A new enhanced IGBTQ-based model for CPU scheduling

Ra'ed M. Al-Khatib[1], Asef Al-Khateeb[2], Esraa Al-Daom[1], Ishraq T. Al-Dagamseh[1], Ayat Tawalbeh[1], and Laith Abualigah[3,4,5,6]

[1]Department of Computer Sciences, Yarmouk University, Irbid 21163, Jordan.
raed.m.alkhatib@yu.edu.jo, esraaabd200@gmail.com, ishraq500@gmail.com, ayattawalbeh217@gmail.com
[2]MIS department, College of Business Administration, King Faisal University, Al Ahsa 31982, Saudi Arabia, amaalkhateeb@kfu.edu.sa
[3]Computer Science Department, Prince Hussein Bin Abdullah Faculty for Information Technology, Al al-Bayt University, Mafraq 25113, Jordan.
[4]MEU Research Unit, Middle East University, Amman, Jordan.
[5]Hourani Center for Applied Scientific Research, Al-Ahliyya Amman University, Amman 19328, Jordan.

[6]aligah.2020@gmail.com

Abstract. CPU scheduling is concerned with how to efficiently exploit processors by executing the largest number of operations in the shortest possible time. The efficiency of CPU scheduling depends on the algorithm used. Round Robin (RR) is one of the most popular algorithms, which is primarily based on determining the period of execution time for the operation. In this work, we develop a new enhancement algorithm for modifying the RR algorithm based on IGBTQ model to solve the CPU scheduling problem. Therefore, the main objective is to improve the Average waiting Time (AWT) and the Average Turnaround Time (ATT). The final results are successful when reached the lowest scores of AWT and ATT measurements. The experiments used two cases of a different number of processes, which obtained better outputs compared to other state-of-the-art models like; the dynamic RR with Control Preemption (DRRCP), Group Based Time Quantum (GBTQ), and IGBTQ algorithm.

Keywords: artificial Intelligence (AI), CPU scheduling, round robin, average waiting time, average turnaround time.

1. Introduction

The operating system is the central part of a computer system, which performs several tasks for running a client application on the system. The CPU is one of the most important computer resources. It is also one of the basic functions of an operating system. The CPU process scheduling is one of the most essential operations because it properly manages the processor. However, the task of CPU scheduling is a difficult process, because it is very time consuming and needs to be modified and checked to ensure that it is working properly. A single processor system is one that allows only one process to run at a time. If there is another process, it must wait until the CPU becomes available. Therefore, the concept of CPU scheduling is introduced in order to run more than one process at the same time or at any other time. CPU scheduling can be categorized into three stages [1]. i) Long-term scheduling, ii) Mid-term scheduling, iii) Short term scheduling, as shown in Figure 1. These three stages are well known as the three main schedulers.

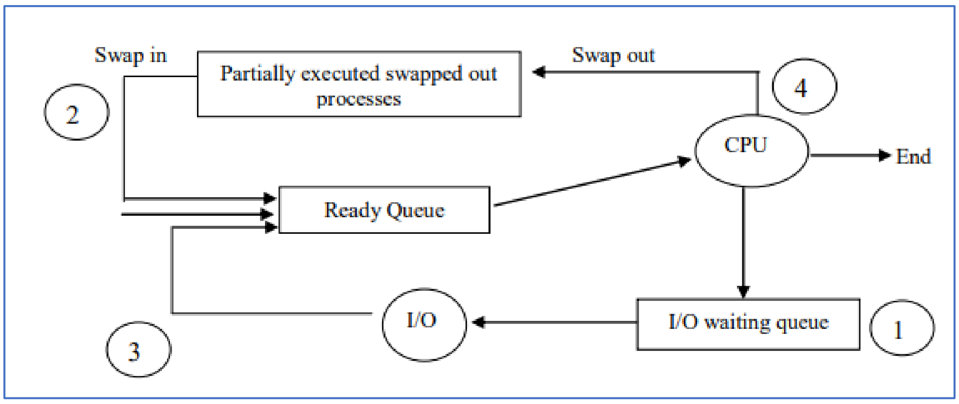


Figure 1. Stages of CPU processes scheduling [1].

