ISSN 1991- 8690 الترقيم الدولي 8690 - 1991

Enhancement of CPU Scheduling using Genetic Algorithm

Khawla Hussein Ali

Dept. of Computer Science - College of Education - University of Basrah

Abstract:

CPU scheduler makes a sequence of "moves" that determines the interleaving of processes to be allocated by a CPU. Programs use synchronization to prevent "bad moves", but scheduling choices appear to the program to be non deterministic, the scheduler's moves are dictated by a "scheduling policy ". Different CPU scheduling algorithms have different properties and may favor one class of processes over another. Many criteria have been suggested for comparison can make a substantial difference in a determination of the best algorithm such as CPU utilization, Throughput, turnaround time, waiting time, response time. Because such a wide variety of scheduling algorithms are available (such as FCFS, SJF, SRT, RR), so we present a genetic algorithm for minimize the turnaround time and comparison with these algorithms, the results are good.

Keywords: *CPU scheduling algorithms, Genetic algorithm.*

تعزيز كفاءة جدولة المعالج باستخدام الخوارزمية الجينية

خولة حسين علي

جامعة البصرة – كلية التربية - قسم علوم الحاسبات

الخلاصة :

ان جدولة المعالج CPU تعني سلسلة من التحركات التي يقوم بها المعالج بين المعالجات Processesالموجودة في الطابور الجاهز وذلك لحصول المعالجة processعلى المعالج CPU ليتم تنفيذها . توجد خوارزميات عديدة تحدد الجدولة واي من المعالجات يتم اختيارها ليتم تنفيذها علما انها موجودة جميعا في الطابور الجاهز بأنتظار التنفيذ من قبل المعالج CPU منها FCFS, SJF, SRT, RR . توجد معايير مختلفة لتحديد اي من الخوارزميات هي الافضل , من هذه المعايير زمن الانتظار, الزمن الدوري, زمن الاستجابة , . في هذا البحث تم تقديم الخوارزمية الجينية لاستخدامها في جدولة المعالجة لتقليل زمن الدوري للمعالجات . تم مقارنة الخوارزمية الجينية مع بقية الخوارزميات وكانت النتائج جيدة لاسيما اذا كان عدد المعالجات كبيرا.

كلمات المفاتيح : خوارزميات جدولة المعالج ، الخوارزمية الجينية .

1- Introduction:

CPU scheduling is the basis of multiprogrammed operating systems [1]. By switching the CPU among processes, the operating system can make the computer more productive. The objective of multiprogramming is to have some process running at all times, in order to maximize CPU utilization. In a uniprocessor system, only one process may run at a time, any other processes must wait until the CPU is free and can be rescheduled. The scheduling is fundamental of operating system function. Typically scheduling problems are NP Hard problems(in complexity theory, a decision problem is one whose output is "yes" or "no" otherwise known as a Boolean Value . NP is a set of decision problems, if the answer to an NP problem is "yes" there is a proof of the answer that can be checked in reasonable or polynomial time, so the problem can described as NP hard if no polynomial or "fast" algorithm for its solution exits). [2]. It is necessary to find out robust and flexible solution for real world scheduling problem. Genetic algorithms (GAs) were first proposed by John Holland in 1960s [3]. The GA is a heuristic search technique that simulates the processes of natural selection and evolution. GA is a promising global optimization technique [4, 6]. A genetic algorithm has the capability to find the optimal job sequence which is to be allocated to the CPU. This research proposes a genetic algorithm based technique to find out the optimal job (process) sequence. We will examine that whether genetic algorithm based scheduling will maximize the operating system performance. The algorithm starts with a population which is consists of several solution to the optimization problem. A member of population is called an individual. a fitness value is associated with each individual. Each solution in the population is encoded as a string of symbols. These symbols are known as genes and the solution string is called a chromosomes .Several pair of parents in the population mate to produce offspring by applying the genetic operator crossover. Selection of parents is done by repeated use of a choice function. A member of individuals and offsprings are passed to a new generation such that the number of individual in the new population is the same as old population. A selection function determines which string forms the population in the next generation. Each serving string undergoes inversion with a specified probability.

2. Problem Description:

In order to schedule the process in a uniprocessor operating system, let us suppose that there are N processes [1, 2, 3, ….., N] and these processes have their static burst time processed on a single CPU at a time. The aim is to minimize the time of turnaround of CPU.

2.1 Assumption for process scheduling problem

1- There is a pool of ready processes waiting for the allocation of CPU [1].

2-the job of the proposed scheduler (using genetic algorithm) is to distribute the CPU to different processes fairly so that performance criteria must be optimize i.e. minimum average turnaround time.

