Volume 1, No. 5, December 2010
Journal of Global Research in Computer Science
ISSN-2229-371X
RESEARCH PAPER
Available Online at www.jgrcs.info

PREMATURE CONVERGENCE AND GENETIC ALGORITHM UNDER OPERATING SYSTEM PROCESS SCHEDULING PROBLEM

Er.Rajiv Kumar*[1] and Er. Ashwani Kaushik[2]
[1]PhD. Scholar, Singhania University Jhunjhunu, Rajasthan, India
rajivkumargill1@rediffmail.com

[2]Lecturer,Mechanical Engg Deptt.
N.C.College of Engineering Israna, Panipat INDIA
ashwanikrkaushik@rediffmail.com

Abstract: An important assumption to maximize the performance of genetic algorithm is to study the convergence state of genetic algorithm. Genetic algorithm is a Mata-heuristic search technique; this technique is based on the Darwin theory of Natural Selection. The important property of this algorithm is that it has worked on multiple state of solution. This algorithm is work with some finite set of population. The population contains set of individual, which represent the solution. Each member of the population is represented by a string written over fixed alphabets and also each member has a merit value associated with it, which represent its suitability for the problem under consideration. There are many coding techniques have been implemented for genetic algorithm. In this paper we study the effect of crossover and inversion probability on the convergence of genetic algorithm .The convergence of genetic algorithm is depends upon the parameter setting of genetic algorithm.

Keywords: Genetic algorithm, NP-hard, Process, Scheduling, Inversion probability.

INTRODUCTION

The work outline in this paper involve the optimization of scheduling problem by using genetic algorithm. The efficiency of the scheduler depends upon the algorithm used to develop the scheduler. The genetic algorithm is a robust algorithm . so it is better to understand the parameter setting of genetic algorithm. our mean abut the parameter setting is concern with the probability of operators , population size, selection techniques etc. The main aim of any scheduling technique is to find out the optimal solution with limited no. of constraint to be adopted. The Scheduling problem is consider to be the NP-hard problem. For literature on this area , see [1][2][7]. It is well known that scheduling problems are a subclass of combinatorial problems that arise every where. Genetic algorithms(Gas) are adaptive methods which may be used to solve search and optimization problems. Genetic algorithms (GAs) were first proposed by the John Holland[3] in the 1960s.The performance of the genetic algorithm is limited by some problem, typically premature convergence. This happens simply because of the accumulation of stochastic errors. If by chance , a gene becomes predominant in the population , then it just as likely to become more predominant in the next generation as it is to become less he predominant. If an increase in predominance is sustained over several successive generation and population is finite, then a gene can be spread to all members of the population. Once gene has converged in this way, it is fixed then crossover cannot

introduce new gene values. This produces a ratchet effect, so that as generations go by , each gene eventually becomes fixed. This diverse effect can be minimize by applying the inversion operator with suitable probability. Here we consider the operating system process scheduling for simulation

THE OPERATING SYSTEM PROCESS SCHEDULING PROBLEM FOR ANALYSIS

The performance of the operating system is greatly depends upon the proper process scheduling .Process scheduling in the operating system is the way by which the operating system allocate the CPU to the ready process in the ready queue[4]. Let us consider batch processing system in which there are 1,2,3…N process and these process has their given service time. This problem is concern to find out optimal sequence schedule which has minimum turn around time .In this paper turn around time is used to find out the fitness of the individual in the population. There is a pool of ready processes waiting for the allocation of CPU .These processes are independent and compete for the allocation of resources. The best approach is the maximum utilization of CPU and minimum turn around time.

SCOPE OF PAPER

The major part of this paper, contained in section 2, will explain working of genetic algorithm and their application in process scheduling problem. The GA is robust techniques and it has no. of operators which have their own properties .The parameter setting in the genetic algorithm is concerned with the setting of applicable static values of the operators used. Ie crossover probability , inversion rate , population size etc. Accessible introduction can be found in the books by Davis [5] and Goldberg[6] .Section 3 describe the

proposed structure of genetic algorithm. Section 4 explain the experimental setup for analysis and section 5 is conclusion .