To distinguish between these three types of schedulers, definitions are given as follows. Long-term scheduling, where it always decides which operations to perform at the moment, such as which ones are accepted depending on whether they are currently running, deferred, or authorized. Mid-term scheduling is any operations that are transferred from main memory to secondary memory, and this conversion process is called swap. Short term scheduling is the most successful process over the other two types because it determines whether the operation is ready to be executed directly from the queue without the need for delay [1].

Therefore, the short-term scheduler is responsible for deciding which process will enter the CPU to be executed, and which process(s) will keep waiting. Several algorithms have been used to manage short term scheduler [2]. The main core algorithms that have been presented in literature are: i) First Come First Service (FCFS), which determines the order of entry of processes to the CPU according to the order of entry into the system. ii) Shortest Job First (SJF) that enters the shortest process first or orders the process based on the length of the processes in ascending order. iii) Round Robin (RR) which specifies a pre-quantum time to assign a specific time to the execution of the process executed in the CPU. After the quantum expiration, the RR algorithm will exit this executing process out of CPU, even if it has not completed its work to allow and enter another process to be executed in CPU. iv) Priority scheduling algorithm that enters processes into the CPU based on their priority [2].

This paper mainly focuses on RR algorithm designed explicitly for multi-program operating systems [3]. RR is one of the widely used scheduling algorithms, which is considered to be the most efficient and effective CPU scheduling. We should show that quantum time greatly affects the presentation and implementation of RR algorithm [4]. While executing a process, choosing how much time quantum is critical because they decide and control the time and resources to complete the processes.

There are different types of methods that can be accessed to determine the quantum contact time with RR algorithm. Quantum time can be specified dynamically or statically. This RR algorithm suffers from high Average Waiting Time (AWT) and Average Turnaround Time (ATT). Waiting time is defined as a process time hold that can be differentiated according to the amount of time spent in the ready queue to partition (allocate) the CPU. Response time is the amount of time that takes from its stay to a processed line until it is completed.

RR algorithm is one of the most useful scheduling algorithm. RR is proved to be efficient, especially in terms of response speed. We know that everything has its positive and negative sides, and the positive side has already been explained. We will now explain the downside of this algorithm. Specifies a fixed time limit for operations to execute, resulting in increasing the waiting time and increased context switching, especially for large operations [5]. Other modifications will be described in more detail in the next section.

The RR algorithm was recently adapted to improve CPU scheduler operation. There are many improvements to the RR algorithm, which are the most notable being in the state-of-the-art technology [5][6]. Therefore, we adopted the RR algorithm to improve the proposed method by enhancing the previous IGBQT model [3], which stands for Improved Group Based Time Quantum algorithm for CPU scheduling. So, we develop a new operator to create groups of processes by adapting 80\% of the entire process to be used to determine the final number of groups. This upgraded launcher enhances the final results obtained by reducing the AWT and ATT schedulers of the CPU when two instances of burst time are applied to different processes that were used in [3].

The rest of this paper is organized as follows. Related works are presented in Section 2. The proposed methodology is introduced in Section 3. Section 4 discusses the main cases with experimental results. Finally, the conclusion is in Section 5 with some relevant future directions.

2. Related works

Several studies have been emerged as an improvement to the performance of the traditional RR algorithm. This section summarizes the related works according to the quantum time selection mechanism. It will be divided into two parts: the first type is called improvements using unstable quantum time, and the second type is collectively called the quantum time type.

2.1. Improvements using unstable Quantum Time

In this section, we focus on Simon's study [6], which is the most important study that improved RR algorithm by making quantum time variable based on some considerations. It is intended to reduce the number of context switching, as well as to enhance the values of AWT, ATT and NCS. At the point of protection, the proposed method is based on a guaranteed 95% compute time burst time. Then, the results are compared with quantum time (QT); and if the 95% percentile of process burst time is less than the QT, it should be run before the greater percentile.

