Task Scheduling in Parallel Systems using Genetic Algorithm

Rachhpal Singh
Sr. Asstt. Professor, Department of Computer Sc. and Applications
Khalsa College,
Amritsar-Punjab

ABSTRACT
The common problem of multiprocessor scheduling can be defined as allocating a task graph in a multiprocessor system so that schedule length can be improved. Task scheduling in multiprocessor system is a NP-complete problem. A number of heuristic methods have been cultivated that achieve partial solutions in less than the minimum computing time. Genetic algorithms have obtained much awareness as they are robust and provide a good solution. In this paper, genetic algorithm based on the principles of evolution to obtain an optimal solution for task scheduling is developed. Genetic algorithm is based on three operators: Natural Selection, Crossover and Mutation. The simulation results prove that the method proposed generates better results.

Keywords
Parallel computing, Heterogeneous system, Task scheduling, Task duplication, Schedule length and Load balance.

1. INTRODUCTION
Parallel systems are important resources that are generally shared by communities of users. Users frequently submit jobs to the system, each with unique resource and service-level requirements as well as value to the user and resource owner. The charge of job scheduling is to decide when and how each job should carry out in order to exploit the system's cumulative value to its owners [1]. The difficulty of scheduling a set of circulated resources for parallel execution of tasks connected with a single job arises in a wide range of circulated computer applications, manufacturing systems and communication network environments. The way in which the processors are allocated to the tasks of such parallel applications is fundamental to realizing the high performance [2] of the corresponding systems, such as minimizing mean job response time and maximizing system throughput.

Task scheduling [3] can be classified as static and dynamic. Some study basis makes static scheduling enviable. First, static scheduling occasionally results in lesser execution times than dynamic scheduling. Second static scheduling permits only single process per processor, which leads to reduced process creation and termination overhead. Third, static scheduling can be used to foretell the expedite that can be attained by a meticulous parallel algorithm on a target machine, presuming that no pre-emption of processes occurs. The scheduling problem has - next to search for a most favourable mapping of the task and their sequence of execution and also search for a best possible configuration of the parallel system. The [4] computational convoluted course cannot be executed on the parallel computing machine in an acknowledged interval time. So to avoid such type of hindrance, the big task must be divided into small sub-process and further the sub-process can be executed either in the expensive multiprocessor or in the parallel distributed system. An entire parallel system is superlative and can be employed due to its cost and performance ratio.

There are two different types of scheduling problems, they are given below:

(1) Job scheduling and

(2) Task scheduling.

Job scheduling compacts with the scheduling of autonomous jobs; whereas tasks scheduling is linked to scheduling of tasks belonging to a solitary application program. In general the aim of job scheduling is to have first-rate load balancing amid the processors, whereas for the later minimization of overall execution time is the main concern [5]. The chief intention behind both is to exploit the parallel system's throughput by effecting maximum number of jobs in the given time span. Load balance [6] is considered as a major problem in parallel computing when scheduling is for multiple jobs with limited resources where more number of jobs is required to run on the same processor again and again. The load balance should be minimized to prevent the system throughput and efficiency thus the job is optimized with load balance also to run on the appropriate processors.

Heuristic optimization algorithm [6] is broadly used to solve a diversity of NP-complete problems. Abraham et al and Braun et al [7] proposed three basic heuristics implied by Nature for Grid scheduling, namely Genetic Algorithm [8], Simulated Annealing [9] and Tabu Search [10], and heuristics derived by a combination of these 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 improved solution of generation is estimated according to the fitness value and the candidates with better fitness values are used to generate additional solutions through crossover and mutation processes. Simulated annealing is based on the process 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.

If these criteria's are considered then there will be another issue often occurred in parallel computing, called processor idle time. During the optimization some processors may be left alone by does not allocate any jobs on to them where the act is called as an idle time of a processor. This idle time is subject to discard by assigning task duplication in which a task is duplicated and allocated to respective processors that may have dependent relationship with some other tasks and are executed concurrently. These are the significant concerns takes place in parallel processing which encourage performing various investigations and optimization techniques for scheduling to conquer the problems arise in parallel computation. In this paper, the task scheduling problem

having some specific characteristics are discussed, genetic approach is discussed in detail in the next section and the last section presents experiments and results.

