Efficient CPU Scheduling using Genetic Algorithm Approach

Anu Taneja, Amit Kumar

Dept. of Computer Science, Hindu College of Engineering, Sonepat, Haryana, India

Abstract
Operating system’s performance and throughput are highly affected by CPU scheduling. The scheduling is considered as an NP problem. An efficient scheduling improves system performance. We use genetic algorithms to provide efficient process scheduling. This paper evaluates the performance and efficiency of the proposed genetic algorithm in comparison with other CPU Scheduling algorithms and will prove that proposed GA-based algorithm gives better performance measure and provides optimal solution. So on the basis of above concepts; a genetic scheduling algorithm has been developed. Hence the problem is comparing the performance of different scheduling algorithms and proving that genetic algorithms provide the optimal solution.

Keywords
This Paper contain following keywords:

I. Introduction
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 they have been suitably encoded [1]. 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. They can also be used for online process control, such as in a chemical plant, or load balancing on a multi-processor computer system. This paper uses Genetic Algorithm Approach in optimizing CPU Scheduling Problem. In computer science, scheduling [4] is the method by which threads, processes or data flows are given access to system resources (e.g. processor time, communications bandwidth). This is usually done to load balance a system effectively or achieve a target quality of service. The need for a scheduling algorithm arises from the requirement for most modern systems to perform multitasking (execute more than one process at a time) and multiplexing (transmit multiple flows simultaneously). The scheduler is concerned mainly with:

A. Throughput
The total number of processes that complete their execution per time unit.

B. Latency, Specifically

1. Turnaround Time
Total time between submission of a process and its completion.

2. Response Time
Amount of time it takes from when a request was submitted until the first response is produced.

C. Waiting Time
Equal CPU time to each process (or more generally appropriate times according to each process’ priority).

II. Methodology of Genetic Algorithms
An algorithm is a series of steps for solving a problem. A genetic algorithm is a problem solving method that uses genetics as its model of problem solving. It’s a search technique to find nearby solutions to optimization and search problems [7]. GA handles a population of possible solutions. Each solution is represented through a chromosome, which is just an abstract representation. Coding all the possible solutions into a chromosome is the first part, but certainly is not the most straightforward part of a Genetic Algorithm [1].

A set of reproduction operators has to be determined, too. Reproduction operators are applied directly on the chromosomes, and are used to perform mutations and recombination over solutions of the problem. Appropriate representation and reproduction operators are really something determinant, as the behavior of the GA is extremely dependant on it. Selection is done by using a fitness function. Each chromosome has an associated value corresponding to the fitness of the solution it represents. The fitness should correspond to an evaluation of how good the candidate solution is. The optimal solution is the one, which maximizes the fitness function. Genetic Algorithms deal with the problems that maximize the fitness function. Once the reproduction and the fitness function have been appropriately defined, a Genetic Algorithm is evolved according to the same basic structure. It begins by generating an initial population of chromosomes. The gene pool should be as large as possible so that any solution of the search space can be engendered. Generally, the initial population is generated randomly. The basic procedure of genetic algorithms is as:

1. Choose the initial population of individuals
2. Evaluate the fitness of each individual in that population
3. Repeat on this generation until termination (time limit, sufficient fitness achieved, etc.):
   • Select the best-fit individuals for reproduction
   • Breed new individuals through crossover and mutation operations to give birth to offspring
   • Evaluate the individual fitness of new individuals
   • Replace least-fit population with new individuals
4. Obtain optimal solution.

III. Problem Description
Suppose there are N processes (1, 2, 3...N) each of which has one to be processed one at a time on CPU. Assume that the burst time
of each job on CPU is given. The problem is to find an optimum sequence so that the total waiting time is minimal. Waiting Time is the fitness function for our problem.

IV. Computational Description

A. Initialization
Initially many individual solutions are (usually) randomly generated to form an initial population. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. Traditionally, the population is generated randomly, allowing the entire range of possible solutions (the search space). Occasionally, the solutions may be “seeded” in areas where optimal solutions are likely to be found.

For our problem population is generated randomly using rand () function.

B. Fitness Function
The fitness function is defined over the genetic representation and measures the quality of the represented solution. The fitness function is always problem dependent.

