A Comparative Study on Dynamic Scheduling of Real-Time Tasks in Multiprocessor System using Genetic Algorithms

Sri Raj Pradhan
Department of CSE,
Sikkim Manipal Institute of Technology, Sikkim, India

Sital Sharma
Department of CSE,
Sikkim Manipal Institute of Technology, Sikkim, India

Debanjan Konar
Department of CSE,
Sikkim Manipal Institute of Technology, Sikkim, India

Kalpana Sharma
Department of CSE,
Sikkim Manipal Institute of Technology, Sikkim, India

ABSTRACT
Multiprocessors have evolved as powerful computing tools for executing dynamic real time tasks. The continual evolutions of the multiprocessor and real-time systems in the last few decades have encouraged the research and development of a new and efficient algorithm for dynamic scheduling of real-time task in multiprocessor systems. This paper proposes a compact study on dynamic real time task scheduling in multiprocessor environment using Genetic Algorithm (GA) which is a typically NP-complete problem. GA exploits the power of parallel computing which drives the solution towards optimal one. The GA, inspired by biological genetics and the process of natural selection, comprises fixed size chromosome and biological inspired genetic operators like mutation and crossover. This paper investigates the various scheduling algorithms and compares the simulation result in terms of fitness value and the percentage of success for scheduling real time tasks.

Keywords
Dynamic task scheduling, Genetic Algorithm, Multiprocessor system, NP complete problem, Real time system, Earlier Deadline First (EDF), Shortest Computation First (SCF).

1. INTRODUCTION
In today's computing era multiprocessor system have outperformed the uniprocessors systems for executing the real-time applications as uniprocessor system is not sufficient enough to execute all the tasks while also satisfying their timing constraints. In multiprocessor system there are several processors available wherein all the task could be executed without missing its deadline. Allocating the task in multiprocessor system is much more difficult as compared to the uniprocessor system as determining the optimal solution has an exponential complexity hence falls under the category of NP-hard problem. Thus scheduling algorithm for multiprocessor system should ensure the feasible schedule for the set of real-time task. A valid schedule is called a feasible schedule if the entire tasks in the set are executed without missing its deadline and no task are scheduled before its arrival [1].

1.1 Real time system
Real-time systems can be defined as the software systems in which the efficiency of the system relies more on the time at which the result is obtained rather than its logical correctness. If the system exceeds the specified time bound to produce the result then it may produce undesirable or even fatal result, thus system failure is said to have occurred. Real-time task are said to have arrived when some events are triggered and may occur at a large number of times at random instance which has some timing constraints. These timing constraints need to be considered when scheduling the real-time task [1]. These timing properties that real-time tasks constitute are as follows:

1.1.1 Arrival time (A_i)
The time at which the real-time task is generated due to the occurrence of some specific event.

1.1.2 Ready time (R_i)
The time at which the task is allocated to the processor for execution.

1.1.3 Computational Time (C_i)
The Maximum time taken for the execution of the task after being released.

1.1.4 Deadline (D_i)
The Time by which the execution of the real-time task needs to be completed. Real-time systems constitutes of multiple tasks with different levels of criticality and it is not always necessary that all the real-time task belongs to the same category thus, real-time tasks are further divided into hard, soft and firm real-time task relying on consequences of task missing its deadline. Let us assume that a real-time system consists of a set of tasks $T = \{T_1, T_2, T_{3,\ldots\ldots,}T_n\}$, where the computation time of the task ,$T_i \in T$ is $C_{i..}$ The system is said to be real-time if there exist at least one task $T_i \in T$ which falls into one of the following category.

1.1.5 Hard Real-Time Task
In hard real-time task meeting the deadline is obligatory otherwise results become undesirable, i.e. $C_i \leq D_i$

1.1.6 Soft Real-Time task
Soft real-time task also have timing constraints associated with it however, missing some is tolerable but a penalty is associated with it if computation time C_i exceeds the given deadline D_i. The penalty function $P(T_i)$ can be defined as:

If $C_i \le D_i$ then $P(Ti) = 0$, otherwise $P(T_i) > 0$.

1.1.7 Firm Real-Time Task
A firm real-time task also associates a deadline with it but missing may not adversely affect the system, the late result are merely discarded but if the task finishes its computation earlier than its deadline than it gains a reward. The reward function R (T_i) can be defined as:

If $C_i \ge D_i$, then $R(T_i) = 0$, otherwise $R(T_i) > 0$.

