A Modified Genetic Algorithm for Process Scheduling in Distributed System

Vinay Harsora
B.V.M. Engineering College
Charatar Vidya Mandal
Vallabh Vidyanagar, India

Dr.Apurva Shah
G.H.Patel College of
Engineering & Technology
Charutar Vidya Mandal
Vallabh Vidyanagar, India.

ABSTRACT
The problem of process scheduling in distributed system is one of the important and challenging area of research in computer engineering. Scheduling in distributed operating system has a important role in overall system performance. Process scheduling in distributed system can be defined as allocating processes to processor so that total execution time will be minimized, utilization of processors will be maximized and load balancing will be maximized. The scheduling in distributed system is known as NP-Complete problem. Genetic algorithm is one of the widely used techniques for constrain optimization. Genetic algorithm is basically search algorithm based on natural selection and natural genetics. In this, paper using the power of genetic algorithms. We solve this problem considering load balancing efficiently. We evaluate the performance and efficiency of the proposed algorithm using simulation result.

Keywords
Distributed system, DAG, Genetic algorithm.

1. INTRODUCTION
The computational complicated process cannot be executed on the computing machine in an accepted interval time. Therefore, they must be divided into small sub-process. The sub-process can be executed either in the expensive multiprocessor or in the distributed system. Distributed system is preferred due to better ratio of cost per performance [1]. Scheduling in distributed operating systems is a critical factor in overall system performance. Process scheduling in a distributed operating system can be stated as allocating processes to processors so that total execution time will be minimized, utilization of processors will be maximized, and load balancing will be maximized. Process scheduling in distributed system is done in two phases: in first phase processes are distributed on computers and in second processes execution order on each processor must be determined [2].

The methods used to solve scheduling problem in distributed computing system can be classified into three categories graph theory based approaches [3], mathematical models based methods [4] and heuristic techniques [5].

Heuristic algorithm can be classified into three categories iterative improvement algorithms [17], the probabilistic optimization algorithms and constructive heuristics. Heuristic can obtain sub optimal solution in ordinary situations and optimal solution in particulars.

The first phase of process scheduling in a distributed system is process distribution on computer. The critical aspects of this phase are load balancing. Recently created processes may be overloaded heavily while the others are under loaded or idle. The main objectives of load balancing are to speared load on processors equally, maximizing processors utilization and minimizing total execution time [6].

The second phase of process scheduling in distributed computing system is process execution ordering on each processor. Genetic algorithm used for this phase. Genetic algorithm is guided random search method which mimics the principles of evolution and natural genetics [18]. Genetic algorithms search optimal solution from entire solution space. They often can obtain reasonable solution in all situations. Nevertheless, their main drawback is to spend much time for schedule. Hence, we propose a modified genetic algorithm to overcome from drawback through this paper.

In this paper using the power of genetic algorithms we solve this problem. Process distribution on different processor done based on processors load. The proposed algorithm maps each schedule with a chromosome that shows the execution order of all existing process on processors. The fittest chromosomes are selected to reproduce offspring: chromosomes which their corresponding schedules have less total execution time, better load balance and processor utilization. We assume that the distributed system processes are non uniform and non-preemptive, that is the processors may be different and a processor completes current process before executing a new one the load balancing mechanism used in this paper only schedule process without process migration.

Reset of the paper organized as follows, In section 2 the preliminaries in this section system and problem description and principles of the genetic algorithm are introduced. Section 3 our proposed genetic algorithm is explains. Section 4 for experiment result and section 5 for conclusions.

2. PRELIMINARIES

2.1 System and Process model
The system used for simulation is loosely coupled non-uniform system, all task are non-pre-emptive and no process migration are assumed. The process scheduling problem considered in this paper is based on the deterministic model. A distributed system with m processors, m>1 should be modelled as follows:

$P= \{p_1, p_2, p_3,..., p_m\}$ is the set of processors in the distributed system. Each processor can only execute one process at each moment, a processor completes current process before executing a new one, and a process cannot be moved to another processor during execution. G is an m × m matrix, where the element g_{uv} $1\leq u,v \leq m$ of G, is the communication delay rate between P_u and P_v. H is an m × m matrix, where the element h_{uv} $1\leq u, v \leq m$ of H, is the time required to transmit a unit of data from P_u and P_v. It is obvious that $h_{uu}=0$ and $r_{uu}=0$.

