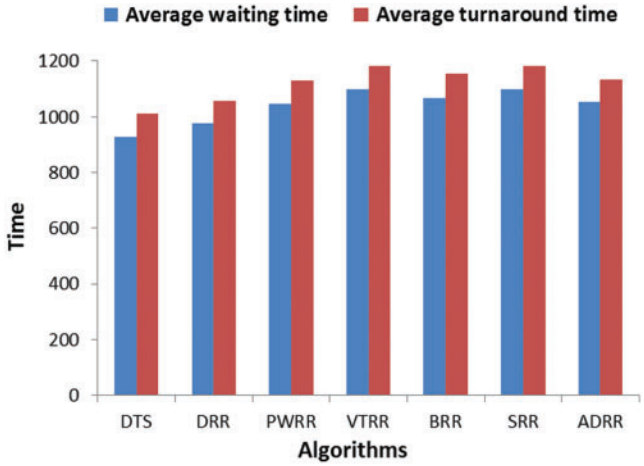

Figure 7: Comparing algorithms' time cost (case 2)

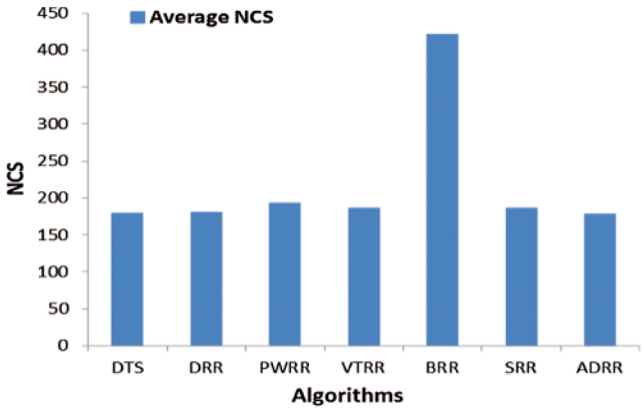

Figure 8: Comparing algorithms' NCS (case 2)

5 Conclusion

This paper introduced a dynamic version of SRR. The proposed algorithm reduces the scheduling time cost (i.e., waiting time and turnaround time). Unlike SRR which uses a fixed slice of time, the proposed algorithm assigns a time slice to a group of similar processes and

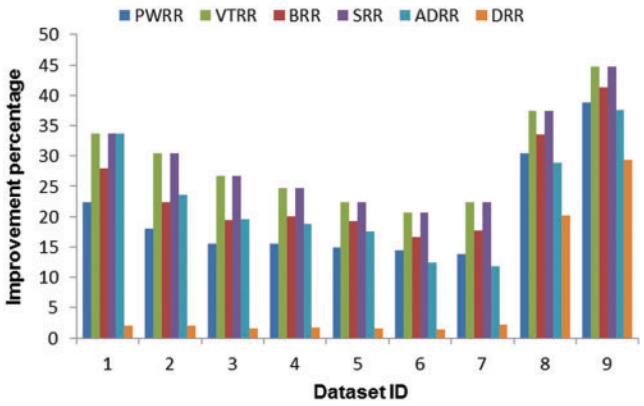

Figure 9: Improvement percentage of the proposed algorithm over the compared algorithms (case 2)

each process in this group runs for this time. The similarity between the processes in a group is determined using the clustering technique depending on the attributes of these processes. The most important attribute is the burst time that determines other attributes (i.e., number of allocations to CPU, weights, and the allowed time slice in a round). Clustering technique uses these attributes to cluster the processes. Every process in a cluster is assigned a time slice equal to the average of all allowed time slices in the group. In a round, some processes may complete their execution times and leave the queue; therefore, in the successive rounds, the number and burst times of survived processes will be updated. If all processes arrived at the same time, the clustering is applied once. On the other hand, if the processes arrive sequentially, clustering is applied in each round. The proposed algorithm endows the process that is close to complete with more time in the current round. In addition, the proposed algorithm gives a process more time in the current and successive rounds according to the condition in Eq. (10). The comparison was done between the proposed algorithm and six common algorithms from the point of view of waiting time, turnaround time, and NCS. The results showed that the proposed algorithm outperformed the compared algorithms.

Funding Statement: The author(s) received no specific funding for this study.

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," *Journal of Advanced Engineering Technologies*, 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 *2019 3rd Int. Conf. on Computing Methodologies and Communication (ICCMC)*, Erode, India, pp. 255–260, 2019.

[4] A. Silberschatz, P. B. Galvin and G. Gagne, *Operating System Concepts-10th*. John Wiley & Sons, Inc., 2018.

