Efficient Tasks scheduling for heterogeneous multiprocessor using Genetic algorithm with Node duplication

Jaspal Singh [1],
[1]P.G. Department of Computer Science, Guru Gobind Singh Khalsa College,
Sarhali (Tarn Taran), Punjab-India
jsingh@ggscollegesarhli.org

Harsharanpal Singh[2]
[2]P.G. Department of Computer Science, SGAD College,
Khadur Sahib (Tarn Taran), Punjab-India
sharansandhu@rediffmail.com

Abstract
The prime function of the effective utilization of heterogeneous multiprocessor system is accurately mapping of tasks and makes their scheduling on different processors for reducing their total finish time. Total runtime is time taken time for all jobs with the individual runtime of tasks and their communication cost among tasks. An optimal scheduling of parallel tasks with some precedence relationship onto a multiprocessor system is considered as NP-complete problem. The scheduling problem considered in this paper is bringing out the optimal mapping of tasks and their efficiently possible execution stream on multiprocessor system configuration. Several solutions and heuristics are proposed to solve this problem. We exhibit efficiency of Node duplication GA based technique by comparing against some of the existing deterministic scheduling techniques for minimizing inter processor traffic communication
.

Keywords: Parallel System with heterogeneous processors, arbitrary search, Task scheduling, genetic algorithm (GA), Node duplication Genetic algorithm (NGA).

1. Introduction

The task scheduling is prime significance of heterogeneous multiprocessor parallel systems. The problem of task scheduling on these type systems is verified the processor, when and on which processor a given task executes. Our objective is to minimize the mean task response time (total finish time) and meet the deadline at the same time. The final intention of this scheduling in a way that the final time and cost of the execution would be minimized and all other constraints such as fundamentals, are satisfied. [1,2,4] In this problem the process of scheduling is static because the finite number of processor, mapped tasks, their dependencies, and inter processor communication cost used for them are known in advanced before going to task schedule. [4, 11] We determine the set of finite tasks T and finite no. of processors
so that map each task to a processor from the set P of processors. Under the task set each task may communicate with zero or more other tasks in set with some communication cost value. So when we consider an execution scheduling in our experiment we focus on three variables which constitute

the all setup like performance factor of non homogenous processors, mapping of different scheduling tasks on working processors and last variable maintain the sequence of execution of tasks under various format like first come first serve, priority scheduling, list scheduling, genetic algorithm and Node duplicated concept with genetic algorithm on processors. The execution and communication costs are represented by DAG (directed acyclic graph). [1, 3] The task scheduling problem is taken as NP-complete problem and genetic algorithms have attracted much attention as class of arbitrarily search algorithm for task scheduling in heterogeneous processor environment. This paper is organized as follows some review of previous works is presented in section 2. In Section 3 overview of problem is given along with solution methodology. Section 3 provides us view about Genetic algorithm for task scheduling. Under section 4 we reveal out the concept of using node duplication. Experiment results and performance analysis are provided in section 5and 6. Section 7 is composing of conclusion.

2. Methodology

We classified the parallel multiprocessor into various classes based on the characteristic of different tasks to be scheduled, multiprocessor system type and knowledge of other correlated data. Our experimental setup is static or deterministic because we are aware of all tasks execution time and its precedence relation in advance. [9] The task setup is completely non-preemptive mean one task execution must be completely occurred before other is task scheduled for get processor control for execution and the processor working environment is fully heterogeneous i.e. all the processors have different processing speed and capabilities. Under this paper we study the task scheduling problem as deterministic and non preemptive in a heterogeneous multiprocessor environment. The main objective is to minimize the total finish time which comprises execution time, waiting time or idle time. The multiprocessor parallel system set consists of n heterogeneous processors

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

They are fully connected with each other via identical links.

Figure 1 shows fully connected three parallel systems with identical links.

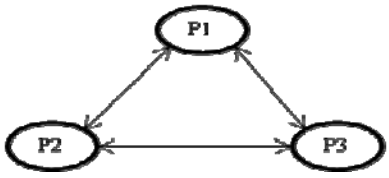


Figure 1 A fully connected parallel processor.