$T = \{t_1,t_2,t_3,...,t_n\}$ is the set of processes to execution. E is an n × m matrix, where the element e_{ij} $1 \leq i \leq n$, $1 \leq j \leq m$ of E, is the execution time of process t_i on processor p_j. In distributed systems the execution time of an individual process t_i on all processors is equal.

F is a linear matrix, where the element f_i $1\leq i \leq n$ of F, is the target processor that is selected for process t_i to be executed on. C is a linear matrix, where the element c_i $i \leq i \leq n$ of C, is the processor that the process t_i is presented on just now.

The problem of process scheduling is to assign for each process $t_i \in T$ a processor $f_i \in P$ so that total execution time will be minimized, utilization of processors will be maximized, and load balancing will be maximized.

The processor load for each processor is the sum of process execution times allocated to that process [2].

$$Load(pi) = \sum_{j=1}^{\substack{No.of\ allocated\\ processes\ on\\ processor\ i}} aj,i + \sum_{k=1}^{\substack{No.of\ new\ assigned\\ processes\ on\\ processor\ i}} ak,i$$

The length of schedule T is the maximal finishing time of all processes or maximum load [2].

$$Scheduling\ length = \max\big(load(pi)\big)$$

The processor utilization for each processor is obtained by dividing the sum of processing times by scheduling length. The average of process utilization is obtained by dividing the sum of all utilizations by number of processors [2].

$$Utilization(Pi) = \frac{Load(pi)}{scheduling\ length}$$

$$Average\ Utilizatio = \frac{\sum_{i=0}^{No\ of\ Processors} Utilization(pi)}{Number\ of\ processors}$$

2.2 Genetic algorithm [20]

Genetic algorithm is guided random search algorithm based on the principles of evolution and natural genetics. It combines the exploitation of the past results with the exploration of new areas of the search space. By using survival of the fittest techniques and a structured yet randomized information exchange, genetic algorithm can mimic some of the innovative flair of human search. Graph algorithm is randomized but not simple random walks. It exploits historical information efficiently to speculate on new search points with expected improvement.

Genetic algorithm maintains a population of candidate solutions that evolves over time and ultimately converges. Individuals in the population are represented with chromosomes. Each individual is numeric fitness value that measures how well this solution solves the problem. Genetic algorithm contains three operators. The selection operator selects the fittest individuals of the current population to serve as parents of the next generation. The crossover operation chooses randomly a pair of individuals and exchanges some part of the information. The mutation operator takes an individual randomly and alters it. As natural genetics, the probability of crossover is usually high, the population evolves iteratively (in the genetic algorithm terminology, through generation) in order to improve the fitness of its individuals.

The structure of genetic algorithm is a loop composed of a selection, followed by a sequence of crossovers and mutations. Probabilities of crossover and mutation are constants and fixed in the beginning. Finally, genetic algorithm is executed until some termination condition achieved, such as the number of iterations, execution time, execution time, result stability, etc.

3. PROPOSED GENETIC ALGORITHM

The newly created process on machine calculate load with it on each processor. These processes are assigned to less loaded or under loaded processors. For that each machine maintain table on it, of processor speed and communication channel speed etc. Here, our concentration on process scheduling instead of process distribution. Processes are non- pre-emptive and no process migration is assumed.

3.1 String representation

The genetic representation of individuals is called genotype. The main criteria in selecting the string representation for the search node is that the new string generated from the application of genetic operator must represents legal search node for the problem. A legal search node is one that satisfies (1) the precedence relation among the task. (2) Every process is present and appears only once in the schedule. The string representation used in this paper is an array of n × m digits, where n is the number of processes and m shows the processor that the process is assigned to. Process index shows the order of execution on that processor.

3.2 Initial population

Genetic algorithm search many nodes in search space. This requires us to generate an initial population of the search node randomly. The population size is typically problem dependent and has to be determined experimentally. Here, each solution *i* is generated as, one of the unscheduled processes is randomly selected, and then assigned to one of the processors. This operation is repeated until all of processes have been assigned.

An initial population with size of *POPSIZE* is generated by repeating this method.

3.3. Fitness function

The fitness function is essentially the objective function for the problem. It provides a mean to evaluate the search nodes and also controls the reproduction process. For the process scheduling in distributed system problem, we can consider factor such as load-balancing, processors utilization etc. We take into account this objective in following equation. Fitness function of schedule T is

$$f(T) = \frac{Average\ Utilization}{scheduling\ length}$$

this equation shows that a fitter solution has less scheduling length and higher processor utilization.

