Scheduling in Multiprocessor System Using Genetic Algorithm

Hadis Heidari[1] and Abdolah Chalechale[2]

Razi University, Kermanshah, Iran
[1]h.heidari@pgs.razi.ac.ir , [2]chalechale@razi.ac.ir (corresponding author)

Abstract

Task scheduling is essential for the suitable operation of multiprocessor systems. The task scheduling is prime significance of multiprocessor parallel systems. In this paper, an efficient method based on genetic algorithms is developed to solve the multiprocessor scheduling problem. The paper also aims to provide a comparative study of incorporating heuristics such as 'Earliest Deadline First (EDF)' and 'Shortest Computation Time First (SCTF)' separately with genetic algorithms. We exhibit efficiency of Node duplication Genetic Algorithm (NGA) based technique by comparing against some of the existing deterministic scheduling techniques for minimizing inter processor traffic communication. A comparative study of the results obtained from simulations shows that genetic algorithm can be used to schedule tasks to meet deadlines, in turn to obtain high processor utilization. Performance analysis of NGA is compared with GA, FCFS and List Scheduler. Experimental results show that the NGA is betters than the others. Performance of NGA, GA, List Scheduling, and FCFS is almost 64.1%, 55.55%, 52.08, and 41.033% respectively.

Keywords: *Genetic Algorithm, Task Scheduling, Node Duplication Genetic Algorithm*

1. Introduction

Multiprocessor task scheduling is an important and computationally difficult problem. The problem of task scheduling is verified the processor, when and on which processor a given task performs. Our objective is to reduce the mean task total finish time. The final purpose of this scheduling in a way that the finish time and cost of the execution can minimized. Real-time systems can be categorized into two important groups: hard real-time systems and soft real-time systems. In hard real-time systems, meeting all deadlines is obligatory, while in soft real-time systems missing some deadlines is tolerable [1]. Real-time systems are vital to industrialized infrastructure such as command and control, process control, flight control, space shuttle avionics, air traffic control systems and also mission critical computations. Tasks can be classified as periodic or aperiodic. A periodic task is a kind of task that occurs at regular intervals, and aperiodic task occurs unpredictably. The length of the time interval between the arrivals of two consecutive requests in a periodic task is called period. A preemptive scheduler can suspend the execution of current executing request in favor of a higher priority request. However, a nonpreemptive scheduler executes the currently running task to completion before selecting another request to be executed [2].

Srinivasan and Anderson [3] have studied the dynamic scheduling of task on multiprocessor systems. Page and Naughton [4] presented a scheduling strategy which makes use of a good algorithm as genetic algorithm to dynamically schedule heterogeneous tasks on heterogeneous processors in distributed systems. In [5-7] several approached based on

stationary scheduling are presented. In [8] a scheduling method for single processor systems is introduced.

Till now, solutions for system scheduling cannot solve scheduling problem with high performance. In this paper, an efficient method based on genetic algorithms is developed to solve the multiprocessor scheduling problem.

We exhibit efficiency of NGA based technique by comparing against some of the existing deterministic scheduling techniques for minimizing cost of the execution in inter processor traffic communication. A comparative study of the results obtained from simulations shows that genetic algorithm can be used to schedule tasks to meet deadlines, in turn to obtain high processor practical use. Performance analysis NGA shows NGA with almost 64.1%. The paper also aims to provide a comparative study of incorporating heuristics such as EDF and SCTF separately with genetic algorithms. The execution and communication costs are represented by directed acyclic graph (DAG).

The rest of the paper is organized as follows: genetic algorithm is presented in Section 2. Section 3 reviews task scheduling models based on genetic algorithm. Section 4 provides experimental results and performance analysis. Conclusion and further work are debated in Section 5.

