A New Approach to CPU Scheduling Algorithm: Genetic Round Robin

Maria Ulfah Siregar
Informatics Department,
Faculty of Science and Technology,
State Islamic University Sunan Kalijaga
Yogyakarta
Indonesia

ABSTRACT
CPU scheduling should preserve fairness and avoid processes from do not ever obtain CPU. Modern operating system era faces multitasking on computer operational environment. If CPU scheduling is efficient, high computation could be done correctly and system could be retained stable. One criterion that must be achieved by scheduling algorithm is minimization average waiting time for a set of processes in gaining CPU allocation. There are several algorithms for CPU scheduling; one of them is Round Robin. Round Robin supplies quantum that is same for each of processes. However, there is no standard for quantum. Inevitably, if quantum is very high, response/waiting time for each process could be high, and otherwise, there is an increasing CPU overhead for context switching. This research concerns with improving Round Robin performance. Our approach is to combine Round Robin with Genetic algorithm. In this approach, an individual is a quantum that will be iterated for achieving best quantum that will produce minimal average waiting time. We use integer number for representing a chromosome with length three. Furthermore, we use roulette wheel method for parent selection and steady state replacement technique for survival selection. By using one point crossover and flip mutation, this approach can result better average waiting time than that were found in references used.

General Terms
CPU Scheduling, Round Robin Algorithm, Genetic Algorithm.

Keywords
Genetic Round Robin.

1. INTRODUCTION
One interesting topic in Operating System is CPU Scheduling. This scheduling relates with CPU allocation to execute processes in a computer system. CPU scheduling is a main task of an operating system [1]. Scheduling should be done correctly for keeping fairness and avoiding processes from do not ever be allocated CPU (process starvation).

CPU scheduling is necessary, particularly in computer networking system which is formed from group of workstations and servers. Next, in this modern operating system, multitasking computer is a goal and this is relied on algorithm for CPU scheduling. The reason for this, CPU is an effective or important part a computer [1].

Moreover, in this era, with help from VLSI (Very Large Scale Integrated circuit), it is possible to produce high powered processor [2]. This amazing power should be utilized so as it is not useless. Along with the power of processor's computation, there is an increasing in applications which use that power.

One criterion that should be fulfilled by scheduler is to minimize average waiting time for whole of processes in obtaining CPU allocation. There are several algorithms for CPU scheduling; one of them is Round Robin (RR).

Basic concept in RR is a usage of time-sharing [3]. Each process will obtain the same CPU time, namely quantum time, which function as a limitation in processing time, generally in range 1-100 millisecond. After quantum time for a process is finished, the process will be stopped from its execution and putted on the ready queue. Next, the next process will be chosen to be executed. These steps will run several times until all processes have been served completely by CPU.

Although there is a range value for quantum time, yet there is no standard. Meanwhile, if the quantum time is very high, time needed to response/wait (how much time it needed to be served) is quite high. Moreover, if it very low, it makes overhead for CPU.

Searching for the best quantum time has goal which is to minimize average waiting time for a group of processes. It is hope that each process can finish its job in a reasonable time. The quicker a process finishes its job impacts in as many processes that can be served by CPU. This will come to better throughput of CPU for it always busy and never be in idle.

Based on introduction above, we think it is necessary for finding the best quantum for achieving better average waiting time, turnaround time and minimal context switch. We propose a Genetic algorithm which is combined with traditional Round Robin.

2. OBJECTIVES
Our research has goals as following:

1) to design and implement a system that can produce the best quantum for come to optimum average waiting time
2) to evaluate GA parameters which can result the best solution

3. PREVIOUS RESEARCHES

Some researches that appropriate with our research are:

H.s. Behera, Sreelipa Curtis and Bijayalaxmi Panda [4] in India proposed a new RR algorithm using a modified mean-deviation. This is addressed to real time system and it is proven that this algorithm is better than traditional RR, SMDRR and SRBRR in which it can reduce context switch, average waiting time and average turnaround time.

Mehdi Neshat, Mehdi Sargolzaei, Adel Najaran and Ali Adeli [5] in Iran used Fonseca and Fleming's Genetic Algorithm (FFGA) multiobjective optimization to yield an adaptive CPU scheduling. This proposed algorithm is compared to seven classical scheduling algorithms, which are FCFS, RR (either equal or prioritized), SJF (pre-emptive and non-pre-emptive), and Priority (pre-emptive and non-pre-emptive). The results showed that this algorithm is more optimized than other methods.

Supriya Raheja, Reena Dhadich and Smita Rajpal [6] in India proposed a new RR algorithm using Linguistic Synthesis to attain an optimum time quantum. This approach includes Mamdani Fuzzy Inference System and produces LRRTQ Fuzzy Inference System. Based on numerical analysis, this algorithm shows the improvement in the performance of the system by cutting off an unimportant context swithces and an unreasonable turnaround time.