A parallel processor application directed acyclic graph (DAG) is represented as G = (T, E) where T is set of nodes which represents the tasks, and E is set of directed edges which represent the execution dependencies as well as the communication cost between two tasks on different processor. Let us suppose T consist of m non preemptive tasks as:

$T= \{ t_{j:} j= 1,2 ,3 ,4\ldots\ldots ,m\}$

A directed edge set E consist of k edges ranging from k= 1, 2, …, r this represent the precedence relationship among the no. of tasks. Suppose any two tasks t_1 and t_2 € T having a directed edge e_1 i.e. edge from t_1 to t_2 which mean that t_2 cannot scheduled until t_1 has been completed, t_1 is predecessor of $t_{2,}$ t_2 successor of t_1, under the relation of dependency on multiprocessor system as shown in Figure 2 tasks t_8 cannot start before tasks t_4 and t_5 finishes execution and gathers all the communication cost data from t_4 and $t_{5.}$

The weight of vertices are denoted by $W_{.}$

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

This represent the execution duration of the corresponding tasks and they are varies for same task on different processor due to heterogeneous environment.

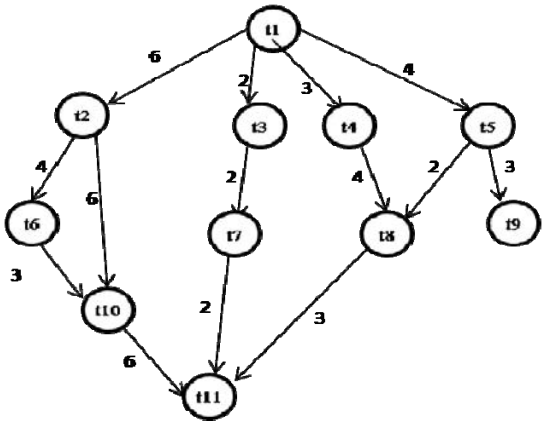


Figure 2: A DAG with task preference.

During experiment we have consider one another factor which represent the communication cost between two processor, if they are scheduled on different processor and notice if both tasks are scheduled on same processor then this factor is null, denoted by C as:

Table 1: Task execution matrix on different processors.

W_{ij}	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11
P1	6	7	7	6	5	9	10	8	7	6	4
P2	7	9	9	7	6	11	13	10	9	7	5
P3	8	11	10	8	7	12	14	11	10	9	6

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

Figure 2 show an example of DAGs. It consist of a set tasks $T = \{t_{j:} j = 1, 2, 3, \ldots 11\}$ Figure 1 set of processors $P = \{p_{i:} i = 1, 2, 3\}$ and table 1 shows matrix of execution time of each task on processor $p_{1,}$ $p_{2,}$ $p_{3,}$ because of heterogeneous environment every processor works on different speeds and processing capabilities. We consider the speeding pattern of processors as p_1 is much faster than p_2 and p_3 and relation formed as
$p_1 < p_2 < p_3$.

3. Genetic algorithm

A genetic algorithm (GA) is based on principle of survival of the fittest. And it is progress through a simple cycle of processing stages: initial Population, evaluation of fitness function, operator for selection, crossover, and mutation. The individual are encoded in the population strings known as chromosomes. After the encoding of chromosomes is occurred then calculate the fitness of individual in a population. For task scheduling, a chromosome represents a solution to scheduling problem as form a schedule S. A schedule consists of the processor allocation and the start of each node of the task graph. [10, 11] An efficient coding scheme for chromosomes makes it beneficial for calculation of fitness operator. This measure is used to select probabilistically individual for crossover. Crossover operator fuses the information continued within pairs of selected parents by placing random subsets of information from both parents. By the impact of random subsets children chromosomes may or may not have higher fitness value than their parents.

3.1 *Initial population*

The first step during GA is initial population creation, requirement for number of processors, number of tasks and size of population. Each individual consists of exactly one copy of each tasks i.e. repetition of task on individual processor is not allowed. And we can arrange them like the length of all individual in an initial population is equal to number of tasks in the target DAG, and during experiment each task is randomly assigned to the processors.

3.2 *Fitness operator function*

The objective of the fitness operator in task scheduling is to find shortest possible schedule, the fitness of chromosome is directly related to length of associated schedule. For task scheduling problem we can consider various factors like throughput, turnaround time, and processor utilization, etc but the fitness function used for GA for finding the shortest possible schedule we focus on total finish time for the required schedule, which include execution time and communication delay time. Fitness function distributes the evolution in two different components. The first component is fitness of task which equipped us with the knowledge all the tasks are executed and scheduled in legal order. Here legal order meant that the scheduling of a pair of task on single processor if the pair is independent to each other. The second component is fitness of processor i.e. which attempt to minimize processing time. By impact of this component of fitness operator we get possible shorter schedule.

