ISSN: 2278-7798
International Journal of Science, Engineering and Technology Research (IJSETR)
Volume 1, Issue 2, August 2012

Task Scheduling for Heterogeneous System Using CGA with Linear Crossover

Neelu Sahu, Sampada Satav

Abstract— **In this paper we present Compact Genetic Algorithms (cGAs) based schedulers for efficiently allocating jobs to resources in a Heterogeneous system. Scheduling in heterogeneous systems has a significant role in overall system performance and throughput. An efficient scheduling is vital for system performance.Task scheduling is a fundamental issue in achieving high performance in Heterogeneous systems. The main purpose of this paper is reducing the repeating of the generations in Genetic Algorithm for reaching higher speed and also considering the communication costs (available in fitness function) with maintaining the fitness efficiency. Scheduling of tasks in a heterogeneous computing (HC) environment is a critical task. It is also a well- known NP-complete problem; the results show the ability of the cGA to locate the optimal solution of problems, considered with a computational effort, comparable to improved population-based GAs and with much fewer storage requirements.**

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

I. INTRODUCTION

Heterogeneous System is a promising approach to meet the ever-increasing computational requirements [3]. Scheduling is the most important issue in the heterogeneous system because the effectiveness of the scheduling directly corresponds to the parallelization obtained. With inappropriate scheduling mechanisms can fail to exploit the true potential of the heterogeneous system. The task scheduling is prime significance of heterogeneous systems. The problem of task scheduling on these type systems is verified the processor, when and on which processor a given task executes. The scheduler has the dual responsibility of minimizing the execution time of the resulting schedule and balancing the load among the processors. Even when the target processors are fully connected and when no communication delay is considered, scheduling is NP complete. Scheduling is usually handled by heuristic methods which provide reasonable solution of the problem. If the number of tasks and computing systems are too high, finding the optimal or sub-optimal task scheduling would be Time-consuming and in some cases it consumes more time than a random execution of tasks. Hence, we must use the heuristic algorithms based on the problem conditions instead of employing the classic methods such as the back tracking and the dynamic programming. Heuristic optimization algorithm is widely used to solve a variety of NP-complete problems. Abraham et al and Braun et al [2] presented three basic heuristics implied by Nature for scheduling, namely Genetic Algorithm (GA) [3-7], Simulated Annealing (SA) [8-9] and Tabu Search (TS) [10], and heuristics derived by a combination of their three algorithms. GA and SA are powerful stochastic optimization methods, which are inspired form the nature. GA is simulated the evolutionary natural selection process. 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. Simulated annealing is based on the processes of annealing about the solid matter in physics. Both methods are valid and have been applied in various fields due to their strong convergence properties. In this paper, Compact genetic algorithm is employed to solve the scheduling problem in a heterogeneous environment. Through a serial of simulated experiments, our results show that cGA is effective for task scheduling in heterogeneous system.

Manuscript received July 15, 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. SCHEDULING ISSUE 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 Ta = {1, 2, 3…n} as n is independent task permutation and Rb = {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:

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

Subject to:

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

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

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.

V.CGA WITH LINEAR CROSSOVER OPERATOR

Compact genetic algorithm is a population based heuristic search technique. Compact genetic algorithm uses iterative process to search the global optima in solution space. Crossover operator with compact genetic algorithm has a property of better exploration in initial generations so by using crossover search area is explored in a relatively better manner even in later generations. CGA has a higher convergence rate, by using crossover with CGA premature convergence is also reduced so that CGA does not get trapped in local optima. Crossover can help the particles to jump out of the local optima by sharing the others information. Two Particles generated by CGA are randomly selected for crossover operation and two new offspring are formed. The best offspring (in terms of fitness) is selected from the new offspring. This new best offspring replaces the worst parent particle which is selected for crossover. The replacement is done if the new best offspring has the good fitness value than the parent particle.

LINEAR CROSSOVER

Linear crossover operator produces two off springs from a pair of parents by randomly selecting a cross site between 1 and n, parents solution dimension and replacing the former and latter half of each parents from the cross site. Let (X1, X2, X3, X4…….. Xn) and (Y1, Y2, Y3, Y4……...Yn) are two parent solutions. A cross site of value will produce the offspring shown in below
(Y1, Y2, Y3, Y4………Yn)
(X1, X2, X3, X4………Xn).

VI. 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.

```
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.
```

ISSN: 2278-7798
International Journal of Science, Engineering and Technology Research (IJSETR)
Volume 1, Issue 2, August 2012

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:

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

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.