Debashree Nayak, Sanjeev Kumar Malla and Debashree Debadarshini [7] in India conducted a research that tends to improve RR scheduling using Dynamic Time Quantum. That concept reduces context switching, average waiting time and average turnaround time. Processes are arranged in ascending order according their burst time. After that, median is calculated to find an optimal burst time.

Sanjay Kumar Panda and Sourav Kumar Bhoi [8] in India proposed an effective RR algorithm using Min-Max dispersion measure of remaining CPU burst time. This algorithm performs better than RR algorithm in terms of average turnaround time, average waiting time and number of context switches.

Vishnu Kumar Dhakad, Saroj Hiranwal and K. C. Roy [9] in India proposed a new algorithm in scheduling which priority driven according to burst time. Results show that this algorithm can solve problem of fixed quantum and it support a development of self-adaptive system.

Abbas Noon, Ali Kalakech and Seifedine Kadry [1] in Lebanon proposed a new algorithm namely AN, a dynamic quantum. An operating system should manage quantum appropriately with burst time of a set of processes in ready queue. This algorithm can improve performance of RR algorithm.

Jeegar A. Trivedi and Priti Srinivas Sajja [2] in India has goals how a RR algorithm can optimize a multitasking environment by increasing throughput and decreasing waiting time of a process. This objective is achieved by combining RR method with Neuro Fuzzy approach.

Samih M. Mostafa, S. Z. Rida and Safwat H. Hamad [10] in Egypt proposed usage of Integer Programming for finding better quantum.

Rakesh Kumar Yadav, Abhishek K. Mishra, Navin Prakash and Himanshu Sharma [11] in India proposed a new algorithm which is a combination of RR and SJF (Shortest Job First) algorithm. From experiments, results show that this combination is better than pure RR.

In a reference entitled "Finding Time Quantum of Round Robin CPU Scheduling Algorithm Using Fuzzy Logic" [12] which is done in India, it is used a Fuzzy Logic method to decide a value for a quantum which is neither high nor small for obtaining reasonable response and good throughput system.

Meanwhile, our proposed research relates with an integration GA into RR. This has goal to find the best quantum time which produces the best average waiting time. Therefore, we contribute to propose a new approach to classic RR algorithm, in which we combine GA and classic RR to yield Genetic RR (GRR) algorithm. In line with terms in GA, quantum time will be an individual and iterated to produce better average waiting time. The uniqueness our algorithm is that the quantum time does not defined by us, rather it will be created by our proposed algorithm.

4. RESEARCH METHOD

In this research, an individual is a quantum time and decoded by using integer representation to create one chromosome. Every chromosome consists of three digit integers (genotype) and each chromosome's value (phenotype) is laid between 1 and 100. We use one point crossover operator and flip mutation for reproduction offspring. Fitness is based on average waiting time for a set of processes. Our system tends to minimize average waiting time, so the fitness is formed as:

$$fitness(q) = \frac{1}{awt(q)} \quad (1)$$

Next, selecting of several pairs of individual which act as parent in recombination (crossover) processes uses roulette wheel method. We will select better individual to put into mating pool. Next generation is built by using steady state replacement method. In this method, size of population is preserved the same. Therefore, there is a competition between parents and children to survive. Iteration stops whether or not it reaches its maximum value which is the number of generation.

Our system uses Java language. Its user interface is figured out in figure 1.

Set of processes will be iterated in GA cycle until it reaches number of generation. Every burst time is inputted in dialog as showed by figure 2.