3-turnaround time: from the point of view of a particular process, the important criteria is how long it takes to execute that process. The interval from the time of submission of a process to the time of completion is the turnaround time (in milliseconds).

3. The proposed GA-Based Algorithm:

Generally a genetic algorithm consists of five basic components as follows [5, 6]:

1- An encoding method that is a genetic representation (genotype) of solutions to the program.
2- Create an initial population of chromosomes.
3- An evaluation function (fitness), rating solution in terms of their fitness and a selection mechanism.
4- The genetic operators (crossover & inversion) that alter the genetic composition of offspring during separation.

5- Values for the parameter of genetic algorithm.

3.1 Genotype

In the GA based algorithms each chromosome corresponds to a solution to the problem. The genetic representation of the individual is called Genotype. Many Genotypes have been proposed in [3]. In this research we represent a chromosome with the combination of seven decimal numbers, here sequence of seven decimal numbers is the number of processes, E.g. (2, 5, 6, 3, 4. 7. 1) sequences of seven processes.

3.2 Initial Population

The genetic algorithm starts with a set of chromosomes called initial population. Most algorithms generate initial population randomly. in the simulation initial population with size of POPSIZE is generated with function [4, 8].

3.3 Fitness Function

The main objective of GA is to find an optimal solution, in order to find the optimal solution, a fitness function must be devised for the problem under consideration i.e., fitness. For a particular chromosome, the fitness function returns a single numerical fitness, which is supposed to be proportional to the ability of the individual which that chromosome represent. In this research the minimum average turnaround time is fittest as compared to other**.** The fitness individuals have for capability to participate in the reproduction cycle. The fitness function of a schedule S_j is given by (1, 7):

$$fitness(S_j) = \frac{\sum_{i=1}^{N} TRi}{N} \qquad \ldots\ldots(1)$$

(i=1, 2, 3,….N)

TR_1 is the turnaround time of process J_1, TR_2 for J_2 and so on. N is the total number of processes.

3.4 Selection

The selection process used here is based on spinning the roulette wheel. Which each chromosome in the population has slot size in population to its fitness. Each time we require an offspring, a simple spin of the weighted roulette wheel gives a parent chromosome, the probability Pi that a parent Sj is given by (2):

$$P_i = \frac{F(S_i)}{\sum_{j=1}^{POPSIZE} F(S_J)} \qquad \ldots\ldots\ldots \qquad (2)$$

Where F (S_i) is the fitness of chromosome S_i.

3.5 Crossover

Crossover is generally used to exchange portions between strings. Several crossover operators are descr**i**bed in the literature [4**].** Crossover is not always affected, the invocation of the crossover depends on the probability of the crossover (pc). Modified crossover operator i.e., **(**Mcrossover**)** is an extension of the one point crossover for permutation problems. A cut position is chosen at random on the first parent chromosomes. Then an offspring an offspring (as shown in Fig. 1) is created by appending the second chromosome to the initial part of the first parent(before the cut point) and eliminating the duplicates.

Parent 1:	1	3	4	|	5	7	2	6
Parent 2:	1	4	5		7	6	2	3
Offspring	1	3	4		5	7	6	2

Figure 1: Modified Crossover

3.6 Inversion(Muting Process)

Inversion is a process in which the order of two gene position swapped with respect to each other. in inversion operator i.e., two points are selected along the length of the chromosome, the chromosome is cut at those points and the end points of the section cut, gets reversed (swapped) . to make it clear, we consider a chromosome of length 7, where two inverse points are selected randomly (the points are 2 and 4 denoted by _ character as shown in Fig. 2)

Offspring	2	3	6	1	5	4	7
Offspring	2	1	6	3	5	4	7

Figure 2. inversion (muting)

3.7 Replacement Strategy

When genetic operators(Modified crossover, inversion) are applied on the selected parents S_1, S_2 one new chromosome is generated. This chromosome is added to the existing population. After that fitter chromosomes are selected from the new population. This process remain continue until we not get optimal solution.

3.8 Termination Condition

There are multiple choices for termination condition; max number of generation, algorithm convergence, equal fitness for fittest chromosomes in respective iteration. In this research we use algorithm convergence termination algorithm.

4. Skelton of proposed GA- Based Algorithm:

```
Begin
    Initialize population (randomly generated);
    Fitness evaluation;
    Repeat
        Selection (Roulette Wheel Selection)
        Modified Crossover;

    Fitness Evaluation;
        Elitism replacement with filteration;
    Until the end condition is satisfied;
        Return the fittest solution found;
End.
```

Our proposed GA-Based algorithm starts with a generation of individuals. A certain fitness function is used to evaluate the fitness of each individual. Good individuals survive after selection according to the fitness of individuals. Then the survived individuals reproduce offspring through crossover and inversion operators. This process iterates until termination condition is satisfied.