Real-time task are generated on the occurrence of some events and may reoccur over a period of time, based on this they are further classified into three main categories: periodic, sporadic and aperiodic task.

1.1.8 Periodic Task
The instance of a real-time task which repeats after a fixed time interval determined by clock interrupts is known as periodic task.

1.1.9 Sporadic Task
The instance of a real-time task whose reoccurrence cannot be predicted as it occurs at random time periods is known as Sporadic Task.

1.1.10 Aperiodic Task
The real-time task that arise at random instance similar to a sporadic but aperiodic task are generally soft real-time task whose two or more instance of may occur at the same time instant and therefore it may be very hard to meet their deadlines for the aperiodic tasks.

1.2 Scheduling
Scheduling is a process of assigning task to a processor for execution. The main objective of scheduling is to assign processor to a task while generating an efficient makespan (Total time to complete the task list) and valid schedules. The task scheduling can be further classified into static and dynamic scheduling.

1.2.1 Static Scheduling
The algorithm is also known as offline deterministic scheduler, the execution times of the task and data dependencies between them are known a prior. These types of scheduling are done during compile time therefore the run time overhead of these schedulers are very low [2]. The main disadvantage of these schedulers is they cannot acceptably handle aperiodic and sporadic task since the exact time of occurrence cannot be predicted. [4]

1.2.2 Dynamic Scheduling
The execution times of the task and data dependencies between them are not known a prior and thus scheduling decisions and processors are allocated on run time. Dynamic scheduling thereby furnishes a faster and flexible system [2].
The remainder section of the paper consist of Related works in section 2, multiprocessor scheduling algorithm in section 3, Genetic Algorithm in section 4 and conclusion on section 5.

2. RELATED WORKS
Various heuristic approaches have been precisely studied for overcoming the problem of scheduling the time critical real time application on multiprocessor platform which includes Earliest Deadline First (EDF)[1], Shortest Computational Time First (SCTF)[1] . The use of genetic algorithm (GA) for real-time task scheduling has also been studied and GA was very successful compared to the earlier approaches. Genetic operators such as crossover and mutation have been the key features of genetic algorithm. Bohler, M. et al.[3] proposed a genetic algorithm for scheduling task in multiprocessor system. This algorithm generated a schedule for 20, 40, 50 task taking crossover probability 1.0 and mutation probability as 0.1 and 0.2. For 20 tasks they generated the solution in 20 generations and were within 1.2% of the optimal solution. For larger problems of 40 tasks they required 100 generations to generate a solution and for 50 tasks it required 500 generations. Heidari, H. and Chalechale, A.[4] implemented GA and have further improvised GA to obtain a node Duplication Genetic Algorithm (NGA) and have produced better results with less execution time than GA. Roy, P. et. al.[5] implemented a heuristic based task scheduling algorithm for multiprocessor system to obtain an efficient way of selecting processors in order to have less execution time. They have compared Elitism algorithm and their proposed scheduling algorithm and have shown that their algorithm take much less time to reach the termination condition than Elitism algorithm except on task that were scheduled using 16 processors . Dahal, K.[6] et al. obtained feasible solution using genetic algorithm combined with well-known heuristics as Earliest Deadline First (EDF) and Shortest Computation Time First (SCF) obtaining high processor utilization. They were able to schedule 90% or more of the task taking into consideration task queue of 100,200,400 and 600. Mostafa R. M, Medhat H, A.[7] have proposed two algorithms namely Modified List Scheduling Heuristic (MLSH) and hybrid approach composed of genetic algorithm and compared them to list scheduling heuristic algorithm and bipartite GA, surpassing them in both execution time and processor utilization also comparing the types of chromosome formats and their impact on the algorithm. They concluded that chromosome format containing both task list as well as processor list generated better results. Dhingra, S. et al [8] proposed a genetic algorithm for multiprocessor system and were able to minimize the make span and total completion time. Sachi Gupta et al implemented multiprocessor scheduling algorithm using genetic algorithm and provided with better solution than Heterogeneous –Earliest Finish Time. It also stated that genetic algorithms are robust since the average schedule continuously decreases as more generations are evolved. Thus it reveals that as generations passes the quality of solution increases [9]. Shu-Chen Cheng et al. [10] suggested a dynamic real-time scheduling for multi-processor tasks using genetic algorithm combined with a feasible energy function which drives the solution towards better quality schedules.