For our problem fitness function is the waiting time i.e. we want to minimize the waiting time so that CPU cycles can be effectively utilized.

C. Selection
During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as the former process may be very time-consuming [1-2].

For our problem Roulette Wheel Selection Method is used. This method is implemented as follows:
1. Sum the total expected value of the individuals in the population. Let it be T.
2. Repeat N times:
3. Choose a random integer ‘r’ between 0 and T.
4. 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.

Roulette Wheel Selection Method is shown as:

![Fig. 1: Roulette Wheel Selection](image)

D. Reproduction
The next step is to generate a second generation population of solutions from those selected through genetic operators: crossover (also called recombination) and mutation.

For each new solution to be produced, a pair of “parent” solutions is selected for breeding from the pool selected previously. By producing a “child” solution using the above methods of crossover and mutation, a new solution is created which typically shares many of the characteristics of its “parents”.

New parents are selected for each new child, and the process continues until a new population of solutions of appropriate size is generated. Although reproduction methods that are based on the use of two parents are more “biology inspired”, some research suggests that more than two “parents” generate higher quality chromosomes.

These processes ultimately result in the next generation population of chromosomes that is different from the initial generation. Generally the average fitness will have increased by this procedure for the population, since only the best organisms from the first generation are selected for breeding, along with a small proportion of less fit solutions.

For our problem Cyclic Crossover Method is used and mutation is done after fixed number of iterations to overcome the problem of local minima.

E. Termination
This generational process is repeated until a termination condition has been reached.

Common terminating conditions are:
- A solution is found that satisfies minimum criteria.
- Fixed number of generations reached.
- Allocated budget (computation time/money) reached.
- The highest ranking solution’s fitness is reaching or has reached a plateau such that successive iterations no longer produce better results.
- Manual inspection.
- Combinations of the above.

The genetic algorithm flow cycle is shown as:

![Fig. 2: Genetic Algorithm Cycle](image)

V. Results
There are four jobs (1,2,3,4,5,6) which requires processing time (5,15,8,20,9). So the sequence of the jobs may vary and also we analyze the GA by changing the process time of different jobs. We are comparing FCFS, SJF, Round Robin and Genetic Algorithms, by applying the genetic operator’s selection, crossover, fitness
function and inversion, we get the final population, and out of the final population we get the job schedule, which have the minimum waiting time.

Fig. 3, focus on comparison of waiting time of various scheduling algorithms.

![Fig. 3: Comparison of Scheduling Algorithms](image)

The bar graph shown here is about the results showing the total waiting time of various algorithms over a number of iterations. Again the results that came out are in favor of genetic algorithms.

VI. Conclusion and Future Work

Optimization is the main point of discussion in this paper. Actually this paper is based on implementing a “Genetic Algorithm Approach to CPU Scheduling Problem”. It includes making a comparison of Genetic Algorithms with CPU Scheduling Algorithms i.e. FCFS, SJF, Round Robin Scheduling Algorithms and proving that Genetics Algorithms provide optimal solution in comparison to CPU Scheduling Algorithms.

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 job sequencing problem. The Genetic algorithms outperform the conventional Initial population procedures in solving optimization problems. The new representation has initially been tested on a data to evaluate its effectiveness. Quite promising results are obtained. The simulation results clearly show that the proposed approach is able to find optimized solution. The experiment carried out is efficient to find best sequence.

After all the experimentation and implementation, Genetics Algorithms provides the optimal solution in comparison to existing CPU Scheduling Algorithms. The results clearly show that proposed genetic algorithm is able to find the optimal solution. In future, further this work may be enhanced to incorporate multiprocessor system environment. This work can also help us to design the effective algorithm for dynamic process scheduling in future.

References


Anu Taneja received her M.C.A degree from MDU, Rohtak, India in 2010 and M.Tech degree in Computer Science & Engineering from MDU, Rohtak, in 2012. Now working as an Assistant Professor in CSE Department of Hindu College of Engineering, Sonepat. Her research interest includes optimization and approximation based algorithms.

Amit Kumar received his B.E. degree in Computer Science and Engineering from MDU Rohtak, India in 2008, and M.E. degree in Computer Science Engineering, Chandigarh, India, in 2010. Now working as an Assistant Professor in CSE Department of Hindu College of Engineering, Sonipat since Aug 2010 to till date.