Otherwise, the authors in [4] presented an algorithm that was designed based on the average rate of processes. It considers the largest variance by comparing adjacent processes in the ready queue. This algorithm was introduced with the goal of extending scheduling implementation in the range of maximum throughput, most extreme CPU usage, reduced Turnaround Time (TT), reduced Average Waiting Time (AWT) and Context Switching (CS).

In [7], the authors introduced the featured QT for process grouping. It reduced Context Switches (CS) by upgrading the implementation of RR algorithm. Their test showed that the implementation of the Group Based Quantum Time (GBQT) for RR algorithm is better when compared to the current RR algorithm for Average Turnaround Time (ATT), Average Waiting Time (AWT) and context switches. In this study [7], the GBQT algorithm divided the operations into groups by 0.25%, and for each group of varied QT.

2.2. Improvements with a combined type of Quantum Time

These types of work are based on the original RR algorithm. They generally applied the fixed quantum time but under some circumstances they changed these values to a dynamic value. One such work was presented in [5], which they first applied FCFS for ordering the operations, and then they used two types to QT: i) QT is constant until the condition satisfies the difference between burst time, and ii) QT is less than or equal to the product of the threshold and the quantum time. Therefore, the value of the QT will be equal to the burst time. Compared to the original RR algorithm, this algorithm proved to be effective, as it reduced context switching and waiting time.

In [4], the authors presented another method based on RR with how much time is skilled. This amount of quantum time was determined by considering the maximum difference between subsequent and contiguous processes in the ready queue. Burst time of all processes was sorted in ascending order in the ready queue using this method. Then, the time quantum was calculated by adding the time of the middle burst and the maximum difference. The strategy of this method focused on reducing turnaround time, reducing waiting time, and increasing productivity.

3. Proposed method

Our proposed method for solving the CPU scheduling problem is fully explained in this section. The main algorithm of our proposed method is a new enhancement for the improved group based quantum time, which is called (AE-IGBQT) method. It is a new improvement from [3], in order to reduce the number of groups. On other words, it dynamically limits the number of groups in this new version of developed algorithm based on IGBQT algorithm [3]. The evaluation model for our new proposed AE-IGBQT method uses two cases, which contain the burst time for processes.

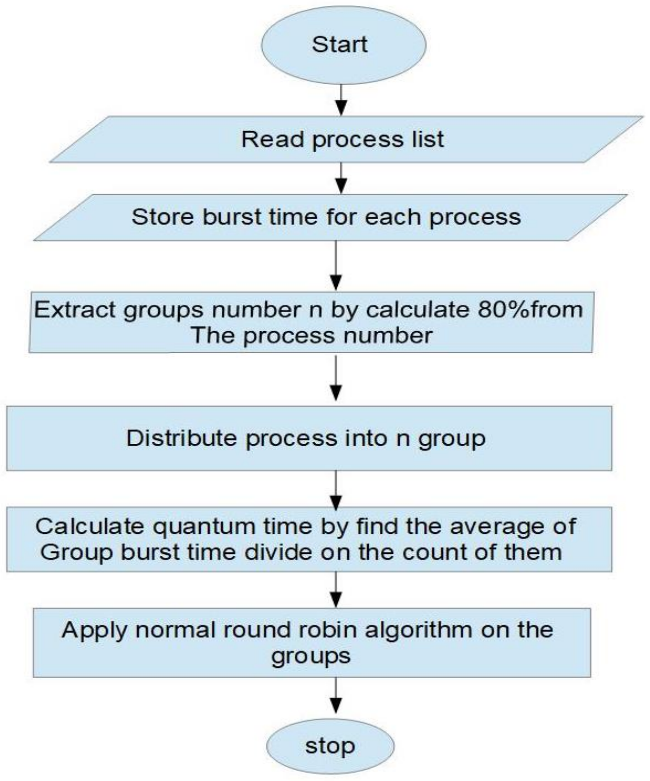

Start
Read process list
Store burst time for each process
Extract groups number n by calculate 80%from The process number
Distribute process into n group
Calculate quantum time by find the average of Group burst time divide on the count of them
Apply normal round robin algorithm on the groups
stop

Figure 2. Flowchart of our proposed AE-IGBQT method.