The paper is structured as section 2 specifies the related task scheduling works in literature; the genetic algorithm is explained in section 3 and the experimental results are discussed in the next section. The work is finally concluded in section 5.

2. RELATED WORKS

Moraglio et.al [11] have presented a new technique that uses a population of Taboo Search runs in a Genetic Algorithm structure: GAs focuses high-quality areas of the solution space so that TS can start its search with capable initial solutions. The curiousness of the Genetic Algorithm they propose consists in a natural demonstration which covers all and only the feasible solution space and assurances the transmission of meaningful description. The results show that this technique outperforms many others generating best worth results in less time.

Ratan Mishral and Anant Jaiswal [12] have proposed this paper based on Ant Colony optimization to resolve the problem of load balancing in cloud environment. In this paper, a heuristic algorithm based on ant colony optimization has been anticipated to instigate the service load distribution underneath cloud computing architecture. The pheromone update mechanism has been proved as an efficient and effective tool to balance the load. This adaptation ropes to minimize the make span of the cloud computing based services and portability of checking the request also has been congregating using the ant colony optimization technique.

Z. Pooranian et.al [13] proposed this paper for the rationale of task scheduling. In this paper the researchers unite the genetic algorithm and GELS (GAGELS) as a scheme to resolve scheduling trouble by which concurrently pay attention to two factors of time and number of missed tasks. Results illustrate that the anticipated algorithm can diminish make span while decreasing the number of missed tasks contrasted with the conventional methods.

Dervis Karaboga and Bahriye Basturk [14] have proposed this paper to demonstrate the evaluation results on the presentation of the Artificial Bee Colony (ABC) algorithm for embarrassed optimization exertions. The ABC algorithm has been initially proposed for unhindered optimization troubles and showed that it has better performance on these kinds of problems. In this paper, the ABC algorithm has been unmitigated for solving inhibited optimization problems and functional to a set of constrained problems.

Mohammad Shojafar et.al [15] have proposed this work by focus of the present involvement, where they have urbanized a new hybrid scheduling algorithm GGA that coalesces GA and the gravitational emulation local search (GELS) algorithm. The remarkable feature of the proposed best possible scheduler is that it reduces runtime and the number of acquiesced tasks whose deadlines are missed. An evaluation of the concert of their proposed joint optimal scheduler to similar techniques shows that it produces more optimal computation time.

Rizos Sakellariou and Viktor Yarmolenko [16] have proposed this paper, they squabble for the need to afford more flexibility in the level of service accessed by Grid-enable high-performance, parallel, supercomputing possessions. It foresees that such need could be contented by making detach Service Level Agreements (SLAs) between the resource owner and the user who wants to submit and run a job on these resources. A number of issues related to the materialization of this vision are highlighted in the paper.

Vishnu Kant Soni et.al [17] have proposed this paper and evaluates an extension from Computational-Communication to Computational- Communication-Memory based Grouping Job Scheduling strategy. This approach exploits the consumption of Grid resources, decreases processing time of jobs and network delay to programmed and execute jobs on the Grid. The sculpt exchanges light weight jobs into coarse-grained job or grouped job according to the necessity jobs and source capacity. This Grouping technique courts the processing power, memory-size and bandwidth desires of each job to recognize the real grid system. The investigational grade shows that the proposed scheduling algorithm resourcefully reduces the processing time of jobs in comparison to others.

U. Karthick Kumar et.al [18] have proposed this paper, that a Load balancing algorithm for fair scheduling, and they compare it to other scheduling schemes for a computational grid. It addresses the fairness issues by using mean waiting time. It scheduled the task by using fair completion time and rescheduled by using mean waiting time of each task to attain load balance. This algorithm method tries to offer optimal solution so that it diminishes the execution time and expected price for the execution of all the jobs in the grid system is minimized.

S. Selvi et.al [19] have proposed this paper in which they introduce a novel approach based on Differential Evolution algorithm for scheduling jobs on computational grid. The proposed approach creates an optimal programmed so as to fulfill the jobs within a minimum period of time and exploiting the possessions resourcefully. Grid computing refers to the amalgamation of computer assets from multiple secretarial provinces to reach common goal. Grids offer a way of using the information technology possessions optimally inside an association. Grid environments assist distributed estimation. Hence the scheduling of grid jobs should be measured as an important issue.