2. Genetic Algorithm

Genetic algorithm is a learning way that is based on biologic evolution. This method introduced by John Holland in 1970. Genetic Programming Inc Company used parallel computer with 1000 node for implementation genetic algorithm in 1999. This algorithm is a technique for finding approximate solution for optimization. Genetic algorithm has initial population, fitness operator function, selection operator, crossover operator, and mutation operator. 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. In [9] an efficient coding scheme for chromosomes is introduced. Crossover operator fuses the information continued within pairs of selected parents by placing random subnets of information from both parents. By the impact of random subnets children chromosomes may or may not have higher fitness value than their parents. The first step during genetic algorithm is initial population creation, requirement for number of processor, number of tasks and size of population. Repetition of task on individual processor is not allowed. The goal of the fitness operator in task scheduling is to find shortest possible schedule. Genetic algorithm pseudocode is in bellow:

```
Genetic Algorithm
begin
        Choose initial population
        Repeat
                Evaluate the individual  fitnesses of a certain proportion of the populatio
              Select pairs of best- ranking individuals to reproduce
       Apply crossover operator
       Apply mutation operator
  until terminating condition
end
```

3. Tasks Scheduler Models with Genetic Algorithm

Each task T_i is characterized by: A_i is arrival time, R_i is ready time, C_i is worst case computation time, and Di is deadline. The scheduler determines the scheduled start time and finish time of a task. If $st(T_i)$ is the scheduled start time and $ft(T_i)$ is the scheduled finish time of task T_i, then the task T_i is said to meet its deadline if$(R_i \leq st(T_i) \leq D_i - C_i)$ and $(R_i + C_i \leq ft(T_i) \leq D_i)$. That is, the tasks are scheduled to start after they arrive and finish execution before their deadlines. A set of such tasks can be said to be guaranteed.

The incoming tasks are held in the task queue and then passed on to the scheduler for scheduling of tasks. It is the central scheduler that allocates the incoming tasks to other processors in the system. Each processor has a dispatch queue associated with it. The processor executes tasks in the order they arrive in the dispatch queue. The communication between the scheduler and the processors is through these dispatch queues. The scheduler works in parallel with the processors. The scheduler schedules the newly arriving tasks and updates the dispatch queue while the processors execute the tasks assigned to them. The scheduler makes sure that the dispatch queues of the processors are filled with a minimum number of tasks so that the processors will always have some tasks to execute after they have finished with their current tasks. The set consists of n processors:

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

Figure 1 show a model than scheduler.

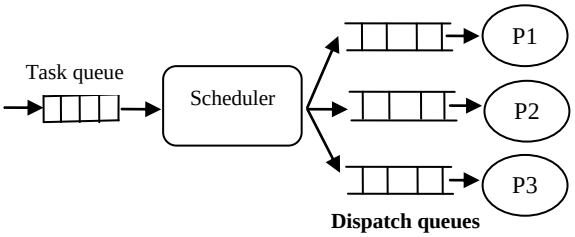

Figure 1. The Scheduler Model

Processors with same links are connected and show in Figure 2.

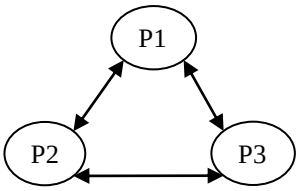

Figure 2. A Fully Connected Parallel Processor

A processor application acyclic graph is represented as G=(T,E) where T is a set of nodes which represents the tasks, and E is the set of edges which represent the execution dependencies as well as the communication cost between two tasks on different processor. Suppose T consists of m non preemptive tasks as: $T = \{t_j : j=1, 2, 3, 4, \ldots, m\}$. A directed edge set E consists of k edges ranging from k=1, 2, …, r. Suppose

any two task t1 and t2 € T having a directed edge e1 i.e. edge from t1 to t2 which mean that t2 cannot schedule until t1 has been completed, t1 is predecessor of t2, t2 is the successor of t1, under the relation of dependency on multiprocessor system as shown in Figure 3 task t8 cannot start before task t4 and t5 finishes execution and gather all the communication cost data from t4 and t5. The weight of vertices is denoted by W: W= $\{w_{ij}: i= 1, 2, \ldots, n; j=1, 2, \ldots, m\}$.

