ISSN 2047-3338

Task Scheduling using Performance Effective Genetic Algorithm for Parallel Heterogeneous System

Jasbir Singh[1] and Gurvinder Singh[2]
[1]Department of Computer Science and IT, Guru Gobind Singh Khalsa College, Sarhali (Tarn Taran), Punjab-India
[2]Department of Computer Science and Engineering, Guru Nanak Dev University (Amritsar), Punjab, India
jasbir2gill@yahoo.com, gsbawa71@yahoo.com

***Abstract*– The scheduling problem deals with the optimal assignment of a set of tasks onto parallel multiprocessor system and orders their execution so that the total completion time is minimized. The efficient execution of the schedule on parallel multiprocessor system takes the structure of the application and the performance characteristics of the proposed algorithm. Many heuristics and approximation algorithms have been proposed to fulfill the scheduling task. It is well known NP-complete problem. This study proposes a genetic based approach to schedule parallel tasks on heterogeneous parallel multiprocessor system. The scheduling problem considered in this study includes - next to search for an optimal mapping of the task and their sequence of execution and also search for an optimal configuration of the parallel system. An approach for the simultaneous optimization of all these three components of scheduling method using performance effective genetic algorithm is presented and its performance is evaluated in comparison with the First Come First Serve (FCFS), Shortest Job First (SJF) and Round Robin (RR) scheduling methods.**

***Index Terms*– First Come First Serve (FCFS), Shortest Job First (SJF), Round Robin (RR), Performance Effective Genetic Algorithm (PEGA), Directed Acyclic Graph (DAG), Heterogeneous Processor and Simultaneous Optimization**

I. INTRODUCTION

TASK assignment and scheduling [1], [2], [3], [4] can be defined as assigning the tasks onto a set of processor and determining the sequence of execution of the task at each processor. While the total finish time of the tasks is determined by the performance of the processors and the sequence of the tasks, therefore, an execution scheduling consists of three components:

- Performance of the heterogeneous processor
- Mapping of the tasks onto the processors
- Sequence of the execution of the tasks on each processor

All three components of this optimization problem [5], [30] are highly dependent on each other and should not be optimized separately.

A Performance Effective Genetic Algorithm (PEGA) approach is being proposed to handle the problem of parallel system task scheduling. A GA [7], [8] starts with a generation of individual, which are encoded as strings known as chromosome. A chromosome corresponds to a solution to the problem. A fitness function is used to evaluate the fitness of each individual. In general, PEGAs consists of selection, crossover and mutation operations [15] based on some key parameters such as fitness function, crossover probability and mutation probability.

This study is divided into following sections: In section 2 an overview of the problem is given along with brief description of the solution methodology. Section 3 provides a detailed improved parallel genetic algorithm. Experimental results and performance analysis are provided in section 4 and conclusion is followed in section 5.

II. PROBLEM DEFINITION

Parallel Multiprocessor system scheduling can be classified into many different classes based on the characteristics of the tasks to be scheduled, the multiprocessor system and the availability of the information [6], [9], [11]-[14]. The strategy behind the execution of the tasks on parallel multiprocessor system environment is to efficiently partitioning the huge task into set of tasks of appropriate gain size and an abstract model of the partitioned tasks that can be represented by Directed Acyclic Graph (DAG) [25], [28], [29]. The focus is on a deterministic scheduling problem in which there exist precedence relations among the tasks to be scheduled. A deterministic scheduling problem [16] is one in which all information about the tasks and the relation to each other such as execution time and precedence relation are known to the scheduling algorithm in advance and the processor environment is heterogeneous [23], [24], [26], [27]. Heterogeneity of processors means that the processors have different speeds or processing capabilities.

In this study, a study has been done regarding the task scheduling problem as a deterministic on the heterogeneous multiprocessor environment. The main objective is to minimize the total task finish time (execution time + waiting time or idle time).

The multiprocessor computing environment consists of a set of m heterogeneous processor:

$$P = \{p_i: i = 1, 2, 3 \ldots m\}$$

They are fully connected with each other via identical links.