Jim Blythe et.al [20] have proposed this paper by recognizing two families of resource allocation algorithms: task-based algorithms, that greedily allocate tasks to resources, and workflow-based algorithms, that search for a proficient allotment for the complete workflow. They evaluate the performance of workflow-based algorithms and task-based algorithms, using simulations of workflows drawn from a real submission and with varying ratios of computation cost to data transfer cost. They examine that workflow-based advances have a probable to work better for data-intensive functions even when assess about future tasks are erroneous.

Angelos Michalas et.al [21] have proposed this work of task scheduling problem in Grid computing surroundings has been addressed. To determine the problem a set of Grid Services are definite and implemented conforming to the OGSA standards. The proposed scheduling architecture is semantically improved and affords the most apposite obligation of tasks to computing resources, given the current load conditions of each computing resource and the network status. The Ant Colony Optimization algorithm (ACO) was used to efficiently assign tasks to computing resources.

3. MULTIPROCESSOR TASK SCHEDULING

Several parallel applications contain various functional units. Whereas the execution of some of the tasks depends on the output of the additional tasks, others can be executed individually at the same time, which escalations parallelism of the problem. The task scheduling problem is the difficulty of conveying the tasks in the multiprocessor system in a manner that will improve the overall performance of the application, while improving the efficiency of the outcome. Multiprocessor scheduling problems can be categorized into many different sorts based on characteristics of the program and tasks (R) to be arranged, the multiprocessor system, and the accessibility of information shown in fig 1.

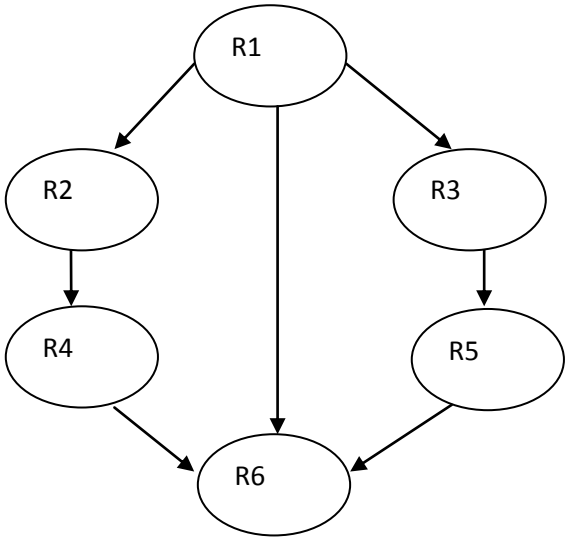

Fig 2: A sample directed acyclic graph

The two main categories of Multiprocessors Task scheduling are: Static and dynamic task scheduling. A static or deterministic task scheduling is one in which preference constrictions and the associations mid the assignment are known well in progress while non-deterministic or dynamic scheduling is one in which these evidence is not known in advance or not known till run time. Static task scheduling algorithms can be categorized into two parts: Heuristic Based and Guided random Search Based Algorithms. Heuristic based algorithm quests a path in the solution space based on the heuristic used while snubbing other probable paths. List scheduling algorithms, clustering and duplication based algorithms come in the same category. In List Scheduling algorithms, each query is allocated significance then added to a queue of waiting queries in order of diminishing importance. As processors become free, the task with the highest priority is removed from the queue and allotted to the most suited processor. In Clustering Heuristic, tasks of a given task graph are mapped into an unrestricted number of clusters. In this heuristic, each reiteration hones the foregoing clustering by merging some clusters. If two tasks are dispensed to the same cluster, they will be accomplished on the same processor. In replication based algorithm, scheduling of a task graph is done by mapping some of its task excessively, which reduces the inter process communication overhead.

4. PROBLEM FORMULATIONS

Scheduling problems with the below given characteristics are considered in this work:

1. Tasks are non initiative in general. Preference relations between the tasks exist.
2. Cost for communication does not exist.
3. In a Multiprocessor System, all the processors are heterogeneous denotation thereby a task may take unlike execution time on each processor.