[5] J. Sunil, V. G. Anisha Gnana and V. T. Karthija, *Fundamentals of Operating Systems Concepts*. Germany, Saarbrucken: Lambert Academic Publications, 2018.

[6] C. McGuire and J. Lee, "The adaptive80 round robin scheduling algorithm," in *Transactions on Engineering Technologies*. Dordrecht, The Netherlands: Springer, pp. 243–258, 2015.
[7] T. Wilmshurst, *Designing Embedded Systems with Pic Microcontrollers*, 2nd ed., Oxford: Elsevier, 2010.
[8] P. Singh, A. Pandey and A. Mekonnen, "Varying response ratio priority: A preemptive CPU scheduling algorithm (VRRP)," *Journal of Computer and Communications*, vol. 3, no. 4, pp. 40–51, 2015.
[9] M. U. Farooq, A. Shakoor and A. B. Siddique, "An efficient dynamic round robin algorithm for CPU scheduling," in *2017 Int. Conf. on Communication, Computing and Digital Systems (C-CODE)*, Islamabad, Pakistan, pp. 244–248, 2017.
[10] A. A. Alsulami, Q. A. Al-Haija, M. I. Thanoon and Q. Mao, "Performance evaluation of dynamic round robin algorithms for CPU scheduling," in *2019 Southeast Conf.*, Huntsville, AL, USA, pp. 1–5, 2019.
[11] A. Singh, P. Goyal and S. Batra, "An optimized round robin scheduling algorithm for CPU scheduling," *International Journal on Computer Science and Engineering*, vol. 2, no. 7, pp. 2383–2385, 2010.
[12] 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.
[13] 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.
[14] 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 New Computer Architectures and their Applications*, vol. 5, pp. 64–71, 2010.
[15] 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.
[16] 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.
[17] A. Lengyel and Z. Botta-Dukát, "Silhouette width using generalized mean—A flexible method for assessing clustering efficiency," *Ecology and Evolution*, vol. 9, no. 23, pp. 13231–13243, 2019.
[18] A. Starczewski and A. Krzyżak, "Performance evaluation of the silhouette index bt: Artificial intelligence and soft computing," in *Artificial Intelligence and Soft Computing, Lecture Notes in Computer Science*, vol. 9120. Cham: Springer, pp. 49–58, 2015.
[19] D. Xu and Y. Tian, "A comprehensive survey of clustering algorithms," *Annals of Data Science*, vol. 2, no. 2, pp. 165–193, 2015.
[20] J. Wu, *Cluster Analysis and K-means Clustering: An Introduction, Advances in K-means Clustering*. Berlin Heidelberg: Springer, pp. 1–16, 2012.
[21] Y. Liu, Z. Li, H. Xiong, X. Gao and J. Wu, "Understanding of Internal Clustering Validation Measures," in *2010 IEEE Int. Conf. on Data Mining, ICDM*, Sydney, NSW, Australia, pp. 911–916, 2010.
[22] A. Harwood and H. Shen, "Using fundamental electrical theory for varying time quantum uniprocessor scheduling," *Journal of Systems Architecture*, vol. 47, no. 2, pp. 181–192, 2001.
[23] T. Helmy, "Burst round robin as a proportional-share scheduling algorithm," in *Proc. of the 4th IEEE-GCC Conf. on Towards Techno-Industrial Innovations*, Bahrain, pp. 424–428, 2007.
[24] 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.
[25] S. Zouaoui, L. Boussaid and A. Mtibaa, "Improved time quantum length estimation for round robin scheduling algorithm using neural network," *Indonesian Journal of Electrical Engineering and Informatics*, vol. 7, no. 2, pp. 190–202, 2019.
[26] A. Pandey, P. Singh, N. H. Gebreegziabher and A. Kemal, "Chronically evaluated highest instantaneous priority next: A novel algorithm for processor scheduling," *Journal of Computer and Communications*, vol. 4, no. 4, pp. 146–159, 2016.

[27] N. Srinivasu, A. S. V. Balakrishna and R. D. Lakshmi, "An augmented dynamic round robin CPU," *Journal of Theoretical and Applied Information Technology*, vol. 76, no. 1, pp. 118–126, 2015.

[28] 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.

[29] 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. 1–4, 2020.

