An Efficient Round Robin Task Scheduling Algorithm Based on a Dynamic Quantum Time

Chunhong Zhang, Ping Luo, Yuye Zhao and Jianqiang Ren

Abstract**—**In the round robin (RR) algorithm, the length of the static quantum time is difficult to determine, too long or too short is not appropriate. If the time slice is too long, the RR algorithm is degraded to First Come First Served (FCFS) algorithm. Each process is executed in one time slice, resulting in a long response time. If the time slice is too short, a user's request will need more time slices to process. The number of context switches will increase and the response time will be longer. There are still some shortcomings in the existing algorithms of dynamic round robin. Therefore, a dynamic round robin scheduling algorithm based on median is proposed, which is called the Median-based Dynamic Round Robin (MDRR) algorithm. The algorithm treats scheduling tasks in ascending order according to the size of their burst time. Then the burst time of the next task that is adjacent to the median is selected as the quantum time for each round scheduling. Each round of scheduling needs to calculate a time slice, instead of calculating the time slice for each task, so the algorithm complexity is low. The simulation experiment showed that the MDRR algorithm can maintain good performance in many cases. It has a good balance between the scheduling overhead, system waiting time, system performance and fairness. The MDRR has better performance than other improved round robin algorithms**.**

Keywords**—**task scheduling, round robin (RR) algorithm, burst time, dynamic quantum time.

I. INTRODUCTION

AS modern operating systems allow multiple tasks to execute concurrently, the multiple tasks simultaneously compete with the CPU. When assigning the CPU to tasks, system require try to achieve the minimum turnaround time, response time and context switching times, and to ensure the fairness of all tasks. The round robin algorithm is widely used as one of the most classical fairness scheduling algorithms. In practical applications, many systems require multiple tasks to run at the same time. For example, in the application of detection of dangerous signals, multiple tasks simultaneously detect dangerous signals, and these multiple tasks should be run at the same time. We can solve the problem with the round robin algorithm.

This work was supported by the Scientific Research and Originality Team of Computational Intelligence and Intelligence System in Langfang Normal University (B-04), the Natural Science Foundation of Langfang Normal University (LSLQ201409), the Natural Science Foundation for Doctors of Langfang Normal University (Grant No.LSLB201409).

Chunhong Zhang is with the College of Mathematics and Information, Langfang Normal University, Langfang 065000, Hebei, China (corresponding author; e-mail: zch0913@126.com).

Ping Luo, Yuye Zhao and Jianqiang Ren are with the College of Mathematics and Information, Langfang Normal University, Langfang 065000, Hebei, China.

In the round robin scheduling algorithm, the size of the quantum time has a great impact on system performance. However, the size of the quantum time is not easy to set. If the quantum time is set too big, the round robin algorithm degenerates into the First Come First Served (FCFS) algorithm. If the quantum time is set too small, it will cause too many times of context switching. Thus it will extend the waiting time for short tasks, resulting in increased system overhead. In recent years, a lot of research on dynamic quantum time has been carried out by many scholars, and they also proposed some improved algorithms [1]-[7]. For example, Hu Sai et al. proposed the performance models of the round robin algorithm and the dynamic round robin algorithm are built by using the theory of birth-death process [8]. But the premise of the model is that the number of all the tasks in the waiting queue must obey the poisson distribution of the parameter "λ", so it does not have universal adaptability. The DQRRR algorithm calculates the quantum time based on the average of two highest burst times and average of two lowest arrival times [9]. The result only reduces the number of context switches, the average waiting time and average turnaround time increased. The DABRR algorithm uses the average of the burst time of all tasks as the quantum time, the algorithm is simple, but the execution effect is not ideal [10]. By setting two sub-queues, Q1 for tasks less than median, Q2 for tasks larger than median. the SRDQ algorithm reduced the response time of all the tasks and solved the problem of task starvation [11]. But compared with the same kind of algorithm, the turnaround time and the waiting time of the tasks increased. Moreover, each task needs to calculate time slice before scheduling. The algorithm is complex and system overhead is large. The SARR algorithm calculates the median of all tasks as the quantum time of each round, and improved the performance of the round robin algorithm [10]. Other improved RR algorithms [1]-[7], [12], [13], some are too complex, some have large system overhead, some are poor in fairness, and some are easy to lead to the starvation of long tasks. These cannot be applied in the practical application. Short Job Priority (SJF) scheduling algorithm has good performance, but it can easily lead to the starvation of long tasks and not timely response to urgent tasks. The FCFS algorithm is simple, but the performance is not ideal.