3.4 Reproduction

The reproduction process is typically based on the fitness values of the strings. The principal is that string with higher fitness value will have higher chance of surviving to the next generation. Here, to reduce searching time of genetic algorithm. We calculate fitness of the entire individual in population, picking the best individual of them and then form a group. We can make a slight modification to basic reproduction operation by always passing the best string in the current generation to the next generation. This modification will increase the performance of genetic algorithm.

3.5 Crossover

Crossover is generally used to exchange portions between strings. The crossover operation picks top two strings from best group. Random task Ti from one of these two strings picks and put on the new string with care of precedence relation. If Ti task at same position then Ti is coping on the same place for new string. If we randomly choose two same parents (A=B) and if one of parents e.g. A the best string we use operator mutation on the second B string. It is elitism. Otherwise we mutate the first set and child generate randomly.

3.6 Mutation

Mutation is used to change the genes in the chromosomes. Mutation replaces the value of a gene with a new value from defined domain for that gene. The mutation operator, first generate two random number *r* and *c* for the processor. The condition for that is a) r # c and b) set r isn't empty.
After that from set *r*, choose one task at random and remove them in the set *c*.

3.7 Termination condition

We can apply multiple choices for termination condition: maximum number of generation, equal fitness for fittest selected chromosomes in respective iterations.

We can now combined all the component discussed above to form the genetic algorithm for distributed process scheduling.

```
Modified Genetic Algorithm
{
        Randomly Create an initial population
        Assign a fitness value to each individual
        Form Group of best individual
  WHILE NOT termination criteria DO
  {
    Assign a priority value to the individual in group
    Choose two best individual from the group;
    Crossover surviving individuals;
    Mutation child;
    Recorded best individual in group and eliminate    worst one
in group.
   }
}
```

Fig-1 Modify genetic algorithm steps

4. EXPERIMENT RESULT

We have simulated our proposed algorithm on Pentium IV with 2.8 MHz Intel processor and 512 MB of RAM. For experiment test, we have used ready standard task graph from [21]. We have implemented program to simulate proposed algorithm. We have tried different values of population size, number of generation to find which values would steer the search towards the best solution. The measurement of performance of these algorithms was based on total completion time means scheduling length and average processor utilization. In our experiment, we have considered population size is 1000 and group size is 20. The default parameters were varied and the results collected from test runs were used to study the effects of changing these parameters.

We have studied the effect of increasing number of processes on scheduling length and average processor utilization. The obtain result are shown in Fig. 2 and Fig 3. A important point in Fig-3 is that when number of process is increased, higher utilization is obtained.

When the number of generations was increased our proposed algorithm had a better function. The obtained result are shown in Fig-4 and Fig-5. While the number of generation was increased the scheduling length was reduced, it is because the quality of the generated process assignment improves after each generation. A important point in these is that when number of generation was increased higher utilization is obtained.

5. CONCLUSIONS

Scheduling in distributed operating system has a significant role in overall system performance and throughput. We have present and evaluate new genetic algorithm based method to solve this problem in a way that simultaneously minimizes scheduling length and maximize average processor utilization and also minimize the computation time of genetic algorithm. Most of existing approaches tend to focus on one of the objectives. Experimental results prove that our proposed algorithm tend to focus on more objective simultaneously and optimize them.

