International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org, editorijaiem@gmail.com
Volume 2, Issue 7, July 2013 ISSN 2319 - 4847

A Genetic Approach for Multiprocessors Task Scheduling

Savita Narang[1], Sunita Dhingra[2]

[1]M. Tech Student,Department of CSE, UIET, MDU Rohtak, Haryana (India)
[2]Assistant Professor, Department of CSE, UIET, MDU Rohtak, Haryana (India)

ABSTRACT

Multiprocessor Task Scheduling Problem is a very challenging problem in computer world. Multiprocessors have emerged as a powerful computing means for running real time applications; especially that a uni-processor system would not be sufficient enough to execute all the tasks. In such practical problems that require a task to go through more than one stage where each stage have several parallel processors, the processing environment requires an efficient algorithm to determine when and on which processor a given task should execute. The Multiprocessor task scheduling problem is a NP-hard problem, hence it is very important to find its solution in minimum time. Many heuristic methods have been given by various researchers to solve this problem. Genetic algorithm is a powerful optimization tool used to solve this problem. The present work considers the multiprocessor task scheduling problem where number of processor as well as number of tasks are taken as variable with an objective to minimize the make span (completion time of the last task). In the present work we considered a problem consisting of 5 processors and 10 processes. All the permutations of task are sequenced on 10 different randomly generated sequences of processors separately. The objective function was to find the optimal sequence of processor and sequence of processes that gave minimum make span with different randomly generated burst times. It was found that a particular sequence of processor gives minimum make span with most of the burst times although the task sequence was different in most of the cases.

Keywords: Genetic algorithm, Multiprocessor, Task scheduling, optimization.

1. INTRODUCTION

Multiprocessor task scheduling is a generalization of classical processor scheduling problem that allows the task to run on more than one processor. Generally the task scheduling is done to overcome the drawbacks of the uni-processor systems [1]. When all task run on uni-processor systems it is very slow. So a number of processors are used to get parallelism and pipelining. The purpose of general task scheduling is fairness which means that the computer's resources must be shared out equitably among users [2].

1.1 GENETIC ALGORITHM

The GA is a stochastic global search method that mimics the metaphor of natural biological evolution. GAs operates on a population of potential solutions applying the principle of survival of the fittest to produce (hopefully) better and better approximations to a solution[3]. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics. This process leads to the evolution of populations of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation [4].Genetic Algorithm (GA) is based on the basic phenomena of chromosome gene code interchange. A process fully based on random selection of data and modification. In most of the cases we use fitness function to select the best suited chromosomes[5].

1.2 WORKING PRINCIPLE OF A SIMPLE GA

1. Begin
2. INITIALIZE population with random candidasolutions.
3. EVALUATE each candidate
4. REPEAT UNTIL (TERMINATION CONDITION satisfied)
5. DO
   a. SELECT parents;
   b. RECOMBINE pairs of parents;
   c. MUTATE the resulting offspring;
   d. EVALUATE new candidate;
   e. SELECT individuals for the next generation;
6. DO
7. END

In genetic algorithm we initialize the population by some sequence of tasks or schedules these are called chromosomes. From this population we select the best fit schedules and apply GA operators on them called crossover and mutation. The offsprings produced as a result of this are evaluated for their fitness and suitable offsprings are added to the population list. In this way after every generation we get a population more suited to the fitness function. When we get a schedule having a minimum makespanstoping criteria is met and GA stops[6]

2. PROPOSED WORK

The work deals with the multiprocessor task scheduling problem(MPTS) where number of processors as well as number of processes both are taken as variable. The work is based on the deterministic model i.e. the number of processors execution of task on different processors is known in advance. In addition to it , the communication cost between two task is considered to be negligible and the multiprocessor system is non-pre-emptive. The processors are homogeneous and each processor completes the current task before the new one starts its execution. The assumptions for the MPTS are given below:

2.1 Assumptions:

1. Processors are kept homogeneous.
2. No pre-emption is allowed.
3. All the jobs and machines are available at time Zero.
4. Machines never break down.
5. The processing time of each task on all the machine are known.
6. Each machine is continuously available for assignment.
7. The first machine is assumed to be ready whichever and whatever job is to be processed on it first.
8. Processors may remain idle.
9. Task cannot be cancelled or broken down into smaller tasks.