In modern real-time operating systems, it is necessary to reduce the average turnaround time and average waiting time, and to meet the fairness requirements of the tasks. Therefore, using the advantages of SJF algorithm, FCFS algorithm and RR algorithm, we proposed an improved RR algorithm, called Median-based Dynamic Round Robin (MDRR). The algorithm focuses on improving the setting of dynamic quantum time. It makes a good balance in the overhead, performance and fairness of the scheduling.

II. The Proposed Algorithm

Because of the good performance of the SJF scheduling algorithm, the MDRR algorithm sorted the tasks in the ready queue in ascending order of the task's burst time. By allowing shorter tasks to run prioritized, the average waiting time and average turnaround time can be reduced as much as possible.

According to the Median theory, all tasks are sorted in ascending order of the task's burst time. This forms an ordered ready queue. The value of the variable in the middle of the queue is called the median, which is expressed as *Me*. When the number of tasks in a queue is odd number, the variable that is located in the middle of the queue is the median. When the number of task is an even number, the median is equal to the average of the two variables in the middle of the queue. The advantage of the median is that it is not affected by big or small data. Therefore, it can be used to represent the general level of the entire group of data.

When the task is submitted to the system, it is ordered by the size of the burst time, and the index number is marked with the subscript *i*. The sequence of *n* tasks is represented by using (1).

$$x_1 \le x_2 \le x_3 \le \cdots \le x_n \qquad (1)$$

where $x_i(1 \le i \le n)$ refer to the burst time of task *i* in the ready queue. *i* and *n* refer to the index number of the ordered task. We calculated *Me* using (2).

$$Me = \begin{cases} x_{\frac{n+1}{2}} & (\frac{n}{2} \ne 0) \\ \dfrac{x_{\frac{n}{2}} + x_{\frac{n+1}{2}}}{2} & (\frac{n}{2} = 0) \end{cases} \qquad (2)$$

We selected the median of burst time in the ready queue as the quantum time. If the number of tasks in the queue is even number, the median (*Me*) is equal to the average value of the two burst times in the middle position of the queue. That will cause the task adjacent to the median to add a round of scheduling, so as to increase one task switching and increase the total waiting time and turnaround time. Therefore, select the time larger than the median as the quantum time. It will reduce the number of context switching, turnaround time and waiting time. In this paper, the quantum time *Qt* is the next burst time adjacent to the median. The *Qt* was calculated by using (3).

$$Qt = \begin{cases} x_{\frac{n+1}{2}+1} = x_{\frac{n+3}{2}} & (\frac{n}{2} \ne 0) \\ x_{\frac{n}{2}+1} & (\frac{n}{2} = 0) \end{cases} \qquad (3)$$

If the number of tasks (we use *n* here to represent the number of tasks.) is even number, we select the burst time of the task with index number $\frac{n}{2}+1$ as the quantum time. If the number of tasks is an odd number, we select the burst time of the task with index number $\frac{n+3}{2}$ as the quantum time.

A. Algorithm Pseudocode

Algorithm: Proposed Median-based Dynamic Round Robin (MDRR)

```
    ready queue $T_i$, $T_i = T_1, T_2, T_3, \cdots, T_n$
    $Bt_i =$ Burst time of $i_{th}$ task
    $Qt =$ Quantum time
    $n =$ number of tasks in ready queue
1.  Assign $T_i(0 < i \le n)$ to the ready queue in ascending order
2.  while (ready queue != 0)
3.  Sort all tasks in ascending order based on their burst time
4.  if ( $\frac{n}{2} \ne 0$ )
5.   $Qt = x_{\frac{n+3}{2}}$
6.  Execute the tasks with same $Qt$ in the round
7.  else
8.   $Qt = x_{\frac{n}{2}+1}$
9.  Execute the tasks with same $Qt$ in the round
10. for each $Ti$
11. if ( $Bt_i < Qt$ )
12. Execute the task i
13. Take Ti out of the ready queue
14. else
15. Execute the task $Ti$ for a time interval up to 1 $Qt$
16.  Insert the remaining tasks into the next round of ready queue in ascending order
17.  $Bt_i = Bt_i - Qt$
18. for next round
19. Repeat until all $T_i = T_1, T_2, T_3, \cdots, T_n$ are completed
```

B. Algorithm Performance Evaluation Index

Average Turnaround Time (*ACT)*: The time between submission of the task and completion of the task is the turnaround time. The average turnaround time of n tasks ACT was calculated by using f (4).

$$ACT = \frac{1}{n}\sum_{i=1}^{n} ACT_i = \frac{1}{n}\sum_{i=1}^{n}(Ct_i - Art_i) \qquad (4)$$

