ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology
Volume 1, Issue 3, May2012

Task Scheduling Using Compact Genetic Algorithm for Heterogeneous System

Neelu Sahu, Sampada Satav

Abstract— **In heterogeneous system task scheduling is a critical problem because tasks are distributed in many networks; the scheduling problem is known to be NP-complete. The task scheduling problem is the problem of assigning the tasks in the system in a manner that will optimize the overall performance of the application, while assuring the correctness of the result. In this paper we introduce the compact genetic algorithm (cGA) to solve the task scheduling problem in heterogeneous system. The cGA represent the population as a probability distribution over the set of solution. It processes each gene independently and requires less memory then the simple genetic algorithm. Therefore, it can be used to give a quick estimate of a problems difficulty. The goal of using this technique is use the given resources optimally and assign the task to the resources efficiently. The simulated result, show that compact genetic algorithm has better then genetic algorithm and cGA only needs a small amount of memory then GA.**

Index Terms— Compact genetic algorithm, Genetic algorithm, Heterogeneous system, Task scheduling, Scheduling.

I. INTRODUCTION

In heterogeneous system, task scheduling is prime significance problems. Scheduling is the most important issue in these type system and the problem of task scheduling on these type systems is verified the processor, when and on which processor a given task executes. Our goal is to minimize the mean task response time (total finish time) and meet the deadline at the same time. The scheduler has the dual responsibility of minimizing the execution time of the resulting schedule and balancing the load among the processors [4]. Even when the target processors are fully connected and when no communication delay is considered, scheduling is NP complete. Scheduling issue is handled by heuristic methods which provide reasonable solution of the problem [6]. Many of the techniques for this problem, such as Abraham et al and Braun et al presented three basic heuristics implied by Nature for Grid scheduling, namely Genetic Algorithm (GA), Simulated Annealing (SA) and Tabu Search (TS), and heuristics derived by a combination of their three algorithms. GA and SA are powerful stochastic optimization methods, which are inspired form the nature. In this paper we compare two algorithms first GA is simulated the evolutionary natural selection process [7]. The better solution of generation is evaluated according to the fitness value and the candidates with better fitness values are used to create further solutions through crossover and mutation processes, the GA can be viewed as implicitly modeling of the solutions seen in the search process. The poor behavior of genetic algorithms in some problems, sometimes attributed to designed operators, has led to the development of other types of algorithms. One of the simplest algorithms of the no dependencies model is the compact Genetic Algorithm (cGA). In this paper simulated results prove that compact genetic algorithm is proving to be better when it is applied for resource allocation in the field of heterogeneous system. We start by discussing the inspiration of this work from a random walk model. Then, we present the cGA and describe the mapping of the sGA parameters into those of an equivalent cGA. Along the way, computer simulations quality and speed.

Manuscript received May, 2012.

Neelu Sahu, *M.E (C.T.A), Department of Computer Science & Engineering,CSVTU / SSTC, Bhilai, India, 09827180937.*

Sampada Satav *Assistant Professor Department of Computer Science & Engineering,CSVTU/SSTC,Bhilai, India.*

II. TASK SCHEDULINGISSUE IN HETEROGENEOUS SYSTEM

Scheduling problem is significant problem of heterogeneous environment because tasks are distributed in many networks. The problem of task scheduling arises in a situation where there is more number of task then the available resources [13]. Consider a scenario where there are A= {1, 2, 3…n) task to be done and there are B= {1, 2, 3…m} resources available. In this condition task are not allowed to migrate between the resources. In such a situation if we have number of resources is greater than the number of task (B>A) then there is no reason for developing new algorithms for task scheduling because then resources can be allocated to the tasks on first come first serve basis, but if number of resource is less than the number of task (B<A) then we need to develop new algorithms for task scheduling because now inefficient resource allocation can greatly hamper the efficiency and throughput of the scheduler. To formulate the problem, define T_a = {1, 2, 3…n} as n is independent task permutation and R_b = {1, 2, 3…m} as m is computing resources [1]. For example a set of printers which are used for printing a set a documents. The overall objective of task scheduling is to minimize the completion time and to utilize the resources effectively and usually it is easy to get the information about the ability to process data of the available resource [4].

Suppose that the processing time $P_{a,b}$ for task 'a' computing on 'b' resource is known. The completion time P(x) represent the total cost time of completion. The objective is to find a permutation matrix y= ($Y_{a, b}$), such that:

$Y_{a, b}$=1 if resource 'b' performs task 'a'.