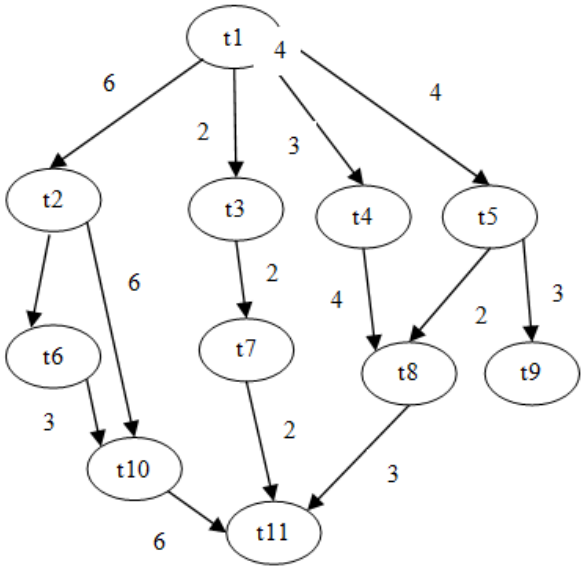

Figure 3. A DAG with Task Preference

If tasks are scheduled on different processor and notice if both tasks are scheduled on same processor then this factor is null, denoted by C as: C= $\{c_k: k=1, 2, 3, \ldots, r\}$. Figure 3 consists of a set tasks T= $\{t_j: j=1, 2, 3, \ldots, 11\}$. In Figure 2 the set of processor is P= $\{P_i: i=1, 2, 3\}$ and Table 1 shows matrix of execution of each task on processors P1, P2, P3.

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

Table 2. Arbitrarily Assigned Tasks to Processors

Processors	Order of Tasks (legal)	Order of Tasks (illegal)
P1	t1, t4, t16, t19	t4, t6, t1, t9
P2	t2, t3, t7, t11	t3, t7, t2, t10
P3	t5, t8, t10	t5, t8, t10

Table 2 shows that P1 start execution from t1, whereas in case of neither 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 consider some general schedule of tasks S1 and S2 on processor system for uniprocessor and multiprocessor. S1 scheduler all tasks execute on P1 (uniprocessor system) and in S2 schedule tasks are randomly distributed on all processors.

$S1: t1 \rightarrow t2 \rightarrow t3 \rightarrow t4 \rightarrow t5 \rightarrow t6 \rightarrow t7 \rightarrow t8 \rightarrow t9 \rightarrow t10 \rightarrow t11$

Total finish time is 75 and communication cost is 0.

$S2: P1: t1 \rightarrow t4 \rightarrow t6 \rightarrow t9 \quad P2: t2 \rightarrow t3 \rightarrow t7 \rightarrow t11 \qquad P3: t5 \rightarrow t8 \rightarrow t10$

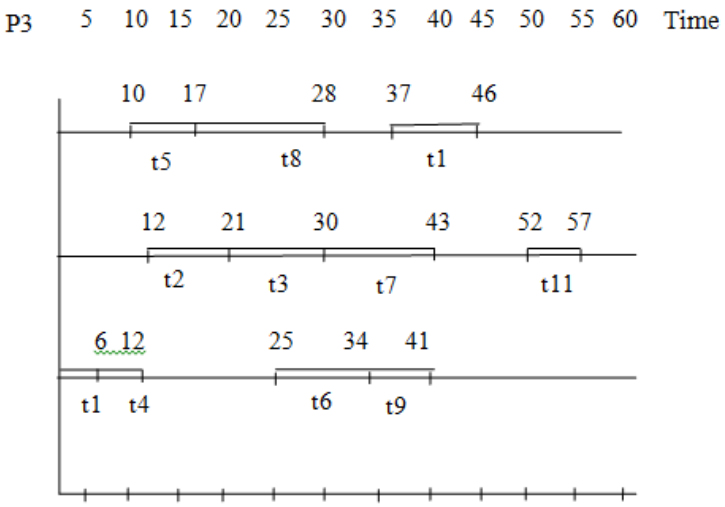

Figure 4. Execution Scheduler of S2

In Figure 4 vertical axes shows the processor and horizontal axis shows time. Single line represents the waiting and idle time of processors and bar with numeric data shows the total finish time equals to sum of execution time and communication time which is equal to 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.

Genetic algorithm duty is selection operator emphasis to find a good chromosome vector and reject the bad ones. New chromosome is generated with crossover operator by combination of two randomly selected parental chromosomes by inheriting ancestor genetic material. Crossover operator has two types: one point crossover and two point crossover.
Figure 5 shows crossover points in genetic algorithm.