Where Ct_i refers to the completion time of task *i*, Art_i refers to the arrival time of task *i*, the turnaround time ACT_i of task *i* was calculated by using (5).

$$ACT_i = Ct_i - Art_i \qquad (5)$$

Average Waiting Time (*AWT)*: From the task submitted to the system to completed, the total waiting time is called the task waiting time *AWT*. Average waiting time is the average value of waiting time of *n* tasks. The average waiting time *AWT* was calculated by using (6).

$$AWT = \frac{1}{n}\sum_{i=1}^{n} AWT_i = \frac{1}{n}\sum_{i=1}^{n}(Ct_i - Art_i - Bt_i) \qquad (6)$$

Where Ct_i refers to the completion time of task *i*, Art_i refers to the arrival time of task *i*, Bt_i refers to the burst time of task *i*. The waiting time for task *i* was calculated by using (7).

$$AWT_i = Ct_i - Art_i - Bt_i \qquad (7)$$

Context Switching Times (*CST*): If using the round robin algorithm, CPU is assigned to each task a certain service time. When the task is switched, we need save the current state of the task and load the next task at the same time, such an operation is called one context switch [14].

C. Two Examples

Example 1: If the number of tasks is even number (for example, there are 6 tasks), assume the 6 tasks are submitted to the system at random at the zero time, as in Table 1. (The time unit in this paper is "*ms*".)

Table 1. The tasks submitted randomly at zero time

Tasks	Arrival time (*Art*)	Burst time (*Bt*)
T1	0	12
T2	0	8
T3	0	23
T4	0	10
T5	0	30
T6	0	15

Sort all tasks in the ready queue in ascending order. Each ordered task is marked with the index S_n, as in Table 2.

Table 2. Even number of tasks in ascending order

Tasks	Arrival time (*Art*)	Burst time (*Bt*)	Index (S_n)
T2	0	8	1
T4	0	10	2
T1	0	12	3
T6	0	**<u>15</u>**	**<u>4</u>**
T3	0	23	5
T5	0	30	6

If the number of tasks is even number, the index number of the quantum time is $S_{nQt} = \frac{n}{2} + 1 = \frac{6}{2} + 1 = 4$, $Qt_1 = 15$. At the end of the first round, the task T2, T4, T1, and T6 have been completed. The remaining running times of T3 and T5 are 8 and 15 respectively, they waited for a second round of scheduling, as in Table 3.

Table 3. Ready tasks in the second round

Tasks	Remaining time (*Rt*)	Index (S_n)
T3	8	1
T5	**<u>15</u>**	**<u>2</u>**

In the second round of scheduling, there are two remaining tasks, and the corresponding index are $S_{nQt} = \frac{n}{2} + 1 = \frac{2}{2} + 1 = 2$ and $Qt_2 = 15$. The Gantt chart is shown in Fig.1.

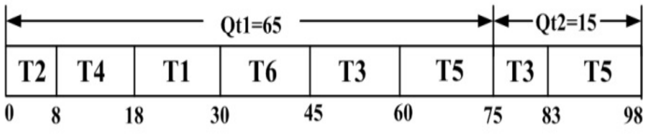

Fig. 1. Gantt chart of Example 1

Using Fig.1, we calculated that the number of context switching is 7. Using (1), (2), (3), and (4), we calculated the waiting time and turnaround time for each task, as in Table 4.

Table 4. Waiting time and turnaround time

Tasks	Arrival Time (*Art*)	Burst Time (*Bt*)	Waiting Time (*Wt*)	Turnaround Time (*Tt*)
T1	0	12	18	30
T2	0	8	0	8
T3	0	23	60	83
T4	0	10	8	18
T5	0	30	68	98
T6	0	15	30	45
Average			30.7	47.0

Example 2: If the number of tasks is odd number (for example, there are seven tasks), assume the seven tasks are submitted to the system at random at zero time, as in Table 5.

Table 5. Tasks submitted at random at zero time

Tasks	Arrival time (*Art*)	Burst time (*Bt*)
T1	0	65
T2	0	72
T3	0	50
T4	0	43
T5	0	78
T6	0	20
T7	0	30

All tasks are sorted in ascending order of burst time. Each ordered task is marked with the index S_n, as in Table 6.

Table 6. Odd number of tasks in ascending order

Tasks	Arrival time (*Art*)	Burst time (*Bt*)	Index (S_n)
T6	0	20	1
T7	0	30	2
T4	0	43	3
T3	0	50	4
T1	0	**65**	**5**
T2	0	72	6
T5	0	78	7

If the number of tasks is odd number, the index number of the quantum time is $S_{nQt} = \frac{n+3}{2} = \frac{7+3}{2} = 5$, $Qt_1 = 65$. At the end of the first round, tasks T6, T7, T4, T3, and T1 have been completed. The remaining running time of T2 and T5 are 7 and 13, respectively. In the second round of scheduling, there are two remaining tasks, and the corresponding index number is $S_{nQt} = \frac{n+2}{2} = \frac{2+2}{2} = 2$, $Qt_2 = 13$. The Gantt chart is shown in Fig.2.