The three main components of Scheduling are: A multiprocessor system, an application and an objective for scheduling. The multiprocessor system consists of a limited number of fully connected heterogeneous processors (HP1, HP2... HPm). An application comprises tasks and their dependencies on each other. It can be represented as a directed acyclic graph (DAG), (see Fig.2) G = (V, E, W), where the vertices set V consists of v non-initiative tasks, and v_i denotes the i^{th} task. The edge set E represents the precedence Relationship among tasks. A directed edge e_{ij} in E indicated that v_j cannot begin its execution before receiving data from v_i. W is a matrix of vxm, and w_{ij} in W represents the estimated execution time of v_i on j^{th} processor A directed edge e_{ij} in E indicated that v_j cannot begin its execution before receiving data from v_i. W is a matrix of vxm, and w_{ij} in W represents the estimated execution time of v_i on j^{th} processor.

.

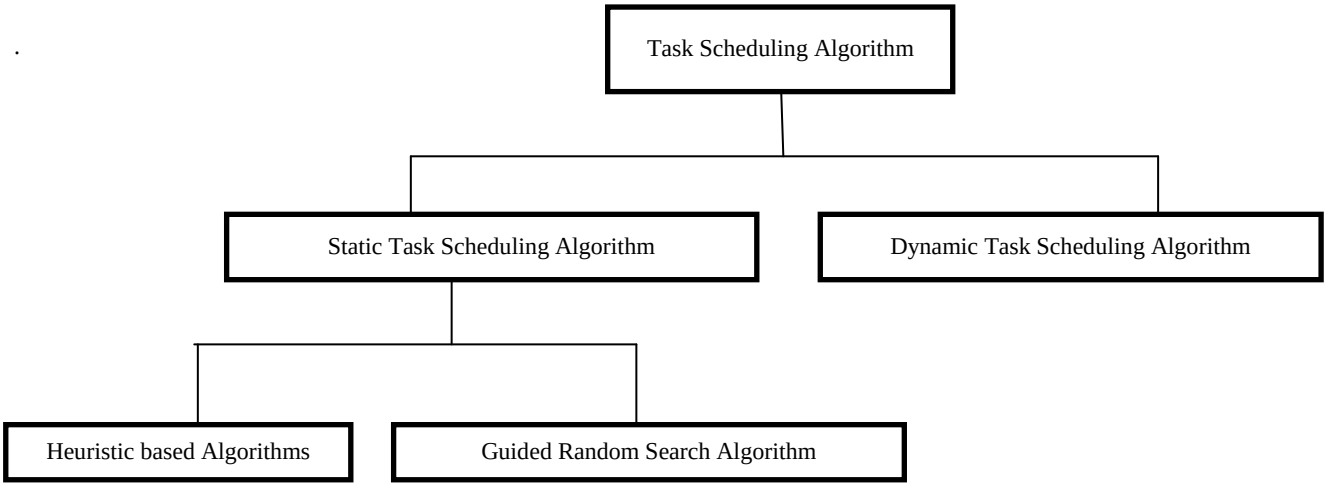

Fig 1. Classification scheduling algorithm

5. GENETIC ALGORITHM STRUCTURE

Genetic algorithm structure for scheduling problem depends on five things which are as follows:

- Representation of chromosomes.
- Construction of genetic operators.
- Selection of Fitness Function.
- Probabilities that can control genetic operators.
- Generation count.

The output obtained and the performance of genetic algorithm is greatly influenced by the above objectives A Genetic Algorithm Structure consists of the following steps:

- Initialization – Defining the population.
- Evaluation – estimate the chromosome using fitness function.
- Genetic operations –Selection of parent chromosomes, perform genetic operators on them to generate new children chromosome.

5.1 Population Initialization

In Genetic Algorithm initializing the chromosome is an important task. The chromosome structure is defined as a combination of two strings SQ and SP of length same as the number of tasks. SQ (scheduling queue) maintains superiority constraints between tasks, and an item in TS denotes a task to be scheduled. An entry in SP (scheduling processor) represents the processor the corresponding task is scheduled onto. The generation of chromosome is explained in the below steps:

1: A task is selected randomly from the entire entry tasks. This task is scheduled as the first task in SQ.

2: Repeat step 3 for (v-1) times.

3: Select a task randomly that is not in SQ with its predecessors all have been in SQ, this task is added to SQ.