Else

$Y_{a,\ b}$=0 that means number of resource are not perform number of task. Which minimize total cost:

$$P(X) = \sum\sum P_{a,\ b} * Y_{a,\ b}$$

Subject to:

$$\sum Y_{a,\ b} = 1;\ \forall b \in T$$

$$Y_{a,\ b} \in \{0, 1\},\ \forall a \in R;\ \forall b \in T$$

The minimal $P(x)$ represents the length of schedule whole tasks working on available resources. The scheduling constraints guarantee that each task is assigned to exactly one resource. We will discuss that a new optimal schedule is able to find the minimal completion time. To solve the task scheduling problem we have used the Compact genetic algorithm (cGA). We set an initial population by selecting a random starting sequence from the set of x! Sequences; where x is the total number of tasks. After getting the initial solution we calculate fitness value of each solution, according to equation. After that we calculate best among the entire solution and set it as an initial global best.GA update equation is used to update old population and generate new sequences and then their resources are calculated. These sequences, along with their resources are then used to find the fitness value of each individual of each solution of the population. After this if crossover criteria is satisfied, then crossover operation performed over two randomly selected parents and as a result a new sequence is generated. Then the resource of this offspring is calculated. Using the sequence and its resources the fitness value of the offspring is calculated. Based on the fitness value, if the offspring is better than its worst parent then this solution replaces that parent.

III. GENETIC ALGORITHM

Genetic Algorithms are search and optimization techniques based on Darwin's Principle of Natural Selection. They can be used to solve Classification Problems. Genetic algorithm (GAs) is stochastic search mechanisms [2]. They are inspired by the mechanics of (Darwinian) natural selection and genetics. They have been successfully used in a wide variety of applications in business, engineering, fuzzy logic and science. All the issues are used with GAs—such as population size, chromosome, encoding methods and genetic operators, etc. Genetic operators are selection, crossover, and mutation [1]. The population size that guarantees an optimal solution quickly enough has been a topic of intense research. This is because large populations generally result in better solutions, but at increased computational costs and memory requirements. GA combines the exploitation of 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. Lee and Cheng Chen [10], use partitioned GA and incorporates random heuristic. Individuals are feasible schedule, which are represented as task-processor pair. Traditional one-point Crossover operator is used. Mutation operators are involves swapping task. R.Deepa, T.shrinivasan [6], developed the first population-sizing equation based on the variance of fitness. Georges, Goldberg [12], are introduce compact genetic algorithm and compares two algorithms and I am using in my project some application with compact genetic algorithm and also compare both genetic and compact algorithm. Harik *et al.* Exploited the similarity between the gambler's ruin problem and the selection mechanism of GAs for determining an adequate population size that guarantees a solution with the desired quality. Furthermore, the analytic model started from the assumption that the fitness values of a pair of chromosomes can be ordered. This effectively implies tournament selection without replacement. Moreover, Ahn and Ramakrishna [11], further enhanced and generalized the population sizing equation, so as to dispense with any (problem dependent) stochastic information such as signal or collateral noise of competing BBs. In an attempt to understand the real importance of population in evolutionary algorithms (EAs), He and Yao showed that the introduction of population increases the first hitting probability, so that the mean first hitting time is shortened. Although all algorithms for the no dependencies model have low efficiency in solving difficult problems, it is still important to study them due to their simplicity in terms of memory usage and computational complexity and with respect to the fact that the computational complexity of the bivariate dependencies model and the multiple dependencies model is high. Lei and chen [13] showed scheduling algorithm based on PSO for grid computing. For solving any problem by genetic algorithm, eight components must be defined:

Representation (definition of individual): Representation represents each chromosome in the real world. A chromosome is a set of parameters which define a proposed solution to the problem that the genetic algorithm is trying to solve. In the context of task scheduling, a chromosome would contain the processor ids and task ids as sets of parameters [6]. Fitness function: These function shows the fitness of each chromosome [2]. It is used to evaluate the chromosome and also controls the genetic operators. Population: The role of the population is to hold possible solution [4]. Parent selection mechanism: The role of parent selection is to distinguish among individuals based on their quality, in particular, to allow the better individuals to become parents of the next generation [3]. Recombination operators: Recombination operator selects two chromosomes and then produces two new children from them [4], [5]. Mutation operators: Mutation operator selects one chromosome and then produces one new child from it by a slight change over the parent. Survivor selection mechanism: The role of survivor selection is to distinguish among individuals based on their quality. Termination Condition: The condition to ending the running of genetic algorithm [2].