The present work is the special case of multiprocessor task scheduling in which the processes and processors both are taken as a variable and our goal is to get an optimal schedule for the n tasks on m processors.

2.2 Problem Statement

In multiprocessor task scheduling, there are n number of processes denoted as Ji , where i=1,2,.. n, that are to be processed on m processors denoted as Mk, where k= 1,2,..,m. Number of processors as well as number of processes both are taken as variable. Operation of ith process on kth processor will be denoted by Oki with the processing time Pki. Operation sequence is same for all the processes on all the processors.

We are to find a sequence of the processes on a sequence of processor that give minimum makespan.Once a machine starts to process a job, no interruption is allowed. The time required for all operations to complete their processes is called make span. Our intention is to minimize this makespan value. All the jobs and machines are available at time Zero.

2.3 Methodology:

We have generated 10 random permutation sequences of processors. These processor sequences were fixed in a processor sequence list(PL). We selected processor sequences one by one from the PL and applied the following procedure on each sequence. We took a randomly generated processing time pki for task i and processor k. Picked one processor sequence from the PL x1 , x2 , x3 ,....xm. . Different permutations of n tasks (j=1, 2, 3 , ...n) are sequenced through these m processors x1 , x2 , x3 ,....xm. Then the problem was to find best sequence of task for the given sequence of processor.

Given the processing time pki for task i and processor k , and a task permutation$\pi 1$, $\pi 2$, $\pi 3$ ....πn and a processor permutation x1 , x2 , x3 ,....xm.where n tasks (j=1, 2, 3 , ...n) are sequenced through m processors(k=1, 2, 3 , ...m) using the same permutations. The completion time C(xk, πi) for ith task of the given permutation π and the kth processor of the given permutation x can be calculated as:

$C(x_1,\pi_1) = P_{x_1\pi_1}$ (1)

$C(x_1,\pi_i) = C(x_1,\pi_{i-1}) + P_{x_1\pi_i}$ i=2,…,n (2)

$C(x_k,\pi_1) = C(x_{k-1},\pi_1) + P_{x_k\pi_1}$ k=2,…,m (3)

$C(x_k,\pi_i) = \max\{C(x_k,\pi_{i-1}),C(x_{k-1},\pi_i)\} + P_{x_k\pi_i}$ i=2,…,n, k=2,…,m (4)

Under these specifications, the value of the objective function, the makespanC_{max} is: $\mathbf{C_{max} = C(x_m, \pi_n)}$

Where C_{max} =Completion time for the last operation on last processor. In this way we can get makespan for all other permutations of tasks on same sequence of processor and select minimum C_{max} for that sequence of processor.

Similarlysequence of task and minimum C_{max} for all other 9 sequences of processors from the PL were calculated for the same processing time $p_{k,i}$.

The sequence of processor with the sequence of processes and makespan corresonding to` the minimum out of all minimum C_{max} for that processing time $p_{ki}$is displayed as the result for that processing time.
The whole process is repeated with different processing times(also callled burst time) and we are to find best processor sequence that give best results with most of the processing times.
In this present work , multiprocessor task scheduling is implemented using GA, in which following work is to be done:

- Initial population generation.
- Fitness function to evaluate each individual of the feasible solution
- Crossover and mutation for reproduction.
- Selection of individuals for the next generation.

Fitness of an individual is defined by the makespan of the schedule(i. e.completion time of the latest task). To minimize the makespan the basic criteria is the the time, when the processor finishes the last task[6].

$$Makespan = \min_{i \in schedule} \{max\, c_i\}$$

In this way the individuals from the current population are selected based on their fitness and mating pool is created for the reproduction stage.

3.EXPERIMENTAL RESULTS

Experiment has been performed taking following
Number of Processors=5
Number of Tasks=10
First of all random burst time matrices is generated for 5 processor and 10 tasks then, 10 random permutations of processors are found and fixed. GA is applied on each permutation of processor with all permutation of tasks separately. For every permutation of processor we run GA 6 times and minimum make span for each combination of
processor is found then same is done for the other combination of processors and at the end we come to know processor combination that gives minimum make span .
Various GA options used are:

- crossover probability= 0.8
- elite count=2
- no. of generations = 20
- population size = 5
- popinit range = num of tasks
- Crossover operator = PMX crossover
- Mutation = Swap mutation