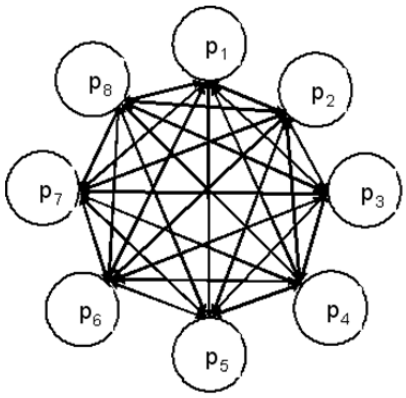

Fig. 1: A fully connected parallel processor

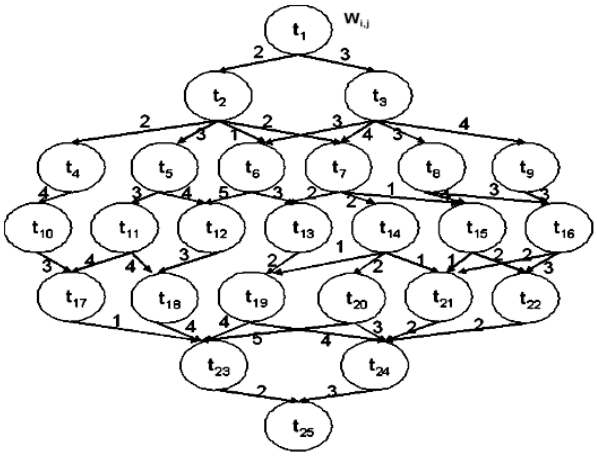

Fig. 2: Directed acyclic graphs of task size=25 with task precedence. Where wi,j = execution time of different tasks on different processors as shown in Table 1

Fig. 1 shows a fully connected eight parallel system with identical link. The parallel application can be represented by a directed acyclic graph (DAG), G = (T, E, W, C), where the vertices set T consist of n tasks as:

$$T = \{t_{j:}\ j = 1, 2, 3 \ldots n\}$$

A directed edge set E consist of k edges as:

$$E = \{e_{k:}\ k = 1, 2, 3 \ldots r\}$$

This represents the precedence relationships among tasks. For any two tasks t_i, $t_{i+1} \in T$ with having directed edge e_k (i.e., edge from task t_i to t_{i+1}) means that task t_{i+1} cannot be scheduled until t_i has been completed, t_i is a predecessor of t_{i+1} and t_{i+1} is a successor of t_i. In other words t_i sends a message whose contents are required by t_{i+1} to start execution.
The elements set W are the weights of the vertices as:

$$W = \{w_{i,j:}\ i = 1, 2, 3 \ldots m, j: 1, 2, 3, \ldots n\}$$

It represents the execution duration of the corresponding task and are varies from processor to processor because of heterogeneous processor environment.

The elements set C are the weights of the edges as:

$$C = \{c_{k:}\ k = 1, 2, 3 \ldots r\}$$

It represents the data communication between the two tasks, if they are scheduled to different processors. But if both tasks are scheduled to the same processor, then the weight associated to the edge becomes null. Figure 2 show example of DAGs. It consist of a set of tasks T: $\{t_{j:}\ j = 1, 2 \ldots\}$, Fig. 1 indicates set of processors $P = \{p_i : i = 1, 2, 3 \ldots\}$ and Table 1 show a matrix of execution time of each task on processor p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, because of heterogeneous environment every processor works on different speeds and processing capabilities. It is assumed that processor p_1 is much faster than p_2, p_3 and so on. Processor p_2 is faster than p_3, p_4 and so on. (i.e., the order of speed and processing capabilities can be expressed as $p_1 > p_2 > p_3 > p_4 > p_5 > p_6 > p_7 > p_8$). As given in Table 1 task t_1 takes 6 time units to complete their execution on processor p_1 and takes 11 time units and 12 time units to complete their execution on processor p_7 and p_8 respectively. This execution time has been calculated on the basis of the size of the tasks by processing on different processors.

