استخدام الخوارزميات الجينية لإعادة ترتيب طابور الانتظار في خوارزمية دائرة روبن للجدولة الزمنية ذات المعدل الموزون للشريحة الزمنية

د. ندى الحكاك

كلية بغداد للعلوم الاقتصادية الجامعة

Using Genetic Algorithm to Reorder Ready Queue in Round Robin Scheduling Algorithm with Weighted Average Time Slice Genetic Weighted Round Robin (GWR) algorithm

Dr. Nada M. Al-Hakkak
Computer Science Faculty

الملخص

احد اهم المجالات البحثية في نظم التشغيل هي جدولة المعالج ، خوارزمية دائرة روبن للجدولة الزمنية تعتبر احد خوارزميات جدولة المعالج الشائعة الاستخدام. الخوارزميات الجينية تعتبر احد طرق التحسين ولذلك تستخدم مع خوارزمية جدولة المعالج لغرض الحصول على اقل معدل انتظار للعمليات في حالة انتظار بأن يخصص لها وقت المعالج. هذا البحث اقترح خوارزمية تستفيد من محاسن الطرق اعلاه مع حساب وقت الشريحة الزمنية حسب مبدأ المعدل الموزون ، وذلك للوصول الى الاستخدام الامثل لجدولة المعالج مع اقل معدل انتظار للعمليات.

الكلمات المفتاحية: جدولة المعالج ، دائرة روبن ، الخوارزميات الجينية ، المعدل الموزون.

Abstract

One of the most important research topics in operating systems is CPU scheduling, different algorithms are available and new once were proposed by researchers in order to enhance CPU's performance. Round Robin (RR) is one of CPU scheduling algorithms that depends on specific calculated time slice given to waited processes in order to allocate the CPU, Genetic Algorithms (GA) is a heuristic method used for optimization. This paper proposed Genetic Weighted Round Robin (GWR) algorithm; which is an optimization tool to RR that uses GA to reorder ready queue for processes of different weights, using weighted average for time slice calculation. GWR helped in enhancing allocation strategy.

Keywords: CPU, Round Robin (RR), Genetic Algorithm (GA), heuristic, Genetic Weighted Round Robin (GWR).

Introduction

Operating system is a program that acts as intermediary between user of computer and computer hardware; it's much like user-interface and a basis for other application programs [1], and one of Operating system's responsibilities is resource management which is done by the help of scheduling.

Scheduling algorithm is a resource allocation method CPU scheduling deals with allocating CPU-time to a specific process [2]. Those algorithms differ from each other in the way that CPU selects the process from the ready queue.

However, there are many scheduling methods; FCFS, SJF, priority, and Round Robin (RR); Where RR is used in time sharing operating system [3]. The simple RR gives an equal time for all processes in the ready queue; which is called time slice. Each process will use its part of time slice many times (rounds) until it finish its execution [2].

Moreover, selecting scheduling algorithm depends on some factors or criteria's: CPU utilization, throughput, response time, turnaround time, waiting time, correctness, overhead, preventability, efficiency, fairness, and load balancing.

Furthermore, waiting time refers to the processes' waiting time spent in ready queue waiting for CPU allocation. While, turnaround time refers to the total time the process spend in the system, it is the summation of waiting time to get memory location, waiting in ready queue, executing in the PU, and I/O operations [2]. The rest of this paper is organized as follows: description to round robin and genetic algorithm, related work, proposed idea and implementation, conclusion with future work, and references.

Round Robin (RR) and Genetic Algorithms

Round Robin depends on time slice that refers to the execution of job for a specific amount of time so the scheduler goes around the ready queue allocating the CPU to all the processes, as shown in the following figure, figure (1) [4].

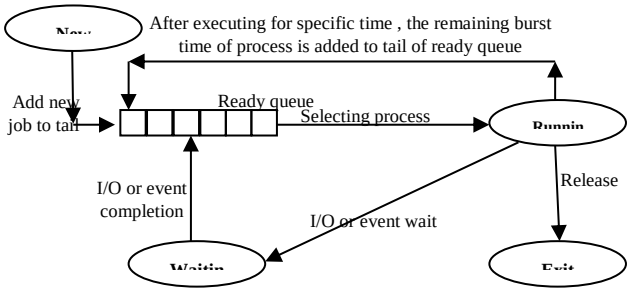

Figure (1): Round Robin and Ready Queue

However, many researchers were interested with the constant time quantum problem [5, 6, 7, and 9] to minimize the waiting time and turnaround time; While others reorganized the ready queue in increasing or decreasing order, [8 and 9] to get better performance.

Mainly, an algorithm composite of multiple steps, a genetic algorithm is one of problem solving methods that uses genetic in its work [11].

Genetic Algorithm (GA) is an optimized tool that solves complex problems (NP-hard problems) like scheduling problems and manufacture systems [12]; it works randomly in searching for resources [13]. Mainly, Genetic algorithm composite of the following main steps:

1- Chromosome's creation.
2- Primary population.
3- Creating fitness function.
4- Describe the strategy of selection.
5- Define genetic operators.
6- Define parameters. [14]

However, the selection of individuals is made depending on the ability of generating offspring's with high quality. It contains many iterations, those called generations [15], and it's work starts by creating a population that is a reflection to the problem to be solved, followed by the fitness step that contains an evaluation of fitness function for chromosome evaluation, then a new generation should be created by selecting two parents for crossover (depending on crossover probability), and the new offspring should be mutated (also depending on mutation probability finally we should check the satisfaction of end conditions to decide [16]. Moreover, it contains two main operations: selection and reproduction, where the selection feeds the reproduction with the best individuals [17].It has many advantages, like:

1- No need for function optimization.
2- All generations work in parallel.
3- Depend on probability (crossover and mutation).

Furthermore, genetic algorithms have the following main steps:

Step-1 representation methods (data representation)

1- Adjacency representation
2- Ordinal representation
3- Path representation

Step-2 generating initial population

1- Randomly way
2- Heuristic way

Step-3 selection (also called reproduction); it's done by roulette wheel selection, depending on probability value.

Step-4 crossover (to generate better offspring); it's done with the use of ordered crossover.

Step-5 mutation, with many types:

1- Towers mutation.

2- Reverse sequence mutation (RSM)
3- Centre inverse mutation (CIM)
4- Throes mutation
5- Partial shuffle mutation (PSM) [17 and 22].

Mainly, the fitness of the chromosome determines if it will survive for next generation or not. It's like the objective function, the greater the fitness the greater the surviving probability [18 and 19]. In this paper the fitness function is the minimum value of the average waiting time. And the fitness step comes before the crossover.

