Exploiting Genetic Algorithm towards CPU Scheduling

Monika Mangla[1] Rakhi Akhare[2] Smita Ambarkar[3]
[1,2,3]Assistant Professor
[1,2,3]LTCE, Navi Mumbai

Abstract— Scheduling has become a problem of keen interest as a result of its widespread application in all areas. Effective scheduling approaches results in maximum utilization of resources. As scheduling in problem of interest to various researchers, various novel techniques are being proposed by various researchers to use the resources in an efficient and effective manner. Various heuristic have also been combined with different techniques to solve scheduling problems. Here, in this paper, we discuss how genetic algorithm can be used to solve CPU scheduling problems in an innovative manner. The same technique can be used for scheduling any resource. To attempt the problem of CPU scheduling, we use efficient encoding, and design appropriate crossover and mutation operators. Each individual is an encoded version of a proposed solution. The proposed approach consists of the individuals evaluation, their selection, which will contribute to the next generation using recombination of the parents by means of crossover, mutation and other operators.

Key words: CPU Scheduling, NP Hard, Genetic Algorithm, Crossover, Mutation, Selection

I. INTRODUCTION

There exist many NP-complete problems that invite rigorous research in the area. Along with other problems Scheduling of any resource also falls into the category of 'NP-complete' problems. For all NP-complete brute force approaches takes exponential time to find optimal solution thus some heuristics are required to reach optimal solution in polynomial time. This thirst for achieving the optimal solution in polynomial time results into various innovative approaches being proposed by researchers. Here in this paper, we discuss scheduling of CPU which is the most critical factor that affects the execution time. Execution time is the most important parameter for any algorithm thus there is a critical demand to have efficient CPU scheduling algorithms. The objective of process scheduling is to allocate all requesting processes to a processor in a manner that maximizes the throughput and efficiency of system. Over the time various techniques have been proposed to solve the problem of CPU Scheduling.

Here, in this paper, a technique based on Genetic Algorithm has been discussed to find out the optimal job sequence in a single-processor environment. Usage of Genetic algorithms was initiated by the Holland in the 1960s [1] to solve scheduling problems. Genetic algorithms are based on the principle of genes and its evolutions. To solve a problem using Genetic Algorithm, an initial solution space or solution population is created. Each member of this population is called chromosome or individual. Then there exists a fitness function which is evaluated for all chromosomes in the solution space. Now the objective is to optimize the value of fitness function. Various chromosomes in the solution space are combined using various genetic operators. Some of these genetic operators are mutation and cross over etc. A chromosome is then subdivided into genes. A gene is the GA's representation of a single factor for special control factor. The values taken by genes are called alleles. Several pair of individual are selected from the solution space and then these selected chromosome acts as parents that produce offspring by applying the genetic operator. These parents are selected by repeated use of a choice function [2]. Then individuals in the population are replaced with these new offspring such that the number of individuals in the population remains. The candidate for replacement is chosen that has minimum value for fitness function. Each serving string undergoes inversion with a specified probability. The research specifically tries to find a genetic algorithm that makes the process of iterative scheduling automatic. The same is practical for modern but relatively low cost computing equipment. This may be achieved by using an efficient encoding, and designing appropriate crossover and mutation operators for problem at hand.

To formally define the problem of CPU Scheduling we consider that there exists N processes {p1, p2, ...pn} and a single processor. The objective is to allocate all waiting processes to the CPU in a manner such that resulting waiting time is minimized. To simplify the problem, we assume that all process are ready for execution and waiting for CPU allocation. It is also assumed that all processes have equal priority.

II. HOW GENETIC ALGORITHM WORKS

As discussed in previous section, Genetic Algorithms starts with a solution universe that consists of numerous chromosomes. The initial solution space may be generated randomly. Encoding technique is used to encode the solution space which is followed by finding fitness function for each individual. Then the chromosomes with higher fitness function value are chosen as parents for the next iteration. Various Genetic operations are performed on the selected candidates and the obtained chromosome replaces a chromosome in the previous population set. The objective of replacing a chromosome by newly found offspring is to maintain the same number of chromosomes in the solution space. The process of finding new offspring and replacement of weak parent continues until no more improvement in fitness function is possible. The working of the Genetic Algorithm can be best described by Fig. 1

