International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 16
June 2018

A Genetic Algorithm Approach to Cpu Scheduling

Aradhana Dahiya & Shabnam Sangwan
[1]Department of Computer Science Sat Kabir Institute of Technology and Management (SKITM),
Maharishi Dayanand University (MDU), Rohtak
[1]aruhce123@gmail.com
[2]Department of Computer Science Sat Kabir Institute of Technology and Management (SKITM),
Maharishi Dayanand University (MDU), Rohtak
[2]shabnam022@gmail.com

Abstract— CPU scheduling is a 'NP-complete' problem. i.e., algorithms require exponential time to reach a solution. Moreover, it is a critical factor that effect the operating system efficiency in process scheduling we allocate processes to processor in such an order so that throughput and efficiency of the resulting system is maximized. This NP-complete nature of the problem requires innovative solutions for the problem. Genetic algorithms over the years have proved itself for finding innovative solutions for scheduling. In this work we study how genetic algorithm is used to find an innovative solution for CPU Scheduling.

Keywords— *CPU Scheduling, Genetic Algorithm, Fitness Function, Roulette Wheel Selection*

I. INTRODUCTION

Single Processor Scheduling Problems are classical combinatorial problems. These problems fall under the category of NP-complete problems. And a number of methods have been devised to solve them. Among them FCFS, SJF, Priority Scheduling and RR are of much importance and are widely used for scheduling of jobs in a processor. This study is an effort to develop a simple general algorithm (genetic algorithm based) for obtaining optimal or near optimal schedules for Single Processor Scheduling Problems with minimum computation effort even for large sized problems. The genetic algorithm is a search algorithm based on the mechanics of natural selection and natural genetics [GOL89]. As summarized by Tomassini [TOM99], the main idea is that, in order for a population of individuals to adapt to some environment, it should behave like a natural system. The genetic algorithm belongs to the family of evolutionary algorithms, along with genetic programming, evolution strategies, and evolutionary programming. Evolutionary algorithms can be considered as a broad class of stochastic optimization techniques. An evolutionary algorithm maintains a population of candidate solutions for the problem at hand. The population is then evolved by the iterative application of a set of stochastic operators. The set of operators usually consists of mutation, recombination, and selection or something very similar.

Using Tomassini's terms [TOM99], genetic algorithms (GA's) consider an optimization problem as the environment where feasible solutions are the individuals living in that environment. The degree of adaptation of an individual to its environment is equivalent of the fitness function evaluated on a solution. Similarly, a set of feasible solutions takes the place of a population of organisms. An individual is a string of binary digits or some other set of symbols drawn from a finite set. Each encoded individual in the population may be viewed as a representation of a particular solution to a problem.

II. GENETIC ALGORITHM SOLUTION

The Holland first proposed genetic algorithms in the 1960s. Genetics algorithm is based on the principle of genetics and evolution using efficient encoding, crossover and mutation operators. The algorithm requires population of the feasible solution. The algorithm is encoded with the suitable encoding scheme. Then we evaluate the fitness of each purposed solution followed by selection to select the parent solution for producing the offspring. Selection of parents is done by repeated use of method used for selecting the individual. Then crossover is used to produce offspring and then is used mutation to prevent the solution to trap in to local minima. The procedure is repeated until termination a criterion is met.

A. Encoding scheme

The first step for solving a problem in GA is to encode the problem at work into a state which can be solved using GA. The basis of genetics in nature is a chromosome. Each individual in the search space, i.e. each solution to the problem worked upon, needs to be encoded so that it can be modeled as a chromosome. The application of a genetic algorithm to a problem starts with the encoding.

The encoding specifies a mapping that transforms a possible solution to the problem into a structure containing a collection of decision variables that are important to the problem worked upon.
A particular solution to the problem can then be represented by a specific assignment of values to the decision variables. The set of all possible solutions is called the search space and a particular solution

represent a point in that search space. Various types of encoding schemes are:

1. Binary encoding
2. Permutation encoding
3. Value/Real encoding
4. Tree encoding
5. Octal encoding &
6. Hexadecimal encoding.

