(IJIASE) 2015, Vol. No. 1, Jan-Dec **e-ISSN: 2454-9258; p-ISSN: 2454-809X**

EVALUATION OF CPU SCHEDULING ALGORITHM WITH GENETIC APPROACH

Er. Amit Gupta

ABSTRACT

There are many factors that affecting the efficiency of data processing systems. Understanding the operating system performance is a big issue. Operating system performance issues commonly involve process management, memory management, and scheduling. In computer System, Processor scheduling divides a computer processor's work between multiple programs so that it is continually switching from one opened application to another. This gives the appearance that the computer is running a number of different programs simultaneously as the whole process are very fast. Our study is an effort to develop an algorithm (genetic algorithm based) for obtaining optimal solution or near to optimal schedules for CPU Scheduling Problems with minimum computation effort even for large sized problems. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. In this paper we compare the result of average waiting time of general algorithms with the genetic algorithm.

Keywords— Genetic Algorithm, Encoding, Crossover, Mutation, Selection

INTRODUCTION

Scheduling is the process of creating a plan to use limited resources that involves creating a time-based schedule. Scheduling involving deciding what resources to commit to a task and when. Processes maintained in the computers main memory are considered to be executing concurrently even though the CPU is (usually) capable of executing only one instruction at a time. The number of processes that can be held in memory is dependent on the amount of memory available to the system. problem of finding an optimal schedule for a set of jobs is NP-complete[1]. This means that the problem can be solved in Polynomial time using a Non-deterministic Turing machine (like a regular Turing machine but also including a non-deterministic "choice" function). Basically, a solution has to be testable in poly time. If that's the case, and a known NP problem can be solved using the given problem with modified input (an NP problem can be reduced to the given problem) then the problem is NP complete. It is not an easy task to find the optimal solution n polynomial time. Actually scheduling is not an easy task. For efficient scheduling and optimization following criteria are followed:

Scheduling Criteria:

- CPU utilization – keep the CPU as busy as possible.
- Throughput – number of processes that complete their execution per time unit.

(IJIASE) 2015, Vol. No. 1, Jan-Dec **e-ISSN: 2454-9258; p-ISSN: 2454-809X**

- Turnaround time – amount of time to execute a particular process; which is the interval from time of submission of a process to the time of completion (includes the sum of periods spent waiting to get into memory, waiting in ready queue, executing on the CPU, and doing I/O).
- Waiting time – sum of the periods spent waiting in the ready queue.
- Response time – the time from the submission of a request until the first response is produced (i.e., the amount of time it takes to start responding, but not the time that it takes to output that response).

Optimization Criteria:

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time

There are a number of scheduling algorithms based on which we can evaluate the various scheduling criteria. First-Come, First-Served (FCFS), Shortest-Job-First (SJF), Priority, Round-Robin (RR), Multilevel Queue, Multilevel Feedback Queue. A genetic or evolutionary algorithm applies the principles of evolution found in nature to the problem of finding an optimal solution to a Solver problem. In a "genetic algorithm," the problem is encoded in a series of bit strings that are manipulated by the algorithm; in an "evolutionary algorithm," the decision variables and problem functions are used directly. Most commercial Solver products are based on evolutionary algorithms. First, it relies in part on random sampling. This makes it a nondeterministic method, which may yield somewhat different solutions on different runs, even if you haven't changed your model. Second, where most classical optimization methods maintain a single best solution found so far, an evolutionary algorithm maintains a population of candidate solutions. Third, inspired by the role of mutation of an organism's DNA in natural evolution. An evolutionary algorithm periodically makes random changes or mutations in one or more members of the current population. Fourth, inspired by the role of reproduction in the evolution of living things. An evolutionary algorithm attempts to combine elements of existing solutions in order to create a new solution, with some of the features of each "parent" using crossover. Fifth, inspired by the role of natural selection in evolution. An evolutionary algorithm performs a selection process in which the "most fit" members of the population survive, and the "least fit" members are eliminated..

The most popular technique in evolutionary computation research has been the genetic algorithm and operators common are selection operators, reproduction operators. In this thesis comparison of the average waiting time of scheduling algorithm (FCFS and SJF) and scheduling using GA is done to find the solution near to the optimal solution.

(IJIASE) 2015, Vol. No. 1, Jan-Dec **e-ISSN: 2454-9258; p-ISSN: 2454-809X**

OPTIMIZATION USING GENETIC ALGORITHM

Genetic algorithms are a rather straightforward, flexible optimization-solving technique that is modeled after evolutionary forces in nature. In the computer science field of artificial intelligence, genetic algorithm is a search heuristic that mimic the process of natural selection.