Table1: The randomly generated combinations of processors:

Number of the processor sequence	Processor sequence
1	2 1 5 4 3
2	2 4 3 1 5
3	1 2 5 4 3
4	3 1 2 5 4
5	3 2 1 4 5
6	1 5 3 2 4
7	4 5 3 1 2
8	5 3 2 4 1
9	2 4 5 3 1
10	5 1 4 2 3

The same experiment is repeated with 5 different randomly generated burst times and results of all experiments are as under:

Table 2: Result of Burst time 1

Processor Sequence	Makespan #Run1	Makespan #Run2	Makespan #Run3	Makespan #Run4	Makespan #Run5	Makespan #Run6	Minimum Makespan Value
1	833	811	851	796	825	803	796
2	813	752	760	788	814	831	752
3	935	914	848	857	914	912	848
4	784	800	835	816	797	867	784
5	797	813	900	838	795	881	795
6	813	789	849	844	873	835	789
7	844	795	799	852	840	783	783
8	837	834	911	877	860	834	834
9	813	822	857	818	845	857	813
10	880	857	895	830	845	899	830

Result: Minimum makespan:752, Processor sequence: 2 4 3 1 5, Task sequence: 8 7 6 4 10 1 3 9 2 5

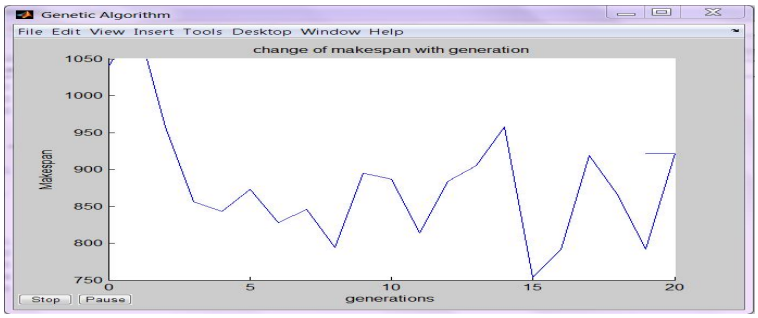

Figure 1: The change of makespan with generation(Burst Time 1)

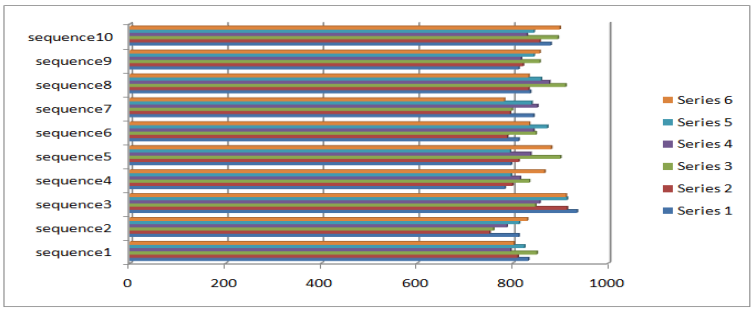

Figure 2: Different runs at different processor sequence(Burst Time 1)

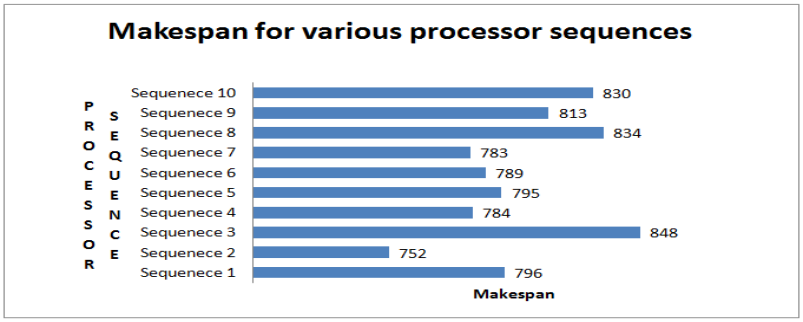

Figure 3: Minimum Make span of different processor sequences(Burst time1)