Appendix A

Table A1: Average waiting time and turnaround time comparison between the proposed algorithm and six scheduling algorithms (case 1)

Dataset	DTS			DRR			PWRR			VTRR			BRR			SRR			ADRR		
	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS
1	393.8	534	122	398.41	538.61	124	450.85	591.05	141	485.90	626.10	124	467.60	607.80	306	485.90	626.10	124	485.9	626.1	124
2	548.289	656.356	141	552.37	660.44	140	605.22	713.28	154	653.27	761.33	144	622.20	730.27	341	653.27	761.33	144	625.933	734	139
3	701.157	792.257	156	704.30	795.40	156	760.63	851.73	169	812.35	903.45	160	779.50	870.60	371	812.35	903.45	160	778.3	869.4	154
4	847.421	927.941	169	852.07	932.59	171	918.54	999.06	182	968.24	1048.76	175	943.52	1024.04	401	968.24	1048.76	175	935.36	1015.88	169
5	999.808	1073.08	183	1004.37	1077.63	185	1078.26	1151.53	195	1125.67	1198.93	190	1107.80	1181.07	430	1125.67	1198.93	190	1094.77	1168.03	184
6	1150.1	1217.87	199	1154.23	1222.00	199	1236.42	1304.20	210	1280.57	1348.34	205	1254.31	1322.09	455	1280.57	1348.34	205	1222.97	1290.74	194
7	1265.88	1329.16	207	1275.34	1338.61	210	1356.38	1419.65	220	1423.53	1486.80	219	1388.58	1451.85	479	1423.53	1486.80	219	1341.85	1405.13	204
8	1247.56	1307.07	219	1383.72	1443.23	221	1468.75	1528.26	230	1533.09	1592.60	229	1498.91	1558.42	499	1533.09	1592.60	229	1454.76	1514.27	214
9	1274.58	1330.88	226	1487.44	1543.74	228	1575.99	1632.29	238	1637.48	1693.78	239	1605.92	1662.22	519	1637.48	1693.78	239	1563.62	1619.92	224
Average	**936.51**	**1018.73**		**979.14**	**1061.36**		**1050.11**	**1132.34**		**1102.23**	**1184.45**		**1074.26**	**1156.48**		**1102.23**	**1184.45**		**1055.94**	**1138.16**	
Improvement%				**4.35**	**11.76**		**10.82**	**17.29**		**15.04**	**20.93**		**12.82**	**19.02**		**15.04**	**20.93**		**11.31**	**17.72**	

Table A2: Improvement percentages of the proposed algorithm over six scheduling algorithms (case 1)

	DRR			PWRR			VTRR			BRR			SRR			ADRR		
	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS
1	1.16	0.86	1.61	12.65	9.65	13.48	18.95	14.71	1.61	15.78	12.14	60.13	18.95	14.71	1.61	18.95	14.71	1.61
2	0.74	0.62	−0.71	9.41	7.98	8.44	16.07	13.79	2.08	11.88	10.12	58.65	16.07	13.79	2.08	12.40	10.58	−1.44
3	0.45	0.39	0.00	7.82	6.98	7.69	13.69	12.31	2.50	10.05	9.00	57.95	13.69	12.31	2.50	9.91	8.87	−1.30
4	0.55	0.50	1.17	7.74	7.12	7.14	12.48	11.52	3.43	10.19	9.38	57.86	12.48	11.52	3.43	9.40	8.66	0.00
5	0.45	0.42	1.08	7.28	6.81	6.15	11.18	10.50	3.68	9.75	9.14	57.44	11.18	10.50	3.68	8.67	8.13	0.54
6	0.36	0.34	0.00	6.98	6.62	5.24	10.19	9.68	2.93	8.31	7.88	56.26	10.19	9.68	2.93	5.96	5.65	−2.58
7	0.74	0.71	1.43	6.67	6.37	5.91	11.07	10.60	5.48	8.84	8.45	56.78	11.07	10.60	5.48	5.66	5.41	−1.47
8	9.84	9.43	0.90	15.06	14.47	4.78	18.62	17.93	4.37	16.77	16.13	56.11	18.62	17.93	4.37	14.24	13.68	−2.34
9	14.31	13.79	0.88	19.13	18.47	5.04	22.16	21.43	5.44	20.63	19.93	56.45	22.16	21.43	5.44	18.49	17.84	−0.89

Appendix B

Table B1: Running times comparison between the proposed algorithm and six scheduling algorithms (case 2)

Dataset	DTS	DRR	PWRR	VTRR	BRR	SRR	ADRR
1	0.037	0.027	0.011	0.01	0.017	0.008	0.01
2	0.041	0.03	0.012	0.01	0.018	0.008	0.012
3	0.047	0.042	0.013	0.012	0.02	0.009	0.013
4	0.049	0.044	0.02	0.021	0.023	0.012	0.02
5	0.05	0.046	0.021	0.023	0.026	0.014	0.021
6	0.053	0.051	0.022	0.025	0.029	0.017	0.023
7	0.056	0.052	0.028	0.027	0.035	0.02	0.026
8	0.07	0.063	0.033	0.03	0.039	0.023	0.03
9	0.1	0.072	0.04	0.033	0.04	0.026	0.032