Genetic algorithms was first proposed in the 1960s by John Holland. As summarized by Tomassini [11], the main idea is that in order for a population of individuals to adapt to some environment, it should behave like a natural system. This means that survival and reproduction of an individual is promoted by the elimination of useless or harmful traits and by rewarding useful behavior and feasible solutions are the individuals living in that environment.

A. ***Optimization***

Finding an alternative with the most cost effective or highest achievable performance under the given constraints, by maximizing desired factors and minimizing undesired ones. Optimization techniques are used to find a set of design parameters that give the best possible result. There are two key components in an optimization problem:

- Objective function
- Constraints (optional)

The objective function calculates the desired quantity to be minimized or maximized. Constraints can be added that limit the possible values for the design parameters. An optimization problem can be represented in the following way:

Given: a function f: A R from some set A of some elements to the real numbers.

Sought: an element x0 in A such that $f(x0) \leq f(x)$ for all x in A ("minimization") or such that $f(x0) \geq f(x)$ for all x in A ("maximization").

Such a formulation is called an optimization problem or a mathematical programming problem. Many real-world and theoretical problems may be modelled in this general framework.

B. ***CPU Scheduling***

A major task of an operating system is to manage a collection of processes. In some cases, a single process may consist of a set of individual threads. In both situations, a system with a single CPU or a multi-processor system with fewer CPU's than processes has to divide CPU time among the different processes/threads that are competing to use it. This process is called CPU scheduling.

A CPU scheduling algorithm should try to maximize the following:

• CPU utilization

• Throughput

(IJIASE) 2015, Vol. No. 1, Jan-Dec **e-ISSN: 2454-9258; p-ISSN: 2454-809X**

A CPU scheduling algorithm should try to minimize the following:
• Turnaround time
• Waiting time
• Response time

C. Genetic algorithms

Genetic algorithms are the adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics. GA is a part of evolutionary computing, a rapidly growing area of artificial intelligence inspired by Darwin's theory "Survival of the Fittest". GA, is randomized, exploit historical information to direct the search into region of better performance with in the search space. Genetic Algorithms (GA's) are adaptive methods which may be used to solve search and optimization problems. They are based on the genetic processes of biological organisms. Over many generations, natural populations evolve according to the principles of natural selection and survival of the fittest, first clearly stated by Charles Darwin in The Origin of Species. By mimicking this process, genetic algorithms are able to find solutions to real world problems, if encoded suitably. For example, GA's can be used to design bridge structures, for maximum strength/weight ratio, or to determine the least wasteful layout for cutting shapes from cloth.

The Algorithm

1. Randomly initialize population(t)
2. Determine fitness of population(t)
3. Repeat
    1. select parents from population(t)
    2. perform crossover on parents creating population(t+1)
    3. perform mutation of population(t+1)
    4. determine fitness of population(t+1)
4. Until best individual is good enough

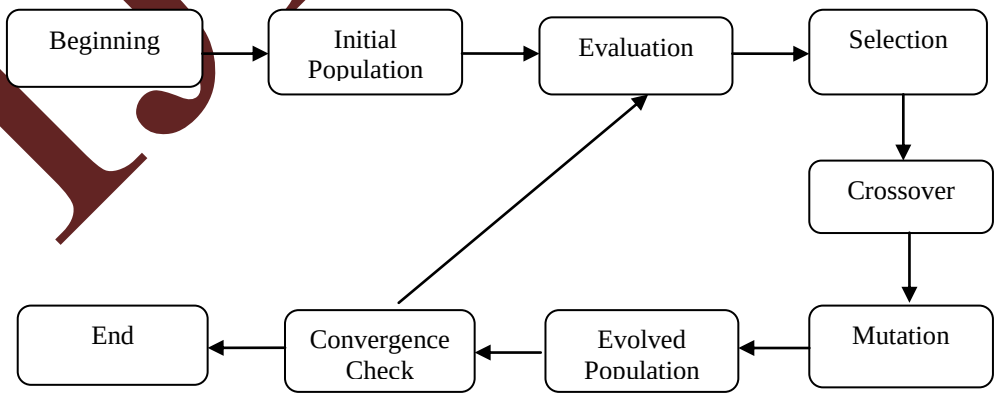

Fig: 1.1 Basic cycle of genetic algorithms

(IJIASE) 2015, Vol. No. 1, Jan-Dec **e-ISSN: 2454-9258; p-ISSN: 2454-809X**

Effects of Genetic Operators

- Using selection alone will tend to fill the population with copies of the best individual from the population
- Using selection and crossover operators will tend to cause the algorithms to converge on a good but sub-optimal solution
- Using mutation alone induces a random walk through the search space.
- Using selection and mutation creates a parallel, noise-tolerant, hill climbing algorithm

III PROBLEM STATEMENT

The objective is to find the schedule, which follows following constrains:

i. Processor has to process every job, without leaving any job unprocessed.
ii. Once a job is executed completely, processor is allocated to next job. Each job should be processed only one time.
iii. No pre-emption is allowed, i.e. processor can't switch between processes before completion of currently executing process.
iv. The total waiting time that for all the jobs till processor runs all the jobs in the ready queue should be minimum.