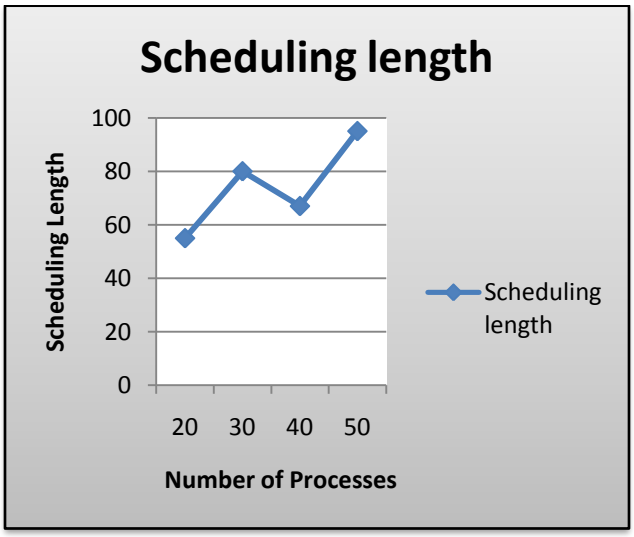

Fig-2 Relation of Scheduling Length with Number of Processes

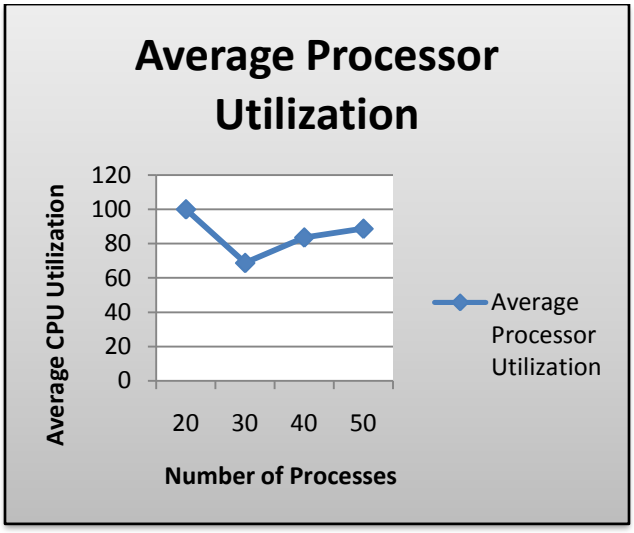

Fig-3 Relation of Average CPU Utilization with Number of Processes

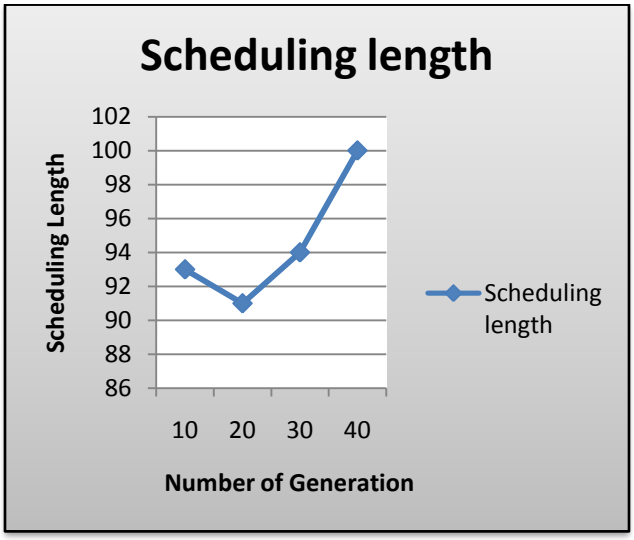

Fig-4 Relation of Scheduling Length with Number of Generation

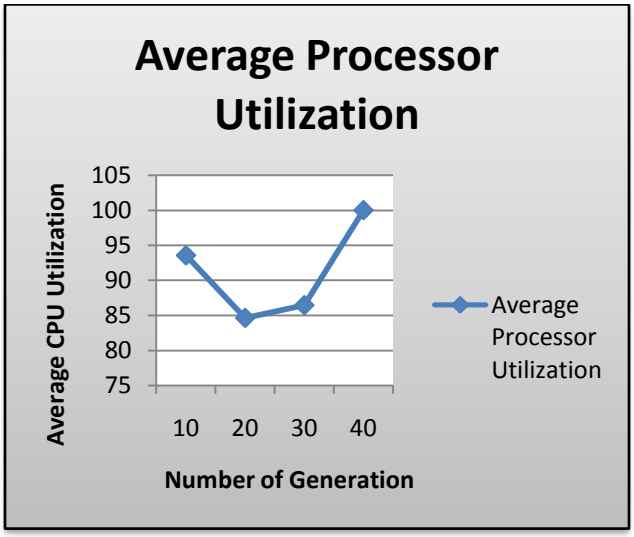

Fig-5 Relation of Average CPU Utilization with Number of Generation

6. REFERENCES

[1] Amir Masoud Rahmani and Mijtaba Rezvani " A Novel Genetic Algorithm for Static task scheduling in distributed systems", *International Journal of Computer Theory and Engineering* Vol. 1, No. 1 , April 2009 1793-8201.