Table B2: Average waiting time and turnaround time comparison between the proposed algorithm and six scheduling algorithms (case 2)

Dataset	DTS			DRR			PWRR			VTRR			BRR			SRR			ADRR		
	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS
1	393.8	534	122	398.41	538.61	124	450.85	591.05	141	485.90	626.10	124	467.60	607.80	306	485.90	626.10	124	485.9	626.1	124
2	544.489	652.556	141	550.47	658.54	140	603.32	711.38	154	651.37	759.43	144	618.40	728.37	341	651.37	759.43	144	624.033	732.1	139
3	695.757	786.857	156	701.60	792.70	156	757.93	849.03	169	809.65	900.75	160	774.10	867.90	371	809.65	900.75	160	775.6	866.7	154
4	841.221	921.741	169	848.97	929.49	171	915.44	995.96	182	965.14	1045.66	175	937.32	1020.94	401	965.14	1045.66	175	932.26	1012.78	169
5	992.008	1065.28	183	1000.47	1073.73	185	1074.36	1147.63	195	1121.77	1195.03	190	1100.00	1177.17	430	1121.77	1195.03	190	1090.87	1164.13	184
6	1140.9	1208.67	199	1149.63	1217.40	199	1231.82	1299.60	210	1275.97	1343.74	205	1245.11	1317.49	455	1275.97	1343.74	205	1218.37	1286.14	194
7	1256.08	1319.36	207	1270.44	1333.71	210	1351.48	1414.75	220	1418.63	1481.90	219	1378.78	1446.95	479	1418.63	1481.90	219	1336.95	1400.23	204
8	1235.56	1295.07	219	1377.72	1437.23	221	1462.75	1522.26	230	1527.09	1586.60	229	1486.91	1552.42	499	1527.09	1586.60	229	1448.76	1508.27	214
9	1258.98	1315.28	226	1479.64	1535.94	228	1568.19	1624.49	238	1629.68	1685.98	239	1590.32	1654.42	519	1629.68	1685.98	239	1555.82	1612.12	224
Average	**928.76**	**1010.98**		**975.26**	**1057.48**		**1046.24**	**1128.46**		**1098.36**	**1180.58**		**1066.50**	**1152.61**		**1098.36**	**1180.58**		**1052.06**	**1134.29**	
Improvement%				**4.77**	**12.17**		**11.23**	**17.70**		**15.44**	**21.33**		**12.92**	**19.42**		**15.44**	**21.33**		**11.72**	**18.12**	

Table B3: Improvement percentages of the proposed algorithm over six scheduling algorithms (case 2)

	DRR			PWRR			VTRR			BRR			SRR			ADRR		
	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS	WT	TT	NCS
1	1.16	0.86	1.61	12.65	9.65	13.48	18.95	14.71	1.61	15.78	12.14	60.13	18.95	14.71	1.61	18.95	14.71	1.61
2	1.09	0.91	-0.71	9.75	8.27	8.44	16.41	14.07	2.08	11.95	10.41	58.65	16.41	14.07	2.08	12.75	10.87	-1.44
3	0.83	0.74	0.00	8.20	7.32	7.69	14.07	12.64	2.50	10.12	9.34	57.95	14.07	12.64	2.50	10.29	9.21	-1.30
4	0.91	0.83	1.17	8.11	7.45	7.14	12.84	11.85	3.43	10.25	9.72	57.86	12.84	11.85	3.43	9.77	8.99	0.00
5	0.85	0.79	1.08	7.67	7.18	6.15	11.57	10.86	3.68	9.82	9.50	57.44	11.57	10.86	3.68	9.06	8.49	0.54
6	0.76	0.72	0.00	7.38	7.00	5.24	10.59	10.05	2.93	8.37	8.26	56.26	10.59	10.05	2.93	6.36	6.02	-2.58
7	1.13	1.08	1.43	7.06	6.74	5.91	11.46	10.97	5.48	8.90	8.82	56.78	11.46	10.97	5.48	6.05	5.78	-1.47
8	10.32	9.89	0.90	15.53	14.92	4.78	19.09	18.37	4.37	16.90	16.58	56.11	19.09	18.37	4.37	14.72	14.14	-2.34
9	14.91	14.37	0.88	19.72	19.03	5.04	22.75	21.99	5.44	20.83	20.50	56.45	22.75	21.99	5.44	19.08	18.41	-0.89