When the program of our proposed algorithm starts, it reads a list of the burst time of processes and then stores the burst time in the queue. Then, it extracts the group number (*n*) by calculating 80% of the process number. Then, it distributes the process into group (*n*). From now on, it calculates the quantum time by finding the mean value. Thus, it will apply RR algorithm, and finally it will calculate the AWT and ATT values. Figure 1 illustrates the flowchart for the main steps of our proposed algorithm.

4. Results and discussions

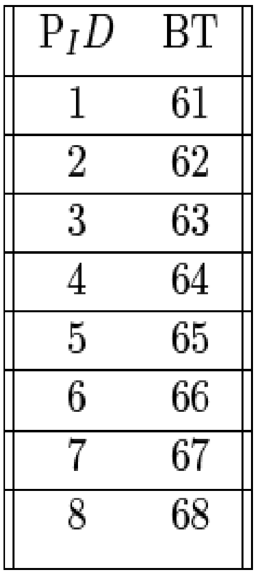
This section will show the results of applying the new proposed AE-IGBQT algorithm to eight processes in which will have determined their burst time from the previous version in the work [3]. Next, we will compare the obtained results with the results from previous works like original Round Robin, DRRCP [6], GBTQ [7], and IGBTQ [3] algorithms.

4.1. Case 1

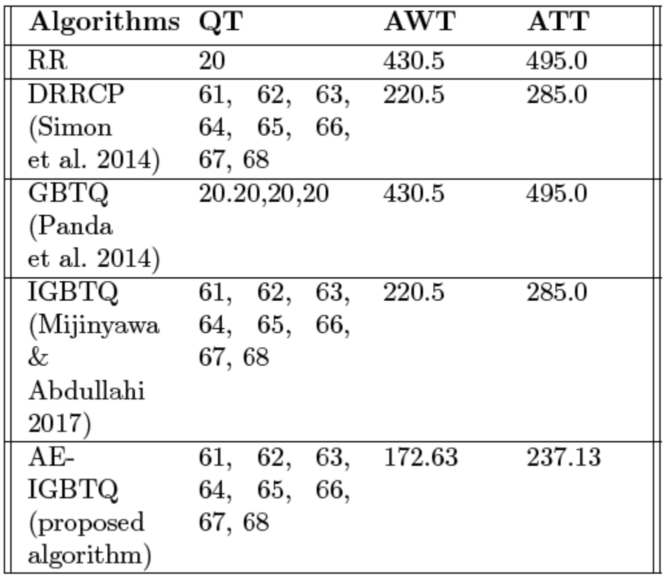
In this case, we adapt numbers for the burst time of eight processes that have previously been used in IGBQT algorithm [3]. Table 1 explains the details of used data in Case 1.

By applying Case 1 to our proposed algorithm. it achieved 172.625 for AWT, and 237.125 for ATT. Summarizes the results of our proposed algorithm work, comparing them with other results obtained from original RR, DRRCP [6], GBTQ [7], and IGBTQ [3] respectively. Finally, our proposed AE-IGBQT algorithm enhanced the results in this case compared to results of previous algorithms, as shown in Table 1.

Figure 3 explains the results from Case1, and show that the results of our proposed algorithm are the best in terms on the AWT and ATT scales.