4.1 pseudo-code of GA Based Scheduling

(1) Begin
(2) generate random population of candidate sequence of processes to be allocated to CPU.
(3) evaluate each individual using fitness function (fitness) that minimize turnaround time.
(4) WHILE NOT finished Do
BEGIN /* Produce New generation */
(5) FOR Population_size / 2 DO
BEGIN / * reproduction cycle */
(6) select two individual from the old population for mating .
(7) recombine the two individuals by apply modified crossover and inversion Operator to give offspring.
(8) compute the fitness of offspring and insert the offspring into the new population
END
(9) if population has converged THEN
Finished = TRUE
END
END

5. Experimental Description:

Individual solutions are randomly generated to form an initial population. Successive generations of reproduction and crossover produce increasing numbers of individuals. The algorithm favors the fittest individuals, as parents will have more offspring in the next generation. To achieve minimum turnaround time the fitness function defined here is based on Round Robin (with quantum =2) algorithm. The algorithm was programmed in visual c++ version 2008.

There are 7 processes which are to be considered. The number of possible sequences are 7!. The total 20 sequence is selected out of 5040 for the 7 processes. Considering the number of processes as 7 and the crossover point is 2. Let us consider following two individual which are marked as fit to generate next generation.

3 1 2 7 4 5 6 and 1 4 5 6 7 3 2
After crossover 3 1 5 4 5 6

This offspring is not a valid sequence because process 7 is not there in new individual and process 5 appears twice such individual is discarded. So to avoid this type of individual after crossover, we are using the modified crossover. In the modified crossover we get proper sequence order individual. Let assume two individual which are marked as fit and use for the next generation.

3 1 2 7 4 5 6 and 1 4 5 6 7 3 2

After modified crossover 3 1 4 5 6 7 2

This individual is accepted because there is no repetition of any process in this sequence.

6. Simulation Results:

There are seven processes (1, 2, 3, 4, 5, 6, 7) which requires processing time (burst time) (17, 2, 26, 10, 16, 20, 17) . So the sequence of the processes may vary and also we analyze the GA by changing the process time of different processes as shown in the table 2. We are comparing FCFS (First Come First Service), SJF (Short Job First), RR (Round Robin) (quantum = 2) and GA, by apply the genetic operator's crossover, fitness function and inversion, we get the final population, and the process schedule, who have the minimum average turnaround time.