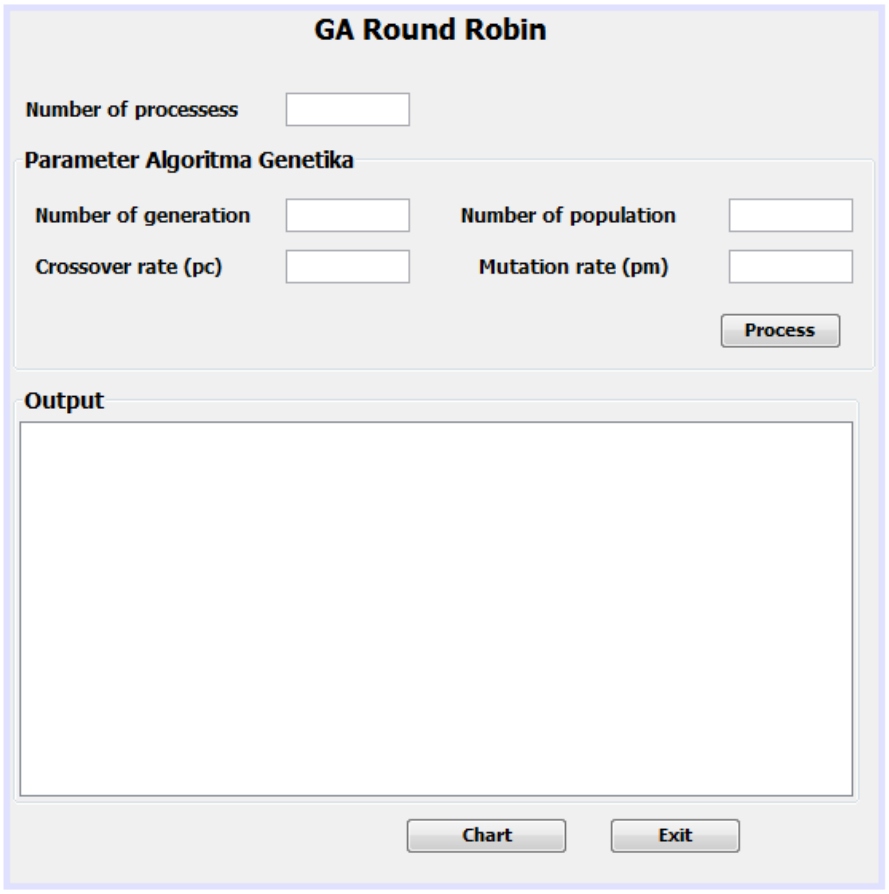

Fig 1: User interface of Genetic RR

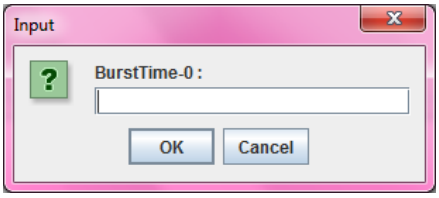

Fig 2: Dialog window for inputting burst time

For each generation, we show its individual value (quantum time) and its average waiting time. Moreover, the best individual for each generation is figured out in XY chart.

5. RESULTS AND DISCUSSIONS

We conduct several experiments which can be divided into two categories. The first is a comparison by using data which are mentioned in lecturer hand-out. Second is comparison with data from previous researches. For both of categories, we assume that each process has the same arrival time. Furthermore, each process is equal to each other. This section consists of two subsections where each section is arranged to each category of experiments.

5.1 Experiment with the First Category

For the first category, we have two data. Data 1 which are obtained from [3] are depicted in figure 3. As can be seen from figure 3, Data 1 consists of three processes.

Process	Burst Time
P_1	24
P_2	3
P_3	3

Fig 3: Data 1

According to operating system lecturer hand-out, for these data in traditional RR, quantum-time is defined as 4 ms. Its Gannt chart is showed in figure 4.

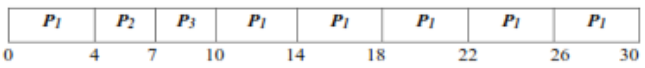

Fig 4: Scheduling three processes of data 1

If we calculate, waiting time for P_1 is 6 ms, P_2 is 4, and P_3 is 7 then its average waiting time is (6 + 4 + 7)/3 = 5.67 ms.

For each data in first category, we do experiments by using 16 schemas. Based on [13], crossover rate is typically in range [0.6, 0.9]. We hope that 60% - 90% of chromosomes will go through crossover [14]. Meanwhile, for mutation, its rate is between 1/pop_size*L and 1/L [15], where L is length of chromosome. We choose 0.8 and 0.9 as crossover rates, 0.033

(for smallest number of population, 10) and 0.33 as mutation rates. The whole schemas are listed in table 1 below:

Table 1. Schemas of experiment with category 1

Schemas	Parameters			
	Number of generation	Number of population	pc	pm
I	10	10	0.8	0.033
II	50	50	0.8	0.033
III	100	50	0.8	0.033
IV	100	100	0.8	0.033
V	10	10	0.9	0.033
VI	50	50	0.9	0.033
VII	100	50	0.9	0.033
VIII	100	100	0.9	0.033
IX	10	10	0.8	0.33
X	50	50	0.8	0.33
XI	100	50	0.8	0.33
XII	100	100	0.8	0.33
XIII	10	10	0.9	0.33
XIV	50	50	0.9	0.33
XV	100	50	0.9	0.33
XVI	100	100	0.9	0.33

Table 2. Results of data 1

Schemas	Values	
	Quantum time	Average waiting time
I	8	8.33
II	3	5
III	3	5
IV	3	5
V	4	5.67
VI	3	5
VII	3	5
VIII	3	5
IX	11	10.33
X	3	5
XI	3	5
XII	3	5
XIII	1	5.67
XIV	3	5
XV	3	5
XVI	3	5

Each schema is done five times. The best result among five experiments for each schema of data 1 is given in table 2.