Table 2: arbitrarily assigned tasks to processor

Processors	Order of Tasks (legal)	Order of Tasks (illegal)
P1	t1, t4, t6, t9	t4, t6, t1, t9
P2	t2, t3, t7, t11	t3, t7, t2, t10
P3	t5, t8 ,t10	t5, t8, t10

In above table legal order because the P1 start execution from t1, whereas in case of illegal order nor a single processor start their execution from task t4, t3, t5 until task t1 is execute. So initiation of illegal task needs some information from t1 task. Here we consider some general schedule of tasks S1 and S2 on processors system for uniprocessor and multiprocessors. Consider in S1 scheduler all tasks execute on P1 (uniprocessor system) and in S2 schedule tasks are randomly distributed (as above legal order) on all processors as execution time shown in table 1.

S1: $t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_6 \rightarrow t_7 \rightarrow t_8 \rightarrow t_9 \rightarrow t_{10} \rightarrow t_{11} \rightarrow$

Total finish time = 75, Communication cost = 0

S2: $P_{1:}\ t_1 \rightarrow t_4 \rightarrow t_6 \rightarrow t_9$ $P_{2:}\ t_2 \rightarrow t_3 \rightarrow t_7 \rightarrow t_{11}$
$P_{3:}\ t_5 \rightarrow t_8 \rightarrow t_{10}$

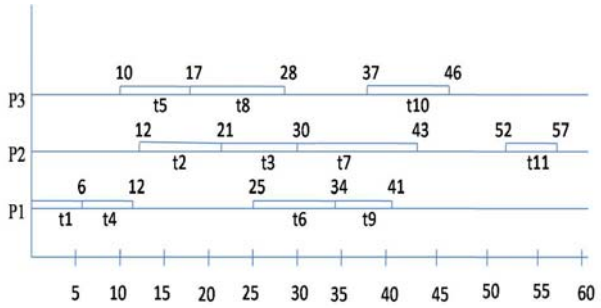


Figure 3: Execution Schedule of S2.

In the figure 3 vertical axes shows the processor and horizontal axis shows time. In the figure single line represents the represents the waiting and idle time of processors and bar with numeric data (above) shows the starting and finishing time of task and name of task (below).

Total finish time = Execution time + comm. Cost
= 57 time unit.

The schedule scheme in S1 represent total finish time of 75 units and S2 shows 57 units, i.e. the proper calculation of fitness operator function reduces the total finish time.

3.3 *Selection operator*

Under the working of GA the selection operator emphasis for selection of good chromosome vector and reject the bad ones. The criteria for picking good or bad vector depends upon the fitness function value, if the chromosome pattern is good then they have better chances to reproduce by selecting best chromosomes from parents and offspring.[13] This procedure is performed by rotating roulette wheel strategy.

3.4 *Crossover operator*

The presence of this operator in GA is more significant. It implements the principle of evolution. New chromosome is generated with this operator by combination of two randomly selected parental chromosomes by inheriting ancestor genetic material. The crossover rate gives the probability that a pair of parents will undergo crossover. But this newly generated offspring chromosome vector may have a fitness value less than or greater than their parental chromosome. Basically they are of two types: one point crossover and two point crossover. [5, 12, 14]

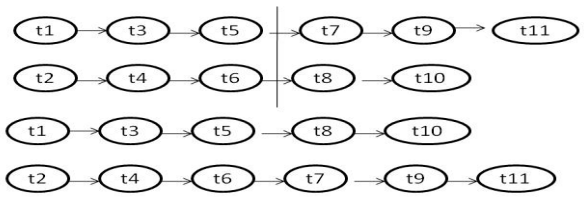


Figure 4: One point crossover

In case of one point crossover portion towards right of chromosome vectors for parents 1 and 2 are exchanged to form two offspring as shown in figure 3. and in two point crossover the chromosome vector is divided with two cross point into three portion left, right and middle for both parents, then in offspring generation left and right remain as intact but middle portion is exchanged as shown in figure 4. In the both cases the vertical lines

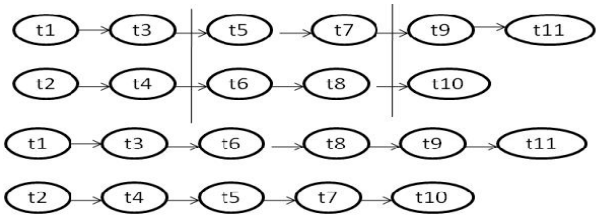


Figure 5: Two point crossover

depict the crossover point. Here the above two bars of oval shape represents the parents and below two bars represents the offspring.