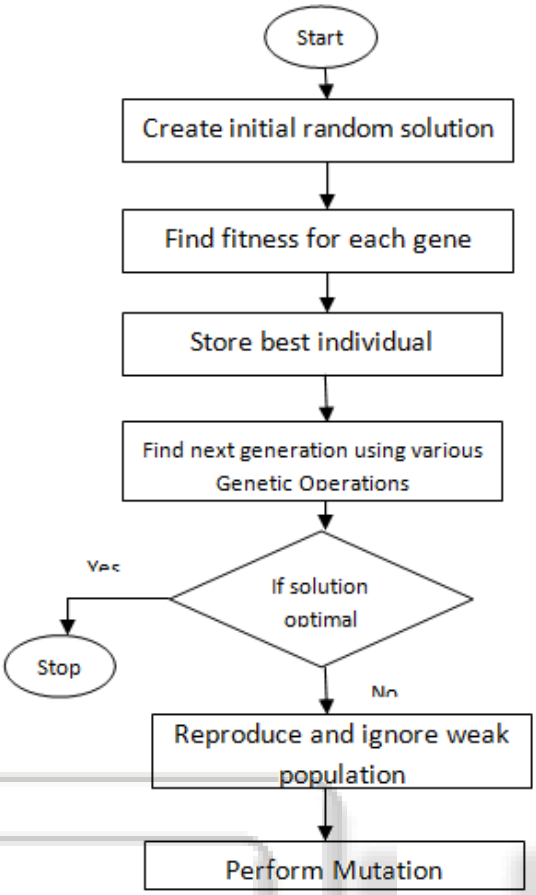

Fig. 1: Showing the Steps Involved In Genetic Algorithm

III. KEY ELEMENTS OF GENETIC ALGORITHMS

Following are the various key elements of Genetic Algorithms:

A. Individual:

An individual is a single solution [3] during that iteration and also known as chromosome. Individuals group together to obtain solution during subsequent iteration using any Genetic operation e.g crossover, mutation etc.

Search Space (Population):

A population is a collection of various individuals or chromosomes also used as feasible solutions.

B. Encoding:

Encoding is the process of representing individual chromosomes using some method. One of the encoding methods is permutation encoding where every solution is a string of numbers, which represents the sequence of events E.g.

Chromosome A 1 5 3 2 6 4 7 9 8
Chromosome B 8 5 6 7 2 3 1 4 9

Permutation encoding is only useful for ordering problems. The problem at hand is also a scheduling problem hence permutation encoding can also be used here.

C. Fitness Function:

To evaluate any solution in terms of its performance, a fitness function must be defined for the problem in consideration. Now the fitness function is evaluated for all feasible solutions and the value of fitness function represents its closeness to the optimal solution. Thus during subsequent iteration, chromosome having minimum fitness function values is replaced by offspring generated by applying Genetic operations to the chromosome of previous iteration. For example, in the problem of CPU scheduling, fitness function should calculate the average waiting time of the given solution. The solution, which is having the minimum value as, calculated from the fitness function, will be the one that is having the minimum waiting time and will be the fittest.

The fitness function of a Solution s_m is given by

$$s_m = \frac{\sum_{i=1}^{n} w_i}{n} \qquad (1)$$

As shown in equation 1, w_i represents the waiting time for process p_i and n represents the total number of processes.

D. Selection:

After deciding on an encoding, the next step is to decide how to perform selection i.e., how to choose individuals in the population that will create offspring for the next generation. During Selection phase, two parents are chosen from the population for Genetic Operation like crossing thus generating offspring [4].

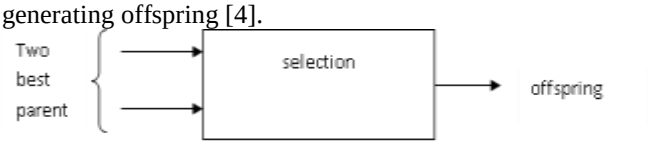

Fig. 2: Process of Selection

E. Roulette Wheel selection:

According to Roulette Wheel selection, each chromosome is assigned a portion of the wheel, where the size of the slice is proportional to the individual's fitness. The wheel is spin N times, N being number of individuals in the population. After every spin, the individual under the wheel's marker is selected that becomes parent for the next generation

This method is implemented as follows:

1) Sum the total fitness value of the individuals in the population. Let it be f.
2) Repeat N times:

– Choose a random integer 'p' between o and f.
– Loop through population, summing the fitness values, until the sum is greater than or equal to 'p'. The individual whose expected value puts the sum over this limit is the one selected.

F. Crossover

In Crossover, two parents are selection to produce an offspring.

G. Ordered Crossover

Ordered two-point crossover operation is used to solve ordering or sequencing problem for example CPU Scheduling problem. Given two parent chromosomes, these are divided ino left, middle and right portion using random partitions. One example is that offspring inherits left and right portion as it is and middle portion shuffles. One such illustration is given as:

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

Here in this paper, we are considering CPU scheduling thus the Ordered Crossover operation is modified resulting Modified Ordered Crossover. In this Modified Ordered Crossover, selected parents are the processes having the longest and shortest burst time to generate offspring.