As explanation for this experiment, here we give two figures, which are figure 5 and figure 6. Both of them are for first schema. Best quantum means quantum which can result the best average waiting time.

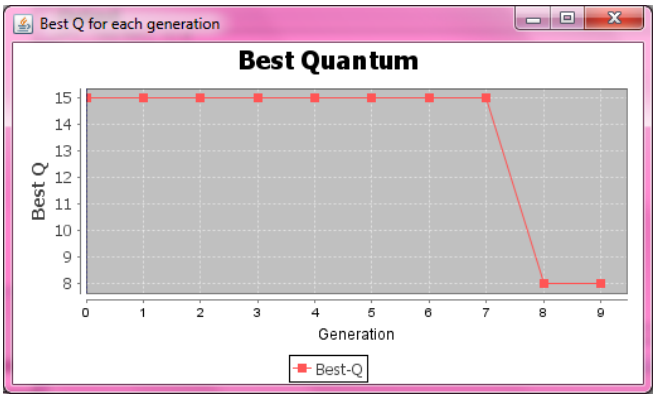

Fig 5: Best quantum for each generation of schema 1

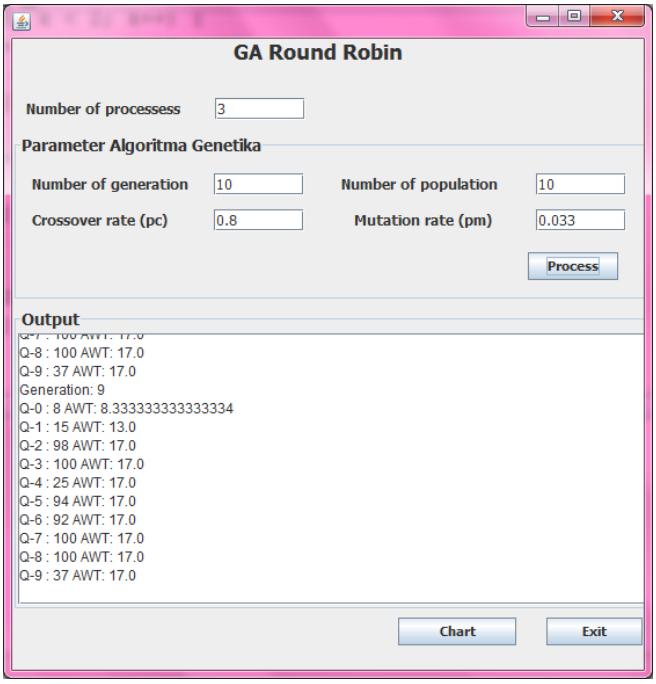

Fig 6: Input and output for an experiment of schema 1

Next is an experiment with data 2 (as showed by table 3). Data 2 is obtained from [16]. On that book, value for quantum time is 20 ms. Average waiting time resulted with traditional RR is 73 ms. Gannt chart for this data can be seen in figure 7.

Table 3. Data 2

Process	Burst time
P1	53
P2	17
P3	68
P4	24

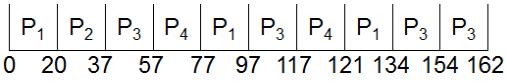

Fig 7: Gannt chart of data 2

Like data 1, we conduct five experiments for each schema and we choose the best among the five. Our result is listed in table 4.

Table 4. Result of data 2

Schemas	Values	
	Quantum time	Average waiting time
I	83	65.25
II	27	65
III	27	65
IV	27	65
V	27	65
VI	27	65
VII	27	65
VIII	27	65
IX	92	65.25
X	27	65
XI	27	65
XII	27	65
XIII	27	65
XIV	27	65
XV	27	65
XVI	27	65

5.2 Experiment with the Second Category

In second category, we compare our algorithm with some of algorithms which are proposed by researchers in our references. Our experiments for second category differ from first one. Here, we use schemas 16 only. The reason is concluded from first category, for the 16th schemas, its average waiting times is approximately better than other schemas. Moreover, we do only once experiment for this schema.

Data 3 which is obtained from Case 1 of Noon's paper [1] is figured out in figure 8. His algorithm is called AN. Here we just concern with waiting time and our algorithm's result is 50 ms for average waiting time. We can offer result that is the same with AN.

Case 1: Assume four processes arrived at time = 0, with burst time (P1 = 20, P2 = 40, P3 = 60, P4 = 80):

	Fixed Quantum=20ms	Dynamic method [2]	AN
Turn-around time	120	112.5	100
Waiting time	70	77.5	50
Context switch	9	6	5

Fig 8: Case 1 of Noon's paper

Figure 9 shows us our result and it lists down the last generation.