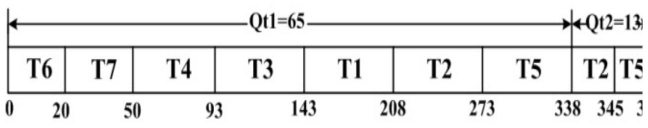

Fig. 2. Gantt chart of Example 2

Using (1), (2), (3), and (4), we calculated the average waiting time and the average turnaround time of the seven tasks, as in Table 7. Assume that these seven tasks arrive at the system at the same time. Their burst times are a random size.

Table 7. Waiting time and turnaround time

Tasks	Arrival Time (*Art*)	Burst Time (*Bt*)	Waiting Time (*Wt*)	Turnaround Time (*Tt*)
T1	0	65	143	208
T2	0	72	208	345
T3	0	50	93	143
T4	0	43	50	93
T5	0	78	280	258
T6	0	20	0	20
T7	0	30	20	50
Average			113.0	160.0

III. Comparision Analysis

In this paper, the performance of the improved MDRR algorithm is verified through simulation experiments. We compared with the traditional RR, three improved RR algorithms, DQRRR, SARR, and DABRR. In order to compare the performance of each scheduling algorithm, we use the three indexes of context switching times, average waiting time and average turnaround time. A good algorithm should have a smaller value. The last column of the following contrast diagrams refer to the algorithm MDRR.

Case 1 (as in [13]): The five tasks arrive at the same time in ascending order, as in Table 8.

Table 8. Tasks submitted in ascending order at zero time

Tasks	Arrival time (*Art*)	Burst time (*Bt*)
T1	0	40
T2	0	55
T3	0	60
T4	0	90
T5	0	102

For the tasks in Table 8, five algorithms RR, DQRRR, SARR, DABRR and MDRR are scheduled, respectively. Fig.3, Fig.4, and Fig.5 showed the results of the comparison between the new algorithm MDRR and the other four round robin algorithms respectively.