3. MULTIPROCESSOR SCHEDULING ALGORITHM
The main goal of real-time scheduling are completing the task within a specified time constraint and preventing them from simultaneously accessing the shared resources and devices. Multiprocessor platforms contain several processors upon which task can be scheduled. Some assumptions that can be taken into consideration while designing multiprocessor scheduling algorithms are:

3.1 Preemptive task

A Task is allowed to be removed (preempted) from the processor prior to its completion and can later be resumed if a task with higher priority arrives for execution [7].

3.2 Task migration

A Task preempted from a processor may execute on a different processor [7].
Multiprocessor Scheduling falls under two general categories:

3.3 Global scheduling algorithms

A queue is shared among all processors which store task that have arrived but not yet completed their execution. If there exist m processors, than the highest n priority task are selected for assigning to n processors [7].

3.4 Partitioning scheduling algorithms

A set of Task are partitioned such that all the task of a partition are assigned to the same processor. This schedule is similar to having many uniprocessor scheduling problem since task are not allowed to migrate [7].

Static algorithms are employed to schedule periodic tasks whose ready time are known a prior whereas dynamic scheduling algorithms are used to schedule sporadic and the aperiodic tasks whose characteristics could not be known prior to its arrival. Dynamic scheduling can either be centralized or distributed. Each processor present in a distributed architecture has its own local scheduler that decides whether the requirement of the incoming tasks could be met or not. If the scheduler fails to satisfy the requirement then it tries to find an acceptable processor for the task so it can be completed within the required condition. In the centralized scheme there is the central processor called the scheduler whose job is to determine the acceptable processor for execution. The two main intents of task scheduling in real-time systems should be fulfilling all timing constraints and gaining high resource utilization.

Some centralized algorithm for static tasks such as: Utilization Balancing algorithm, where task are maintained in a queue in an increasing order of their utilization and removed one by one from the head of the queue and allocated to the least utilized processor each time. This algorithm is appropriate when the number of processors is fixed and the tasks at individual processors are scheduled using Earliest Deadline First (EDF) [1]. Next Fit Algorithm attempts to use as less processors as possible and can be implemented on any arbitrary number of processors. The tasks are divided into a particular number of classes. If the task are divided into m classes, task Ti belongs to class j, $0 \leq j < m$ assigned according to: $(2^{1/j+1-1} -1) < e_i/p_i \leq (2^{1/j-1}-1)$. The main methodology behind this algorithm is to assign task with similar utilization value to the same class of task [1].

Some Decentralized algorithm for dynamic task such as: Focussed Addressing and Bidding, have two tables called status table and system load table. These two tables store information's regarding the task and the load of the processors respectively. The task on arrival is scheduled locally if it is free else offloaded to another processor (focussed processor) selected on the basis of the table. The table are periodically updated over a fixed interval of time. However this algorithm incurs a high communication overhead since it needs to maintain the system load table at each processor [1]. Buddy Algorithm is similar to Focussed Addressing and Bidding algorithm but differ in the manner it selects the processor. Only processors with utilization values less than a threshold values are used for selection and the table information are only distributed among a set of processors called the buddy list [1].

4. GENETIC ALGORITHM (GA)

Genetic algorithms are search methods that implement methodologies based on biological evolution [7]. The method was introduced by John Holland in 1970. Genetic Programming Inc. company used parallel computer with 1000 node for implementation of genetic algorithm in 1999 [4]. To find potential solutions that approach a specific criterion this algorithm search or operate on a given set of potential solutions. To do this, the algorithm applies the principle of survival of the fittest in order to a find a better approximation [10]. At each generation, by the process of selecting potential solutions on the basis of their level of fitness and breeding them together with operators borrowed from natural genetics a new set of approximations is created. Due to these processes just as in natural adaption a new set of individuals are generated that are better suited for their environment.

