Genetic Algorithm Optimal Approach For Scheduling Processes In Operating System

Manu Sharma
Shobhit University
Meerut

Preeti Sindhwani
Assistant Professor
N.I.E.T.
Gr. Noida

Vijay Maheshwari
Assistant Professor
Shobhit University
Meerut

ABSTRACT

To enhance the performance of CPU, Many scheduling approaches are used to get maximum throughput in terms of scheduling the processes, waiting in the ready queue in minimum waiting time. This paper presents the genetic algorithm approach to provide the optimal solution for scheduling the processes and compares the results with other traditional scheduling algorithms. CPU scheduling is an NP hard problem. Genetic algorithm provides the optimal solution to these NP hard problems. These algorithms handle a population of possible solutions and the selection of these solutions is based on the fitness function. Crossover & other operators are used to generate new population.

General Terms

Genetic Algorithm, NP hard, Crossover, Inversion

Keywords

Ordered Crossover

1. INTRODUCTION

CPU Utilization and Throughput can be maximized by effectively allocating processes to the processor. It requires an optimal solution for allocating these processes in an Operating System. Genetic Algorithm uses recombination operators, able to mix good characteristics from different solutions and it is considered helpful to find optimal solutions of the problem. Genetic Algorithm uses genetics as its model of problem solving. Genetic Algorithm handles a population of possible solutions. Each solution is represented through a chromosome. Select two parent chromosomes from a population according to their fitness (the better fitness the bigger the chance to get selected.) with a crossover probability crossover the parents to form new offspring. Each string undergoes with the inversion operator[1].

2. Genetic Algorithm

Steps involved in genetic algorithm used in this paper are:

[Start] Generate Random Population of n chromosomes (Suitable solutions)

[Fitness] Evaluate the Fitness F(x) of each chromosome x in the population.

[New Population] Create new Population by Repeating following steps until the new population is complete.

[**Selection**] Select to Parent chromosomes from a population according to their Fitness.

[**Crossover]** With a crossover probability crossover the parents to form a new offsprings.

[Replace] Place new offspring in the new population.

[**Termination]** If the End condition is satisfied, stop and return the best solution in the current population.

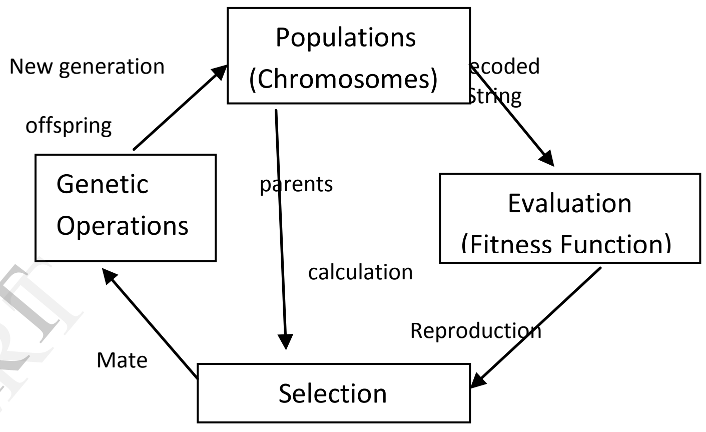

Fig 1: Genetic Algorithm Cycle

2.1 Comparison of Genetic Algorithm with other Traditional optimization techniques

* Genetic algorithms operate with coded versions of the problem parameters rather than parameter themselves i.e. GA works with the coding of the solution set not with the solution itself.
* Almost all conventional optimization techniques search from a single point but GAs always operate on a whole population of points (strings).GA uses population of solutions rather than a single solution. This plays a major role to the robustness of GAs. It improves the chance of reaching the global optimum.
* GA uses fitness functions for evaluation rather than derivatives. They can be applied to any kind of continuous or discrete optimization problem[3].

3. Assumption for CPU scheduling problem

Consider N processes with their static Burst time. These processes have been processed on CPU to process in such a way that their waiting time should be minimized. The initial population of individual solutions is randomly generated. Processes are waiting in the ready queue i.e. gene pool.

3.1 Fitness Evaluation

For calculating fitness the chromosome has to be first decoded and abject function has to be evaluated. The fitness not only indicates how good the solution is, but also corresponds to how close the chromosome is to the optimal one (Lesser the Burst time of process more fit it will be).

Fitness $F(x) = \frac{\sum_{i=0}^{N} Wi}{N}$ (i =1, 2, 3.....N)

.W*i* is the waiting time of processes.