III. PERFORMANCE EFFECTIVE GENETIC ALGORITHM

PEGAs operate through a simple cycle of stages: creation of a population strings, evaluation of each string, selection of the best strings and reproduction to create a new population. The individuals are encoded in the population string known as chromosomes. Once the chromosome has been coded, it is possible to evaluate the performance or fitness of individuals in a population. A good coding scheme [17], [18] will benefit operators and make the object function easy to calculate.
During selection, each individual is assigned a fitness value given by the objective function and choose the fittest individual of the current population to serve as parent of the next generation. Reproduction involves two types of operators namely crossover and mutation.
The crossover operator chooses randomly a pair of individuals among those selected previously and exchange some part of the information. The mutation operator takes an individual randomly and alters it.

A. Creation of the Population String

The first step in the PEGAs algorithm is the creation of the initial population. Number of processors, number of tasks and population size are needed to generate initial population. The initial population is initialized with randomly generated individuals. The length of all individuals in an initial population is equal to the number of tasks in the DAG. Each task is randomly assigned to a processor.

B. Evaluation of the Fitness Function

The fitness function used for improved parallel genetic algorithm is based on the total completion time for the schedule, which includes execution time and communication

Table 1: Shows a tasks execution matrix on different processors with task size = 25

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{18}	t_{19}	t_{20}	t_{21}	t_{22}	t_{23}	t_{24}	t_{25}
p_1	6	3	8	3	2	4	5	7	8	9	6	11	5	13	9	10	7	11	8	11	7	5	12	10	13
p_2	7	3	9	4	2	5	6	7	9	10	6	12	6	14	10	11	8	12	8	12	8	6	13	11	14
p_3	7	4	9	4	3	6	7	8	9	11	7	13	7	15	11	12	8	13	9	13	9	7	14	12	15
p_4	8	5	10	5	4	6	8	9	10	12	8	13	7	16	12	12	9	13	10	14	9	7	15	13	16
p_5	10	6	10	5	4	7	8	10	11	13	9	14	8	16	13	13	10	14	11	15	10	8	15	13	17
p_6	11	7	11	6	5	8	9	10	12	13	10	15	9	17	13	14	10	15	12	15	11	9	16	14	17
p_7	11	8	12	6	6	9	9	11	12	14	10	15	9	18	14	15	11	16	13	16	12	10	17	15	18
p_8	12	8	13	7	6	10	10	12	13	15	11	16	10	19	15	16	12	17	14	17	13	10	17	16	19

delay time. The fitness function separates the evaluation into two parts: Task fitness and processor fitness. The task fitness focuses on ensuring that all tasks are performed and scheduled in valid order. A valid order means that a pair of tasks is independent if neither task get data output from the other task for execution. The scheduling of a pair of tasks to a single processor is valid if the pair is independent or if the order in which they are assigned to the processor matches the order of their dependency. Table 2 (a) show a valid order of the tasks assigned to the set of processors (p_1, p_2, p_3……) and invalid order in Table 2 (b).

The processor fitness component of the fitness functionattempts to minimize processing time. Consider the following schedule S1 and S2 for single processor and multiprocessor parallel system tasks schedules with task size equal to 25 tasks respectively (here, we consider the case when fitness function assigned all tasks to a single processor and randomly generated tasks to heterogeneous parallel system.) The processor chosen for scheduler S1 is p_1 and the execution time for all task are given in Table 1. The total finish time of scheduler S1 and S2 is:

S1: $t_1 \rightarrow t_2 \rightarrow t_3$ -------- $\rightarrow t_{24} \rightarrow t_{25}$

Total Finish Time = Execution time + Comm. time.

=6+3+8+3+2+4+5+7+8+9+6+11+5+13+9+10+7+11+8+11+7 +5+12+10+13=193time units.

Here comm. Time = 0, because all tasks are executed on same processor. The processors chosen for schedule S2 are same as given in Table 1 and the randomly order sequence is given in Table 2 (a), here the processors chosen for scheduler S2 are p_1, p_2, p_3, p_4 and p_5 . Thus S2: Total finish time = Execution time + Comm. Time = 76 time units.