For our problem value encoding is required. 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.

B. Fitness function

Coming up with an encoding is the first thing that a genetic algorithm implementer has to do. The next step is to specify a function that can assign a score to any possible solution. The score is a numerical value that indicates how well a particular solution solves the problem. The score is the fitness of the individual solution. It represents how well the individual suits to the environment. In this case, the environment is the search space. The task of the GA is to discover solutions that have higher fitness values among the set of all possible solutions.

C. Operators

Once the encoding and the fitness function are specified, the implementer has to choose selection and genetic operators to evolve new solutions to the problem being solved. The selection operator simulates the "survival-of-the-fittest". There are various mechanisms to implement this operator, and the idea is to give preference to better individuals. Selection replicates individuals with high fitness values and removes individuals with low fitness values.

D. 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 [LOB00].

Operator probabilities are chosen in order to maintain the population diversity. For crossover the probability is 100% i.e. with every iteration/algorithm run crossover is performed always. Whereas the mutation operation performed after every five consecutive algorithm runs.

E. Initialization method & stopping criteria

The last steps of applying a GA are the specification of an initialization method and stopping criteria. The genetic algorithm is usually initialized with a population of random individuals, but sometimes a fraction of the population is initialized with previously known (good) solutions [LOB00].

Following the initialization step, each individual is evaluated according to the user's specified fitness function. Thereafter, 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 [LOB00]. A number of criteria can be chosen for this purpose, including among others, a maximum number of generations or time has elapsed, some predefined fitness value is reached, or the population has converged considerably [LOB00]. For our problem, stopping criteria chosen is no. of iterations and initialization method is random generation of population.

III. IMPLEMENTATION & RESULTS

Our problem of interest is to compare the three different scheduling algorithms. In this chapter, various algorithms that are required for solving this stated problem are put forward one after the other.

First algorithm is a SGA (i.e. Standard Genetic Algorithm) as follows:

START GA
Step 1. Initialize nop and populationsize [input no. of processes and population size from the user].

Step 2. Initialize brust time of each process randomly [Using rand function].

Step 3. Initialize population randomly [Using cpu_createperm function].

Step 4. Evaluate the fitness of each individual [Using fitness function].

Step 4. FOR no. of iterations DO

Step 5. Select two parents from the randomly generated population [Perform Roulette Wheel Selection using wheel function].

Step 6. Crossover two parents [Perform cyclic crossover using cyclic_crossover function].

Step 7. If i/5=0, Mutate two offspring produced. [Perform mutation at every fifth iteration using mutation_interchanging function].

Step 8. Evaluate Fitness and return the two best fitted individual to the population [Perform weakparent replacement using replace_weak function].

Step 9. Is done=Optimization criteria met?
Step 10. If (not) then, i = i + 1.
END FOR
[If optimization criteria met, stop the algorithm].

Step 11. Output the best solution
END GA

Here in this algorithm, "nop" represents number of processes and "populationsize" is used for holding the population size. Then the population is initialized randomly for generation first. Then the fitness of the population is evaluated. Then after this the selection operation is performed using the Roulette Wheel Selection. After selecting the parents for reproduction/mating, cyclic crossover is performed resulting in generation of new offspring's. Then if i is equal to 5, then mutation is performed. Otherwise fitness is evaluated for new offspring's and next population is generated after replacement based on the fitness of offspring's.

Now next operation is to check for the optimization or terminating criteria. If it is met then stop the algorithm, else increase the value of i variable with one and continue the same loop with the new population until the ending point has been reached, which results in the maximized values for the function under consideration.

The basic procedure for the selection is as follows:

```
For all members of population
   Cumulative sum += fitness of this individual
End for
 Do this twice
Choose a random number between 1 and cumulative sum(end)
      For all members of population
         If number > cumulative sum of that member
              then you have been selected
      end for
  end loop
```

After selection next operation is to perform crossover, and crossover operator used is cyclic crossover. This is performed when encoding scheme used is real/value encoding. As with our problem the chromosomes are encoded with real values.