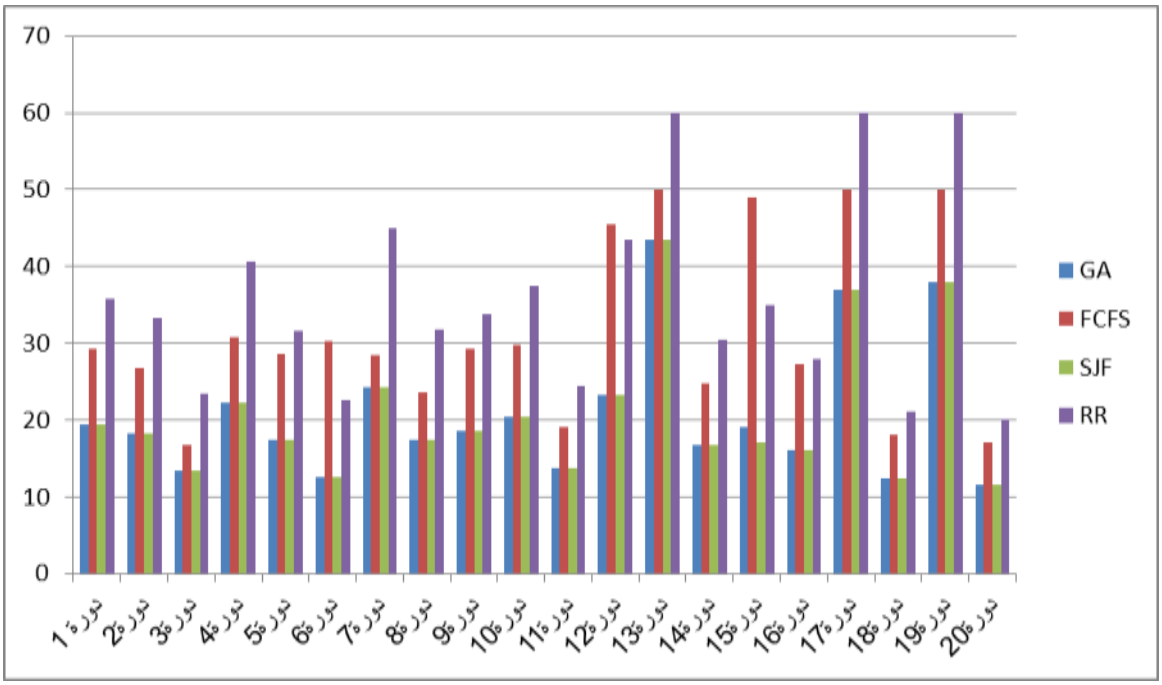

Table 1: parameters and strategies used for GA and CPU Scheduling

Parameter/Strategy	Setting
Population Size	14
Population type	Generational
Initialization	Random
Selection	Roulette wheel
Crossover	Modified crossover
Muting probability	0.1
Replacement strategy	Keep 80% best
Stopping strategy	No improvement for 30 generation
No. of process to be schedule	7
CPU Scheduling Criteria	Minimum average turnaround time

Table 2: comparison of results

No.	Burst time of process							GA	FCFS	SJF	RR=2
	P1	P2	P3	P4	P5	P6	P7				
1	17	2	26	10	16	20	17	19.4	29.2	19.2	33.7
2	15	8	19	2	40	18	10	18.1	26.7	18.2	33.2
3	8	2	15	6	35	7	12	13.2	16.8	13.4	23.4
4	2	25	28	5	32	3	11	22.2	30.7	22.2	40.7
5	19	4	19	2	19	2	4	17.3	28.6	17.4	31.6
6	20	6	20	3	3	3	4	12.5	30.2	12.6	22.6
7	18	3	18	15	18	2	3	24.2	28.4	24.2	45.0
8	16	2	15	8	15	2	2	18.5	29.2	18.6	33.7
9	21	2	21	4	32	4	3	16.0	27.4	16.0	28.7
10	8	11	35	4	30	4	3	17.3	23.6	17.4	31.7
11	9	4	17	3	50	3	4	13.8	19.0	13.8	24.4
12	36	10	20	4	20	4	5	23.2	45.5	23.2	43.5
13	35	30	36	3	39	10	3	23.2	45.5	23.2	43.5
14	5	15	12	25	5	5	4	16.7	24.7	16.8	30.4
15	10	50	25	4	8	4	4	19.0	50.7	19.0	35.4
16	2	25	28	5	32	6	6	22.1	30.7	22.2	40.7
17	20	25	40	8	36	8	7	37.5	50.5	37.5	72.0
18	2	10	15	12	4	2	3	12.3	18.0	12.4	21.6
19	22	40	35	2	38	2	2	38.4	58.0	38.4	73.8
20	4	2	25	3	23	3	2	11.5	17.0	11.6	20.0
Total Avg. tumaround Time								397.4	581.3	397.5	729.6
Total Mean Avg. tumaround Time								19.86	29.06	19.87	36.48

7. Conclusion and Future Work:

The simplicity of the methods used supports the assumption that GA's can provide a highly flexible and user-friendly, near optimal solution to the general process sequence problem. The genetic algorithm performs well to solve optimization problems. The experiment results clearly show that the proposed approach is able to find optimized problem. The experiment carried out is efficient to find best sequence. The work can be extended so that technique can be implemented for other criteria of CPU scheduling to maximize utilization of computer system.

8. References:

1- Abraham S. & Peter B. Galvin & Greg Gagne," *Operating System Concepts*",John Wiley & Sons,Inc. 2007.

2- David E. Goldberg, " *Genetic Algorithms in Search Optimization & Machine learning*", Second Reprint , Person Education Asia Ltd., 2000.

3- Holland J. H., "*Adaptation in neural and artificial systems*", Ann Arbor, the university of Michigan Press, 1975.

4- L.M. Schmitt, " *Fundamental Study Theory of Genetic Algorithms*", International Journal of Modeling and Simulations Theoretical Computer Science, 2001.

5- Greffenstette J. J., "*Optimization of Control Parameters for Genetic Algorithms* ", IEEE Computer Society Press, California., 1992.

6- Dr.Rakesh K. & Rajiv K. & Sanjeev G.& Ashwani K. ," *genetic algorithm approach to operating system process scheduling problem*", international journal of Engineering Science and Technology Vol.2(9), 4247-4252, 2010 .

7- Lnsup Lee & Dianna Xu, "CPU Scheduling", University of Pennsylvania, fall 2003, Lectures notes.

8- J. Sivas, "CPU scheduling algorithms simulation using Java", 2010.