Table 3: Result of Burst time 2

Processor Sequence	Makespan #Run1	Makespan #Run2	Makespan #Run3	Makespan #Run4	Makespan #Run5	Makespan #Run6	Minimum Makespan Value
1	795	788	863	842	838	762	762
2	807	847	797	814	810	790	790
3	914	845	914	873	846	868	845
4	793	810	944	771	808	783	771
5	848	861	851	859	806	776	776
6	849	898	826	828	880	810	810
7	769	817	840	789	816	844	826
8	827	849	859	859	838	857	827
9	841	843	802	857	841	972	802
10	845	959	959	872	995	841	841

Result: Minimum makespan: 762, Processor sequence: 2 1 5 4 3, Task sequence: 10 3 6 12 4 5 9 7 8

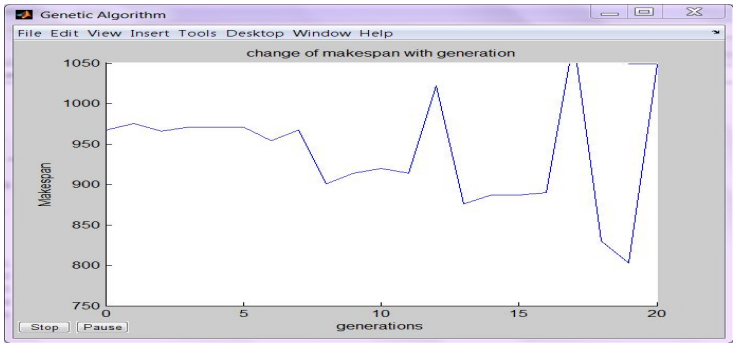

Figure 4: The change of makespan with generation(Burst Time 2)

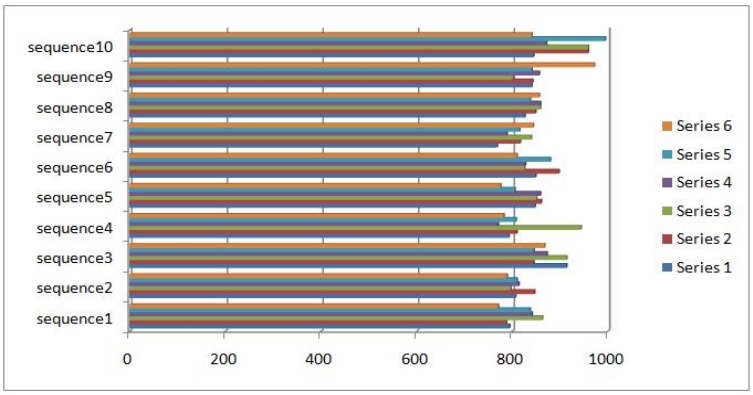

Figure 5: Different runs at different processor sequences(Burst Time 2)

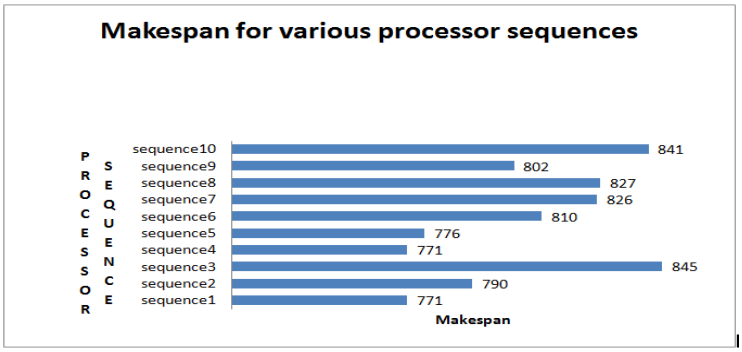

Figure6: Minimum Make span of different processor sequences(Burst Time 2)

Table 4: Result of Burst time 3

Processor Sequence	Makespan #Run1	Makespan #Run2	Makespan #Run3	Makespan #Run4	Makespan #Run5	Makespan #Run6	Minimum Makespan Value
1	1015	988	987	963	1001	1101	963
2	940	963	996	963	986	903	903
3	971	1029	981	1023	941	992	941
4	922	975	958	912	885	949	912
5	932	1029	902	1047	913	967	902
6	982	932	1009	991	958	992	932
7	914	991	999	986	937	1031	914
8	933	940	942	945	940	913	913
9	1030	1032	952	1001	1001	1043	952
10	968	910	971	955	999	974	910

Result: Minimum makespan: 902, Processor sequence: 3 2 1 4 5, Task sequence: 7 9 1 5 3 8 4 10 2 6