One of the main selection tools in genetic algorithm is Rolette wheel selection, it works linearly; the wheel contains weighted slots that are related to individual's fitness values. The target value should be set before the work wheel, so that it will not stop until it reaches its target. Hence, if we have N individuals then the wheel will run N times and each time the selected individual is the one under the wheel's marker [11]. The following figure, figure (2); describe GA's work flowchart:

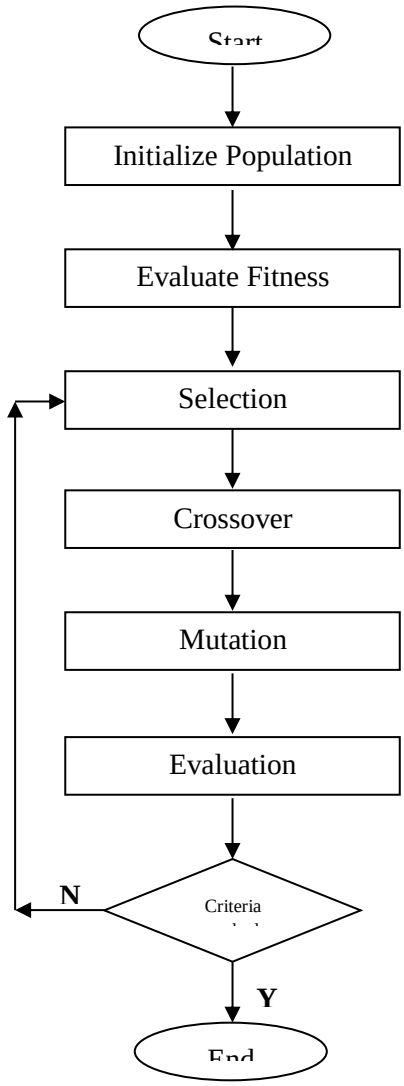

Figure (2): Genetic Algorithm Flowchart

Crossover is a process of creating better offspring by taking two parent chromosomes and chose a cross point or more along the chromosome length to be followed by swapping the substring between the two parents, to get new offspring [11]. There are many types of point crossover, like one point crossover and two-point crossover, and more [20 and 21].

Related Work

Many researches have been done to increase CPU utilization and to get better performance for CPU scheduling algorithms. The following are some of the work related to GA and RR.

In [23] the others decoded the time quantum with length three to create one chromosome. In the same way they created many chromosomes with value between 1 and 100. They used one point crossover operator and flip mutation to get offspring. The main objective was to minimize average waiting time; they used java language to write the code.

In [24] they generated (randomly) a population of ten chromosomes with size of five for each chromosome (length); those chromosomes differ from each other in values. They selected average waiting time as a fitness function of the chromosomes; the fittest chromosome is one with minimum average waiting time, where the time quantum is recalculated on each round. The concluded GA based RR with dynamic quantum is better than static quantum.

In [25] they used GA to find best scheduling solutions; they used a set of individuals where each one is a proposed solution code. With the use of crossover and mutation operations they produced new generations, they wanted to get the best sequence they used order crossover with some modification the crossed two points of min and max values in parent but they will have the same problem of duplicated values.

In [26] they constrain on DPM (Dynamic Power Management) and scheduling as the main factor that affect DPM's performance. They used GA to improve scheduling as the main factor that affects DPM's performance. They used GA to improve scheduling in the real time by minimizing the consumed power.

In [27] they used the function (Evpop popcurrent) to get initial population, and the fitness function (Fit popcurrent) with selection fitness (pickchrams popcurrent) and crossover function (M crossover popcurrent) for one point crossover closed randomly. They applied multiple scheduling algorithm (FCFS, SJF, RR, and GA) so they used GA to calculate the waiting time.

In [28] constraint on crossover of the longest burst in two points in the parents (so they modified the order crossover and compared average waiting time between (FCFS, RR, SJF, GA).

In [29] they combined two algorithms EDF (Earliest Deadline First) and new genetic algorithm (GA), they tested the algorithm under load and unloaded environment to compare the results.

In [30] they proposed optimized RR using dynamic time slice for real time system where new process arrive in each round.

In [31] assumed that RR is not suitable for real time system because it cause large waiting time and context switches. So they proposed a new RR with highest response ratio next (RRHRRN) scheduling algorithm, it depends on highest response ration (HRR) tool for selecting the process, they compared their work with DQRR in previous work and their idea had better results.

Proposed Idea and Implementation

This work proposed a hybrid algorithm; called Genetic Weighted Round robin (GWR) algorithm that combines GA with weighted average and Round Robin scheduling algorithm. Using GA would help in reordering the processes to minimize average waiting time with fitness function as minimum waiting time, and using weighted average to calculate the time slice for the processes in the ready queue. The following is the main steps for the proposed algorithm:

1- Randomly generate bursts for processes, with weight for each process.
2- Re arrange the process in two different ways to get two parents, as in table (2).
3- Use GA to re-order, as in table (3):
a. Use crossover
b. Use mutation
4- Use weighted Average to calculate the time slice.

However, generating data have been done in order to simulate the proposed idea. The simulated data contains ten processes with different burst times and weights; where the weight represents importance if each process. Chromosomes are generated from ten processes in random different orders. Table (1) contains processes' burst and its weight with final calculated value time slice=43,

to be used in Round Robin scheduling algorithm. Table (2) represents the two main chromosomes as parents. While table (3) contains the details for each experiment named order with a number; like order1 means Order one crossover with Insert Mutation, and so on. Hence, table (4) contains all the details about waiting time, for each process. Figure (3) describe the waiting time for all processes in each order type. Figures (4) to (13) describe the waiting time for each process alone in all orders. Figure (14) shows the average waiting time for each order. Finally, table (5) and figure (14) shows the results related to the Turn around Time for all processes.

Process	Burst	Weight	Burst × Weight	
P_1	60	5	300	
P_2	5	1	5	
P_3	13	2	26	
P_4	2	1	2	
P_5	10	1	10	
P_6	19	2	38	
P_7	25	3	75	
P_8	45	5	225	
P_9	72	7	504	
P_{10}	34	4	136	
Sum	285	31	1321	43

Table (1): Processes' Burst and Weight

Parent1	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
	60	5	13	2	10	19	25	45	72	34
Parent2	P_4	P_2	P_5	P_3	P_6	P_7	P_{10}	P_8	P_1	P_9
	2	5	10	13	19	25	34	45	60	72

Table (2): Parents for Genetic Algorithm

Order-No	Order-Name Details		
Order1	Order one crossover	Insert Mutation	Child1
Order2	Order one crossover	Insert Mutation	Child2
Order3	Order one crossover	Swap Mutation	Child1
Order4	Order one crossover	Swap Mutation	Child2
Order5	Order one crossover	Inversion Mutation	Child1
Order6	Order one crossover	Inversion Mutation	Child2
Order7	Order one crossover	Scramble Mutation	Child1
Order8	Order one crossover	Scramble Mutation	Child2
Order9	Cycle Crossover	Insert Mutation	Child1
Order10	Cycle Crossover	Insert Mutation	Child2
Order11	Cycle Crossover	Swap Mutation	Child1
Order12	Cycle Crossover	Swap Mutation	Child2
Order13	Cycle Crossover	Inversion Mutation	Child1
Order14	Cycle Crossover	Inversion Mutation	Child2
Order15	Cycle Crossover	Scramble Mutation	Child1
Order16	Cycle Crossover	Scramble Mutation	Child2
Order17	Partially Mapped Crossove	Insert Mutation	Child1
Order18	Partially Mapped Crossove	Insert Mutation	Child2
Order19	Partially Mapped Crossove	Swap Mutation	Child1
Order20	Partially Mapped Crossove	Swap Mutation	Child2

Order21	Partially Mapped Crossove	Inversion Mutation	Child1
Order22	Partially Mapped Crossove	Inversion Mutation	Child2
Order23	Partially Mapped Crossove	Scramble Mutation	Child1
Order24	Partially Mapped Crossove	Scramble Mutation	Child2