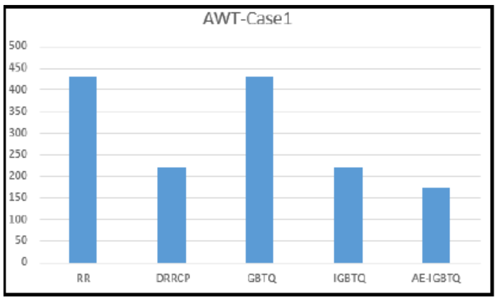


P_ID	BT
1	61
2	62
3	63
4	64
5	65
6	66
7	67
8	68

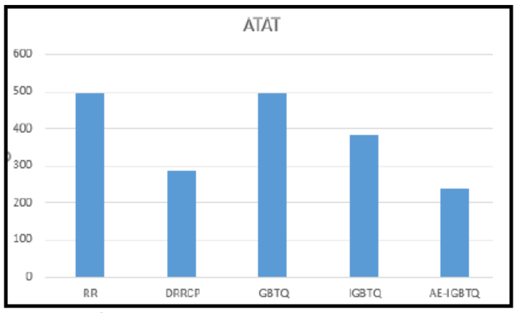


Algorithms	QT	AWT	ATT
RR	20	430.5	495.0
DRRCP (Simon et al. 2014)	61, 62, 63, 64, 65, 66, 67, 68	220.5	285.0
GBTQ (Panda et al. 2014)	20.20,20,20	430.5	495.0
IGBTQ (Mijinyawa & Abdullahi 2017)	61, 62, 63, 64, 65, 66, 67, 68	220.5	285.0
AE-IGBTQ (proposed algorithm)	61, 62, 63, 64, 65, 66, 67, 68	172.63	237.13

Figure 3. Data of Case 1 & Summary results of Case 1.



AWT comparison Results in Case1

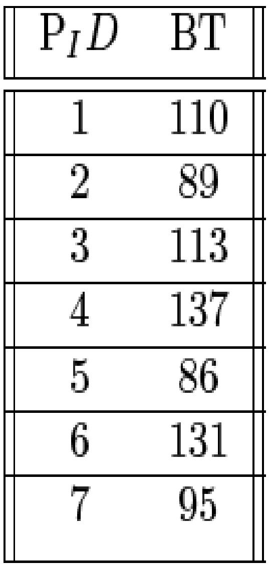


ATT comparison Results in Case1

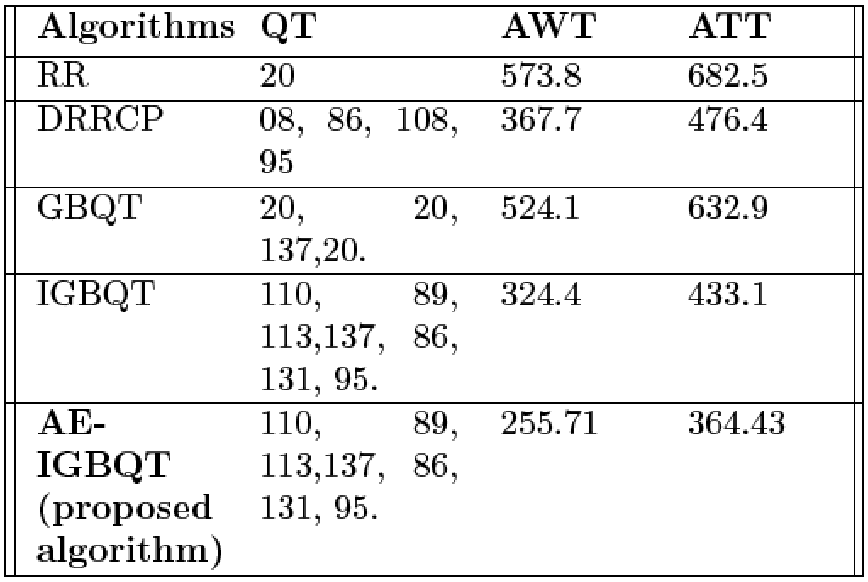
Figure 4. AWT & ATT results from Case 1.

4.2. Case 2

Seven processes are used in Case 2. In addition, these seven processes were used in previous methods like IGBQT paper [3].