3.5 *Mutation operator*

Mutation is used to get away from early convergence. In this paper the uses of mutation operation is not common. We are using partial-gene mutation randomly which selects a chromosome and changes a randomly selected gene (T_i , P_i) to (T_i , P_j) for which avail time(P_j) is minimum over all the processors for task Ti. It is to notify that partial-gene mutation changes the processor which a task is assigned to, whereas crossover assigns a set of tasks to, probably, different processors. In this light, partial-gene mutation was applied in conjunction with crossover operator. [14] This operator is designed to reduce the idle time.

Now the procedure of Genetic algorithm is:

Level1. Let Generation $G_{counter} = 0$
Analysis for number of processor, DAG (task execution matrix), communication cost, count tasks.

Level2. Generate the initial population by choosing
arbitrarily chromosomes.

Level3. Estimate the objective and fitness function
of each Individual for the legal strings.
Based on following
3 (a) Task fitness.
3 (b) Processor fitness.
(A zero fitness value is assigned for illegal solutions as a price to reduce its chance
for reproduction.)

Level4. Selection Operator i.e. sort in a descending
order all the population according to their
fitness.

Level5. Carry out crossover operation (implement

crossover function on chromosome)
and mutation operation (swap of chromosomes with same height)

Level6. Selection of population size of chromosomes from parents.

Level7. Generation counter $G_{counter} = G_{counter} + 1$

Level8. If $G_{counter} = \max G_{counter,}$ and exit with best solution and stop
Else $G_{counter} <$ generation size Go to 3

4. Node duplication

As we know major challenge in scheduling to avoid interprocessor communication. A node often has to wait for entering communication. As shown in fig 2 task t2 and t3 are delayed due to communication from task t1 and both processor P2 and p3 run idle during that time. now after task duplication communication from task t1 is local to on each processor as shown in figure 8 of the target system and tasks t3, t2 , t5 can start immediately after t1 finishes. Node duplication is employed in a large variety of scheduling approaches by Kruatrachue and Lewis, and Ahmed and Kwok. Here we introduced the node duplication concept with the genetic algorithm based heuristic. During scheduling with node duplication, it might happen that some nodes are unnecessarily duplicated. But still schedule remains valid without any effect on its length. [16, 17]

5. Experimental Results

After application of our Genetic algorithm to the DAG with task preference shown in figure 2 onto heterogeneous parallel processors as figured in 1 is shown in figure 6. Here our motive to reduce the total finish is fulfill, the completion time is obtained by our method is 45 time units. Under this experiment we compare the results with traditional task scheduling methods like FCFS and priority scheduling and new genetic algorithm heuristic with node duplication. During FCFS policy tasks are assigned to three different processors are as:

$P_{1:}\ t_1 \rightarrow t_4 \rightarrow t_7 \rightarrow t_{10}$ $P_{2:}\ t_2 \rightarrow t_5 \rightarrow t_8 \rightarrow t_{11}$
$P_{3:}\ t_3 \rightarrow t_6 \rightarrow t_9$

While we analysis execution diagram of FCFS like (figure3 of schedule3) we get the total finish time = 58 time units. [15, 13] and During Priority scheduling (without preemption) we need some more data as priorities of tasks shown in figure1.

Task	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}
Priority	1	2	5	7	6	3	7	8	9	4	8

In case of priority scheduling tasks are assigned priority values in scheduling pipe then according to their priority value tasks occupy their space in different processors as shown below:

$P_{1:}\ t_1 \rightarrow t_2 \rightarrow t_6 \rightarrow t_{10}$ $P_{2:}\ t_3 \rightarrow t_5 \rightarrow t_4 \rightarrow t_8$
$P_{3:}\ t_9 \rightarrow t_7 \rightarrow t_{11}$

During implementation of list scheduling with start time minimization we get the following schedule for processors

P1: $t_1 \rightarrow t_2 \rightarrow t_5 \rightarrow t_6$ $P_{2:}\ t_3 \rightarrow t_7$
P3: $t_4 \rightarrow t_9 \rightarrow t_8 \rightarrow t_{11}$

The above execution schedule reveals that we get the total finish time = 48 time units.