This process is repeated a number of times, and usually leads to better and better individuals.

IV PROPOSED GENETIC ALGORITHM

The steps of applying GA relating our problem are:

1. Pick out an Encoding scheme.
2. Pick out Fitness function.
3. Pick out Operators.
4. Pick out Parameters.
5. Pick out an Initialization and Stopping criteria

1. Encoding Scheme

Encoding of chromosomes is one of the problems, when you are starting to solve problem with GA. Encoding very depends on the problem. Various types of encoding schemes are: Binary encoding, Permutation encoding, Value encoding, octal encoding & Hexadecimal encoding. But for we required only value encoding for our algorithm.

Value Encoding

In value encoding, every chromosome is a string of some values. Values can be anything connected to problem, form numbers, real numbers or chars to some complicated objects.

(IJIASE) 2015, Vol. No. 1, Jan-Dec **e-ISSN: 2454-9258; p-ISSN: 2454-809X**

Chromosome A	1.2324 5.3243 0.4556 2.3293 2.4545
Chromosome B	ABDJEIFJDHDIERJFDLDFLFEGT
Chromosome C	(back), (back), (right), (forward), (left)

Example of chromosomes with value encoding

Value encoding is very good for some special problems. On the other hand, for this encoding is often necessary to develop some new crossover and mutation specific for the problem.

2. Fitness Function

The fitness function basically determines which possible solutions get passed on to multiply and mutate into the next generation of solutions. This is usually done by analyzing the "genes," which hold some data about a particular solution to the problem. Our aim is to find out individual having minimum average waiting time. So the individual who has minimum average waiting time is fittest as compared to other. So, The fitness function of a Solution Si is given by

$$\text{Fitness (Si)} = \frac{\sum_{j=1}^{n} wj}{n}$$

Where wj is the waiting time of the process j. n is the total no. of processes.

3. Selection operator

The key idea of selection operator is to give preference to better individuals (those that are nearer to the solution) by allowing them to pass on their genes to the next generation and prohibit the entrance of worst fit individuals into the next generations [3]. The selection operator mainly works at the level of chromosomes.

Roulette wheel selection

The basic part of the selection process is to stochastically select from one generation to create the basis of the next generation. The requirement is that the fittest individuals have a greater chance of survival than weaker ones. This replicates nature in that fitter individuals will tend to have a better probability of survival and will go forward to form the mating pool for the next generation. Weaker individuals are not without a chance.

This method is as follows:

1. Total expected value of the individuals in the population is summed. Let it be t.
2. Repeat N times:
   - A random integer ‘r’ between o and t is Chosen.
   - Loop through the individuals in the population, summing the expected values, until the sum is greater than or equal to ‘r’. The individual whose expected value puts the sum over this limit is the one selected.

The rate of evolution depends on the variance of fitness in the population.

(IJIASE) 2015, Vol. No. 1, Jan-Dec **e-ISSN: 2454-9258; p-ISSN: 2454-809X**

Crossover

Crossover occurs in the genetic algorithm/program when two members of a population (chromosomes) are selected for reproduction, and is usually based upon their relative fitness in solving the problem. Sometimes called recombination, crossover is the process of combining the attributes of two chromosomes to produce two new chromosomes that inherit some (or all) of their attributes from one (or both) of their parents.

In **ordered crossover** operator two cut points are randomly selected from the parent's chromosomes. To produce the offspring the genes between the cut points are replaced by the genes in the second parent. Consider the following example.

P1	2 1 5 4	7 8 9 3	6 10
P2	1 5 4 6	10 2 8 7	3 9

Finally we have the offsprings as follows:

O1	5 4 9 3	10 2 8 7	6 1
O2	4 6 10 2	7 8 9 3	1 5

Fig 1.2: Ordered Crossover

Mutation

In mutation the two random positions in the chromosomes are chosen and bits corresponding to those positions are interchanged. There are many techniques for mutation like flipping, interchanging, reversing but for our problem interchanging method is used.

In **Interchanging,** two random positions of the string are chosen and the bits corresponding to those positions are interchanged. This is shown below.

Parent	1 0 1 1 0 1 0 1
Child	1 1 1 1 0 0 0 1

Fig 1.3: Mutation Interchanging.