VII. EXPERIMENTAL RESULTS AND DISCUSSION

This section focuses on the efficiency of the proposed algorithm compact genetic algorithm with linear crossover to solve heterogeneous system task scheduling. This section shows the experimental results and the parameter setting of the proposed algorithm.

EXPERIMENTAL SETUP

For every algorithm there are some control parameters which are used for its efficient working. Hence, there are some control parameters for Compact genetic algorithm with Linear Crossover operators also. We did an extensive literature survey and carried out our own experiments for determining the values of these control parameters. From this we found that the values which we have taken in this experiment are standard and they are also suitable for this experiment. The first control Parameter is we use genetic algorithm and take 10 individuals and each individuals contain 10 resources and 30 task each task assign the number of resources and calculate fitness value of each individuals.
The next parameter we use compact genetic algorithm and take parameters are probability vector, winner, looser and fitness value of individuals. We take two population ‘a’ and ‘b’ calculate fitness of both population if the fitness value a> b then we assign ‘a’ is win (winner), and ‘b’ is los (looser).
In this experiment we are using the feature of linear crossover Operator in the Compact genetic algorithm. The control parameter for Crossover operator is Probability. Therefore we need to find the value of this parameter also. We compare previous population T1 and T2 to another population fitness value i1 and i2 with probability vector 0.5 if the fitness T>I them print T1 is “first child” and T2 is “second child” otherwise i1 and 12 print.

EXPERIMENTAL RESULTS

In this section we analyze the result obtained by our algorithm. In this section we analyze the result obtained by our algorithm. To test the efficiency of our algorithm results of compact genetic algorithm with linear crossover is results. In scheduling task we already have the information about the number of resources, number of tasks, and the amount of time that will be taken by a resource to complete a task. We just need to find the sequence which will provide us the optimal results. We conducted the experiment by varying the number Of resources as well as varying the number of tasks. In particular, we have taken three cases in which we have taken different number of resources and tasks.
Here, we are provided with 10 resources and 30 tasks. Given below are the execution time of Genetic algorithm and CGA with linear Crossover operator with probability.
From the table, it can be concluded that by Compact genetic algorithm with linear crossover we get better results when the probability is 0.5.

Table 1. Execution time calculated by Genetic algorithm for 10 resources allocated to 30 tasks.

256	216	232	109	129	259	73	21	284	55
74	51	192	52	116	55	56	265	43	210
60	233	112	215	213	5	279	2	183	295

Shows the fitness value of each 10 task are assign in 10 resources

Table 2. Take 10 individuals from genetic algorithm and apply Compact genetic algorithm with probability 0.5 then

Task string is

24	12	4	3	16	26	5	20	29	11
6	7	0	27	1	8	21	9	13	2
14	10	28	15	25	17	18	23	19	22

Resource string is

4	2	4	3	6	6	5	0	9	1
6	7	0	7	1	8	1	9	3	2
4	0	8	5	5	7	8	3	9	2

Table 2 shows the numbers of tasks are assign the number of resources and calculate fitness value of all individuals.

Table3.**Apply Compact genetic algorithm with linear crossover**

First Child

9	11	17	18	12	10	20	6	4	14
13	21	15	16	19	22	23	24	25	26
8	27	28	29	0	1	2	3	5	7

Second Child

0	1	29	2	3	9	4	18	5	6
7	8	12	19	15	10	11	14	13	20
22	21	16	17	23	27	26	24	25	28

Table 4 Compare both fitness value of both parents with probability vector 0.5 values

Best Task String is

4	**27**	**20**	**16**	**6**	**3**	**7**	**0**	**21**	**22**
25	**29**	**1**	**23**	**2**	**5**	**11**	**8**	**24**	**9**
10	**12**	**17**	**26**	**28**	**13**	**14**	**15**	**18**	**19**

Best Resource String is

1	**3**	**9**	**6**	**0**	**9**	**8**	**2**	**2**	**4**
0	**4**	**1**	**6**	**7**	**5**	**7**	**3**	**8**	**8**
5	**9**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**

Best Fitness is 3199.0

CONCLUSION

In this paper we present compact genetic algorithm with linear crossover with probability vector 0.5and CGA 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. There is no specific value for crossover probability for which we can obtain best results for task scheduling. Show the best task string and best resource string are show best fitness value. It depends upon number of tasks and number of resources. As future work we have the intention to apply other types of nature inspired algorithms to the heterogeneous scheduling problem, comparing their results with the ones accomplished by the CGA with Linear.

VIII.ACKNOWLEDGMENT

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.