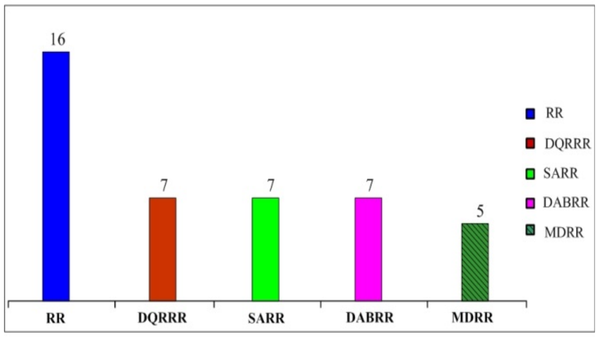

Fig. 3. Context switches times submitted in ascending order

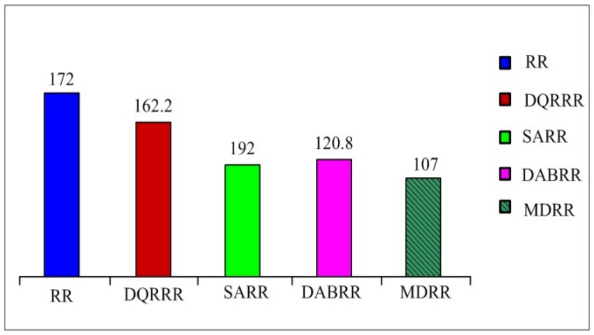

Fig. 4. Average waiting time submitted in ascending order

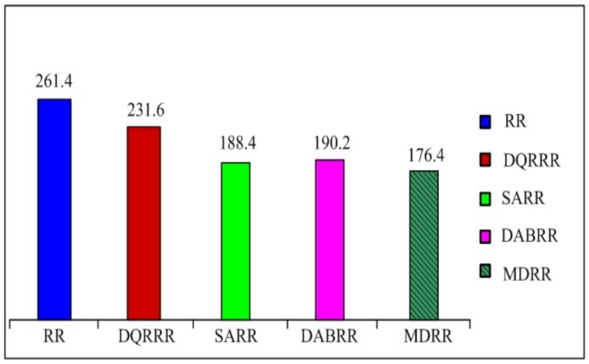

Fig. 5. Average turnaround time submitted in ascending order

From the contrast results of Fig.3, the number of switching times of the improved algorithm MDRR is 5. It obtains the minimum context switching times. The rate of improvement for the number of context switches is 41%, compared with the other four algorithms. It reduces the overhead of the system. From the results of Fig.4, the average waiting time of MDRR is 107. The algorithm MDRR obtains the minimum average waiting time. The ratio of improvement of waiting time is 29%, compared with the other four algorithms. From the results of Fig.5, the algorithm MDRR's average turnaround time is 176.4. The algorithm obtains the minimum average turnaround time. The improvement ratio of the average turnaround time of the MDRR algorithm is 16%. This shows that the MDRR algorithm has better performance, when the five tasks arrive at the same time in ascending order.

Case 2 (as in [13]): If the five tasks are submitted to the system in descending order, as in Table 9.

Table 9. Tasks submitted in descending order

Tasks	Arrival time (*Art*)	Burst time (*Bt*)
T1	0	105
T2	0	85
T3	0	55
T4	0	43
T5	0	35

For the tasks in Table 9, five algorithms RR, DQRRR, SARR, DABRR and MDRR are scheduled, respectively. Fig.6, Fig.7, and Fig.8 showed the results of the comparison between the new algorithm MDRR and the other four round robin algorithms respectively.

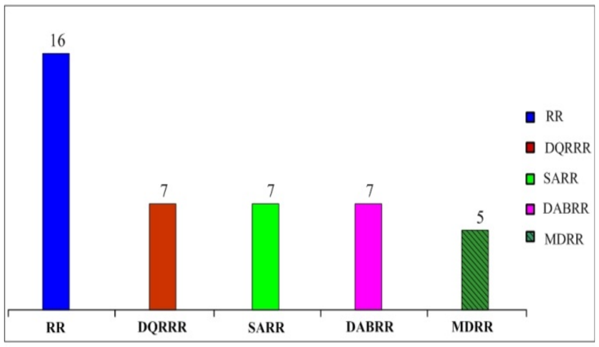

Fig. 6. Number of context switches submitted in descending order

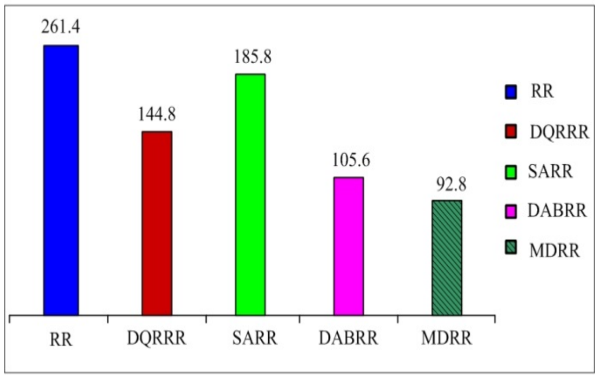

Fig. 7. Average waiting time submitted in descending order

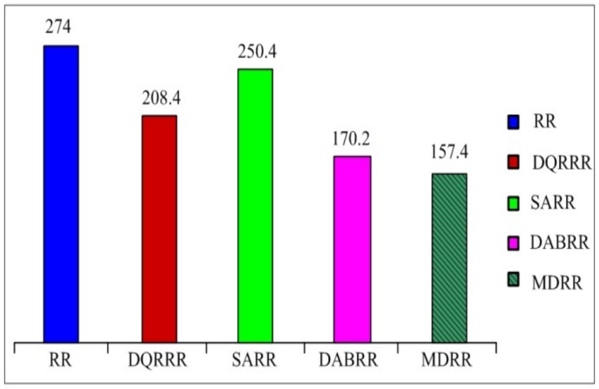

Fig. 8. Average turnaround time submitted in descending order

From the contrast result of Fig.6, the number of switching times of the improved algorithm MDRR is 5. It obtains the minimum context switching times. The rate of improvement for the number of context switches is 41%, compared with the other four algorithms. It reduces the overhead of the system. From the result of Fig.7, the average waiting time of MDRR is 92.8. The algorithm MDRR obtains the minimum average waiting time. The ratio of improvement of waiting time is 41.3%, compared with the other four algorithms. The average waiting time of the tasks is reduced, and the response time of the tasks becomes shorter. From the result of Fig.8, the algorithm MDRR obtains the minimum average turnaround time. Its average turnaround time is 157.4. The improvement ratio of the average turnaround time of the MDRR algorithm is 25.7%. So the improved algorithm MDRR obtains better performance when all tasks arrive at the system in descending order simultaneously.