The scheduler S1 shows a total finish time of 193 time units, where as scheduler S2 shows a total finish time of just 76 time units. Therefore, proper fitness function reduces the total finish time very well.

Table 2(a): Random assignment of tasks to parallel processors (Valid order)

Processor	Ordering of tasks
p_1	t_1, t_3, t_7, t_{14}, t_{20}, t_{21}, t_{24},
p_2	t_2, t_4, t_8, t_{12}, t_{18}, t_{23}.
p_3	t_6, t_9, t_{16}, t_{17}.
p_4	t_5, t_{11}, t_{13}, t_{19}, t_{22}.
p_5	t_{10}, t_{15}.

Table 2(b): Random assignment of tasks to parallel processors (Invalid order), because no processor (p_1, p_2, p_3……) starts execution of tasks t_2, t_3, t_5 t_{15}, t_{10} respectively until the execution of task t_1

Process	Ordering of tasks
p_1	t_2, t_{23}, t_{18}, t_{16}, t_4, t_1.
p_2	t_3, t_8, t_7, t_6, t_{24}, t_{25}.
p_3	t_5, t_9, t_{12}, t_{13}, t_{17}.
p_4	t_{15}, t_{14}, t_{19}, t_{11}.
p_5	t_{10}, t_{20}, t_{21}, t_{22}.

Therefore, the fitness values (task and processor) have been evaluated for all chromosomes and the probability of higher fitness is to be selected for reproduction from current generation to the next generation.

C. Selection Operator

The design of the fitness function is the basic of selection operation, so how to design the fitness function will directly affect the performance of genetic algorithm. PEGAs uses selection operator to select the superior and eliminate the inferior. The individual are selected according to their fitness value. Once fitness values have been evaluated for all chromosomes, we can select good chromosomes through rotating roulette wheel strategy. This operator generate next generationby selecting best chromosomes from parents and offspring.

D. Crossover Operator

Crossover operator randomly selects two parent chromosomes (chromosomes with higher values have more chance to be selected) and randomly chooses their crossover points, and mates them to produce two child (offspring) chromosomes. We examine one and two point crossover operators. In one point crossover, the segments to the right of the crossover points are exchanged to form two offspring as shown in Fig. 3(a) and in two point crossover [19], [20], the middle portions of the crossover points are exchanged to form two offspring as shown in Fig. 3(b).

Randomly selects parent 1 and 2, crossover point 2:

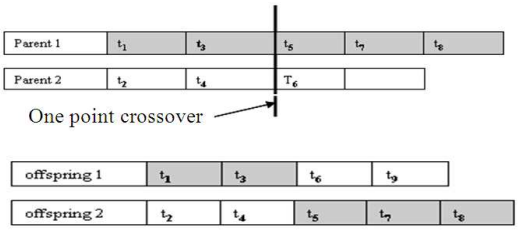

Fig. 3: (a) One point crossover

Randomly selects parent 1 and 2, crossover points 1 and 3:

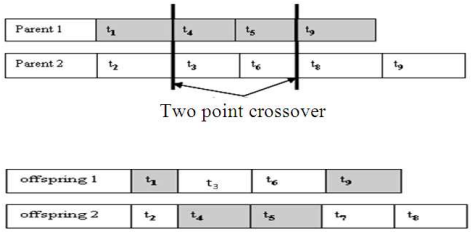

Fig. 3: (b) Two point crossover

E. Mutation operator

A mutation operation is designed to reduce the idle time of a processor waiting for the data from other processors. It works by randomly selecting two tasks and swapping them. Firstly, it randomly selects a processor, and then randomly selects a task [21] on that processor. This task is the first task of the pair to be swapped. Secondly, it randomly selects a second processor (it may be the same as the first), and randomly selects a task. If the two selected tasks are the same task the search continues on. If the two tasks are different then they are swapped over (provided that the precedence relations must satisfy). Consider the following example of six tasks DAG with tasks precedence and the execution times of tasks t_1 to t_6 on processor p_1 and p_2 are given in Table 1. Fig. 4(a), Fig. 4(b) and Fig. 4(c) demonstrates the mutation operation.