3.2 Selection

In this paper the selection process used is based on **Roulette Wheel Process**. The expected value of an individual is the fitness divided by the actual fitness of the population. The wheel spins N times, where N is the number of individuals in the population[2]. On each spin, the individual under the wheel marker is selected to be in the gene pool of parents for next generation.

* Sum the total expected value of the individuals in the population. Let it be T.
* Repeat N times.
 > Choose a random integer r between 0 and T.
 > Loop through the individuals in the population, summing the expected value puts the sum over this limit is the one selected.
Let F(max) be the fitness of the currently available best string. If the next string has fitness of F(x) such that F(x)> F (max), then new string is selected otherwise it is selected with Boltzmann Probability:-

$P = e^{\left[-\frac{Fmax - Fx}{T}\right]}$

Where, T = T0 (1-α)^k and k = (1+100*g/G)
g=current generation number; G=Max Value of g.

α = [0,1] ; T0 =[5,100]

Final state is reached when computation approaches 0 value of T.

3.3 Crossover

Two mating chromosomes are cut once at corresponding points and the sections after the cut are exchanged. Crossover point is selected randomly along the length of the mated string and bits next to the cross points are exchanged[5].

Parent1 :	4	2	1	|	3	6	5
Parent2 :	2	3	1	|	4	5	6
Offspring:	4	2	1		4	5	6

In this paper Modified Ordered Crossover (MOX) is used.

3.3.1 Ordered Crossover (OX)

For given two parent chromosomes, two random crossover points are selected partitioning them into left, middle and right section from parent 1 and it's middle section is determined by the genes in the middle section of parent 1 in the order in which the values appear in the parent 2. Similar process is applied to offspring 2.

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

Modified OX

In modified OX two crossover points are the processes having the two longest burst time.

3.3.2 Inversion Operator

Two points are selected along the length of the chromosome, the chromosome is cut at those points and the end of the selection cut, gets reversed (switched or swapped). In this paper **linear+ end inversion** operator is used[4], which minimizes the property of linear inversion to disrupt the alleles present near the center of the string disproportionately to those of alleles present near the ends.

1	2	3	4	5	6	7	8	9	10
4	8 ^	6	2	7	9 ^	1	10	5	3
4	8	9	7	2	6	1	10	5	3

The gene is to be used in fitness function the same as before inversion and it remains same as after inversion operation also.

3.3.3 Replacement

After inversion new chromosome is generated and is added to the existing population. Fit chromosomes are again selected to produce another new offspring. This process continues until we not get the optimal solution[6]. New population always replaces the existing population to survive.

3.3.4 Termination

Multiple termination conditions are used in genetic algorithm. We will terminate the algorithm by equal fitness for the fittest selected chromosomes in respective iteration.

4. Pseudo code of GA

```
(1) Begin /* MOX GA Algorithm*/
(2) Generate random population of N chromosomes (possible
solutions)
```

```
(3) Evaluate Fitness F(x) of each chromosome x in the population using fitness function
Fitness F(x) = $\frac{\sum_{i=0}^{N} Wi}{N}$          (i =1, 2,3.....N)

(4) WHILE NOT terminate Do
BEGIN /* Produce New generation */
(5) Select two best fitted parents from the existing population using Roulette Wheel selectio
(6) Crossover the two individuals by apply
     Modified OX using OX_C( ) function.
(7) linear +end inversion operator, Inversion( ) to generate offspring.
(8) Compute the fitness of offspring and add the offspring into the new Population using    Replace( ) function
   END
(9) IF Termination criteria is matched THEN
   end:= TRUE

end

end

end
```

5. Experimental Description:

Solutions are randomly generated to form an initial population. New generations are reproduced by crossover. Fittest solutions are chosen as parents to reproduce the offsprings for new generation. Fitness function is evaluated based on FCFS algorithm.

Suppose there are 4 jobs. The number of possible sequences are 4!. Total 10 sequences are selected out of 24 for 4 jobs. Consider the crossover point is 2. Suppose following two individuals are fit to generate next generation.

3 1 2 4 and 4 3 5 1
After cross over 3 **1** 5 **1**
This offspring is not valid because 4 is not there in new individual and job 1 appears twice such
individual is discarded .So to avoid this, we are using the modified OX . In the modified crossover we get proper order individual. let us assume two individual which are marked as fit and use for the next generation .
5 1 2 4 and 3 4 2 1
After modified OX 5 1 3 4
this individual is accepted because there is no repetition of any job.