INTRODUCTION OF GENETIC ALGORITHM

OVERVIEW

The evaluation function, or objective function, provides a measure of performance with respect to a particular set of parameters. The fitness function transforms that measure of performance into an allocation of reproductive opportunities. The evaluation of a string representing a set of parameters is independent of the evaluation of any other string. The fitness of that string, however, is always defined with respect to other members of the current population. In the genetic algorithm, fitness is defined by: fi /fA where fi is the evaluation associated with string i and fA is the average evaluation of all the strings in the population. Fitness can also be assigned based on a string's rank in the population or by sampling methods, such as tournament selection. The execution of the genetic algorithm is a two-stage process. It starts with the current population. Selection is applied to the current population to create an intermediate population. Then recombination and mutation are applied to the intermediate population to create the next population. The process of going from the current population to the next population constitutes one generation in the execution of a genetic algorithm. In the first generation the current population is also the initial population. After calculating fi /fA for all the strings in the current population, selection is carried out. The probability that strings in the current population are copied (i.e. duplicated) and placed in the intermediate generation is in proportion to their fitness.

CODING

Before a GA can be run, a suitable coding (or representation) for the problem must be devised. We also require a fitness function, which assigns a figure of merit to each coded solution. During the run, parents must be selected for reproduction, and recombined to generate offspring. It is assumed that a potential solution to a problem may be represented as a set of parameters (for example, the parameters that optimize a neural network). These parameters (known as genes) are joined together to form a string of values (often referred to as a chromosome. For example, if our problem is to maximize a function of three variables, F(x; y; z), we might represent each variable by a 10-bit binary number (suitably scaled). Our chromosome would therefore contain three genes, and consist of 30 binary digits. The set of parameters represented by a particular chromosome is referred to as a genotype. The genotype contains the information required to construct an organism which is referred to as the phenotype. For example, in a bridge design task, the set of parameters specifying a particular design is the genotype, while the finished construction is the phenotype.

The fitness of an individual depends on the performance of the phenotype. This can be inferred from the genotype, i.e. it can be computed from the chromosome, using the fitness function. Assuming the interaction between parameters is nonlinear, the size of the search space is related to the number of bits used in the problem encoding. For a bit string encoding of length L; the size of the search space is 2^L and forms a hypercube. The genetic algorithm samples the corners of this L-dimensional hypercube. Generally, most test functions are at least 30 bits in length; anything much smaller represents a space which can be enumerated. Obviously, the expression 2^L grows exponentially. As long as the number of "good solutions" to a problem are sparse with respect to the size of the search space, then random search
or search by enumeration of a large search space is not a practical form of problem solving. On the other hand, any search other than random search imposes some bias in terms of how it looks for better solutions and where it looks in the search space. A genetic algorithm belongs to the class of methods known as "weak methods" because it makes relatively few assumptions about the problem that is being solved. Genetic algorithms are often described as a global search method that does not use gradient information. Thus, non differentiable functions as well as functions with multiple local optima represent classes of problems to which genetic algorithms might be applied. Genetic algorithms, as a weak method, are robust but very general.

FITNESS FUNCTION

A fitness function must be devised for each problem to be solved. Given a particular chromosome, the fitness function returns a single numerical "fitness," or "figure of merit," which is supposed to be proportional to the "utility" or "ability" of the individual which that chromosome represents. For many problems, particularly function optimization, the fitness function should simply measure the value of the function.

$$\text{Fitness}(Ta_j) = \frac{\sum_{i=1}^{N} (C_i - A_i)}{N}$$

SELECTION

Individuals are chosen using "stochastic sampling with replacement" to fill the intermediate population. A selection process that will more closely match the expected fitness values is "remainder stochastic sampling." For each string i where fi/fA is greater than 1.0, the integer portion of this number indicates how many copies of that string are directly placed in the intermediate population. All strings (including those with fi/fA less than 1.0) then place additional copies in the intermediate population with a probability corresponding to the fractional portion of fi/fA. For example, a string with fi/fA = 1:36 places 1 copy in the intermediate population, and then receives a 0:36 chance of placing a second copy. A string
with a fitness of fi/fA = 0:54 has a 0:54 chance of placing one string in the intermediate population. Remainder stochastic sampling is most efficiently implemented using a method known as stochastic universal sampling. Assume that the population is laid out in random order as in a pie

graph, where each individual is assigned space on the pie graph in proportion to fitness. An outer roulette wheel is placed around the pie with N equally-spaced pointers. A single spin of the roulette wheel will now simultaneously pick all N members of the intermediate population.

REPRODUCTION

After selection has been carried out the construction of the intermediate population is complete and recombination can occur. This can be viewed as creating the next population from the intermediate population.