After analysis of execution schedule of priority scheduling we get the total finish = 67 time units. Here processors with same priority tasks are arranged in FCFS manner. After applying the genetic algorithm, then the best optimal schedule of tasks we get as

P_1: $t_1 \rightarrow t_5 \rightarrow t_4 \rightarrow t_8 \rightarrow t_9$ P_2: $t_2 \rightarrow t_6 \rightarrow t_{10}$
P_3: $t_3 \rightarrow t_7 \rightarrow t_{11}$

We can analyze that now after applying Genetic Algorithm to collection tasks we performed better than other type of scheduling algorithm. The total finish time of enhanced GAs scheduler is 45 time units as shown in execution graph in figure 7.

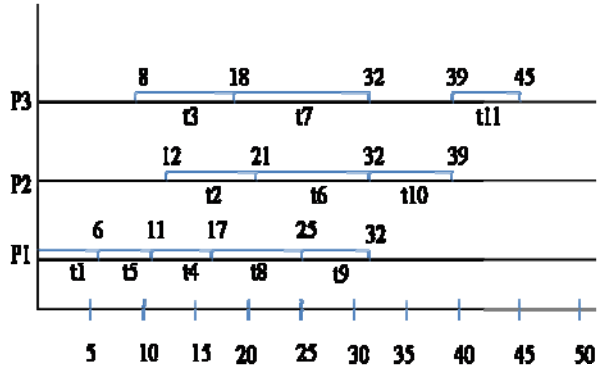


Figure 7: Execution schedule of task with genetic algorithm.
The axes description same as figure 3 schedule.
During the implementation of Genetic algorithm with concept of node duplication (NGA) to collection of task as shown in DAG considered above. The total finish time of NGA is 39 time units as shown in figure 8
The optimal schedules for processors are

P_1: $t_1 \rightarrow t_5 \rightarrow t_4 \rightarrow t_8 \rightarrow t_9$ P_2: $t_1 \rightarrow t_2 \rightarrow t_6 \rightarrow t_{10}$
P_3: $t_1 \rightarrow t_3 \rightarrow t_7 \rightarrow t_{11}$

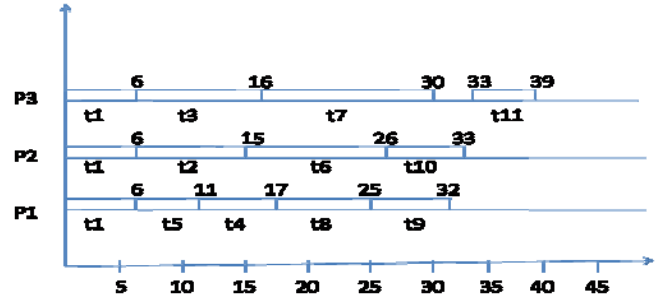


Figure 8: Execution schedule of task with Node duplication Genetic algorithm (NGA).
The axes description same as figure 3 schedule.

6. Performance analysis

Now during performance analysis of above scheduling algorithms we focus on the measure of speedup and efficiency. [15]

Speed up (T_{sp}): Speed up is defined as the ratio of completion time on uniprocessor system to completion time of multiprocessor system.

Node duplication Genetic Algorithm (NGA):
Speed up (T_{sp}) = 75/39 = 1.9230

(¢) = (T_{sp} * 100) / n; where n = number of processors.
Efficiency (¢) = (1.9230 * 100) / 3 = 64.1%

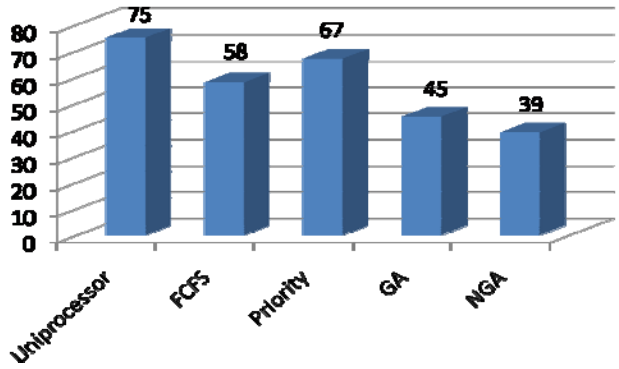
Genetic Algorithm
T_{sp} = 75 / 45 = 1.6667 ¢ = (1.6667 * 100) / 3 = 55.5566 %

List Scheduling with start time minimization:
T_{sp} = 75 / 48 = 1.5625
¢ = (1.5625 * 100) / 3 = 52.08 %