P_ID	BT
1	110
2	89
3	113
4	137
5	86
6	131
7	95

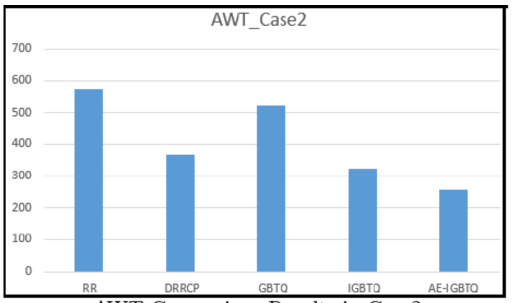


Algorithms	QT	AWT	ATT
RR	20	573.8	682.5
DRRCP	08, 86, 108, 95	367.7	476.4
GBQT	20, 20, 137,20.	524.1	632.9
IGBQT	110, 89, 113,137, 86, 131, 95.	324.4	433.1
AE-IGBQT (proposed algorithm)	110, 89, 113,137, 86, 131, 95.	255.71	364.43

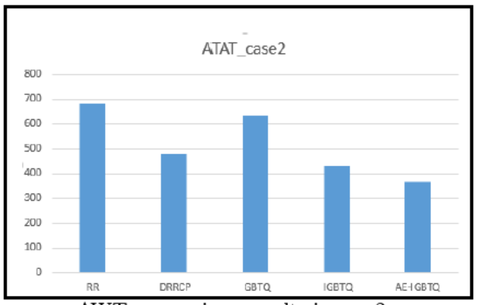
Figure 5. Data of Case 2 & Summary results of Case 2.

As shown in Table 2 from Case 2, our proposed AE-IGBQT algorithm enhanced the results in this case compared to all previous algorithms that were mentioned in Table 2. Moreover, when applied Case 2 on proposed algorithm, it achieved 255.71 for AWT, and 364.428 for ATT. Table 2 is also summarized the obtained results of proposed algorithm compared with results from other algorithms.

From the charts in Figure 5, it can be concluded that the least amount of AWT and ATT was obtained from our proposed AE-IGBQT modified algorithm when applying case2 processes. So, it reached the target with the best results and outperformed other state-of-the-art algorithms in literature; original RR, DRRCP [6], GBTQ [7], and IGBTQ [3].



AWT Comparison Results in Case2



AWT comparison results in case2

Figure 6. AWT & ATT results from Case 2.

5. Conclusion and future work

In this paper, we developed a new enhancement algorithm called AE-IGBQT, for modifying the Round Robin algorithm based on IGBTQ model to solve the CPU process scheduling problem. The main objective of this work is to improve and reduce the final Average waiting Time (AWT), and the Average Turnaround Time (ATT). The results obtained from this work are significantly outperformed the results previous modifications, and especially on the original RR, DRRCP, GBTQ, and IGBQT algorithms. Therefore, our new developed AE-IGBQT algorithm reduced the measurements of AWT and ATT by means of dynamic determination of the number of groups in which processes are distributed, as well as quantum time (QT). We look forward to making improvements to other CPU scheduling algorithms in the future works.

References

[1] S. Zouaoui, L. Boussaid, and A. Mtibaa, “Priority based round robin (PBRR) CPU scheduling algorithm.,” *International Journal of Electrical \& Computer Engineering (2088-8708)*, vol. 9, no. 1, 2019.

[2] S. M. Ali, R. F. Alshahrani, A. H. Hadadi, T. A. Alghamdi, F. H. Almuhsin, and E. E. El-Sharawy, “A Review on the CPU Scheduling Algorithms: Comparative Study,” *IJCSNS*, vol. 21, no. 1, p. 19, 2021.

[3] H. M. Mijinyawa and S. E. Abdullahi, “Improved Group Based Time Quantum (IGBTQ) CPU scheduling algorithm,” in *2017 13th International Conference on Electronics, Computer and Computation (ICECCO)*, 2017, pp. 1–6.

[4] D. Biswas and M. Samsuddoha, “Determining Proficient Time Quantum to Improve the Performance of Round Robin Scheduling Algorithm.,” *International Journal of Modern Education \& Computer Science*, vol. 11, no. 10, 2019.

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

[6] A. Simon, S. Abdullahi, and S. Junaidu, “Dynamic Round Robin with Controlled Preemption (DRRCP),” *International Journal of Computer Science Issues (IJCSI)*, vol. 11, no. 3, p. 109, 2014.

[7] S. K. Panda, D. Dash, and J. K. Rout, “A group based time quantum round robin algorithm using min-max spread measure,” *arXiv preprint arXiv:1403.0335*, 2014.