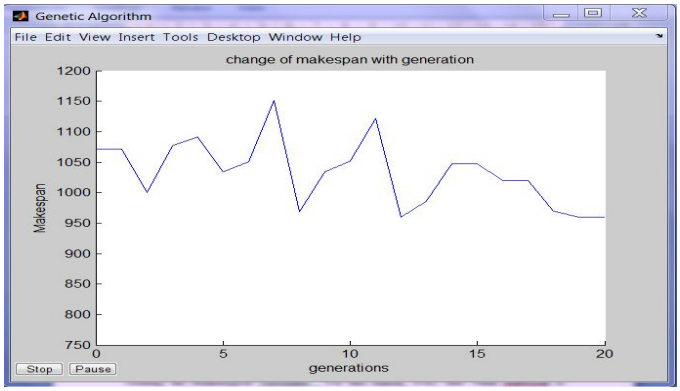

Figure 7: The change of makespan with generation(Burst Time 3)

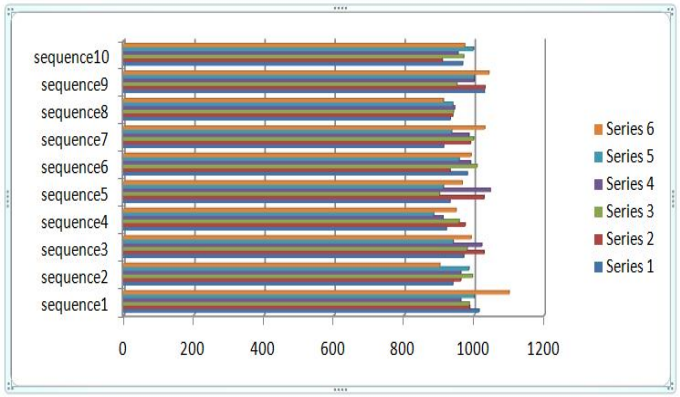

Figure 8: Different runs at different processor sequences(Burst time 3)

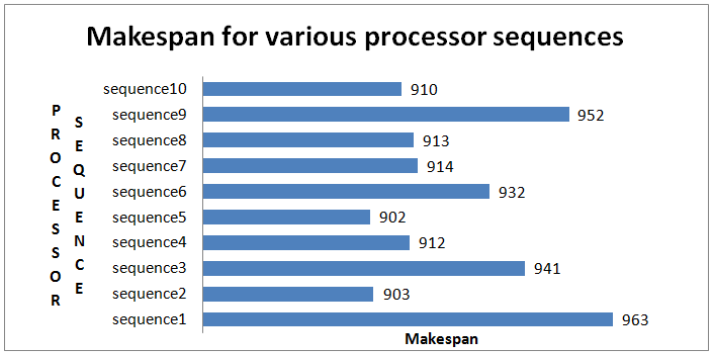

Figure 9: Minimum Make span of different processor sequences(Burst Time 3)

Table 5: Result of Burst time 4

Processor Sequence	Makespan #Run1	Makespan #Run2	Makespan #Run3	Makespan #Run4	Makespan #Run5	Makespan #Run6	Minimum Makespan Value
1	868	866	765	842	866	873	765
2	831	854	841	829	784	821	794
3	864	806	777	767	793	795	767
4	849	865	775	851	866	866	775
5	808	828	792	795	799	829	792
6	806	843	798	799	793	814	793
7	853	803	882	873	871	876	803
8	818	779	795	842	836	833	779
9	804	790	837	871	779	818	779
10	780	841	780	851	835	828	780

Result: Minimum makespan: 765 , Processor sequence: 2 5 4 3 1, Task sequence:3 4 2 6 9 7 5 8 10 1

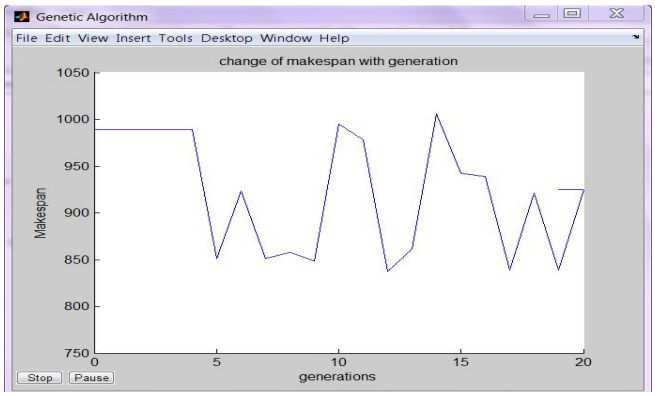

Figure 10: The change of makespan with generation(Burst Time 4)