Steps 2 and 3 are repeated until termination condition reached

4: In SP, an integer number between 1 and m for each task in SQ is randomly generated and add it to SP.

5.2 Estimation and Selection

In genetic algorithm the fitness function is a exclusive objective function that should be enhanced to solve the problem. The chromosomes are evaluated using the fitness function. The fitness function is defined as:

$$F(i) = (maxCT - CT(i) + 1) / (maxCT - minCT + 1) \quad (1)$$

where:

MaxCT and minCT is the maximum and minimum completion time of chromosomes in current generation, respectively. CT (i) is the completion time of the i^{th} chromosome. After evaluating the fitness values of all the chromosomes the higher fitness value chromosome are selected.

5.3 Crossover and Mutation

5.3.1 Crossover

Crossover mechanism reproduces new children chromosomes which have some parts of both parent's chromosomes. The common type of crossover is a single-point crossover. Multipoint crossover uses m randomly chosen crossover positions. Bits between successive crossover points are exchanged producing two new offspring. As the chromosomes comprises two separate parts SP and SQ having dissimilar distinctiveness, for each part here employ different crossover policies. In this randomly select one or the second part and apply two different crossover operators for these two parts.

Details about crossover are given in following steps:

- C1: Input the Crossover probability Pc.
- C2: Randomly select pairs of chromosomes and generate a float number (FLC) between 0 and 1 for each pair.
- C3: If FLC <= Pc, then recur step CR4 to step CR5. Else unswervingly reproduce those two chromosomes to the next generation.
- C4: Arbitrarily produce two crossover points, p and q, between 1 and v and crossover flag CF between 0 and 1.
- C5: If CF=0 then rearrange the order of tasks in SQ between p and q of one chromosome according to the order of tasks of another chromosome, the rest of the two chromosomes are continued. Else exchange the part in SP between p and q of two chromosomes and the rest of the two chromosomes are remained.

5.3.2 Mutation

Mutation is a genetic operator that alters one or more gene values in a chromosome from its primary state. This can result in completely new chromosomes being added to the population. With these new chromosomes, the genetic algorithm may be able to achieve a better solution than was formerly possible. Mutation can be considered as a random alternation of the individual. Then employ two policies to mute the chromosome as given in following steps:

- M1: Input the Mutation probability Pm.
- M2: For each chromosome, generate a float number (FLM) between 0 and 1.
- M3: If FLM <= Pm, then repeat step MT4 to step MT5 Else directly reproduce this chromosome to the next generation.
- M4: Randomly generate a mutation point p between 1 and v and mutation flag MF between 0 and 1.
- M5: If MF=0 then select randomly a location between location of the nearest immediate predecessor and that of successor of sq_p. Then move sq_p to this location. Else change randomly the processor of sq_p between 1 and m as spp.

5.3.3 Performance Evaluations

The implemented system automatically generates the scheduling problems of required sizes. These have been done to avoid biasing in giving values of dissimilar parameters necessary for the problems. The proposed system fits random

values to these parameters in suitable ranges. The produced problems for our experiments with the subsequent uniqueness:

- Size of problem ranges from 25 to 65 with an interval of 5.
- There is no limit on the number of successors of each task except the exit task which does not have any successor.
- The execution time for each task is a random number between 5 and 25.
- Number of processors varies from 4 to 8 according to the size of problems.

As here not put any restriction over the number of successor a task may have, task graph may be much problematical. So, the problems have chosen may be judged complex in comparison to the kind of problems that was normally seen in literature, where a constraint on maximum number of successor tasks has been put.

6. EFFECT OF MUTATION PROBABILITY ON THE PERFORMANCE OF GA

As mutation is the solution to modify the section of search space, mutation probability may have leading role in finding solutions of good quality. Thus, experiments are repeated by fixing crossover probability and changing mutation probabilities from 0.05 to .40 and noted average schedule lengths. Experiments had been done on the problem having size 65. Here observation shows similar trend in the problems of all sizes. The below graph shows the further average of results, mixing the effect of all crossover probabilities which clearly shows that up till mutation probability is .20, increase in mutation probability leading to better results. After .20 results are fluctuating in a small range but normally are not better than that obtained for .20. So, here it was found best mutation probability for our set of problems as .20.