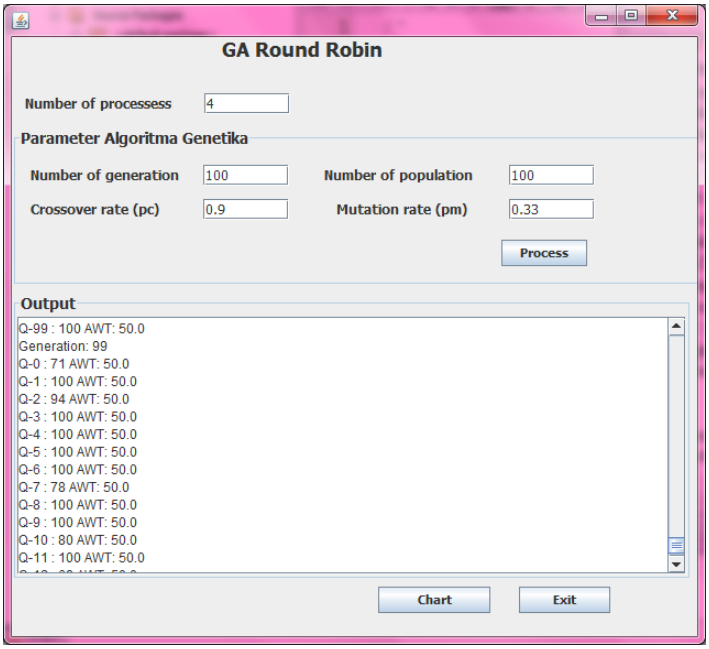

Fig 9: Input and output for case 1

Data in figure 10 is a case 2 in the same paper as previously. For this data, again we get same result as AN, 32 ms.

Case 2: Assume four processes arrived at time = 0, with burst time (P1 = 10, P2 = 14, P3 = 70, P4 = 120):

	Fixed Quantum=20ms	Dynamic method [2]	AN
Turn-around time	100.5	96	85.5
Waiting time	47	42.5	32
Context switch	11	6	5

Fig 10: Case 2 of Noon's paper

Data in figure 11 is obtained from Trivedi's paper [2]. Its average waiting time is 69 ms. Our algorithm can achieve better result which is 65 ms.

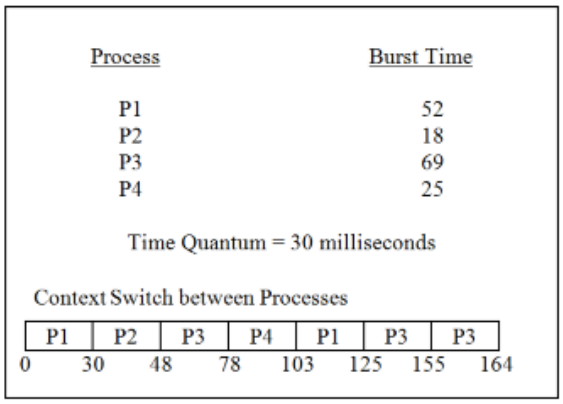

Fig 11: Data from Trivedi's paper

Figure 12 shows our algorithm's result.

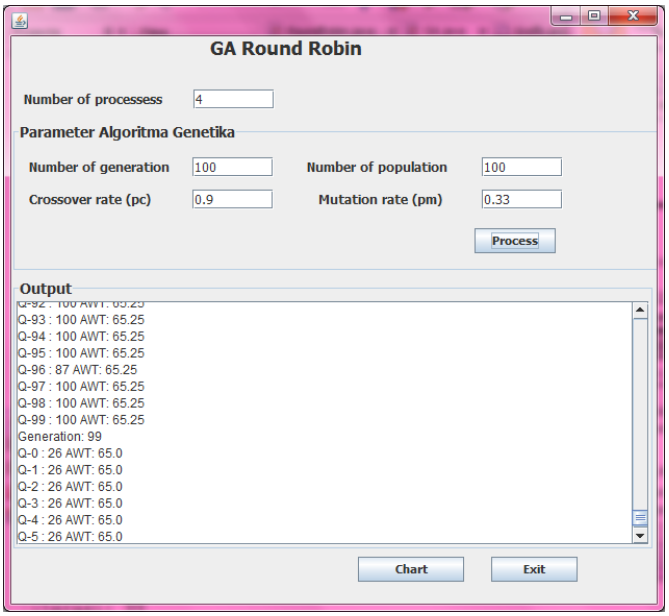

Fig 12: Our result for data in figure 11

Data for next experiment is showed in figure 13. Dhakad's result is 71 ms and ours is 62 ms. Our result is showed in figure 14.