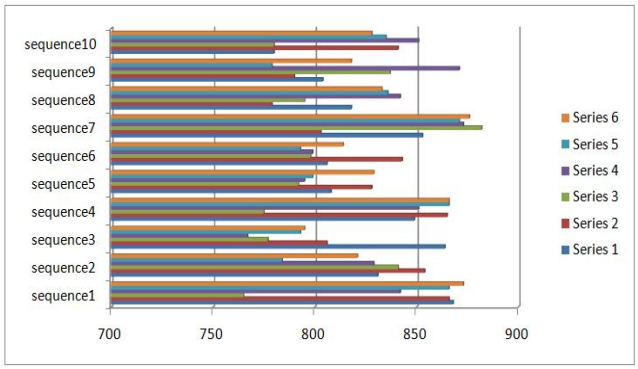

Figure 11: Different runs at different processor sequences(Burst Time 4)

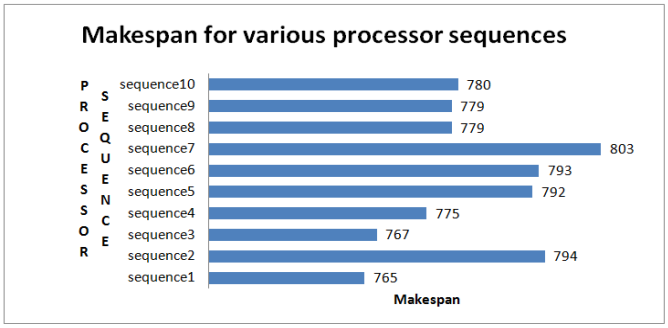

Figure 12: Minimum Make span of different processor sequences(Burst Time 4)

Table 6: Result of Burst time 5

Processor Sequence	Makespan #Run1	Makespan #Run2	Makespan #Run3	Makespan #Run4	Makespan #Run5	Makespan #Run6	Minimum Makespan Value
1	805	790	850	785	840	805	785
2	816	808	841	858	786	827	786
3	834	827	831	834	834	831	834
4	799	839	782	825	822	801	799
5	799	807	778	793	822	822	778
6	834	849	825	787	826	787	787
7	776	783	826	816	839	796	776
8	809	815	832	828	847	850	809
9	786	802	811	805	787	809	786
10	784	827	811	813	828	800	784

Result: Minimum makespan: 776, Processor sequence: 4 5 3 1 2, Task sequence:3 6 5 2 8 9 4 1 7 10

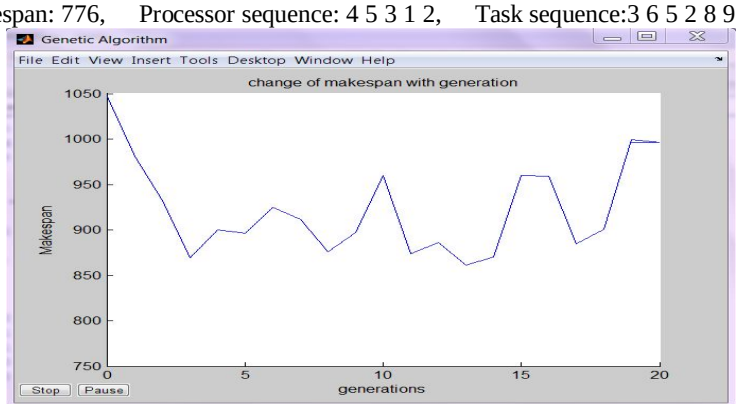

Figure 13: The change of makespan with generation(Burst Time 5)