6. Simulation Results:

Figure 1 shows the comparison graphs of various scheduling algorithms finding that GA based scheduling algorithm gives the same result as in SJF in some cases. Although SJF is optimal, it can not be implemented at the level of short term CPU scheduling[2]. There is no way to know the length of the next CPU burst. Hence GA based CPU scheduling algorithm can be implemented to find the near to optimal solution.

Table 1 shows the average waiting time of FCFS, SJF,RR and GA based scheduling algorithm for 4 processes with their burst times. Waiting time of processes has been minimized in SJF and GA based algorithms, so that the maximum CPU utilization can be achieved

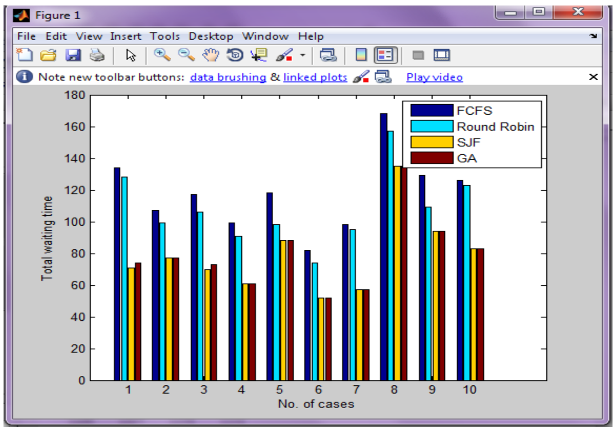

Figure1. Comparison of various scheduling algorithms

Table1: Result Analysis

S.N.	Burst time of processes (P1,P2,P3,P4)	FCFS	Round Robin	SJF	GA
1.	48,28,28,15	134	128	71	74
2.	21,25,31,42	167	99	77	77
3.	20,40,27,23	117	106	70	73
4.	29,10,30,22	99	91	61	61
5.	34,19,35,49	118	98	88	88
6.	8,8,36,48	82	74	52	52
7.	20,20,28,17	98	95	57	57
8.	49,40,49,46	168	157	135	135
9.	25,40,34,35	129	109	94	94
10.	45,6,45,32	126	123	83	83
	Total Avg. Waiting Time	123.8	108	78.8	79.4

7. Conclusion & future scope:

Although SJF is optimal, it can not be implemented at the level of short term CPU scheduling. There is no way to know the length of the next CPU burst. Hence GA based CPU scheduling algorithm can be implemented to find the optimal solution. Research can be extended so that GA can be implemented for dynamic process scheduling. The performance can also be increased by apply other GA based operators to this problem.

8. References

[1] Genetic Algorithm approach to Operating System process scheduling Problem Dr.Rakesh Kumar, Reader Department of Computer Science and Application, Kurkshetra university, Haryana, India.

[2] Genetic Algorithm approach to optimal CPU scheduling Problem Preeti Sindhwani, Assistant Professor , Noida Institute of Engg. & Technology, Gr. Noida, India

[3] Introduction to genetic algorithm by S.N.Shivanandnam and S.N.Deepa.

[4] L.M.Schmitt, "Fundamental Study Theory of Genetic Algorithms" , International Journal of Modelling and Simulation Theoretical Computer Science 259, 2001, 1 – 61.

[5] L. Davis, "Applying Adaptive Algorithms to Epistactic Domains", in Proceedings of the Int. Joint Conf. on Artificial Intelligence (IJCAI'85), Los Angeles, CA, pp. 162-164.

[6] Holland, J.H., 1975. "Adaptations in natural and artificial systems", Ann Arbor: The University of Michigan Press.

Manu Sharma is pursuing her M.Tech. from Shobhit University,Meerut.She is also lecturer in Noida Institute Of Engineering & Technology, Gr. Noida. Her research interest includes Genetic algoritms & problem solving.

Preeti Sindhwani did her M.Tech. from N.C. College of Engineering, Israna(Panipat). She is gold medalist in M.Tech. She is also Assistant Professor in Noida Institute of Engineering & Technology,Gr. Noida. Her research interest includes approximation and optimization based algorithms.

Vijay Maheshwari did his M.Tech. from Shobhit University, Meerut. He is also an Assistant Professor in Shobhit University,Meerut. His research interest includes Genetic Algorithms & Neural Network Design.