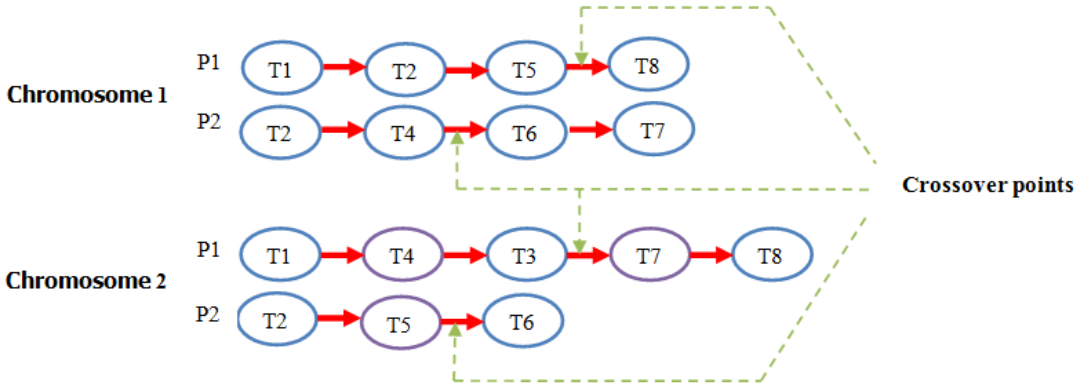

Figure 5. Crossover Points in Genetic Algorithm

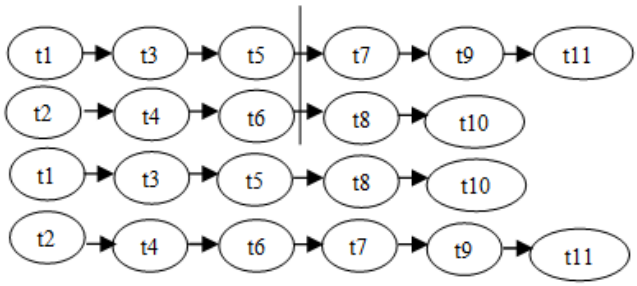

Figure 6. One Point Crossover

In case of one point crossover portion towards right of chromosome vectors for parents 1 and 2 are exchange to form offspring as shown in Figure 6 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 7.

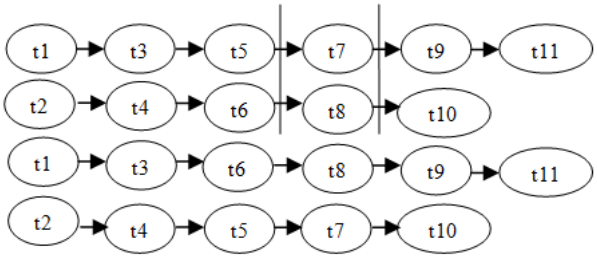

Figure 7. Two Point Crossover

In the both cases the vertical lines depict the crossover point. Mutation operator is used to get away from early convergences. In our study we use 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 available time (P_j) is minimum over all the processors for task T. 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. Figure 8 is an example of one point crossover.