Process	*Arrival Time (ms)*	*Burst Time (ms)*
P1	0	14
P2	0	34
P3	0	45
P4	0	62
P5	0	77

Fig 13: Data from Dhakad's paper [4]

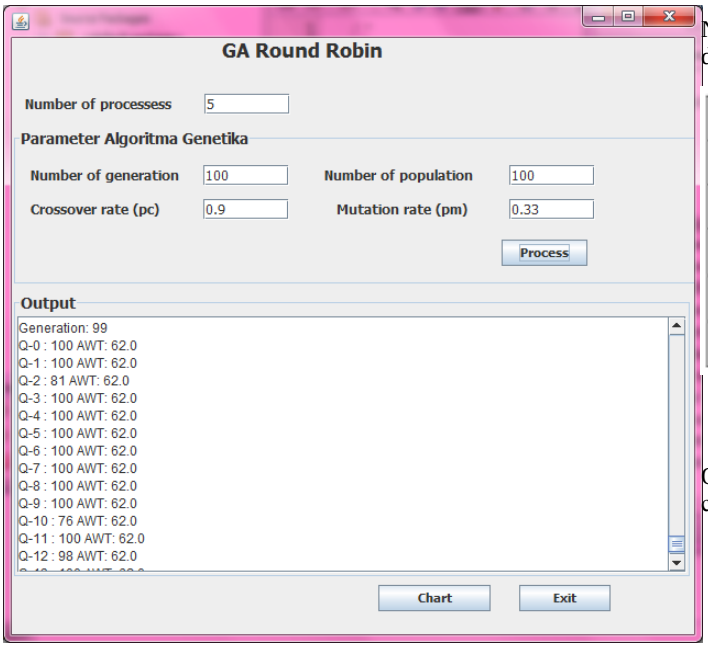

Fig 14: Input and output for data from figure 13

Our next experiment is with Yadav's data. For data in figure 15, Yadav's result is 39.2 ms. Our algorithm achieves better, which is 30.2 ms.

Process	Burst Time in ms
A	24
B	20
C	8
D	10
E	3

Fig 15: Data from Yadav's paper [6]

Figure 16 give result from our algorithm for Yadav's data. As mentioned earlier, for second category, we use last schema.

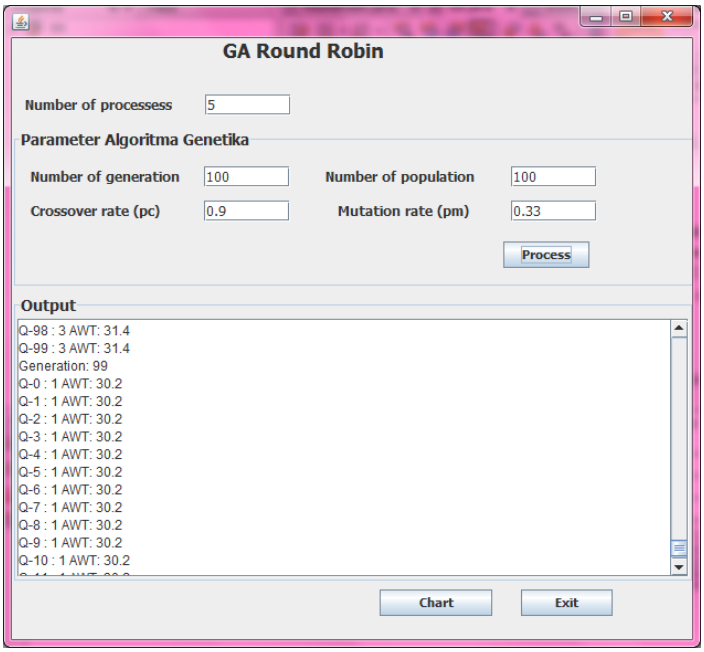

Fig 16: Input and output for data from figure 15

Next, we do experiment with data from [4]. Figure 17 describes the data.

Processes	Arrival Time	Burst Time
P1	0	13
P2	0	35
P3	0	46
P4	0	63
P5	0	97

Fig 17: Data from Behera, H.S. et.al

Our proposed algorithm's result is shown in figure 18. As comparison, Behera's result is 62.4 ms, the same with us.