Crossover is applied to randomly paired strings with a probability denoted pc. (The population should already be sufficiently shuffled by the random selection process.) Pick a pair of strings. With probability pc "recombine" these strings to form two new strings that are inserted into the next population. In the proposed algorithm we use the modified crossover operator.
Good individuals will probably be selected several times in a generation; poor ones may not be at all. Having selected two parents, their chromosomes are recombined, typically using the mechanisms of crossover and

mutation. The previous crossover example is known as single point crossover. Crossover is not usually applied to all pairs of individuals selected for mating. A random choice is made, where the likelihood of crossover being applied is typically between 0.6 and 1.0. If crossover is not applied, offspring are produced simply by duplicating the parents. This gives each individual a chance of passing on its genes without the disruption of crossover.
Mutation is applied to each child individually after crossover. It randomly alters each gene with a small probability. The next diagram shows the fifth gene of a chromosome being mutated: The traditional view is that crossover is the more important of the two techniques for rapidly exploring a search space. Mutation provides a small amount of random search, and helps ensure that no point in the search has a zero probability of being examined.

CONVERGENCE

The fitness of the best and the average individual in each generation increases towards a global optimum. Convergence is the progression towards increasing uniformity. A gene is said to have converged when 95% of the population share the same value. The population is said to have converged when all of the genes have converged. As the population converges, the average fitness will approach that of the best individual. A GA will always be subject to stochastic errors. One such problem is that of genetic drift. Even in the absence of any selection pressure (i.e. a constant fitness function), members of the population will still converge to some point in the solution space. This happens simply because of the accumulation of stochastic errors. If, by chance, a gene becomes predominant in the population, then it is just as likely to become more predominant in the next generation as it is to become less predominant.

If an increase in predominance is sustained over several successive generations, and the population is finite, then a gene can spread to all members of the population. Once o gene has converged in this way, it is fixed; crossover cannot introduce new gene values. This produces a ratchet effect, so that as generations go by, each gene eventually becomes fixed. The rate of genetic drift therefore provides a lower bound on the rate at which a GA can converge towards the correct solution. That is, if the GA is to exploit gradient information in the fitness function, the fitness function must provide a slope sufficiently large to counteract any genetic drift. The rate of genetic drift can be reduced by increasing the mutation rate. However, if the mutation rate is too high, the search becomes effectively random, so once again gradient information in the fitness function is not exploited.

STRUCTURE OF PROPOSED GA-BASED ALGORITHM

Algorithm MCGA (Modified crossover GA)

(1) Begin

(2) Initialize Population (randomly generated);

(3) Fitness Evaluation;

(4) Repeat

(5) Selection(Roulette wheel Selection) ;

(6) Modified crossover;

(7) Inversion();

(8) Fitness Evaluation;

(9) Elitism replacement with Filtration;

(10) Until the end condition is satisfied;

(11) Return the fittest solution found;

(12) End

EXPERIMENTAL SETUP

The Individual solutions are randomly generated to form an initial population. Successive generations of reproduction and crossover produce increasing numbers of individuals . Modified crossover operator with crossover probability Cp is 0.5 and 1.0 is taken. This operator is suitable for permutation coding. Crossover operator exploited the population or you can say that it can diversify the population. But due to the genetic drift some time the population is converge to the local optimal point, At that time crossover operation can not diversify the population. The inversion operator is explorative in nature ,it diversify the population ,but in general the probability of inversion is very low . so in our simulation We first have 0.01 inversion probability then we proceed with .001,.0001. The parameter

setting for proposed genetic algorithm is as shown in table no .1.

Table 1: Parameters and strategies used for proposed genetic algorithm

Parameter / Strategy	**Setting**
Population Size	20
Population Type	Generational
Initialization	Random
Selection	Roulette wheel
Crossover	Two Parents, Modified crossover
Crossover Probability	0.5 and 1.0
Variable Inversion Probability	0.1 0.01 0.001
Replacement strategy	Keep 95 % Best
Stopping Strategy	85 % Population converge
No. of process to be Schedule	5
Fitness criterion	Minimum Turn Around Time