Step 1. Let the two parents selected be: P1 and P2.

Step 2. Select the first Process of P1 and make it as the first process of offspring O1.

Step 3. To find the next process of offspring O1 search current process, which is selected from P1 in P2.

Step 4. Find the location of process in P2 and select the process, which is in the same location in P1.

Step 5. if the process is already present in O1
Then if no process till now selected from step 4 select the next process from P1 and go to step 2.
Else stop the procedure. Copies the process from the parent P2 in the corresponding locations.

After crossover next operation is to perform mutation. For our problem the mutation used is also of special type for value-encoded strings. Mutation performed is interchanging mutation. Algorithm for it is as follows:

Step 1. Select first offspring O1 chromosome.

Step 2. Select two-mutation point at random.

Step 3. Swap values at these two points in first chromosomes.

Step 4. Repeat the procedure for second chromosome.

After performing mutation operation, new population generated is evaluated and based on those Genetic algorithm further proceeds.

RESULTS

As it can be seen in the last chapter, three algorithms for CPU scheduling are implemented. The results that came out after the implementation are being discussed in this chapter. The procedure followed while implementing it is first started with randomly generating the population of chromosomes, for this implementation population chosen is 50 only with each chromosome having length of 50 genes. This process in GA has been called as encoding the input population. Then selection techniques have been employed over the encoded population and after that a special type of crossover is performed, i.e. modified cyclic crossover. After the crossover operation, mutation is performed, but after every 5 iterations. Then this operation is repeated for the required number of iterations. The results generated for our problem at hand are shown in Figure 1 below:

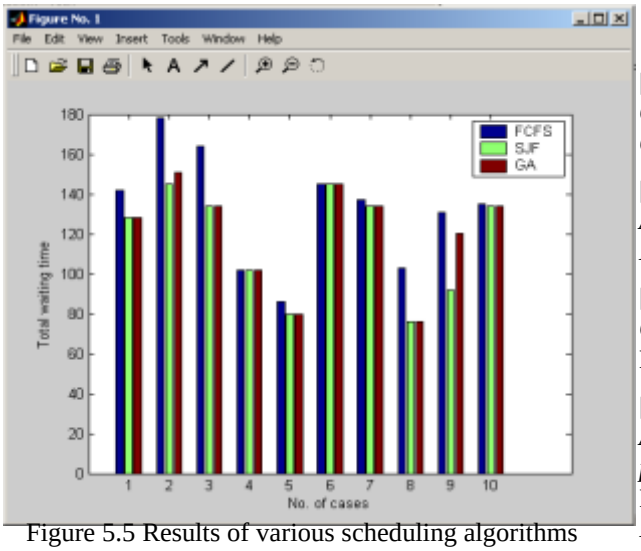

Figure 5.5 Results of various scheduling algorithms

The bar graph shown here is about the results showing the total waiting time of various algorithms. The results that came out are in favour of genetic algorithm. Although in some cases genetic algorithm shows more waiting time as compared to SJF.SJF scheduling algorithm is provably optimal. That is it gives the minimum waiting time for a given set of processes. By executing the short process before the long process total waiting time of the solution decreases. But with SJF we must know in advance the length of next CPU burst, which is not always possible in real world environment.

V. CONCLUSION

In this work we have studied how genetic algorithm is used to find an innovative solution for CPU Scheduling. Genetics algorithm is based on the principle of genetics and evolution using efficient encoding, crossover and mutation operators. The algorithm requires population of the feasible solution. The algorithm is encoded with the suitable encoding scheme. Then we evaluate the fitness of each purposed solution followed by selection to select the parent solution for producing the offspring. Selection of parents is done by repeated use of method used for selecting the individual. Then crossover is used to produce offspring and then is used mutation to prevent the solution to trap in to local minima. The procedure is repeated until termination a criterion is met. The result of genetic algorithm is then compared with the FCFS and SJF scheduling algorithm.

REFERENCES

[1]. [GJO79] Garey, M. & Johnson, D. (1979). *Computers and Intractability: a theory of NP-Completeness.* W.H.Freeman, San Francisco.