Case 3 (as in [12]): If all tasks are submitted to the system at the same time in random order, as in Table 10.

Table 10. Tasks with zero arrival and burst time in random order

Tasks	Arrival time (*Art*)	Burst time (*Bt*)
T1	0	80
T2	0	45
T3	0	62
T4	0	34
T5	0	78

For the tasks in Table 10, five algorithms RR, DQRRR, SARR, DABRR and MDRR are scheduled, respectively. Fig.9, Fig.10, and Fig.11 showed the results of the comparison between the new algorithm MDRR and the other four round robin algorithms respectively.

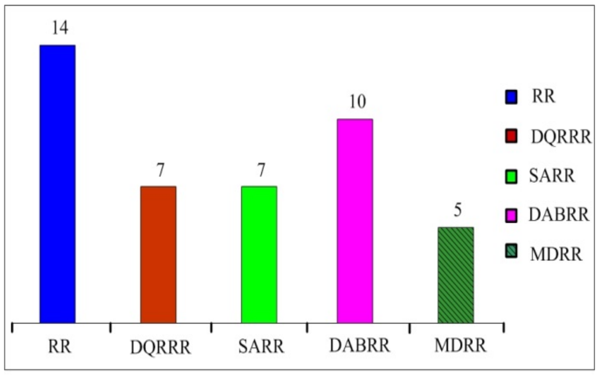

Fig. 9. Number of context switches submitted in random order

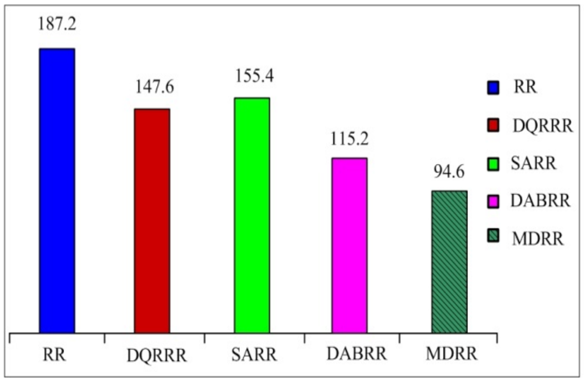

Fig. 10. Average waiting time submitted in random order

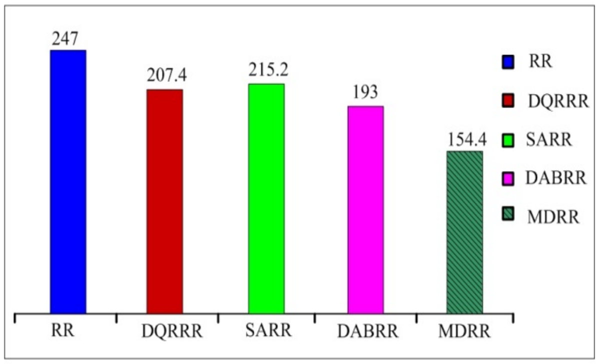

Fig. 11. Average turnaround time submitted in random order

From the contrast result of Fig.9, the number of switching times of the improved algorithm MDRR is 5. It obtains the least context switching times. The rate of improvement for the number of context switches is 41.86%, compared with the other four algorithms. It reduces the overhead of the system. From the result of Fig.10, the average waiting time of MDRR is 94.6. The algorithm MDRR obtains the least average waiting time. The ratio of improvement of waiting time is 32.4%, compared with the other four algorithms. The average waiting time of the tasks is reduced, and the response time of the tasks becomes shorter. From the result of Fig.11, the algorithm MDRR obtains the lowest average turnaround time. Its average turnaround time is 154.4. The improvement ratio of the average turnaround time of the MDRR algorithm is 24.09%. So the improved algorithm MDRR obtains better performance when all tasks arrive at the system in random order.

Case 4 (as in [13]): If five tasks are submitted to the system in a random order, and the five tasks have different arrival times, as in Table 11.

Table 11. Randomly submitted tasks with different arrival times.