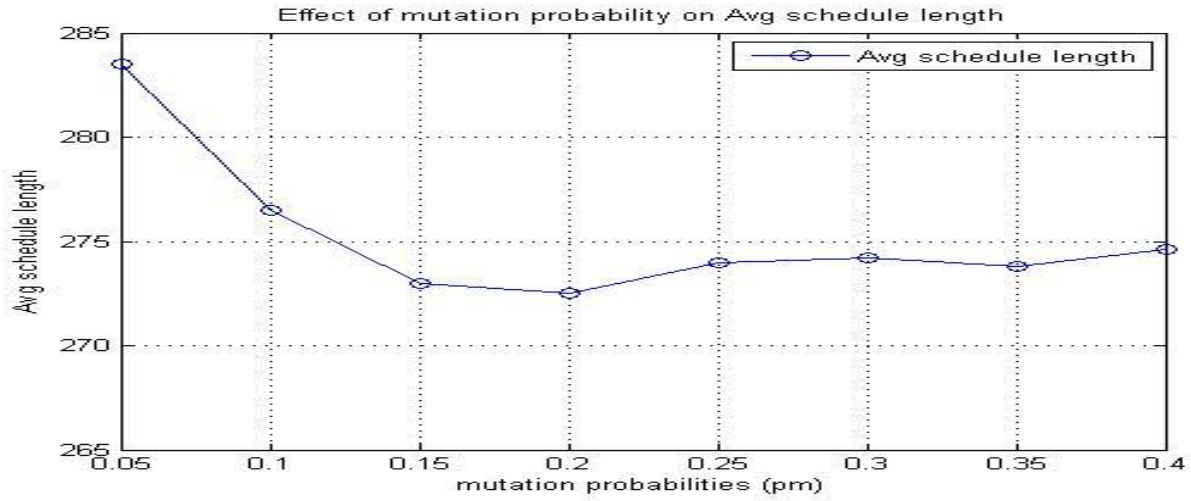

Fig. 3. Effect of mutation probability on avg. schedule length

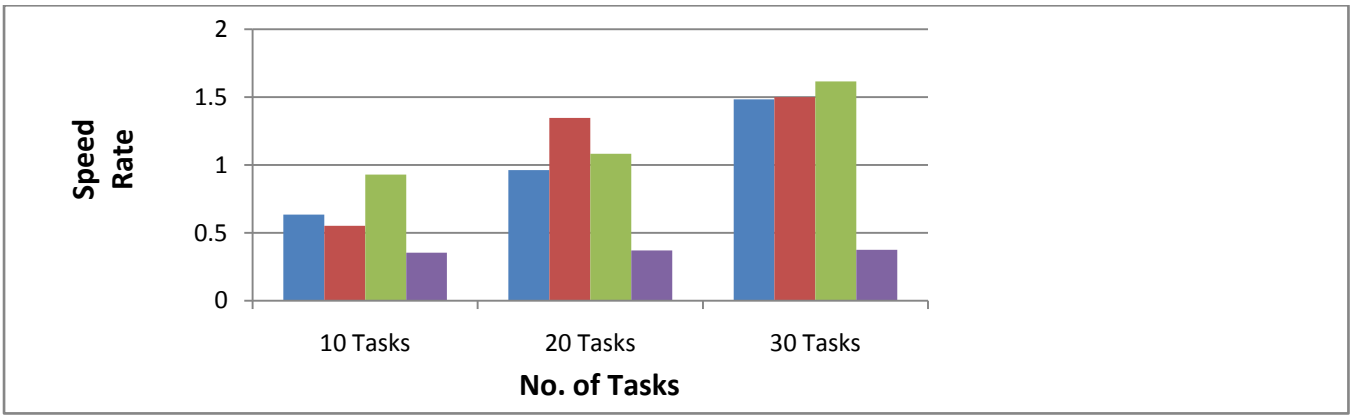

Fig4. Comparison on Speedup with 3 processors

The efficiency, schedule length, speedup and utilization can be computed for three processors with GA approach is shown as below:

i). $Schedule\ Length\ =\ max\ \{E\_Time\ (\ t_i\ )\}$ = 33.0

(Because the three processors have the execution time as 33, 32 and 31 respectively)

ii). $Speedup = \frac{S_L}{Avg\ (P_T)}$ = 33.0/52.0 = 0.6346

(S_L is the schedule length and it is 33.0. Average processor time for three processors is 52.0 (33.0+10+9), which was computed through the simulator.)