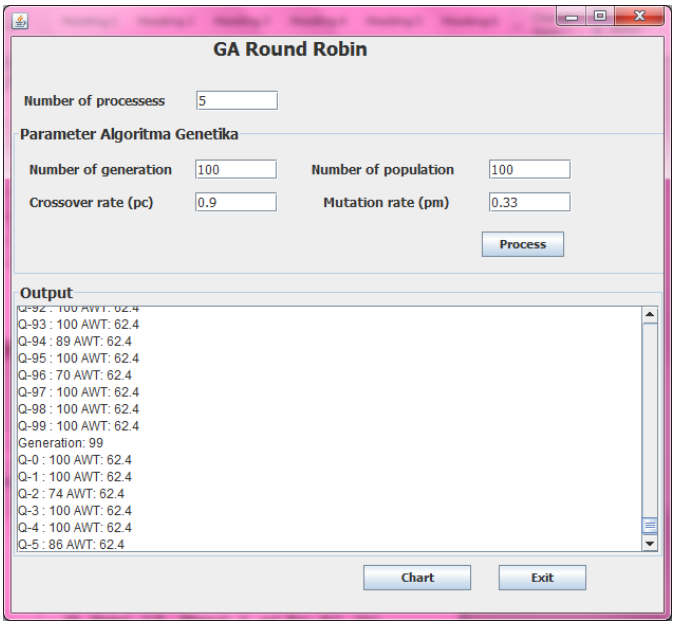

Fig 18: Input and output from figure 17

From [7], we use data as described by figure 19. Here, the sum of processes is five and assumed all the processes arrive at the same time at 0 ms.

No. of process	At	Bt
P1	0	30
P2	0	34
P3	0	62
P4	0	74
P5	0	88

Fig 19: Data in increasing order of Nayak's paper

Nayak's result is 85.6 ms, whereas ours is 84 ms. Our result is shown by figure 20. This result is much better than Nayak's one. The difference is 0.4 ms.

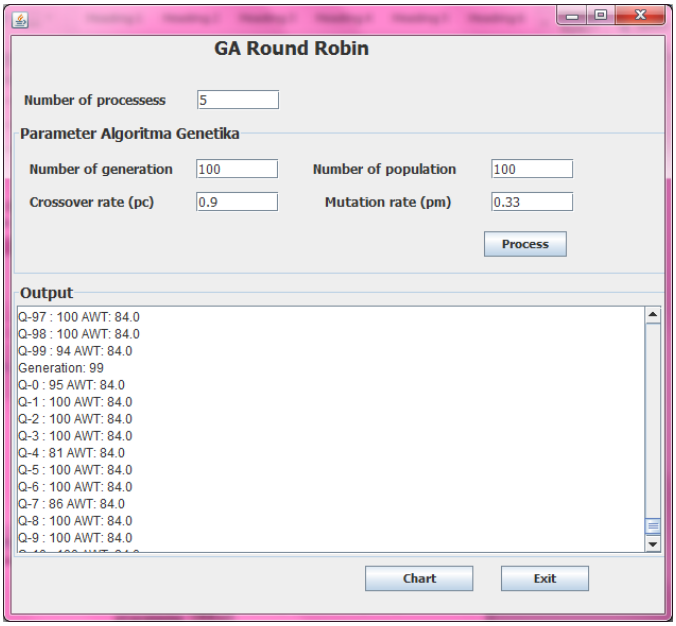

Fig 20: Input and output for figure 19

Our last experiment uses data from [6]. The data is given in table 5 below. For this experiment, we yield much better result than Raheja's one. Raheja's average waiting time is 11.8 ms which is worse than ours which is 9.5 ms. For the explanation of our result, see figure 21 below.