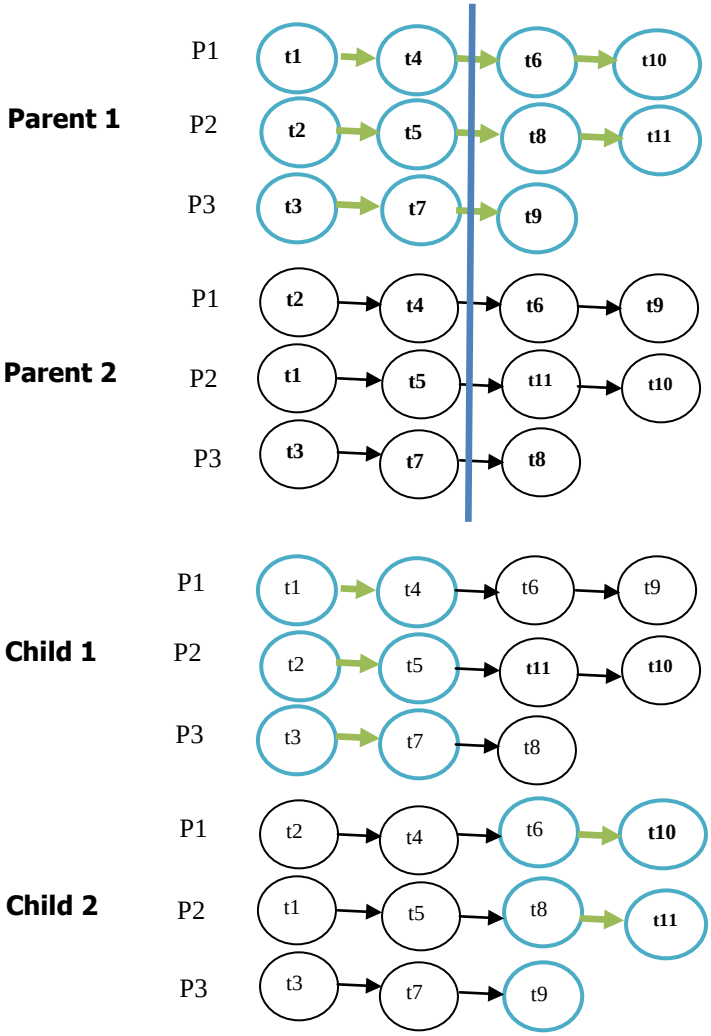

Figure 8. An Example of One Point Crossover

The genetic algorithm is:

Step 1: we set $G_{counter}$ =0

Step 2: generate the initial population by choosing chromosome in form random.

Step 3: appraise the objective and fitness function of each individual.

Step 4: selection operator such as sort in a descending order all the population according to some value such as their fitness.

Step 5: perform crossover operation and mutation operator.

Step 6: choice of population size of chromosomes from parents.

Step 7: increment counter $G_{counter} = G_{counter} + 1$

Step 8: if $G_{counter}$ =max $G_{counter}$ and exit with best solution and stop else $G_{counter}$ < generation size goes to 3 and continues.

A major note in scheduling is to prevent inter processor communication. It means a node sometimes has to wait for entering communication. Node duplication is activated by Kruatrachue and Lewis, and Kwok. During scheduling with node duplication, some of node might happen that some nodes are unnecessarily replicated but still schedule remains prevent without any effect on its length [11]. By attention to Figure 1, scheduling algorithm stages are in below:
1-Generate a task queue 2-Sort the tasks in the increasing order of their deadlines 3-Select a suitable number of tasks for a fixed chromosome size 4-Generate chromosome for the population 5-Sort the genes in each chromosome based on deadline 6-Determine the fitness value of each chromosome in the population 7-Sort the chromosomes within the population depending on fitness value 8-Sort the genes in each chromosome based on deadline, determine the fitness value of each chromosome in the population, sort the chromosomes within the population depending on fitness value 9-Choose the best chromosome.

In run time, task queues of different lengths were generated. The lengths of tasks queues considered were 100, 200, 400 and 600. The worst case computation time, C_i, of a task T_i has been chosen randomly between a minimum and maximum computation time value denoted. The value of minimum for computation time is 30 and the value of maximum for computation time is 60. The value for the deadline of a task T has been randomly chosen between (R_i+2C_i) and (R_i+rC_i) where $r \geq 2$ and the value of r has been chosen to be 3 [12]. The mean of the arrival time was assumed to be 0.05. The number of processors considered was 10. The chromosome size has been assumed equal to the number of tasks considered at a time for scheduling. Depending on this, the value for the chromosome size has been varied between 20 and 60. Fitness value is shown number of tasks that are feasible. The population size for the algorithm has been assumed to be 30. Initially the tasks were assigned to processors based on EDF. After the results have been observed, the tasks were scheduled using the proposed hybrid genetic algorithm. This works repeat for SCFT.

4. Experimental Results

With attention to Figure 4 and doing the following scheduling for processors, comparing node duplication to FCFS shows that the total finish time is 58 time units.

$P1: t1 \rightarrow t4 \rightarrow t7 \rightarrow t10 \quad P2: t2 \rightarrow t5 \rightarrow t8 \rightarrow t11 \quad P3: t3 \rightarrow t6 \rightarrow t9$

In list scheduling, performing below scheduling for processors, the total finish time is 48 time units.

$P1: t1 \rightarrow t2 \rightarrow t5 \rightarrow t6 \quad P2: t3 \rightarrow t7 \quad P3: t4 \rightarrow t9 \rightarrow t8 \rightarrow t11$

Optimal scheduling in genetic algorithm for tasks is shown below:

$P1: t1 \rightarrow t5 \rightarrow t4 \rightarrow t8 \rightarrow t9 \quad P2: t2 \rightarrow t6 \rightarrow t10 \quad P3: t3 \rightarrow t7 \rightarrow t11$

With analysis of the Figure 9 and with execution of the genetic algorithm, the total finish time is 45 time units.