Tasks	Arrival time(*Art*)	Burst time (*Bt*)
T1	0	45
T2	5	90
T3	8	70
T4	15	38
T5	20	55

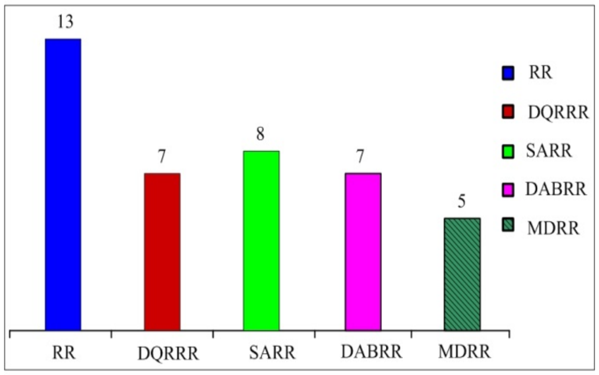

Fig. 12. Comparison of context switches

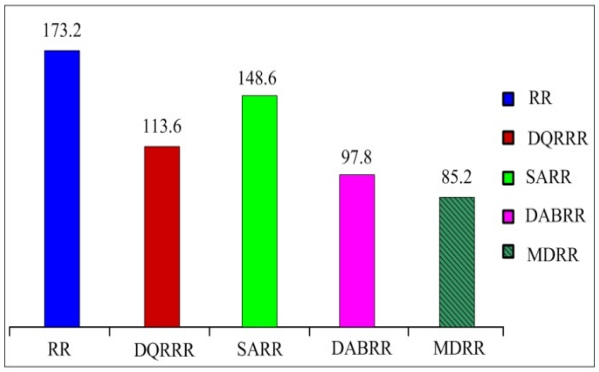

Fig. 13. Comparison of average waiting time

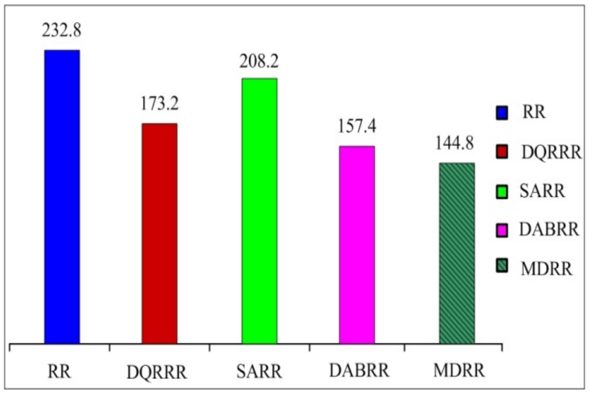

Fig. 14. Comparison of average turnaround time

For the tasks in Table 11, five algorithms RR, DQRRR, SARR, DABRR and MDRR are scheduled, respectively. Fig.12, Fig.13, and Fig.14 showed the results of the comparison between the new algorithm MDRR and the other four round robin algorithms respectively.

From the contrast result of Fig.12, the number of switching times of the improved algorithm MDRR is 5. It obtains the least context switching times. The rate of improvement for the number of context switches is 37.5%, compared with the other four algorithms. It reduces the overhead of the system. From the result of Fig.13, the average waiting time of MDRR is 85.2. The algorithm MDRR obtains the least average waiting time. The ratio of improvement of waiting time is 31.12%, compared with the other four algorithms. The average waiting time of the tasks is reduced, and the response time of the tasks becomes shorter. From the result of Fig.14, the algorithm MDRR obtains the lowest average turnaround time. Its average turnaround time is 144.8. The improvement ratio of the average turnaround time of the MDRR algorithm is 21.0%. The experiments showed that the improved algorithm MDRR obtained the least number of context switches, the lowest average waiting time and the lowest average turnaround time, when tasks with different arrival times were submitted to the system in random order.

From the simulations results, it is obvious that new algorithm MDRR had achieved a good performance compared to the traditional RR, improved DQRR, SARR, and DABRR in number of context switches, average waiting time and average turnaround time. On the other hand, the experiments results had shown that the dynamic quantum time in task had a good impact on improving the performance of the task. We can see that RR had a static quantum for all tasks, RR's quantum time did not change from round to round, its performance was poor. DQRR, SARR, and DABRR also use dynamic quantum time, but their performance were still not ideal compared with MDRR. From all of the above, we can surely conclude that the method in this paper is reasonable for the selection of dynamic time slices. Thus MDRR has relatively better performance.

IV. Conclusion

In recent years, RR algorithms are increasingly used in real-time operating systems. The task scheduling algorithm of real-time operation system requires reducing the waiting time and turnaround time, reducing overhead and giving consideration to fairness on the basis of ensuring time constraints.

In this paper, we proposed an efficient median-based dynamic round robin scheduling algorithm (MDRR). The quantum time of this paper is the next burst time adjacent to the median. The method avoids placing the tasks adjacent to the median in the next round of scheduling, reduced the number of tasks context switches. The quantum time is dynamic. Each round of scheduling needs to calculate a time slice, instead of calculating the time slice for each task, so the algorithm complexity is low. The MDRR algorithm balances the performance, fairness and complexity of the scheduling. Simulation experiments showed that the algorithm performs well in all kinds of cases. The MDRR algorithm has the advantages of better performance, lower overhead and lower complexity, compared with the traditional RR and other improved round robin scheduling algorithms.