iii). $Efficiency = \frac{Speedup}{Total\ number\ of\ processor}$ = 0.6346/3 = 0.2115

Table 1: Parameters Analyzed with three processors

Parameters Analyzed with Four Processors		GA		
		3	4	5
For 10 Tasks	Efficiency	0.1136	0.1052	0.1136
	Schedule Length	20.0	16.0	30.0
	Speed Up	0.4545	0.4210	0.4545
	Utilization	1.0	0.9531	0.9166

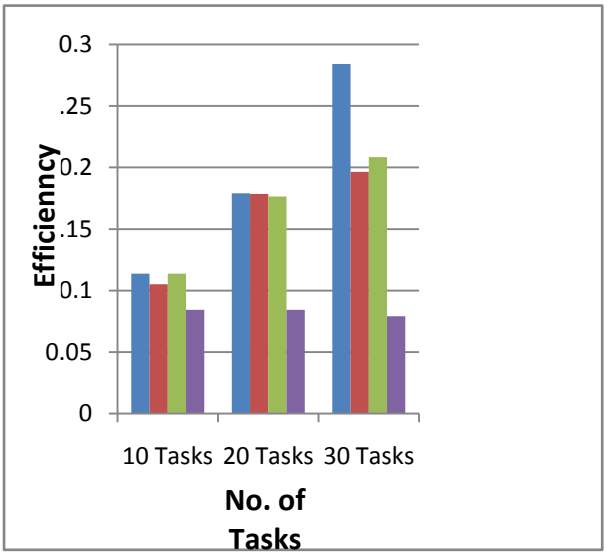

Fig.5. Comparison of Efficiency with 4 processors

Where,

$$Average\ Execution\ time\ of\ p_j = \frac{\sum_{j=1}^{No.\ of\ Processors} Execution\ \ time\ [No.of.Processors\ ]}{No.of.Processors}$$

Average Execution time of three processors = 32.333

Load Balance = 33.0/32.333 = 1.0206

iv). $Utilization = \frac{\sum FEP_{P_i}}{Total\ \ number\ \ of\ processor}$

FEP for processor LB1 = 33/33 =1

FEP for processor LB2 = 30/33 = 0.9090

FEP for processor LB3 = 1.0301

Total FEP for three processors is = 2.9391

Utilization = 2.9391/3 = 0.9797

The above Table shows the efficiency, schedule length, speedup, and utilization. The table is made throughout with four processors with a dynamic number of tasks and duplicated tasks. The efficiency of the parallel processor is calculated as follows:

$$Efficiency = \frac{Speedup}{Total\ \ number\ \ of\ processor}$$

Table 2: Parameters analyzed with four processors

Parameters Analyzed with Three Processors		GA		
		3	4	5
For 10 Tasks	Efficiency	0.2115	0.1842	0.30952
	Schedule Length	33.0	21.0	39.0
	Speed Up	0.6346	0.5526	0.9285
	Utilization	0.9797	0.9682	0.8888

The formula of efficiency derives the speedup of the processor with the all parallel processors which is taken into account.

The efficiency of the parallel processors is high using the proposed GA. Finally, the utilization of the parallel processors can be defined as follows:

$$Utilization = \frac{\sum FEP_{P_i}}{Total\ \ number\ \ of\ processor}$$

Where FEP_{P_i} denotes each processors scheduling time and FEP_{P_i} is calculated as follows:

$$FEP_{P_i} = \frac{P_i^{th} S_L}{S_L}$$

Where $P_i^{th} S_L$ denotes the i^{th} processors schedule length and S_L denotes the processors maximum schedule length among all the parallel processors.

7. CONCLUSIONS

In this paper, genetic algorithm for task scheduling in parallel systems is implemented. The genetic operations like crossover and mutation take place as usual to maintain the diversity among the chromosomes. The parameters taken are schedule length, efficiency, and utilization. And also have assembled a system which automatically generates the problems of required sizes and also fits values to the parameters. The performance of our algorithm for robustness is analyzed. It has been seen that in GA, Average Schedule Length continuously decreases as the number of generation increases. This shows that genetic algorithm is robust and gives a guarantee for good results. Lastly, the effect of mutation probability on the performance of GA is analyzed. It was found that mutation is a mechanism to avoid premature convergence. Mutation can be considered as an occasional random alternation of the value of a string.