IV. COMPACT GENETIC ALGORITHM

the compact(cGA) as an estimation of distribution algorithm (EDA) that generates offspring population according to the estimated probabilistic model of parent population instead of using traditional recombination and mutation operators [3]. The cGA initializes a probability (distribution) vector (PV) over the set of solutions and two solutions are randomly generated by using this PV. The generated solutions are ranked based on their fitness values. The cGA represents the population as a PV over a set of solutions and operationally mimics the order-one behaviour of simple GA (sGA) with uniform crossover using a small amount of memory. When confronted with easy problems (e.g., continuous-unimodal problems involving lower order BBs), the cGA achieves the

performance of the SGA (with the uniform crossover) in terms of the number of fitness evaluations. That the cGA may not be effective in solving real-world problems. In order to obtain better solutions to such difficult problems, the cGA should exert a higher selection pressure. This, in turn, increases the survival probability of higher order BBs, thereby preventing loss of the best solution found so far [12]. In other words, higher selection pressure may play the role of memory. Therefore, it can take care of a finite number of decision errors and some linkage information of genes. Selection pressure of the cGA can be increased by creating a larger tournament size in a simple manner.

Goldberg and Miller analyzed the growth and of a particular gene in the population as a one-dimensional random walk. As the GA progresses, genes fight with their competitors, and their number in the population can go up or down, depending on whether the GA makes good or bad decision. These decisions are made implicitly by the GA when selection takes place. The next section explores the effects of this decision making.

Selection

Selection gives to more copies to better individuals. But it does not always do so for better genes. This is because genes are always evaluated within the context of a larger individual. For example, consider the onemax problem (that counting ones). Suppose individual a competes with individual b.

Individual	chromosome	fitness
a	1011	3
b	0101	2

When these two individuals compete, individuals a will win. However, at the level of the gene, a decision error is made on the second position. That is, selection incorrectly prefers the schema 0 to 1. The role of the population is to buffer against a finite number of such decision errors. Imagine the following selection scheme: pick two individuals randomly from the population and keep two copies of the better one. This scheme is equivalent to a steady-state binary tournament selection. In a population of size n, the proportion of the winning alleles will increase by 1/n. For instance, in the previous example the proportion of 1's will increase by 1/n at gene position 1 and 3, and the proportion of 0's will also increase by 1/n at gene position2. At gene position 4, the proportion will remain the same. This thought experiment suggests that an update rule increasing a gene's proportion by 1/n simulates small steps in the action of a GA with a population of size n. The next section explores how the generation of individuals from probability distributions mimics the effects of crossover.

Crossover

The roll of crossover in the GA is to combine bits and pieces from fit solutions. A repeated application of most commonly used crossover operators eventually leads to a decorrelation of the population's genes. In this decorrelation state, the population is more compactly represented as a probability vector. Thus the generation of individuals from this vector can be seen as a shortcut to the eventual aim of crossover.

V. PROPPSED METHODOLOGY

In this paper we have proposed a solution for heterogeneous system using compact genetic algorithm. For solving any problem we have to first consider task and resources into matrix form.

Pseudo code of the cGA

Compact GA (n, N, fitness)

P= allocate vector of n real number;

For i: =1 to n

do p [i]:= 0.5;

t=0;

Generate two solutions from probability vector

a:= ***generate*** *p[i]; b :=* ***generate*** *p[i];*

Compete both solutions

if (fitness (a)> fitness (b)) then

W = a;

Else

L = b;

t= t+1;

*(*Where W is winner and L is loser*)*

Update the probability vector

d=1/n;

For i: = 1 to n do

If(W [i] >L[i])then

p[i]:=p[i] +d;

else

p[i]:= p[i] -d;

Check if the probability vector has converged.

Go to Step2, if it is not satisfied.

The probability vector represents the final solution.

We take task and resources into matrix form and find the minimum cost. Suppose no of task is T_i= (1, 2, 3……..n) where n is an independent task, and R_j= (1, 2, 3…….m) where m is a computing resources. Find the total cost of matrix by:

$$C(x) = \sum\sum P_{i,j} * M_{i,j}$$

Where $P_{i,j}$ is a processing time for task i computing on resources j and $M_{i,j}$ is permutation matrix, if $M_{i,j}$= 1then resource j perform task i, else $M_{i,j}$= 0.

In cGA first we take number of task and number of resources in matrix form, then each task assign public because any task are used by any resources.

Next step create individual class it is sub task of a genetic algorithm.