[2] M. Nikravan and M.H.Kashani " A Genetic Algorithm For Process Scheduling in Distributed Operating systems considering Load Balancing", *European Conference on Modelling and simulation.*

[3] C.C.Shen, & W.H.Tsai, "A Graph Matching Approach to Optimal Task Assignment in Distributed Computing Using a Minimax Criterion", *IEEE Trans. On Computers, 34*(3), 1985, 197-203.

[4] P.Y.R.Ma, E.Y.S.Lee, & J.Tsuchiya, "A Task Allocation Model for Distributed Computing Systems", *IEEE Trans. On Computers, 31*(1), 1982, 41-47.

[5] W.Yao, J.Yao, & B.Li, "Main Sequences Genetic Scheduling For Multiprocessor Systems Using Task Duplication", *International Journal of Microprocessors and Microsystems, 28*, 2004, 85-94.

[6] G.L.Park, "Performance Evaluation of a List Scheduling Algorithm In Distributed Memory Multiprocessor Systems", *International Journal of Future Generation Computer Systems 20*, 2004, 249-256.

[7] A.T. Haghighat, K. Faez, M. Dehghan, A. Mowlaei, & Y. Ghahremani, "GA-based heuristic algorithms for bandwidth-delay-constrained least-cost multicast routing", *International Journal of Computer Communications 27,* 2004, 111–127.

[8] M. Moore, "An Accurate and Efficient Parallel Genetic Algorithm to Schedule Tasks on a Cluster", *Proceedings of the IEEE International Parallel and Distributed Processing Symposium*, 2003.

[9] V. D. Martino, "Sub Optimal Scheduling in a Grid using Genetic Algorithms", *Proceedings of the IEEE International Parallel and Distributed Processing Symposium*, 2003.

[10] C.I.Park, & T.Y.Choe, "An optimal scheduling algorithm based on task duplication" , *IEEE Trans. on Computers, 51*(4), 2002, 444–448.

[11] A.T. Haghighat, K. Faez, M. Dehghan, A. Mowlaei, & Y. Ghahremani, "Multicast routing with multiple constraints in high-speed networks based on genetic algorithms" , *In ICCC 2002 Conf.*, India, 2002, 243–249.

[12] A.Y.Zomaya, & Y.Teh, "Observations on Using Genetic Algorithms for Dynamic Load-Balancing", *IEEE Trans .On Parallel and Distributed Systems, 12*(9), 2001, 899-911.

[13] K.Qureshi, and M.Hatanaka, "A Practical Approach of Task Scheduling and Load Balancing on Hetrogeneous Distributed Raytracing Systems", *Information Processing Letters 79*, 2001, 65-71.

[14] L.M.Schmitt, "Fundamental Study Theory of Genetic Algorithms" , *International Journal of Modelling and Simulation Theoretical Computer Science 259,* 2001, 1 – 61.

[15] A.Y.Zomaya, C.Ward, & B.Macey, "Genetic Scheduling for Parallel Processor Systems: Comparative Studies and Performance Issues", *IEEE Trans. On Parallel and Distributed Systems*, 10(8), 1999, 795-812.

[16] S. Salleh, & A.Y. Zomaya, "Scheduling in Parallel Computing Systems: Fuzzy and Annealing Techniques", *Kluwer Academic*, 1999.

[17] M.Lin, & L.T.Yang, "Hybrid Genetic Algorithms for Scheduling Partially Ordered Tasks in a Multi-processor Environment", *Proc. of the 6th IEEE Conf. on Real-Time Computer Systems and Applications*, 1999, 382–387.

[18] D.Goldberg genetic algorithm in search optimization and machine learing reading Mass addssion wesly 1989.

[19] Sung-Ho Woo, Sung-Bong Yang, Shin-Dug Kim, Tack-Don Han, "Task scheduling in distributed computing systems with a genetic algorithm", *High-Performance Computing on the Information Superhighway*, HPC-Asia '97, 1997, p. 301.

[20] A.Y. Zomaya, C. Ward, and B.Macey "Genetic Scheduling for Parallel Process System: Comparative studies and Performance Issues", *IEEE Tansaction on Parallel and Distributed Systems*, Vol.10, No.8 pp795-812 Aug. 1999.

[21] Task graph downloaded from site http://www.kasahara.elec.waseda.ac.jp/schedule