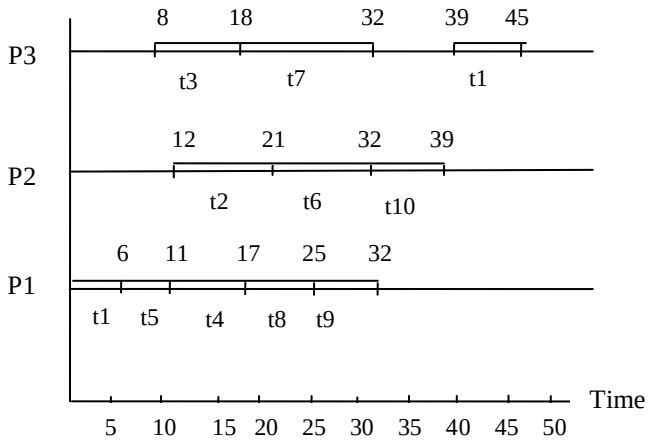

Figure 9. Execution Schedule of Tasks with Genetic algorithm

With analysis of Figure 10 and with implementation genetic algorithm with node duplication, the total finish time is 39 time units. Optimal scheduling in node duplication is depicted below:

$P1: t1 \rightarrow t5 \rightarrow t4 \rightarrow t8 \rightarrow t9 \quad P2: t1 \rightarrow t2 \rightarrow t6 \rightarrow t10 \quad P3: t1 \rightarrow t3 \rightarrow t7 \rightarrow t11$

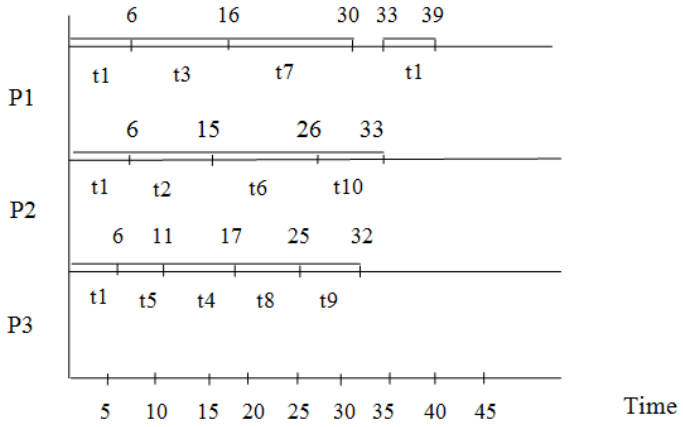

Figure 10. Schedule of Tasks with Node Duplication using Genetic Algorithm

Speedup, shown with $T_{SP,}$ is defined as the ratio of completion time on uniprocessor systems to completion time of multiprocessor system.

$$T_{SP}=\frac{\text{completion time on uniprocessor}}{\text{Completion time of multiprocessor}}$$

$$C=\frac{(Tsp*100)}{n}$$

Where, n is the number of processors. Speedup in FCFS is calculated using Equation 1:

$$T_{SP}= \frac{75}{58} =1.2931 \quad (1)$$

$$C= \frac{(1.2931*100)}{3} =41.033\%$$

Speedup in list scheduling is calculated as follows:

$$T_{SP}= \frac{75}{48} =1.5625 \quad (2)$$

$$C= \frac{(1.5625*100)}{3} =52.08\%$$

Speedup in genetic algorithm is calculated using relation 3:

$$T_{SP}= \frac{75}{45} =1.6667 \quad (3)$$

$$C= \frac{(1.6667*100)}{3} =55.5566\%$$

And the speedup in node duplication genetic algorithm is come from:

$$T_{SP}= \frac{75}{39} =1.923$$

$$C= \frac{(1.9230*100)}{3} =64.1\%$$

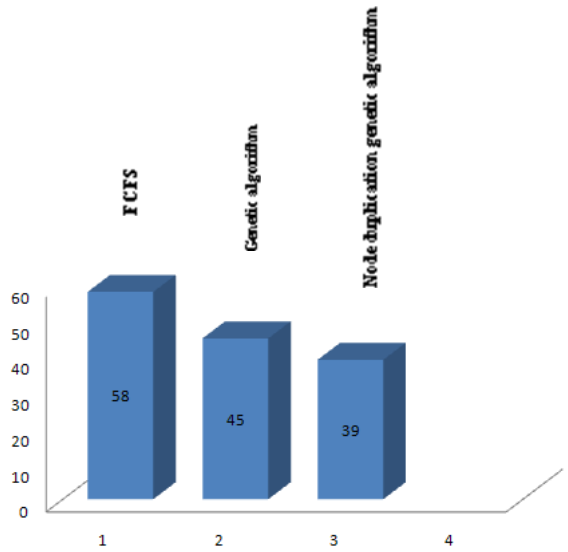

Figure 11. Comparison of Total Finish Time of FCFS, Genetic Algorithm and Node Duplication Genetic Algorithm

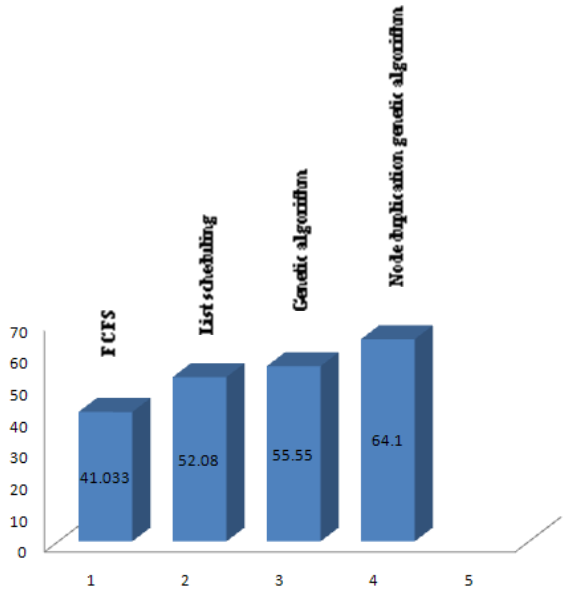

Figure 12. Performance Comparison of FCFS, Genetic Algorithm, List Scheduling and Node Duplication Genetic Algorithm

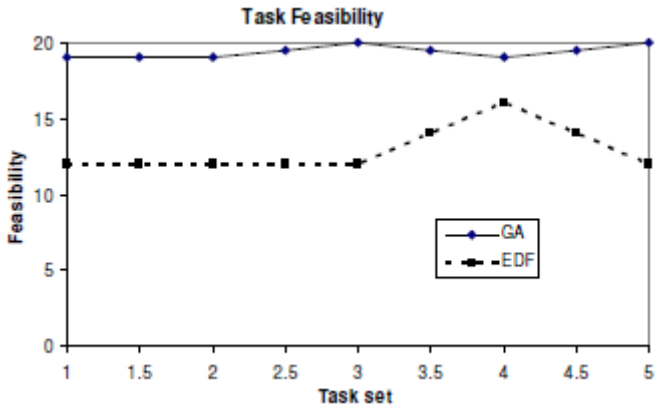

Figure 13. Task Feasibility of EDF and Genetic Algorithm [12]

In EDF 100 tasks were generated randomly and were divided into task sets of 20 for each. The tasks were ordered in the increasing order of their deadlines and assigned to processors considering earliest deadline first. The processors were chosen randomly between 1 and 10. For each task we calculate fitness value. Figure 13 shows feasible tasks in EDF merging with genetic algorithm. The percentage of tasks that are feasible is 95 percent. In SCTF merging with genetic algorithm, the chromosome size is 20 and the length tasks queue is 100. The tasks were sorted in the increasing order of computation time. In this case, the percentage of tasks that are feasible is 90 percent. Figure 12 shows EDF and SCTF with chromosome size 20 and 40.

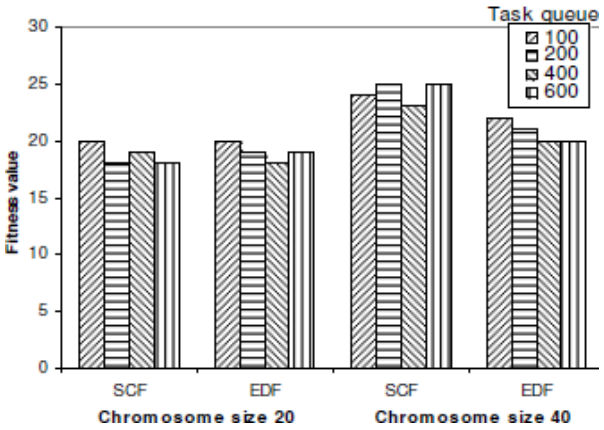

Figure 14. Comparative Fitness Values for Chromosome Size 20 and 40 [12]

Figure 14 shows that there are best results with chromosome size 20 and Table 3 exhibits that if length of queue is 100 then there are the best results.