Table (3): Generations' Details

Order-No	W-P1	W-P2	W-P3	W-P4	W-P5	W-P6	W-P7	W-P8	W-P9	W-P10	m-Wait	vg-Wati
Order1	225	0	48	95	97	107	126	223	194	61	1176	117.6
Order2	225	0	60	5	50	73	92	223	194	117	1039	103.9
Order3	225	232	43	90	92	102	121	223	194	56	1378	137.8
Order4	225	0	58	192	48	71	90	223	194	115	1216	121.6
Order5	196	0	51	15	5	64	83	194	213	17	838	83.8
Order6	225	0	17	5	7	30	189	223	194	135	1025	102.5
Order7	196	0	5	77	141	79	52	194	213	18	975	97.5
Order8	225	232	55	0	45	68	87	223	194	112	1241	124.1
Order9	194	43	139	235	91	160	135	211	213	101	1522	152.2
Order10	223	198	86	99	126	136	101	240	194	203	1606	160.6
Order11	194	43	136	235	48	117	92	211	213	58	1347	134.7
Order12	223	198	86	153	124	134	99	240	194	203	1654	165.4
Order13	194	43	128	91	118	141	93	211	213	203	1435	143.5
Order14	223	86	190	91	137	113	93	240	194	203	1570	157
Order15	194	43	137	48	150	118	93	211	213	160	1367	136.7
Order16	223	198	62	75	77	0	87	240	194	203	1359	135.9
Order17	196	13	0	95	97	107	126	194	213	18	1059	105.9
Order18	194	86	93	91	227	106	125	240	211	150	1523	152.3
Order19	225	13	0	235	95	105	124	223	194	18	1232	123.2
Order20	194	43	91	225	227	104	123	240	211	148	1606	160.6
Order21	196	13	0	149	139	120	95	194	213	18	1137	113.7
Order22	194	34	19	32	227	0	82	211	213	107	1119	111.9
Order23	196	13	0	52	184	97	116	194	213	18	1083	108.3
Order24	194	43	222	235	48	101	120	240	211	145	1559	155.9

Table (4): Generations' Weight-Time

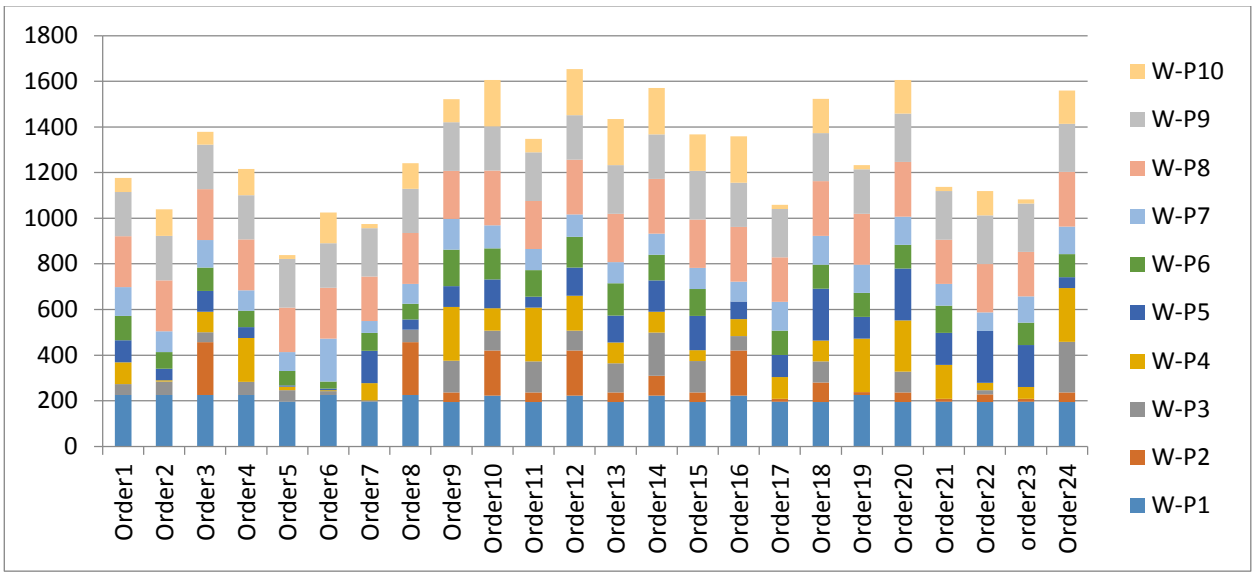

Figure (3): The Waiting Time for all Processes in Each Order

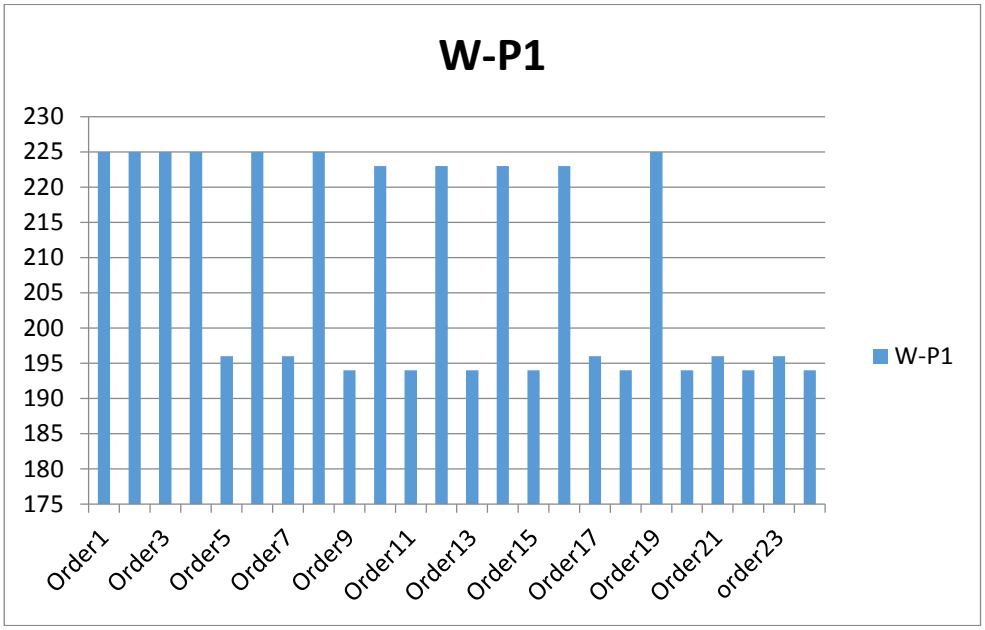

Figure (4): Waiting Time for Process-1 in all Orders

Figure (4) describes the waiting time for process P1 in all order types or chromosomes, as we can see the minimum waiting time was in orders: 9, 11, 13, 15, 18, 20, 22, and 24 with value194ms. So those chromosomes were the best between others