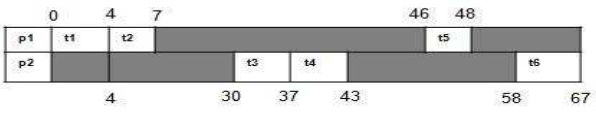

Fig. 4 (a): A DAG with tasks precedence

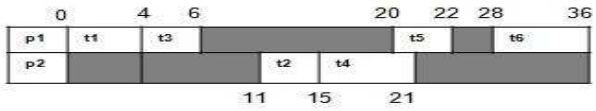

Fig. 4 (b): A Gantt chart before mutation operation, which takes 67 time units to complete the schedule

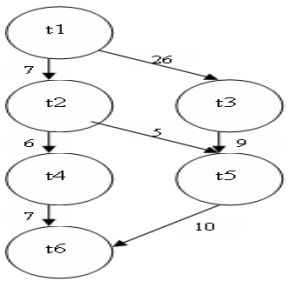

Fig. 4 (c): A Gantt chart after mutation operation, which takes 36 time units to complete the schedule

Here the mutation operation swaps task t_2 on processor p_1 to task t_3 on processor p_2.

The procedure of the Performance Effective Genetic Algorithm (PEGA) is:

Step 1: Setting the parameter
Set the parameter: Read DAG (task execution matrix (number of tasks n, number of processors m) and comm. cost), population size pop_size, crossover probability p_c, mutation probability p_m, and maximum generation maxgen
Let generation gen = 0, maxeval = 0

Step 2: Initialization
Generate pop size chromosomes randomly.

Step 3: Evaluate
Step 3.1: Calculate the fitness value of each chromosomes
Step 3.2: Task fitness
Step 3.2: Processor fitness

Step 4:Crossover
Perform the crossover operation on the chromosomes selected with probability p_c.

Step 5: Mutation
Perform the swap mutation on chromosomes selected with probability p_m.

Step 6: Selection Select pop_size chromosomes from the parents and offspring for the next generation

Step 7: Stop testing If gen = maxgen, then output best solution and stop else gen = gen + 1 and return to step 3

IV. EXPERIMENTAL RESULTS AND PERFORMANCE ANALYSIS

The final best schedule obtained by applying the Performance Effective Genetic Algorithm (PEGA) to the DAG of Fig. 2 with execution time shown in Table 1 onto the parallelmultiprocessor system is shown in Fig. 5.

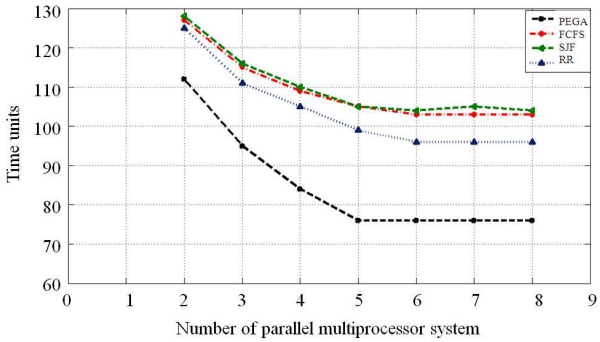

Fig. 5: A Gantt chart of the PEGA for task size =25. It shows a time taken to complete the schedule by applying the PEGA and execution time of 76 time units