[2]. [GOL89] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Boston: Addison-Wesley.

[3]. [DAV91] Davis, L. (1991). *Handbook of Genetic Algorithm*. Von Nostrand Reinhold, Newyork.

[4]. [LAU96] Lau, T. L. & Tsang, E. P. K. (1996). *Applying a mutation-based genetic algorithm to the processor conjuration problem*, in Proceedings of IEEE 8th International Conference on Tools with Artificial Intelligence, pp. 17-24.

[5]. [MIT96] Mitchell, Melanie (1996). *An Introduction to Genetic Algorithms.* MIT Press.

[6]. [BAC97] Back, T., Fogel, David B. & Michalewicz, Z. (Eds.) (1997). *Handbook of Evolutionary Computation*. Computational Intelligence, Library Oxford University Press in cooperation with the Institute of Physics Publishing / CRC Press, Bristol, New York, ring bound edition. ISBN: 0-7503-0392-1, 978-0-75030-392-7, 0-7503-0895-8, 978-0-75030-895-3.

[7]. [JON98]Jong, K. De (1998). *Learning with Genetic Algorithm: An overview*, Machine Learning vol. 3, Kluwar Academic publishers.

[8]. [LAU98] Lau, T. L. & Tsang, E. P. K. (1998). *Guided genetic algorithm and its application to the generalized assignment problem*, Submitted to Computers and Operations Research.

[9]. [TOM99] 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.

[10]. [HAN00] Hanne, Thomas (2000). *Global multiobjective optimization using evolutionary algorithms*. Journal of Heuristics, vol. 6, no. 3, pp. 347-360.

[11]. [LOB00] Lobo, Fernando Miguel (2000).*The parameter-less genetic algorithm: rational and automated parameter selection for simplified genetic algorithm operation.* Paper submitted in International Conference on Genetic Algorithms, in Lisboa.

[12]. [GOL02] Goldberg, D. E. (2002). *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Norwell, MA: Kluwer.

[13]. [YAL04] Yalcinoz, T. & Altun, H. (2004). *A new genetic algorithm with arithmetic crossover to economic and environmental economic research dispatch*. Submitted as a project work at Dept. of Electrical & Electronic Engg.. Nigde University, Turkey.

[14]. [MOL05] Molga, M. & Smutnicki, C. (2005). *Test functions for optimization needs*, in Proceedings of 4th Conference on Genetic Algorithms.

[15]. [OMA06] Omar, M., Baharum, A., & Hasan, Y. Abu (2006). *A Job-Shop Scheduling Problem (JSSP) Using Genetic Algorithm* ,in Proceedings of 2nd IMT-GT Regional Conference on Mathematics, Statistics and

[16]. [INA06] Inazawa, H. & Kitakaze, K. (2006*). Locus-Shift Operator for Function Optimization in Genetic Algorithms*. Complex Systems Publications, Inc.

[17]. [SKA06] Snehal Kamalapur (2006).*Efficient CPU Scheduling: A Genetic Algorithm based Approach*. IEEE, pp.206-207

[18]. [SIV08] Sivanandam, S. N. & Deepa, S. N. (2008). *Introduction to Genetic Algorithms*. Springer.

[19]. [BIR09] Birch, J. B. & Wan, W. (2009). *An Improved Genetic Algorithm Using a Directional Search*. Tech report presented at Virginia Polytechnic Institute and State University, Blacksburg.

[20]. [HNZ09] H. Nazif(2009). *A Genetic Algorithm on Single Machine Scheduling Problem toMinimise Total Weighted Completion Time*. European Journal of Scientific Research,Vol.35 No.3, pp.444-452

[21]. [RKU10] Dr. Rakesh Chawla(2010). *Genetic Algorithm approach to Operating system process scheduling problem*. International Journal of ngineering Science and Technology, pp. 4247-4252

[22]. [SRA10] S. Ramya[2010] *"Window Constrained Scheduling of Processes in Real Time CPU Using Multi Objective Genetic Algorithm"* International Journal of Computer Applications Volume 1, pp.86-90