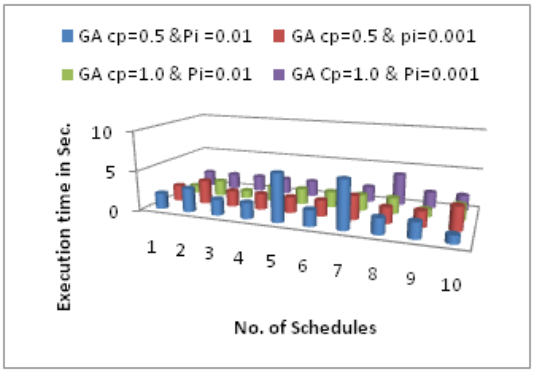

Figure 1. No. of schedules vs exec. time

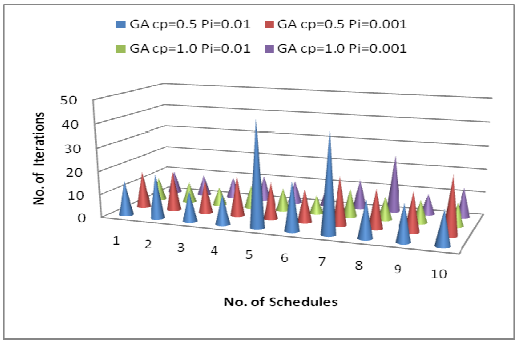

Figure 2: No. of schedules vs No. of iteration

Table 2: Computation results

Sr.No	**Burst Time of Jobs**					**GA $c_{p=0.5}$**		**GA$c_{p=1.0}$**	
	J1	**J2**	**J3**	**J4**	**J5**	Pi=0.01 E.T(sec.)	Pi=0.001 E.T(sec.)	Pi=0.01 E.T(sec.)	Pi=0.001 E.T(sec.)
1	20	39	49	45	43	2	2	1	2
2	45	47	46	44	27	3	3	2	2
3	43	29	45	47	30	2	2	1	2
4	25	46	48	33	42	2	2	2	2
5	28	31	46	54	24	6	2	2	2
6	20	47	50	32	48	2	2	2	1
7	20	48	43	41	21	6	3	2	2
8	29	24	20	44	48	2	2	2	4
9	40	39	42	24	44	2	2	1	2
10	34	39	22	47	40	1	3	2	2
$Total E.T = \sum_{i=1}^{i=10} ET_i$						28	23	17	21
$Mean E.T. = \frac{\sum_{i=1}^{i=10} ETi}{10}$						2.8	2.3	1.7	2.1

Table 3. Computation results

Sr.No	Burst Time of Jobs					GA $c_{p=0.5}$		GA$c_{p=1.0}$	
	J1	J2	J3	J4	J5	Pi=0.01 No.Itr.	Pi=0.001 No.Itr.	Pi=0.01 No.Itr.	Pi=0.001 No.Itr.
1	20	39	49	45	43	15	16	10	10
2	45	47	46	44	27	19	17	9	9
3	43	29	45	47	30	13	14	8	9
4	25	46	48	33	42	13	17	10	11
5	28	31	46	54	24	45	16	10	10
6	20	47	50	32	48	21	14	8	10
7	20	48	43	41	21	42	21	12	13
8	29	24	20	44	48	16	17	10	25
9	40	39	42	24	44	16	17	10	9
10	34	39	22	47	40	15	25	10	13
$Total\,NT = \sum_{i=1}^{i=10} NTi$						215	174	97	119
$Mean = \frac{\sum_{i=1}^{i=10} NTi}{10}$ $\frac{\sum_{i=1}^{i=10} NTi}{10}$						21.5	17.4	9.7	11.9

CONCLUSION

The experiment result shows that the convergence of the genetic algorithm is depend upon the parameter setting of the genetic algorithm. When cross over probability is set to Cp =1.0 and inversion probability is set pi=0.01 then convergence Time of GA is reduced considerably. But when Cp=0.5 and Pi=0.01 the convergence time of GA increased. Same result is getting when iteration is consider. So it is clear that the convergence of genetic algorithm is depends upon the parameter setting of getting algorithm. At a particular parameter setting we get optimal convergence state.

REFERENCES

[1] M. Pinedo,, (2001), Scheduling – Theory, Algorithms and Systems, 2ª edição, Prentice-Hall.

[2] Brucker, P. (2001). Scheduling Algorithms, Springer, 3rd edition, New York.

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

[4] R.Kumar,(2010),"Genetic algorithm approach to operating system process scheduling problem", International journal of Engineering science and Technology, pp 4248-4253

[5] L.Davis. Job-shop scheduling with genetic algorithms. Van Nostrand Reinhold,1990

[6] David E.Goldberg, Genetic Algorithms in Search Optimization & Machine learning, Second Reprint, Pearson Education Asia pte. Ltd.,2000.

[7] M.Srininivas and L.M.Patnaik,"Genetic Algorithms: A Survey", IEEE computer Magazine, pp.17-26, June 1994.