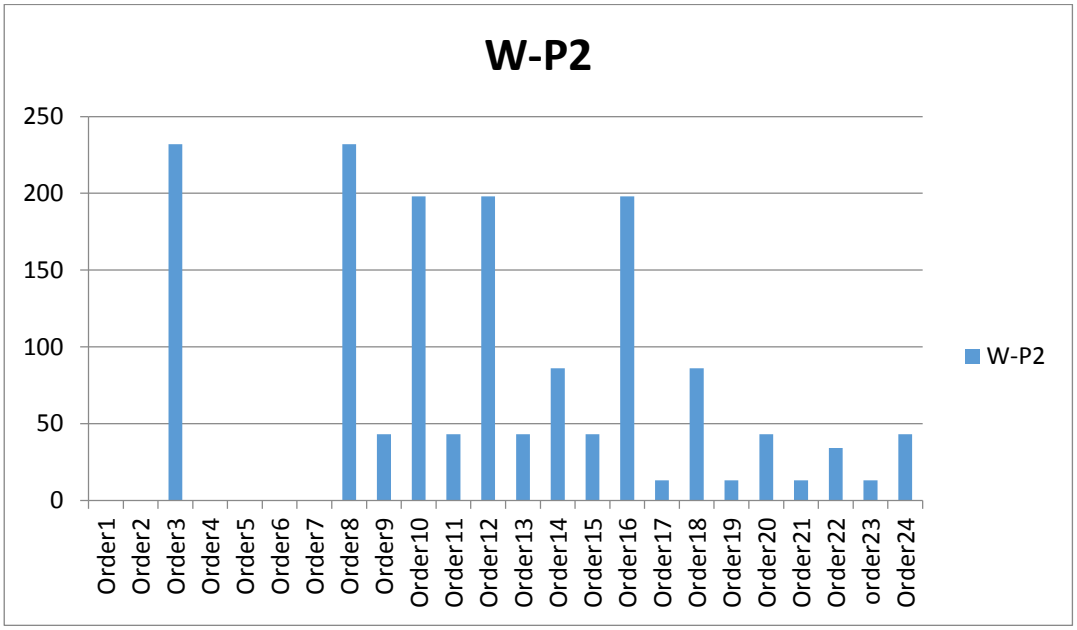

Figure (5): Waiting Time for Process-2 in all Orders

Figure (5) describes the waiting time for process P2 in all order types or chromosomes, as we can see the minimum waiting time was in orders: 1, 2, 4, 5, 6, and 7 with value 0ms. So those chromosomes were the best between others.

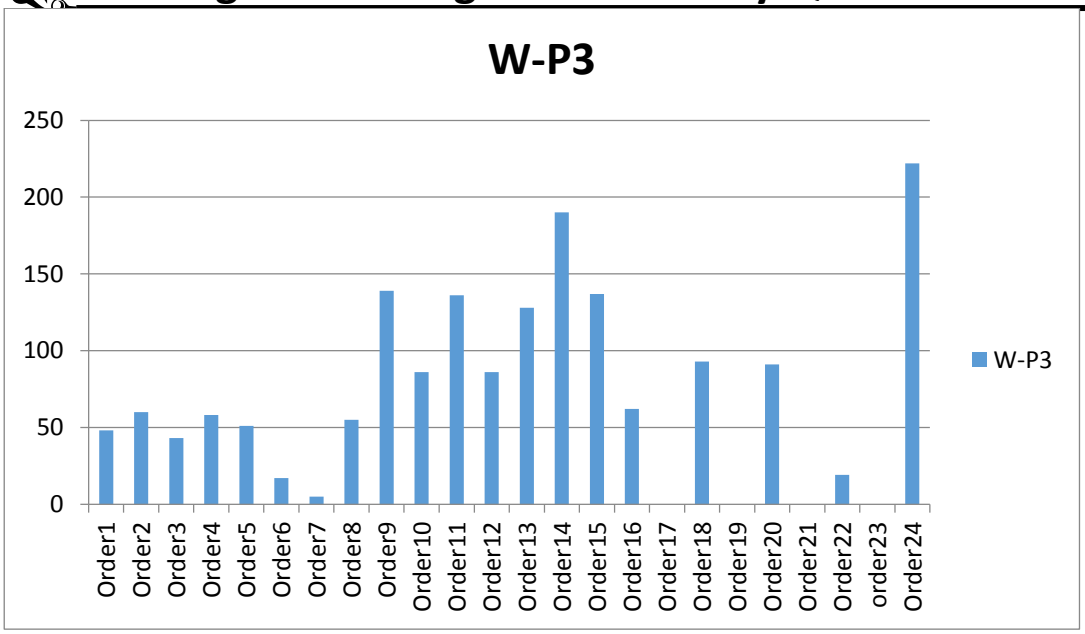

Figure (6): Waiting Time for Process-3 in all Orders Figure (6) describes the waiting time for process P3 in all order types or chromosomes, as we can see the minimum waiting time was in orders: 17, 19, 21, and 23 with value0ms. So those chromosomes were the best between others.

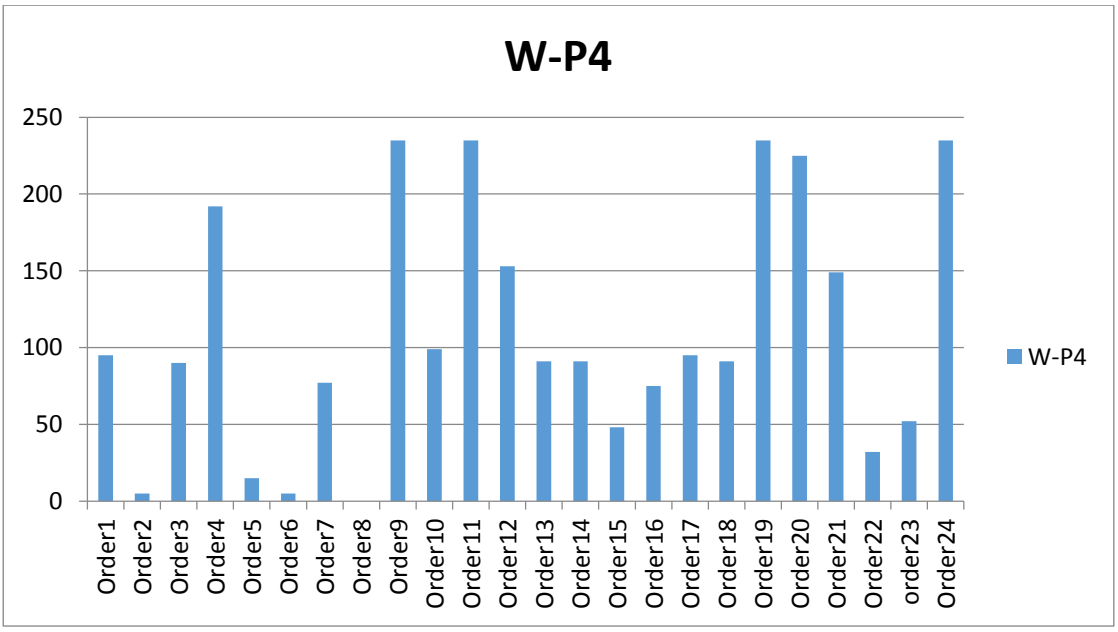

Figure (7): Waiting Time for Process-4 in all Orders Figure (7) describes the waiting time for process P1 in all order types or chromosomes, as we can see the minimum waiting time was in order: 9 only with value 0ms. So this chromosome was the best between others.