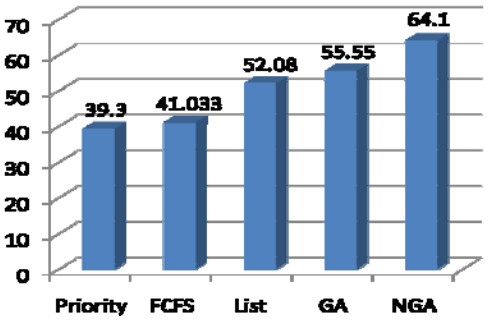
FCFS Scheduler:
T_{sp} = 75 / 58 = 1.2931
¢ = (1.2931 * 100) / 3 = 41.033 %

Priority Scheduler:
T_{sp} = 79 / 67 = 1.1791
¢ = (1.1791 * 100) / 3 = 39.30 %

Time Analysis



Performance Analysis



6. Conclusion

In this paper we have proposed a GA with the node duplication (NGA) based technique for deterministic heterogeneous multiprocessor system which comprises the communication cost in precedence to minimize the total finish time and improve the throughput of system. During comparison with other algorithms our NGA performance is best with finish time. But Node duplication introduces more computation load into the parallel system in order to decrease the cost of crucial communication. The main factor here is which task should be duplicated to reduce the overall time. In future we can attempt to implement this proposed NGA method also for the problem of nondeterministic heterogeneous multiprocessor used for real time, where some benchmarks can

be configured during runtime. Here we can see in the above charts for time analysis and performance analysis NGA is compared with GA, FCFS, Priority and List scheduler with start time minimization shows NGA with 64.1% and GA of 55.55% and List scheduling of 52.08 and 41.033% of FCFS and 39.033 of Priority Scheduling.

7. References

[1] Sara Baase, Allen Van Gelder, "Computer Algorithms" published by Addison Wesley, 2000, Page No.557-562
[2] Sartaj Sahni, " Algorithms analysis and Design", Published by Galgotia Publications Pvt. Ltd., new Delhi, 1996
[3] Anup Kumar, Sub Ramakrishnan, Chinar Deshpande, "IEEE Conferences on Parallel Processing 1994, Page No. III-83-III-87
[4] Mohammad Moghimi Najafabadi, Mustafa Zali, Shamim Taheri, Fattaneh Taghiyareh,"IEEE proceeding 2007, Page No. 226-230.
[5] Elnaz Zafarani, Amir Masoud Rahmani, Mohammad Reza feizi Derakhshi, "IEEE Proceeding 2008, Page No. 248-251
[6] David E. Goldberg, "Genetics Algorithms in Search Optimization and machine Learning", Published by Pearson Education, 2004, Page No. 60-83.
[7] Mitchell, Melanie, "An Introduction to Genetic Algorithms, published Bu MIT Press 1996
[8] Sung-Ho Woo, Sung –Bong Yang, Shin-Dug Kim and Tack-don Han."IEEE Trans on Parallel System, 1997, Page 301-305
[9] M.Salmani Jelodhar, S.N Fakhraie, S.M Fakhraie, M.N Ahmadabadi," IEEE Proceeding",1999, Page No. 340-347.
[10] Man Lin and Laurene Tianruo Yang,"IEEE Proceeding",1999 Page No. 382-387
[11] Yajun Li, Yuhang Yang, Maode Ma, Rongbo Zhu, "IEEE Proceeding", 2008
[12] Yi-Wen Zhong, Jian-Gang yang, Heng-Nlan QI,"IEEE Proceeding ", August 2004, Page No. 2463-2468.
[13] Ceyda Oguz and M. Fikret Ercan, "IEEE Proceeding", 2004, Page No. 168-170
[14] Kai Hwang & Faye A. Briggs," Computer Architecture and Parallel Processing", Published Tata McGraw Hill, 1985, Page No. 445-47, 612
[15] W. Smith, I. Foster, V. Taylor, Scheduling with advanced reservations, Proceedings of the 14th
[16] International Parallel and Distributed Processing Symposium, IEEE Computer Society, Cancun,May, 1–5, 2000, pp. 127–132.
[17] F.E.Sandnes and G.M. Megson. An evolutionary approach to static taskgraph scheduling with task duplication for minimized interprocessor traffic,pages 101-108, Taipai, Taiwan, July 2001
[18] T. Hagres and J. Janeaek. Ahigh performance, low complexity algorithm for compile Time task scheduling in heterogeneous systems, Parallel computing, 31(7): 653-670, 2005