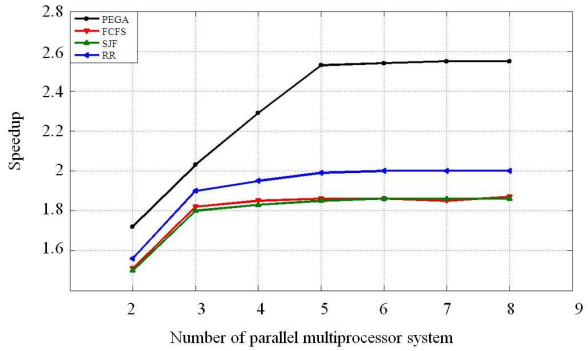

Fig. 6: Time taken to complete the schedule by PEGA, RR, SJF and FCFS Scheduler on parallel multiprocessor system for task size = 25tasks

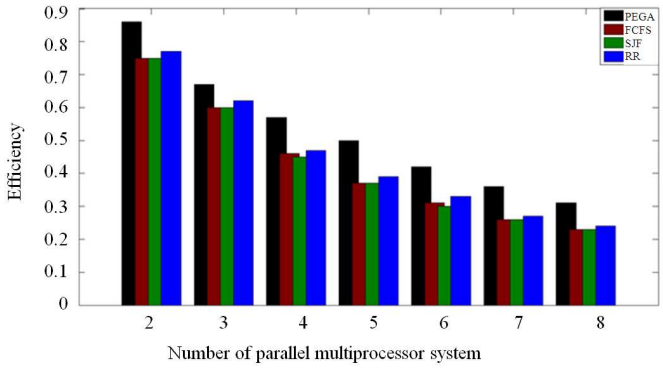

Fig. 7: Speedup v/s number of parallel multiprocessor system for task size = 25tasks

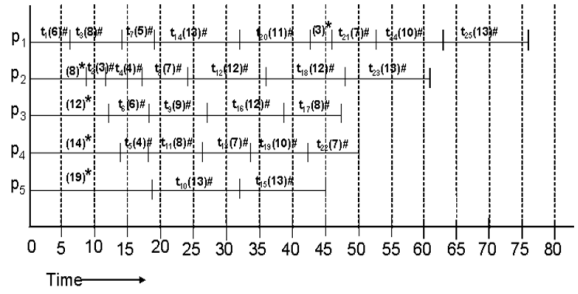

Fig. 8: Performance comparisons of the PEGA, SJF, FCFS and RR for task size = 25tasks

We also compare the results with Shortest job First (SJF), First Come First Serve (FCFS) and Round Robin (RR) scheduling method [22] on parallel systems and execution of the schedules are shown in Fig. 6 for task size equal to 25.

A. Performance Analysis

Speed up (T_{sp}): Speed up (Hwang and Briggs, 1985) is defined as the completion time on a uniprocessor divided by completion time on a multiprocessor. In case of homogeneous system, it is denoted as: T_{sp} = p(1)/p(m). But in case of heterogeneous system, it is denoted as T_{sp}= (min (p(1)) / p(m) i.e., the best uniprocessor completion time divided by the completion time on a heterogeneous multiprocessor system. The speedup is measured with the execution of tasks on single processor which shows 193 time units for task size equal 25 tasks divided by execution time units on PEGA, RR, SJF and FCFS scheduler as shown in Fig. 7.

B. Efficiency (¢)

(T_{sp} / m), where m is the number of processors.

V. CONCLUSION

In this study we have proposed a Performance Effective Genetic Algorithm (PEGA) for task scheduling in heterogeneous parallel multiprocessor system to minimize the finish time including execution time and waiting or idle time and increase the throughput of the system. The proposed method found a better solution for assigning the tasks to the heterogeneous parallel multiprocessor system. Experimental results and performance of the Performance Effective Genetic Algorithm (PEGA) is compared with FCFS, SJF and RR Scheduling methods. The performance study is based on the best randomly generated schedule of the Performance Effective Genetic Algorithm (PEGA).

REFERENCES

[1] Sara Baase, Allen Van Gelder, "Computer Algorithms", Published by Addison Wesley, 2000.

[2] Sartaj Sahni, "Algorithms Analysis and Design", Published by Galgotia Publications Pvt. Ltd., New Delhi, 1996.

[3] Anup Kumar, Sub Ramakrishnan, Chinar Deshpande, Larry Dunning, " IEEE Conference on Parallel processing", 1994Page no.83-87.