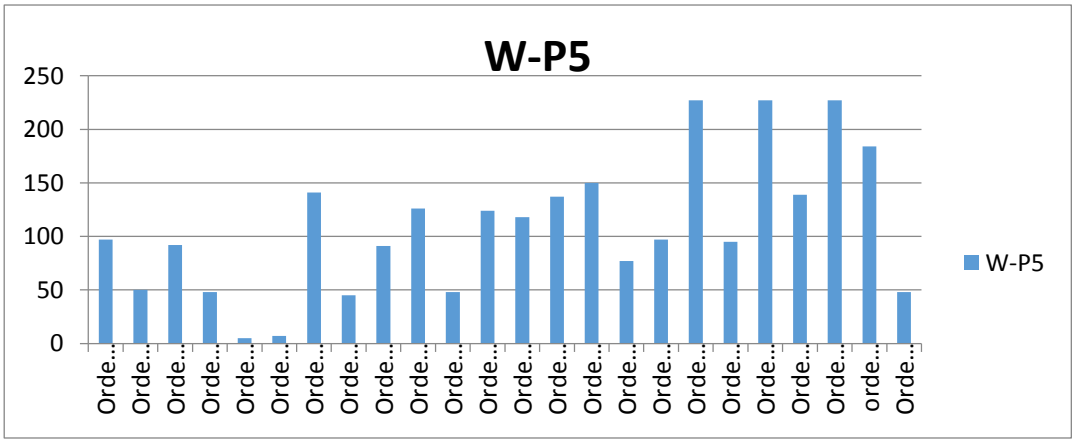

Figure (8): Waiting Time for Process-5 in all Orders Figure (8) describes the waiting time for process P5 in all order types or chromosomes, as we can see the minimum waiting time was in order: 5 only with value 5ms. So this chromosome was the best between others.

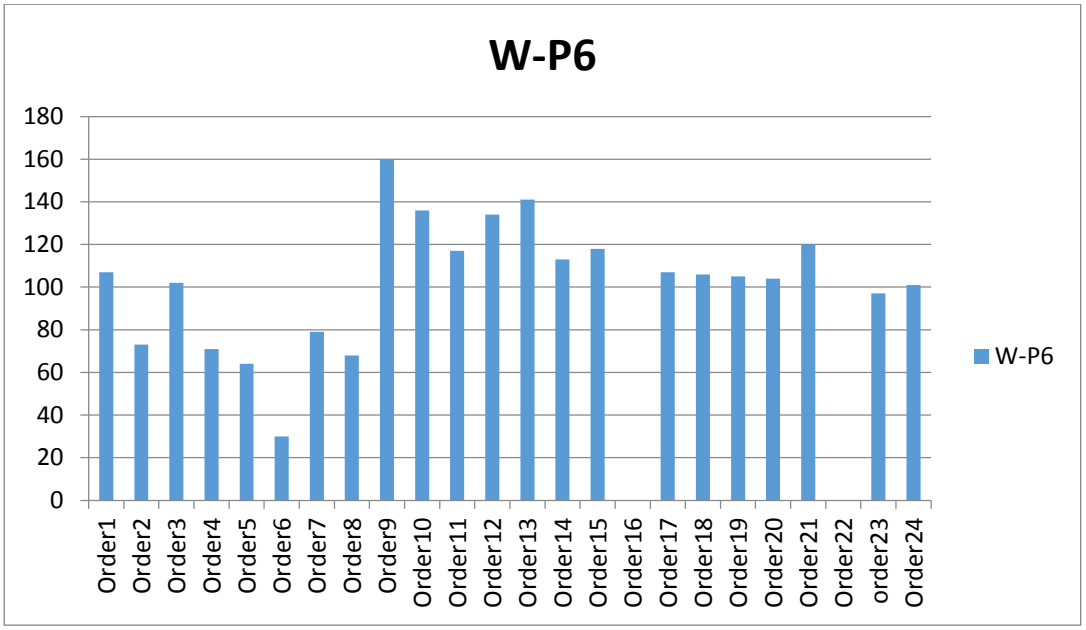

Figure (9): Waiting Time for Process-6 in all Orders

Figure (9) describes the waiting time for process P6 in all order types or chromosomes, as we can see the minimum waiting time was in orders: 16 and 22 with value 0ms. So those chromosomes were the best between others.

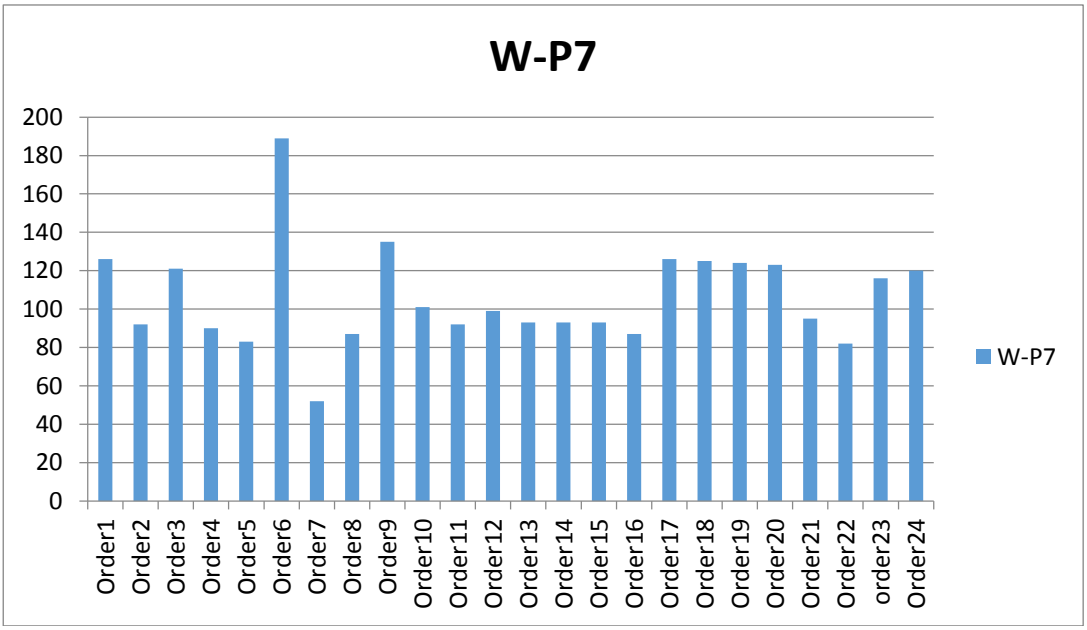

Figure (10): Waiting Time for Process-7 in all Orders Figure (10) describes the waiting time for process P7 in all order types or chromosomes, as we can see the minimum waiting time was in order: 7 with value 52ms. So this chromosome was the best between others.