Table 5. Data for last experiment

Process	Burst time
P1	8
P2	5
P3	4
P4	7

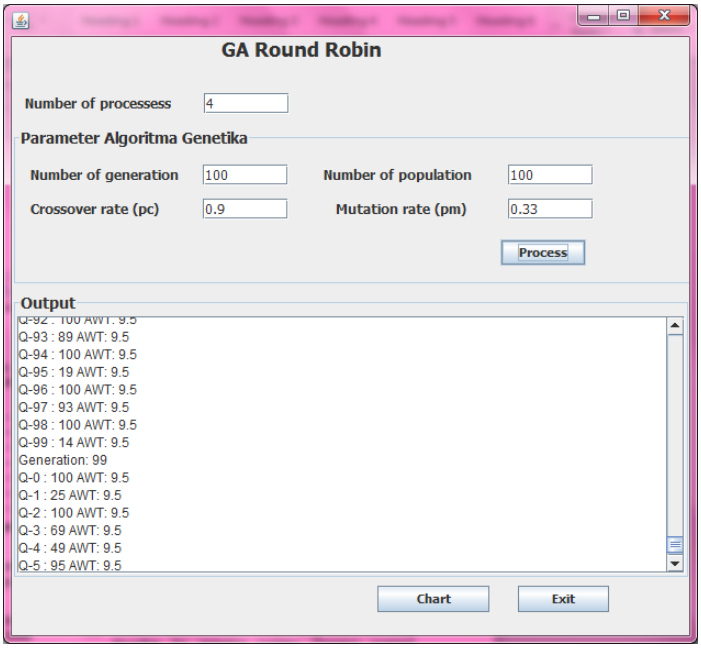

Fig 21: Input and output for data in last experiment

6. CONCLUSION AND FUTURE WORK

Based on our experiments, we can conclude that, integration GA into RR can achieve better result in terms of average waiting time. For the 16th schema that we used in second category, approximately this proposed algorithm can result better than previous researchers. Meanwhile, for the first category, our algorithm can produce quantum time which can result better average waiting time than written in lecturer hand-out. We may assume that better average waiting time is usually achieved for large number of generation and/or population.

However, because of GA's characteristic as stochastic solution searching method, sometimes our algorithm results badly. Therefore, for one data and schema it is better to do more than once experiment. In other words, we should supply repetition on the same data and schema.

In spite of weaknesses such as repetition in running algorithm for more than one to infinite number of experiment, based on our experiences, it took less time to run one experiment. Therefore, although it runs more and more times, it will take insignificant times. Furthermore, because of our proposed

algorithm approximately can yield better average waiting time than other researches in our references, this proposed algorithm should be developed more. So, it can not only produce average waiting time, but also other parameters, such as turnaround time, context switches and etc.

7. ACKNOWLEDGMENTS

We would thank to previous researchers who give us data for experiments. Moreover, we would appreciate our head of department for let us teach Operating System in previous semester.

8. REFERENCES

[1] Noon, A. K. 2011. A new Round Robin based scheduling algorithm for operating systems: Dynamic quantum using the mean average. International Journal of Computer Science Issues, 224-229.

[2] Trivedi, J.A. and Sajja, P.S. 2011. Improving efficiency of Round Robin scheduling using Neuro Fuzzy approach. International Journal of Research and Reviews in Computer Science. 2: 308-311. .

[3] Sugiantoro, B. 2010. Handout for Operating System's Lecture. Yogyakarta: Handout's lecture of Informatics Department, Faculty of Science and Technology, State Islamic University Sunan Kalijaga.

[4] Behera, H.S., Curtis, S. and Panda, B. 2012. Enhancing the CPU Performance Using A Modified Mean-Deviation Round Robin Scheduling Algorithm for Real Time Systems. Journal of Global Research in Computer Science. 3: 9-17.

[5] Neshat, M., Sargolzaei, M., Najaran, A. and Adeli, A. 2012. The New Method of Adaptive CPU Scheduling Using Fonseca and Fleming's Genetic Algorithm. Journal of Theoretical and Applied Information Technology. 37: 1-16.

[6] Raheja, S., Dhadich, R. and Rajpal, S. 2012. An Optimum Time Quantum Using Linguistic Synthesis for Round Robin CPU Scheduling Algorithm. International Journal on Soft Computing. 3: 57-66.

[7] Nayak, D., Malla, S.K. and Debadarshini, D. 2012. Improved Round Robin Scheduling using Dynamic Time Quantum. International Journal of Computer Application. 38: 34-38.

[8] Panda, S.K. and Bhoi, S.K. 2012. An Effective Round Robin Algorithm using Min-Max Dispersion Measure. International Journal on Computer Science and Engineering. 4: 45-53.

[9] Dhakad, V.K., Hiranwal, S. and Roy, K.C. 2011. Adaptive Round Robin scheduling using shortest burst approach based on smart time slice. International Journal of Computer Science and Communication. 2: 319-323.

[10] Mostafa, S.M., Rida, S.Z. and Hamad, S.H. 2010. Finding time quantum of round robin CPU scheduling algorithm in general computing systems using integer programming. IJRRAS.

[11] Yadav, R.K., Mishra, A.K., Prakash, N. and Sharma, H. 2010. An improved Round Robin scheduling algorithm for CPU scheduling. International Journal on Computer Science and Engineering. 02: 1064-1066.

[12] Alam, B., Doja, M.N. and Biswas, R. 2008. Finding time quantum of Round Robin CPU scheduling algorithm using Fuzzy Logic. ICCEE. 795-798.

[13] Eiben, A.E. & Smith, J.E. 2007. Introduction to Evolutionary Computing. Germany: Springer.

[14] Fadlisyah, Arnawan, and Faisal. 2009. Algoritma Genetika. Yogyakarta: Graha Ilmu..

[15] Suyanto. 2008. Evolutionary Computation. Bandung: Informatika.

[16] Silberschatzs, A., Galvin, P.B., Gagne, G. 200 9. Operating System Concepts. Massachusetts: Addison Wesley.