H. Mutation

After crossover, the strings are performed operation of mutation. Mutation prevents the algorithm to be trapped in a local minimum and helps in achieving global optima. Mutation has traditionally considered as a simple search operator. If crossover is supposed to exploit the current solution to find better ones, mutation is supposed to help for the exploration of the whole search space. Mutation is considered as operator to maintain genetic diversity in the population. It introduces genetic structures in the population by randomly modifying some of its building blocks. It also keeps the gene pool well stocked. A search space is said to be ergodic if there is a non-zero probability of generating any solution from any population state.

There are various mutation methods for various kind of representations. For binary representation, a simple mutation can be inverting the value of each gene with a small probability.

I. Inversion:

Inversion operator is a primary natural mechanism to recode a problem [5]. In this operation, two random points are chosen along the length of the chromosome. The Chromosome is cut at those points and then gets swapped. The Inversion operator has been described in example below where the length of chromosome is 8 and two randomly selected inversion points are say 3 and 5.

Parent 4 2 7 6 8 1 3 5
Child 4 2 8 6 7 1 3 5

J. Replacement:

Two parents are drawn from a fixed size population, they breed two children, but not all four can return to the population, so two must be replaced i.e., once off springs are produced, a method must determine which of the current members of the population, if any, should be replaced by the new solutions. The technique used to decide which individual stay in a population and which are replaced in on a par with the selection in influencing convergence. Basically, the method for replacement selection is:

1) Weak Parent Replacement:

In weak parent replacement, a weaker parent is replaced by a strong child [6]. With the four individuals only the fittest two, parent or child, return to population. This process improves the overall fitness of the population when paired with a selection technique that selects both fit and weak parents for crossing, but if weak individuals and discriminated against in selection the opportunity will never raise to replace them.

K. Termination Criteria:

The genetic process ends if there is no change to the population's best fitness for a specified number of generations

IV. PSEUDO-CODE

1) Begin
2) Generate random population using random1 () function
3) Evaluate each individual using fitness fitness() function.
4) WHILE NOT finished Do
   BEGIN /* Produce New generation */
   Select two individual from the old population for mating using Selection () function
   Recombine the two individuals by apply Modified ordered crossover operator using mocrossover () function.
   Apply inversion operator using inversion () function to give offspring
   Compute the fitness of offspring and Insert the two fittest offspring into the new Population using replace() function.
   END

V. EXPERIMENTAL DESCRIPTION

To illustrate the usage of Genetic Algorithm for CPU Scheduling, we have illustrated the results using following data. There exists 6 jobs. Using a brute force approach there exist a total of 6! sequences. Initially randomly 10 sequences are selected out of 720 for the 6 jobs. Considering the number of jobs as 6 and the crossover point is 2 and 4, let us consider following two individuals, which are marked as fit to generate next generation.

3 1 2 4 5 6 and 6 4 3 5 1 2
After cross over Child 1: 3 4 1 2 5 6 Child 2:6 3 4 5 1 2
And let the inversion point be 2 and 5
After inversion Child 1: 3 5 1 2 4 6 Child 2:6 1 4 5 3 2
After that we calculate the fitness of parent1, parent2, child1, child2 and two fittest individual are returned to the solution space.
And above step is repeated until convergence criteria meet.

VI. CONCLUSION

Here in this paper, we have implemented Genetic algorithms for solving CPU Scheduling. It has been observed that various operations defined above can be efficiently used to solve scheduling problem. Using various Genetic operations, best sequence of the processes can be obtained. The work can further be extended so that technique can be implemented for dynamic process scheduling and sequencing.

REFERENCES

[1] Introduction to genetic algorithm by S.N.Shivanandnam and S.N.Deepa.
[2] Genetic Algorithm approach to Operating System process scheduling Problem Dr. Rakesh Kumar, Reader Department of Computer Science and Application, Kurkshetra university, Haryana, India
[3] M.Nikravan,M.H. Kashani,"A Genetic Algorithm for process scheduling in Distributed operating systems Considering load balancing", Proceedings 21st European Conference on Modelling and Simulation Ivan Zelinka, Zuzana Oplatková, Alessandra Orsoni ©ECMS 2007

[4] Blazewicz, J., Domschke, W., and Pesch, E. (1996). "The job shop scheduling problem: Conventional and new solution techniques". European Journal of Operational Research.
[5] Holland, J.H., 1975. "Adaptations in natural and artificial systems", Ann Arbor: The University of Michigan Press
[6] David E.Goldberg, Genetic Algorithms in Search Optimization, Machine Learning, Second Reprint, Pearson Education Asia Pvt. Ltd., 2000.