Now we use compact genetic algorithm and each iteration cGA manage its population as a probability vector is PV,

Probability vector is initialized with parameter 0.5 to represent a randomly generated population. In each generation (i.e. iteration), generate the individuals from the probability vector and find out the best one and then the

position vector is updated to favour the better chromosome (i.e. winner).
Let the best individuals are ‘a’ and ‘b’ then compete both individuals, if both individuals fitness value is same then we assign ‘a’ is winner and update the probability vector along the way. Clearly the best individual wins all the competition. The cGA terminates when all the probabilities converge to zero or one.

VI. CONCLUSIONS

In this paper we present compact genetic algorithm, an algorithm that mimics the order one behavior of simple genetic algorithm with a given population size and selection rate, but that reduce its memory requirement. In this paper we explained of the compact algorithm. I am showing the result of each no. of task is assign by no. of resources and calculates fitness value. I will show my final paper which task is minimum cost that will directly show the value of task and resources.

VII. ACKNOWLEDGEMENT

I would like to thank my project guide for guidance and thank H.O.D sir for using Computer lab and thank Director sir for using resources and thank colleague and friends for supporting and thank the anonymous referees for their helpful comments and suggestion that have improved the quality of this manuscript.

REFERENCES

[1]R.Nedunchelian, K.Koushik, N.Meiyappan, V.Raghu,“Dynamic Task Scheduling Using Parallel Genetic Algorithm for heterogeneous Distributed System”

[2] Edwin S.H. Hou, Nirwan Ansari, Hong Ren, “A Genetic Algorithm for Multiprocessor Scheduling,” IEEE Transaction On Parallel And Distributed System, 1994, Vol.5, No.2, pp.113-120.

[3] Reza Rastegar, Arash Hariri, “A Step Forward in Studying the Compact Genetic Algorithm”, 2006 by the Massachusetts Institute of Technology, Vol.14, No.3, pp.277-289.

[4] Javier Carretero, Fatos Xhafa, “Genetic Algorithm Based Scheduling for Grid Computing Systems”, International Journal of Innovative Computing, Information and Control, Volume 3, Number 6, 2007.

[5] Chatchawit Aporntewan, Prabhas Chongstitvatana,“A Hardware Implementation of the Compact Genetic Algorithm”,IEEE congress of evolutionary computation Seoul Korea, may 2001.

[6] R.Deepa, T.Srinivasan, “An Efficient Task Scheduling Technique in Heterogeneous Systems using Genetic Algorithm”.

[7] Lee Wang, Howard Jay Siegel, Vwani P. Roychowdhury, “Task Matching and Scheduling in Heterogeneous Computing Environments Using a Genetic-Algorithm-Based Approach,” Journal of Parallel and Distributed Computing,1997 ,pp.8-22.

[8] Shijue Zheng, Wanneng Shu, and Shangping Dai,” Task Scheduling Model Design Using Hybrid Genetic Algorithm”, First International Conference on Innovative Computing, Information and Control (ICICIC'06),2006 IEEE.

[9] Vahe Aghazarian, Arash Ghorbannia, Nima Ghazanfari Motlagh, Mohsen Khajeh Naeini,” RQSG-I: An Optimized Real time Scheduling Algorithm for Tasks Allocation in Grid Environments,” 2011 IEEE.

[10]Yi-Hsuan Lee and Cheng Chen, .A Modified Genetic Algorithm for Task Scheduling in Multiprocessor Systems. 2003.

[11] Ahn, C. W. and Ramakrishna, R. S. (2003). Elitism-based compact genetic algorithms. IEEE Trans. Evolutionary Computation, 7(4):367–385.

[12]George, Goldberg and lobo, “The Compact genetic algorithm”, 1998IEEE.
[13]Lei, chen, jing and bo yang, “Task scheduling algorithm based on pso for grid computing”, International Journal of Computational Intelligence Research. Vol.4, No.1 (2008), pp. 37–43.

Neelu Sahu received the B.E degree in computer science & engineering from the Institute of Technology, Guru Ghasidas Vishwavidyalaya, Bilaspur, India**,** in 2010. And Pursuing M.E from the Shri Shankaracharya Group of Institutions, Bhilai, India.

Sampada Satav has done her M.Tech (CSE) in 2011 and B.E(CSE) in 2009 from Rungta College of Engg. and Tech.,Bhilai(C.G),India. She was University 2nd topper in her P.G.. Presently she is working as an Asst. Professor in Dept. of CSE in Shri Shankaracharya.