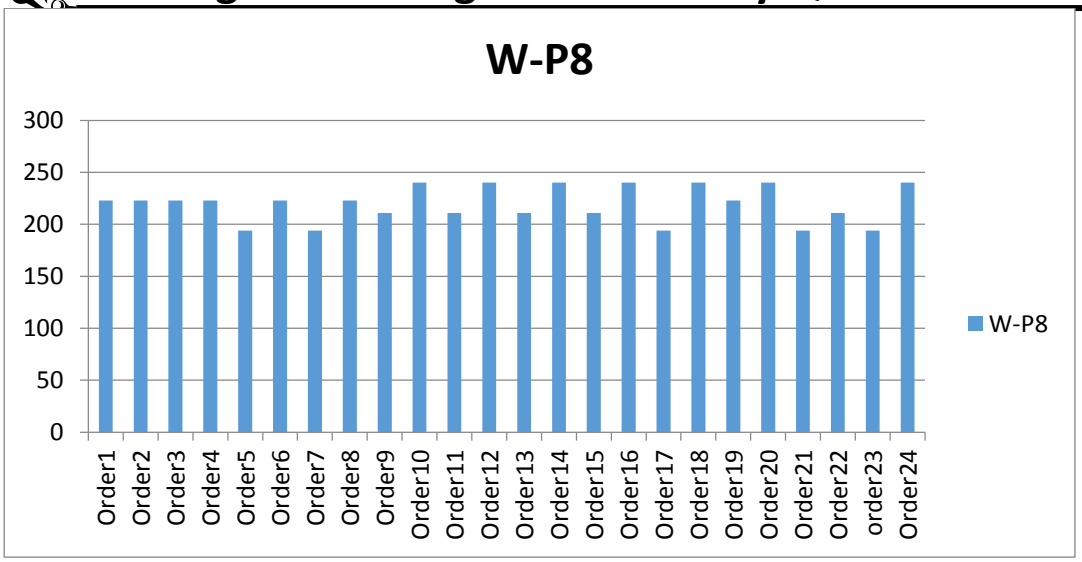

Figure (11): Waiting Time for Process-8 in all Orders Figure (11) describes the waiting time for process P8 in all order types or chromosomes, as we can see the minimum waiting time was in orders: 5, 17, 21, and 23 with value194ms. So those chromosomes were the best between others.

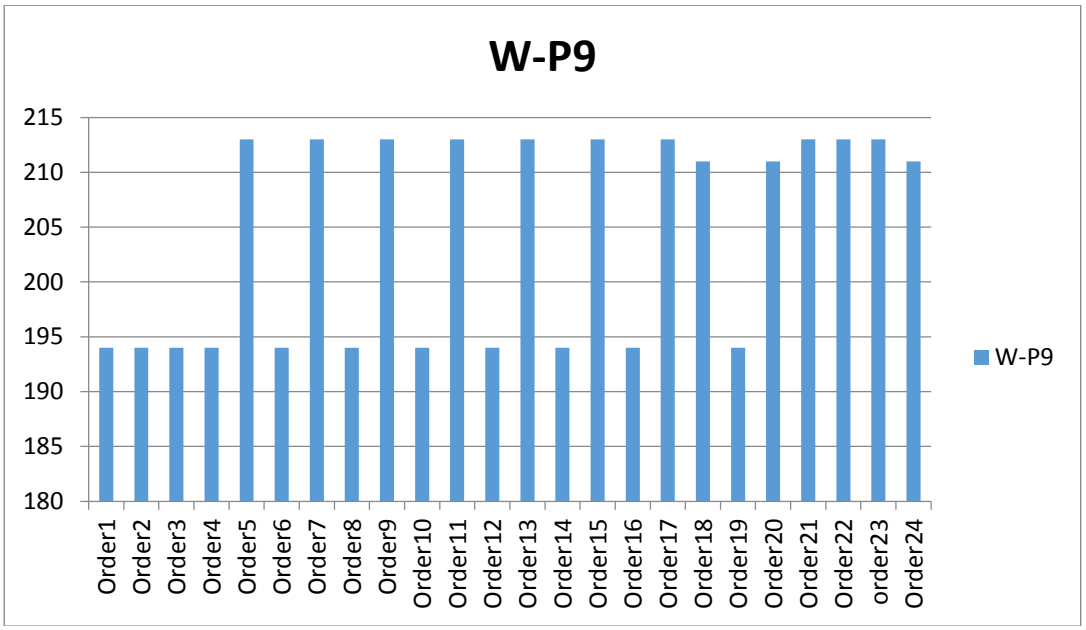

Figure (12): Waiting Time for Process-9 in all Orders Figure (12) describes the waiting time for process P9 in all order types or chromosomes, as we can see the minimum waiting time was in orders: 1, 2, 3, 4, 6, 8, 10, 12, 14, 16, and 19with value194ms. So those chromosomes were the best between others.

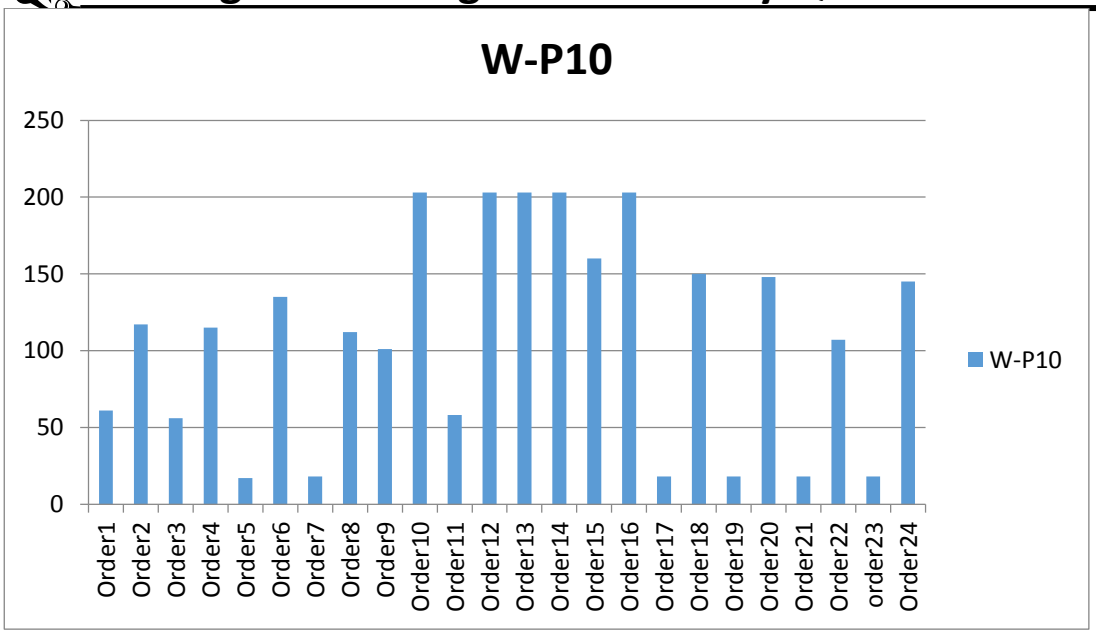

Figure (13): Waiting Time for Process-10 in all Orders Figure (13) describes the waiting time for process P10 in all order types or chromosomes, as we can see the minimum waiting time was in order: 5 with value17ms. So this chromosome was the best between others.