[4] Ananth Grama, Georage Karypis, Anshul Gupta, Vipin Kumar,"Introduction to parallel computing", Published by Pearson Education, 2009.

[5] Kalyanmoy Deb,"Optimization for Engineering Design", Published by PHI, 2003.

[6] Dezso Sima, Terence Fountain, Peter Kacsuk, "Advanced Computer Architectures", Published by Pearson Education, 2009.

[7] David E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Published by Pearson Education, 2004, Page No.60-83.

[8] Mitchell, Melanie," An Introduction to Genetic Algorithm, Published Bu MIT Press 1996

[9] Michael J Qumn, "Parallel Computing Theory and Practices, 2nd Edition", Published by Tata McGraw Hill Education Private Ltd, Page No. 346-364. Michael J. Qumn, "Parallel

Programming", Published by Tata McGraw Hill Education Private Ltd, Page No. 63-89.
[10] John L Hennessy, David A Pattern, "Computer Architecture, 3rd Edition", Published by Morgan Kaufmann & Elsevier India, Page No. 528-590.
[11] David E Culler, "Parallel Computer Architecture", Published by Morgan Kaufmann & Elsevier India.
[12] J P Hayes, "Computer Architecture and Organization", Published by McGraw Hill International Edition.
[13] J D Carpinalli, "Computer System Organization & Architecture", Published by Pearson Education.
[14] Sung-Ho Woo, Sung-Bong Yang, Shin-Dug Kim, and tack-Don Han ," IEEE Trans on parallel System, 1997, Page 301-305.
[15] M.Salmani Jelodar, S.N.Fakhraie, S.M.fakharie, M.N. Ahmadabadi," IEEE Proceeding, 2006, Page No 340-347.
[16] Man Lin and Laurence tianruo Yang,"IEEE Proceeding", 1999, Page No 382-387
[17] Yajun Li, Yuhang yang, Maode Ma, Rongbo Zhu, " IEEE Proceeding", 2008.
[18] YI-Wen Zhong, Jian-Gang Yang, Heng-Nlan QI," IEEE Proceeding", August 2004, Page No. 2463-2468.
[19] Imtiaz Ahmad, Muhammad K. Dhodhi and Arif Ghafoor," IEEE Proceeding", 1995, Page No. 49-53.
[20] Ceyda Oguz and M.Fikret Ercan," IEEE Proceeding", 2004, Page No. 168-170
[21] Kai Hwang & Faye A. Briggs,"Computer Architecture and Parallel Processing", Published by McGraw Hill ,1985.Page No. 445-47, 612.
[22] Andrei R. & Arjan J.C. van Gemund, "Fast and Effective Task Scheduling in Heterogeneous Systems", IEEE Proceeding, 2000.
[23] Michael Bohler, Frank Moore, Yi Pan, "Improved Multiprocessor Task Scheduling Using Genetic Algorithms", Proceedings of the Twelfth International FLAIRS Conference, 1999.
[24] Andrew J. page," Adaptive Scheduling in Heterogeneous Distributed Computing System",
[25] Jameela Al-Jaroodi, Nader Mohamed, Hong Jiang and David Swanson," Modeling Parallel Applications Performance on Heterogeneous Systems", Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03)
[26] Yu-Kwong and Ishraq Ahmad, "Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors", ACM Computing Surveys, Vol. 31, No. 4, December 1999
[27] Wai-Yip Chan and Chi-Kwong Li, "Scheduling Tasks in DAG to Heterogeneous Processor System", Proceeding IEEE 1998.
[28] Wojciech Cencek, "High-Performance Computing on Heterogeneous Systems", Computational Methods in Science and Technology, 1999
[29] C.P Ravikumar, A.K Gupta," IEEE proc. Comput. Digit. Tech., Vol 142, No. 2", March 1995, Page No. 81-86.
[30] Michael J Qumn, "Parallel Programming", Published by Tata McGraw Hill Education Private Ltd, Page No. 63-89.