4. Parameters

With an encoding, a fitness function, and operators in hand, the GA is ready to enter in action. But before doing that, the user has to specify a number of parameters such as population size, no. of processes.

(IJIASE) 2015, Vol. No. 1, Jan-Dec **e-ISSN: 2454-9258; p-ISSN: 2454-809X**

5. Initialization method & stopping criteria

The GA simulates evolution on the artificial population of solutions using operators that imitate the survival-of-the-fittest and principles of natural genetics such as recombination and mutation [9]. Each individual is evaluated according to the user's specified fitness function.

V. RESULT

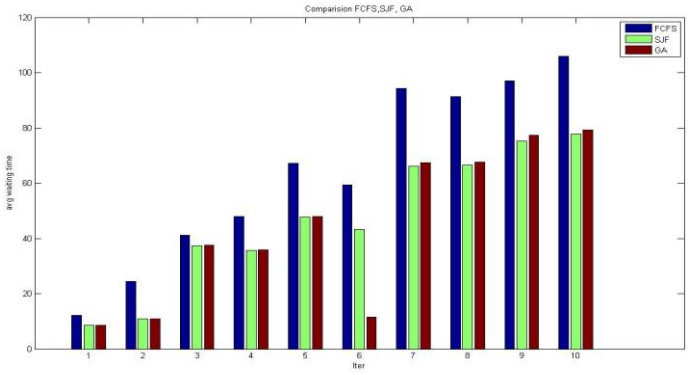

Fig 1.4: Comparison between FCFS, SJF, GA

The bar graph shown here is about the results showing the **average waiting time** of various algorithms. Three algorithms (FCFS, SJF and Algorithm using genetic approach) for CPU scheduling are implemented the procedure started with randomly generating the population. Then a selection technique has been employed over the encoded population and after that crossover is performed. After the crossover operation, mutation is performed, but after every 4 iterations. Then this operation is repeated for the required number of iterations. The result came out is approximate same as for SJF scheduling algorithm.

VI. CONCLUSION

The main objective of this paper is to maximize the efficiency of the processor or minimize the waiting time for the processes. In this paper the average waiting time of scheduling algorithm based on the genetic approach is compared with the average waiting time of the first come first serve scheduling algorithm and shortest job first scheduling algorithm. The average waiting time is considered on different iterations.

Genetic algorithm is better approach to find the optimal solution of the problem. For this the population is randomly generated. After that the encoding scheme (value encoding in this method) is applied and after that the crossover method is applied to get the better offspring. And finally the mutation technique (interchanging) is applied to get the final result.

Our result of average waiting time of GA is almost similar to the average waiting time of the SJF algorithm. After all the implementations and after comparing the average waiting time of all the three algorithms, the Genetic algorithm is the best for scheduling algorithm.

(IJIASE) 2015, Vol. No. 1, Jan-Dec **e-ISSN: 2454-9258; p-ISSN: 2454-809X**

REFERENCES

[1] J.D. Ullman, NP-complete scheduling problems, published in Journal of Computer and System Sciences.

[2] Wemke van der Weij, Sandjai Bhulai, Rob van der Mei. "Optimal scheduling policies for the limited processor sharing queue".

[3] Back, T. (1996). "Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms". Oxford University Press US. [On-line]. Available: http://books.google.de/books?id=EaN7kvl5coYC [2010].

[4] Sivanandam, S. N. & Deepa, S. N. "Introduction to Genetic Algorithms". Springer.2008.

[5] Omar, M., Baharum, A., & Hasan, Y. Abu. "A Job-Shop Scheduling Problem (JSSP) Using Genetic Algorithm". In Proceedings of 2nd IMT-GT Regional Conference on Mathematics and Statistics, 2006.

[6] Snehal Kamalapur. "Efficient CPU Scheduling: A Genetic Algorithm based Approach". In conference proceeding IEEE, 2006, pp.206-207.

[7] Mitchell, Melanie. "An Introduction to Genetic Algorithms", 1st ed., MIT Press, 1996.

[8] Lau, T. L. & Tsang, E. P. K. "Guided genetic algorithm and its application to the generalized assignment problem". Submitted to Computers and Operations Research, 1998.

[9] Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Boston: Addison-Wesley.

[10] Davis, L. (1991). Handbook of Genetic Algorithm. Von Nostrand Reinhold, Newyork.

[11] Tomassini, M. (1999). Parallel and Distributed Evolutionary Algorithms: A Review. In Miettinen, K., Makela, M., & Periaux, J. (Eds.), Evolutionary Algorithms in Engineering and Computer Science (pp. 113 - 133). Chichester: J. Wiley and Sons.