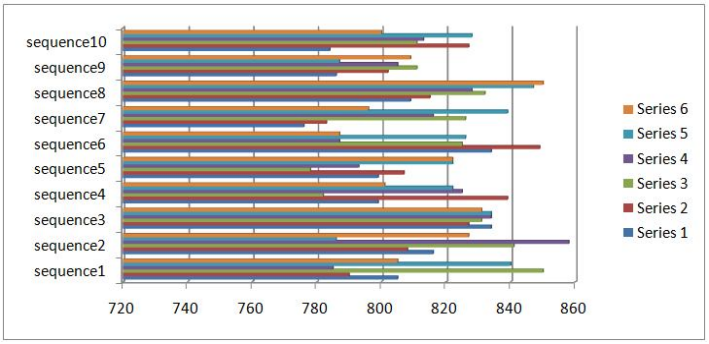

Figure14: Different runs at different processor sequences(Burst Time 5)

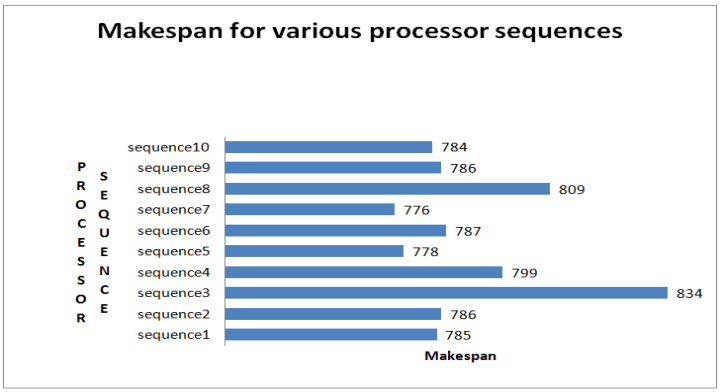

Figure 15: Minimum Make span of different processor sequences(Burst Time 5)

4. COMBINED ANALYSIS

Sequence of processor 2 1 5 4 3 gives minimum makespan twice i.e. 762 and 765.In all other Burst times the processor sequence give the makespan approximately equal to the overall minimum makespan except one case as in burst time 3. So we can conclude that processor sequence 1 i.e. 2 1 5 4 3 gives best results with most of the burst times.Although the minimum makespan values and combination of tasks may be different with different burst times.

5. CONCLUSIONS

In the present work, Multiprocessor task scheduling problem has been implemented using Genetic algorithm. Some of the major conclusions from the present work have been found which are explained below:

1. Makespan of multiprocessor scheduling problem vary with combination of processors as well as combination of processes.
2. Different task combination give different results with different processors combinations
3. Different combination of tasks can give same Makespan values with the same combination of processors.
4. With the same burst time minimum makespan of different combination of processors is different.

Variability of the processors is kept limited to some extent in future all permutations of processors can be considered. In this way performance can further be increased.

REFERENCES

[1.] T.L.Adam , K.M. Chandy, And J.R. Dicson, " A Comparison Of List Schedules For Parallel Processing Systems", Communication OftheAcm, Vol.17,PP.685-690, December 1974.

[2.] Kohler, W.H., "A Preliminary Evaluation Of The Critical Path Method For Scheduling Tasks On Multiprocessor Systems" IEEE Trans. Computers, Vol. C-24, Dec. 1975.

[3.] M.R.GareyAnd D.S. Johnson, Computers And Intractability: A Guide To The Theory Of NP Completeness, San Francisco, CA, W.H. Freeman, 1979.

[4.] Kenneth A. De Jong And William M. Spears, "Using Genetic Algorithms To Solve NP-Complete Problems" Proceedings Of The Third International Conference On Genetic Algorithms, George Mason University, June 4-7, 1989.

[5.] E. Hou, R. Hong, And N. Ansari, "Multiprocessor Scheduling Based On Genetic Algorithms", Dept Of Ece, New Jersey Institute Of Technology, Technical Report, Aug. 1990

[6.] Sandeep Jain And ShwetaMakkar, "Multiprocessor Environment Using Genetic Algorithm" Volume 2, Issue 5, May 2012.

[7.] KirtiNagpal, ShephyBatraAndVaishaliWadhwa, "Proposed Algorithm For Optimization Of Job Scheduling In Multiprocessor Systems Using Genetic Approach" International Journal Of Computer Applications & Information Technology Vol. I, Issue III, November 2012.

[8.] Sachi Gupta, GauravAgarwal, AndVikas Kumar, "An Efficient And Robust Genetic Algorithm For Multiprocessor Task Scheduling" International Journal Of Computer Theory And Engineering, Vol. 5, No. 2, April 2013.