In the next study, we intend to continue to study the calculation method of better task time slice, between static and dynamic time slices, a better balance will be found between algorithm complexity and response time. We also suggest that the MDRR algorithm is applied to embedded real-time system μC/OS-Ⅲ. It will be used to improve the scheduling results of the same priority tasks in the μC/OS-Ⅲ system.

References

[1] M. M. Tajwar, M. N. Pathan, L. Hussaini, and A. Abubakar, "CPU scheduling with a round robin algorithm based on an effective time slice", *Journal of Information Processing Systems*, vol. 13, no. 4, pp.941-950, 2017.

[2] A. E. A. Agha, S. J. Jassbi, "A new method to improve round robin scheduling algorithm with quantum time based on barmonic-arithmetic mean (HARM)", *International Journal of Information Technology and Computer Science*, vol.5, no. 7, pp. 56-62, 2013.

[3] A. Gulati, R. K. Chopra, "Dynamic round robin for load balancing in a cloud computing", *International Journal of Computer Science and Mobile Computing*, vol. 2, no. 6, pp. 274-278, 2013.

[4] M. S. Lraji, "Time sharing algorithm with dynamic weighted harmonic Round Robin", *Journal of Asian Scientific Research*, vol. 5, no. 3, pp. 131-142, 2015.

[5] M. K. Mishra, 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.

[6] J. Khatri, “An improved dynamic round robin cpu scheduling algorithm based on variant time quantum” *IOSR-Journal of Computer Engineering*, vol. 18, no. 6, pp. 35-40, 2016.

[7] R. Verma, S. Mittal, V. Singh, “A Round Robin algorithm using mode dispersion for effective measure”, *International Journal for Research in Applied Science and Engineering Technology*, vol. 2, no. 3, pp. 166-174, 2014.

[8] S. Hu, B. H. Zhao, H. J. Xiong, “A fair dynamic quantum algorithm”, *Journal of Natural Science of Hunan Normal Universit*y, vol. 35, no. 5, pp. 30-36, 2012.

[9] H. S. Behera, R. Mohanty, and D. Nayek, “A New Proposed Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and its Performance Analysis”, I*nternational Journal of Computer Applications*, vol. 5, no. 5, pp.10-15, 2010.

[10] A. R. Dash, S. K. Sahu, and S. K. Samantra, “An optimized round robin cpu scheduling algorithm with dynamic time quantum”, *International Journal of Computer Science, Engineering and Information Technology (IJCSEIT,)* vol. 5, no. 1, pp. 7-26, 2016.

[11] 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: Advances, Systems and Application*s, vol. 6, no. 12, pp.1-12, 2017.

[12] D. Nayek, S. K. Malla, and D. Debadarshini, “Improved round robin scheduling using dynamic time quantum”, *International Journal of Computer Applications*, vol. 38, no. 5, pp. 34-38, 2012.

[13] A. Alsheikhy, R. Ammar, and R. Elfouly, “An improved dynamic Round Robin scheduling algorithm based on a variant quantum time ”, *2015 11th international computer engineering conference (ICENCO)*, 2015, PP.98-104.

[14] C. H. Zhang, J. Q. Ren, P. Luo, “Dynamic round robin scheduling algorithm for μC/OS-III”, *Boletin Tecnic*, vol. 55, no. 7, pp. 8-15, 2017.

Chunhong Zhang was born on May 29, 1978. She received the M. S. degree in computer science and technology from Beijing University of Technology, Beijing, China. She is currently a lecturer at the College of Mathematics and Information Science at Langfang Normal University. Her research interests include artificial intelligence and computer application.

Luo Ping was born on Oct. 19, 1979. She received the M. S. degree in computer science and technology from Tianjin Polytechnic University of China. She is currently a lecturer at the College of Mathematics and Information Science at Langfang Normal University. Her research interests include artificial intelligence and image processing.

Yuye Zhao was born on Jun. 13, 1976**.** She received the M. S. degree in computer science and technology from University of Science and Technology Beijing of China. She is currently a lecturer at the College of Mathematics and Information Science at Langfang Normal University. Her research interests include artificial intelligence and cloud computing.

Jianqiang Ren was born on Apr. 4, 1978. He received the Ph. D. degree in Control Science and Engineering and M. S. degree in Pattern Recognition and Intelligent System from Beijing University of Technology, Beijing, China, in 2016 and 2010. He is currently an Associate Professor at the Institute of Pattern Recognition and Intelligent System at the Langfang Normal University. His research interests cover pattern recognition, computer vision and intelligent system.