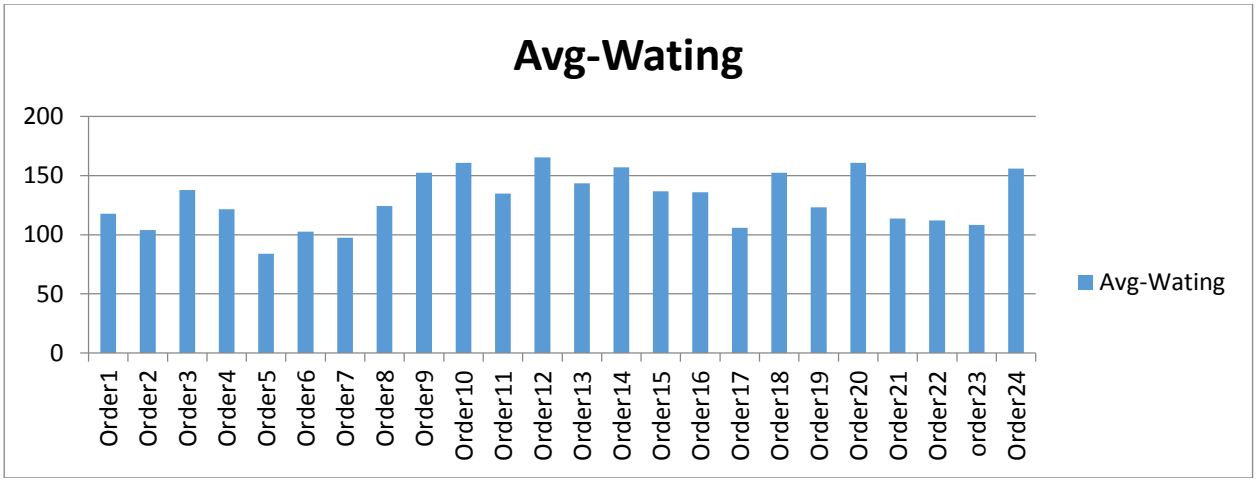

Figure (14): Average Waiting Time for all Process in all Orders Figure (14) shows the average waiting time for the ten processes generated randomly in this paper.

Order-No	AT-P	AT-P	AT-P	AT-P	AT-P	AT-P	AT-P	AT-P	AT-P	AT-P	um-TA	vg-TA
Order1	285	5	61	97	107	126	151	268	266	95	1461	146.1
Order2	285	5	73	7	60	92	117	268	266	151	1324	132.4
Order3	285	237	56	92	102	121	146	268	266	90	1663	166.3
Order4	285	5	71	194	58	90	115	268	266	149	1501	150.1
Order5	256	5	64	17	15	83	108	239	285	51	1123	112.3
Order6	285	5	30	7	17	49	214	268	266	169	1310	131
Order7	256	5	18	79	151	98	77	239	285	52	1260	126
Order8	285	237	68	2	55	87	112	268	266	146	1526	152.6
Order9	254	48	152	237	101	179	160	256	285	135	1807	180.7
Order10	283	203	99	101	136	155	126	285	266	237	1891	189.1

Order11	254	48	149	237	58	136	117	256	285	92	1632	163.2
Order12	283	203	99	155	134	153	124	285	266	237	1939	193.9
Order13	254	48	141	93	128	160	118	256	285	237	1720	172
Order14	283	91	203	93	147	132	118	285	266	237	1855	185.5
Order15	254	48	150	50	160	137	118	256	285	194	1652	165.2
Order16	283	203	75	77	87	19	112	285	266	237	1644	164.4
Order17	256	18	13	97	107	126	151	239	285	52	1344	134.4
Order18	254	91	106	93	237	125	150	285	283	184	1808	180.8
Order19	285	18	13	237	105	124	149	268	266	52	1517	151.7
Order20	254	48	104	227	237	123	148	285	283	182	1891	189.1
Order21	256	18	13	151	149	139	120	239	285	52	1422	142.2
Order22	254	39	32	34	237	19	107	256	285	141	1404	140.4
order23	256	18	13	54	194	116	141	239	285	52	1368	136.8
Order24	254	48	235	237	58	120	145	285	283	179	1844	184.4

Table (5): Generations' TAT-Time

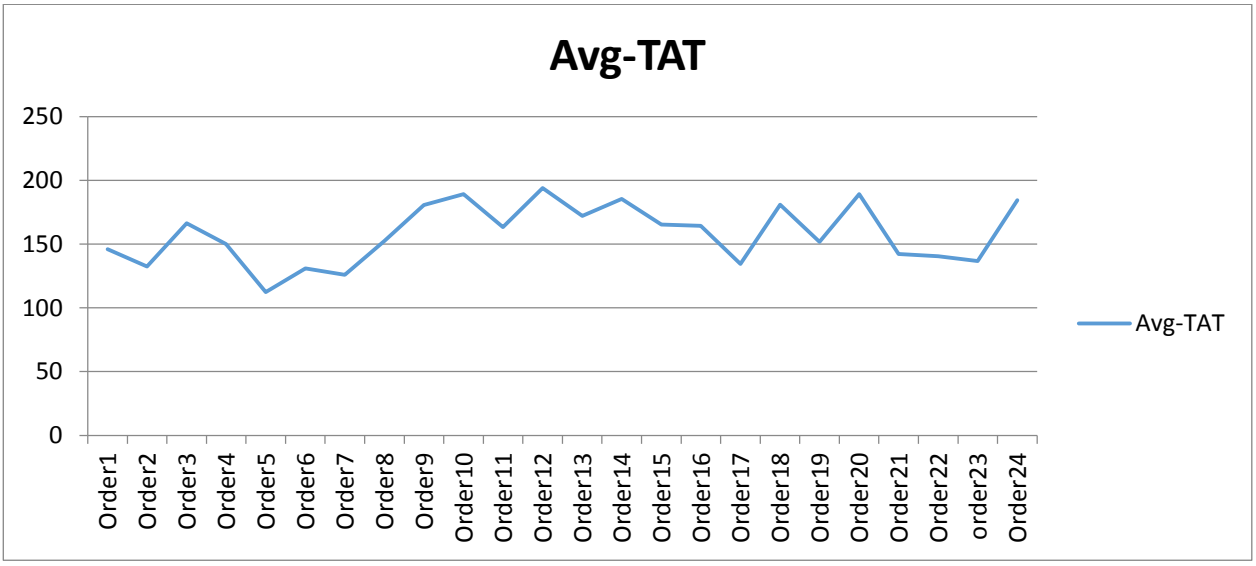

Figure (14): Average Turnaround Time for all Process in all Orders

Conclusion and Future Work

This paper proposed (GWR) algorithm for RR's optimization by minimizing processes' average waiting time, which is the fitness function. Two main chromosomes are designed to represent random orders of processes bursts' in ready queue. Multiple chromosomes are generated from using different crossover and mutation operations. The experiments shows that chromosome named (order-5) was the fittest chromosome because it gave the minimum average waiting time, where the time quantum was calculated once at the begin using weighted average method that means each process had a weight of importance.

For future work, GA should be used in each round, the time slice should be dynamic rather than static, and if remaining burst time was less than time slice value then no need for switching to another process.

References

[1] Jain Sh. and Jain Sa. A Review Study on the CPU Scheduling Algorithm Department of Computer Science, Gujarati Profession Institute, India, IJARCCE International Journal of Advanced research in computer and Communication Engineering, ISSN (online) 2278-1021, Vol.5, Issue 8, 2016.