8. REFERENCES

[1] J Weinberg, "Job Scheduling on Parallel Systems", Job Scheduling Strategies for Parallel Processing, 2002.

[2] CH Xia, G Michailidis, N Bambos, "Dynamic on-line task scheduling on parallel processors", Performance Evaluation, Elseiver 2001.

[3] Esquivel S.C., Gatica C. R., Gallard R.H, "Solving the parallel task scheduling problem by means of genetic algorithm", National Agency to Promote Science and Technology.

[4] Rachhpal Singh, "Genetic Algorithm for Parallel Process Scheduling", International Journal of Computer Applications & Information Technology Vol. 1, 2012.

[5] U.Karthick Kumar, "A Dynamic Load Balancing Algorithm in Computational Grid Using Fair Scheduling", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 5, No 1, 2011.

[6] Lei Zhang, Yuehui Chen, Runyuan Sun, Shan Jing and Bo Yang, "A Task Scheduling Algorithm Based on PSO for Grid Computing", IEEE, vol 2, 2006.

[7] Abraham, R. Buyya and B. Nath, Nature's Heuristics for Scheduling Jobs on Computational Grids, The 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), pp. 45-52, 2000.

[8] S. Song, Y. Kwok, and K. Hwang, "Security-Driven Heuristics and A Fast Genetic Algorithm for Trusted Grid Job Scheduling", IEEE International Parallel and Distributed Processing, pp.65-74, 2005.

[9] J.E. Orosz and S.H. Jacobson, Analysis of static simulated annealing algorithm, Journal of Optimization theory and Applications, pp. 165-182, 2002.

[10] R. Braun, H. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. Reuther, J. Robertson, M. Theys, B. Yao, D. Hensgen and R. Freund, "A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems", pp. 810-837, J. of Parallel and Distributed Computing, vol.61, 2001.

[11] A. Moraglio, H.M.M. Teneikelder, R. Tadei, "Genetic Local Search for Job Shop Scheduling Problem", Technical Report CSM, 2005.

[12] Ratan Mishra1 and Anant Jaiswal, "Ant colony Optimization: A Solution of Load balancing in Cloud", International Journal of Web & Semantic Technology, Vol.3, 2012.

[13] Z. Pooranian, A. Harounabadi, M. Shojafar and N. Hedayat"New Hybrid Algorithm for Task Scheduling in Grid Computing to Decrease missed Task", World Academy of Science, Engineering and Technology, Vol-5, 2011.

[14] Dervis Karaboga and Bahriye Basturk, "Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems", IFSA, pp. 789–798, 2007.

[15] Zahra Pooranian, Mohammad Shojafar, Reza Tavoli, Mukesh Singhal, Ajith Abraham, "A Hybrid Metaheuristic Algorithm for Job Scheduling on Computational Grids", Informatica, pp 157–164, 2013.

[16] Rizos Sakellariou and Viktor Yarmolenko, "Job Scheduling on the Grid: Towards SLA-Based Scheduling".

[17] Vishnu Kant Soni, Raksha Sharma, Manoj Kumar Mishra, "Grouping-Based Job Scheduling Model In Grid Computing", World Academy of Science, Engineering and Technology, Vol: 4, 2010.

[18] U.Karthick Kumar, "A Dynamic Load Balancing Algorithm in Computational Grid Using Fair Scheduling", International Journal of Computer Science Issues, Vol. 8, 2011.

[19] S.Selvi, Dr. D.Manimegalai and Dr.A.Suruliandi, "Efficient Job Scheduling on Computational Grid with Differential Evolution Algorithm", International Journal of Computer Theory and Engineering, Vol. 3, 2011.

[20] Jim Blythe, Sonal Jain, Ewa Deelman, Anirban Mandal, and Ken Kennedy "Task Scheduling Strategies for Workflow-based Applications in Grids".

[21] Angelos Michalas, and Malamati Louta, "Adaptive Task Scheduling in Grid Computing Environments".