Table 3. Fitness Values Obtained from Different Chromosome Sizes

Task queue	Fitness values			
	Chromosome size 20		Chromosome size 40	
	SCF	EDF	SCF	EDF
100	20	20	24	22
200	18	19	25	21
400	19	18	23	20
600	18	19	25	20

5. Conclusions

This paper has described a useful genetic algorithm with the node duplication for multiprocessor systems. The communication cost of the proposed algorithm is the minimum in comparison with three other approaches (GA, FCFS, and List Scheduler). Moreover, its throughput in multiprocessor systems is better than the others. Based on simulation results, our NGA algorithm is the best regarding final time but it has more computation load in the parallel systems. Importantly, in order to minimize the finish time the task should be replicated. Also, NGA performance is compared with GA, FCFS and List Scheduler (see Figure 11 and 12). NGA performance is 64.1%, GA performance is 55.55%; Performance of List Scheduling and FCFS are 52.08% and 41.033%; respectively.

In the future, we will work on reducing the calculation time which is a drawback of the proposed approach.

References

[1] K. Ramamrithm and J. A. Stankovic, "Scheduling algorithms and operating systems support for real-time systems", Proceedings of the IEEE, vol. 82, no. 1, **(1994)** January, pp. 55-67.
[2] J. Goossens and P. Richard, "Overview of real-time scheduling problems", Euro Workshop on Project Management and Scheduling, **(2004)**.
[3] A. Srinivasan and J. Anderson, "Fair scheduling of dynamic task systems on multiprocessor", The Journal of Systems, vol. 77, **(2005)**, pp. 67-80.
[4] A. Page and J. Naughton, "Dynamic task scheduling using genetic algorithms for heterogeneous distributed computing", The proceedings of the 19th International Parallel & Distributed Processing Symposium, Denver, USA IEEE Computer Society, **(2005)**.
[5] S. Baase and A. Gelder, "Computer Algorithm", Published by Addison Wesley, **(2000)**, pp. 557-562.
[6] S. Sahni, "Algorithm analysis and Design", Published by Galgotia Publications Pvt. Ltd, New Delhi, **(1996)**.
[7] M. Moghimi, M. Zali, S. Taheri and F. Taghiyareh, IEEE proceeding, **(2007)**, pp. 226-230.
[8] B. Andersson and J. Jonsson, "Fixed-priority preemptive multiprocessor scheduling: to partition or not to partition", Seventh International Conference on Real-Time Computing Systems and Applications (RTCA'00), **(2000)**.
[9] M. Lin and L. Yang, IEEE Proceeding, **(1999)**, pp. 287-382.
[10] E. Zafarani, A. Rahmani and M. Derakhi, IEEE Proceeding, **(2008)**, pp. 248-251.
[11] F. Sandnes and G. Megson, "An evolutionary approach to static task graph scheduling with task duplication for minimized interprocessor traffic", Taiwan, **(2001)** July, pp. 101-108.
[12] K. Dahal, A. Hossain, B. Varghese, A. Abraham, F. Xhafa and A. Daradoumis, "Scheduling in Multiprocessor System Using Genetic Algorithm", International Joint Conference on Computer.

Authors

Hadis Heidari

Hadis Heidari is currently an MSc student in Razi University of Kermanshah, Iran. She obtained her B.Sc. in Computer Science from Razi University, Kermanshah, Iran in 2011. She was the first rank of students in computer engineering in the Razi University. Her research interests are in the field of operating systems.

Abdollah Chalechale

Born in Kermanshah, Iran, received his BS and MSc degrees in Electrical Eng (Hardware) and Computer Eng. (Software) from Sharif University of Technology, Tehran, Iran, respectively. He received his PhD degree from Wollongong University, NSW, Australia in 2005 and currently is with Razi University, Kermanshah, Iran. His research interests include image processing, machine vision and human-machine interactions.