[2] Hiranwal S. and Roy K. Adaptive Round Robin Scheduling using Shortest Burst Approach Based on Smart Time Slice Computer Science and Engineering, International Journal of Data Engineering (IJDE), Volume 2, Issue. 3, 2012.
[3] Datta K., Jana M., Mazumdar A An Effective Dynamic Quantum Round Robin (EDQRR) CPU Scheduling Algorithm Halida Institute of Technology, ICARE Complex, India, International Journal of Computer Applications (0975-8887), Vloume 130, No.6, 2015.
[4] Kaur A. and Kinger S. A Survey of Resource Scheduling Algorithms in Green Computing Department of Computer Science and Engineering, SGGSWU, India, IJCSIT International Journal of Computer and Information Technologies, Vol.5, ISSN 0975-9646, 2014.
[5] Noon A., Kalakech A., and Kadry S. A new Round Robin based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average Faculty o Computer Science, Arts Sciences and Technology University, Lebanon, IJCSI International Journal of Computer Science Issues, Vol.8, Issue 3., No.1, ISSN (online) 1694-0814, 2011.
[6] Lenka R. and Ranjan P. A 2LFQ Scheduling with Dynamic Time Quantum using Mean Average CSED,MNNIT, India, International journal of Computer Applications (0975-8887), Volume 47-No.23, 2012.
[7] Rajput N. and Kumar A. A Task set Based Adaptive Round Robin (TARR) Scheduling algorithm for improving performance Deptt of Computer Science & Engineering, IEC College of Engineering & Technology, India, 1st International conference on futuristic trend in computational analysis and knowledge management, 2015.
[8] Mishra M. and Khan A. An Improved Round Robin CPU Scheduling Algorithm Department of Infromation Technology, Haramaya University, Ethiopia, Journal of Global Research in Computer Science, Vol. 3, No. 6, ISSN-2229-371X, 2012.
[9] Kathuria S., Singh P., Tiwari P., and Prashant A Revamped Mean Round Robin (RMRR) CPU Scheduling Algorithm International Journal of Innovative Research in Computer and Communication Engineering, ISSN2320-9801, Vol.4, Issue 4, 2016.
[10] Dash A., Sahu S., and Samantra S An Optimized Round Robin CPU Scheduling Algorithm with Dynamic Time Quantum Department of Computer Science, NIST, India, International Journal of Computer Science, Engineering and Information Technology (IJCSEIT), Vol.5,No.1,2015.
[11] Gaba V. and Prashar A. Comparison of Processor Scheduling Algorithms using Genetic Approach Department of Computer Science and Engineering, Haryaa college of Technology and Management, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 8, ISSN 2277 128X,2012.
[12] Mansour M. A Genetic Algorithm for Scheduling n Jobs on a Single Machine with a Stochastic Controllable Processing, Tooling Cost and Earliness-Tardiness Penalties Industrial Engineering Department, Zagazig University, Egypt, American J. of Engineering and Applied
[13] Sawant S. A Genetic Algorithm Scheduling Approach for Virtual Machine Resources in a Cloud Computing Environment master's Projects, Computer Science Department, San Jose State
[14] Vaghefinezhad S. and Wong K. A Genetic Algorithm Approach for Solving a Flexible Job Shop Scheduling Problem Manufacturing department, Universiti Teknologi Malaysia, IJCSI, International Journal of Computer Science, Vol. 9, Issue 3, No. 1, ISSN (Online): 1694-0814, May
[15] Harmanani H., Drouby F., and Ghon S. A Parallel Genetic Algorithm for the Open-Shop Scheduling Problem Using Deterministic and Random Moves Computer Science and Mathematics Department, Lebanese American University, Lebanon, 2010.
[16] Rani P. and Nagpal S. Dynamic Process Scheduling and Sequencing Using Genetic Algorithm CSE Department, Geeta College of Engineering Naultha, Panipat, IOSR Journal of Computer Engineering (IOSR-JCE), e-ISSN: 2278-0661, p-ISSN:2278-8727, Volume 16, Issue 3, 2014.
[17] Abdoun O., Abouchabaka J., and Tajani C. Analyzing the Performance of Mutation Operators to Solve the Travelling Salesman Problem Ibn Tofail University, Morocco, 2012.

[18] Li Y. and Chen Y. A genetic Algorithm for Job-Shop Scheduling Transportation Management College, Dalian Martime University, China, Journal of Software, Vol. 5, No. 3, 2010.
[19] Prashanth S. and Andresen D. Using Implicit Fitness Functions for Genetic Algorithm-based Agent Scheduling Department of Computing and Information Sciences, Kansas State University,
[20] Obaid O., Ahmad M., Mostafa S., and Mohammed M Comparing performance of Genetic Algorithm with Varying Crossover in Solving Examination Timetabling Problem College of graduate studies, University Tenaga National, Malaysia, Journal of Emerging Trends in Computing and Information Sciences, VoL. 3, NO. 10, ISSN 2079-8407, 2012.
[21] Lakshmi R. and Vivekanandan K. A Novel Hybrid Crossover Operator for Genetic Algorithm Pondicherry University, India, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 3, ISSN: 2277 128X, 2014.
[22] Al-Angari N. and AlAbdullatif A. Multiprocessor Scheduling Using Parallel Genetic Algorithm Computer Science Department, College of Computer and Information Science, King Saud University, Saudi Arabia, IJCSI International Journal of Computer Science Issues, Vol.9, Issue 4, No3, ISSN (Online): 1694-0814, 2012.
[23] Diregar M. A New Approach to CPU Scheduling Algorithm: Genetic Round Robin Informatics Department, State Islamic University Sunan Kalijaga,Indonesia, International Journal of Computer Applications (0975-888), Vol 47, No.19, 2012.
[24] Dhumal R., Maktum T., and Ragha L Dynamic Quantum based genetic Round Robin Algorithm Computer Engineering, TEC, India, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 3, Issue 3, ISSN 2278-1021, 2014.
[25] Wadhwa V. and Sindhwani P. N.C. College of Engineering, India, Genetic Algorithm Approach for Optimal CPU Scheduling IJCST International Journal of Computer Science and Technology, Vol. 2,Issue 2, ISSN 2229-4333, 2011.
[26] Arslan T. and Tian L. Genetic Algorithm for Energy Efficient Device Scheduling in real-Time Systems School of Engineering and Electronics, Edinburgh University, UK, IEEE, 2003.
[27] Kumar R., Kumar R., Gill S., and Kaushik A. Genetic Algorithm Approach to Operating Systems Process Scheduling Problem Reader Department of Computer Science and Application, Kurkshetra University, India, International Journal of Engineering Science and Technology, Vol.
[28] Sharma M., Sindhwani P., and Maheshwari V. Genetic Algorithm Optimal Approach For Scheduling Process In Operating System Shobhit University, International Journal of Engineering Research & technology (IJERT), Issn:2278-0181, Vol.2, Issue 5, 2013.
[29] Shakhare S. and Ali M. Genetic Algorithm Based Adaptive Scheduling Algorithm for Real Time Operating Systems Department of Information Technology, VIIT, India, International Journal of Embedded Systems and Applications (IJESA), Vol.2, No.3, 2012.
[30] Muraleedharan A., Antony N., and Nandakumar R. Dynamic Time Slice Round Robin Scheduling Algorithm with Unknown Burst Time Department of Computer Science & IT, Amrita School of Arts and Sciences Kochi, India, Indian Journal of Science and Technology, Vol9(8),
[31] Behera H., Swain B., Parida A., and Sahu G. A New Proposed Round Robin with Highest Response Ratio Next (RRHRRN) Scheduling Algorithm for Soft Real Time Systems International Journal of Engineering and Advanced Technology (IJEAT), ISSN 2249-8958, Vol. 1, Issue 3,