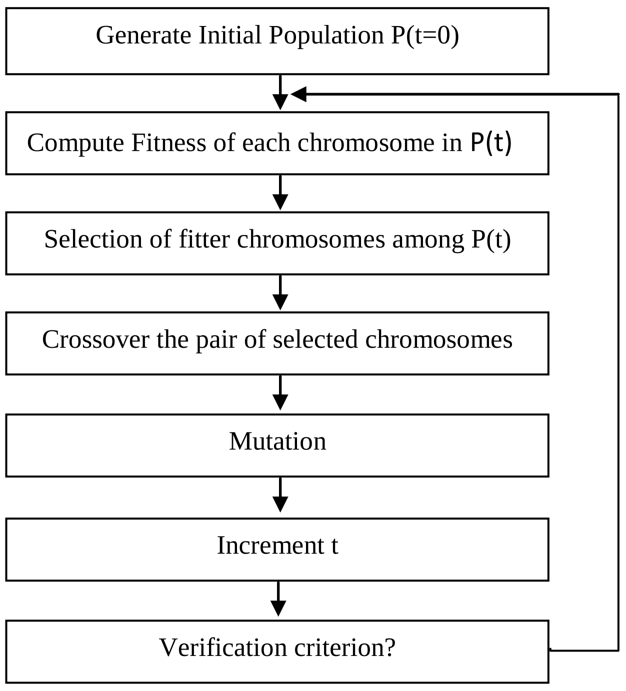

Generate Initial Population P(t=0)
Compute Fitness of each chromosome in P(t)
Selection of fitter chromosomes among P(t)
Crossover the pair of selected chromosomes
Mutation
Increment t
Verification criterion?

Fig 1: Genetic Algorithm

The following subsections explain the basic concepts of GA.

4.1 Selection

This operator selects individuals from a set of population for reproduction. There are many selection methods. Some are described below:

4.1.1 Roulette Wheel Selection

The parents are selected according to their fitness. Chromosomes that have higher fitness values have more chance to be selected. Imagine a roulette wheel where the chromosomes are placed and the size of the place occupied in the wheel is considered according to their fitness shown in Fig2.

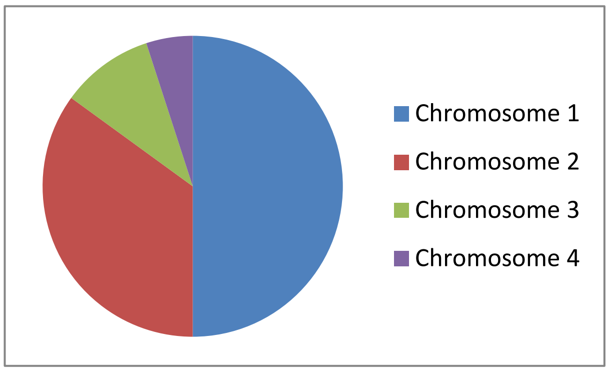


Fig 2: Roulette wheel selection

Then a marble is thrown and the position where the marble stops that chromosome is selected. Therefore, we can see that chromosomes with higher fitness values are likely to be submitted.

The above selection can be simulated using the following Pseudo Code:

```
 Chromosome  =  RouletteWheel(Population)

       Generate Cumulative Sum of all the
fitness value of Population.

       Generate random number between
interval [0,S] – r

`      Select Chromosome whose cumulative
sum is greater than the random number r.

end
```

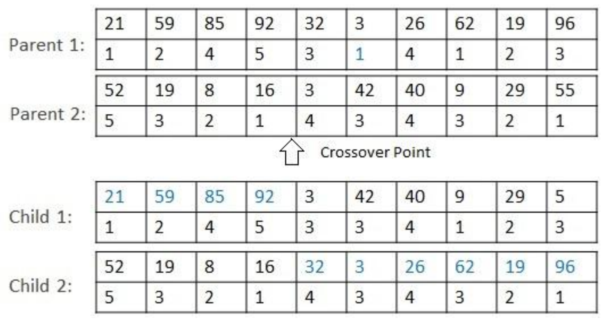
4.2 Crossover

This operator randomly selects a point on an individual string and exchanges the subsequence before and after that point to create two new offspring.

There are basically two types of crossover:

4.2.1 *Single - Point Crossover*

Fig 3. Implements single point crossover. This type of crossover includes only one crossover point. The entire chromosome from the crossover point to the end of the chromosome is replaced by another string selected from another chromosome taken from the same positions.



Parent 1:	21	59	85	92	32	3	26	62	19	96
	1	2	4	5	3	1	4	1	2	3
Parent 2:	52	19	8	16	3	42	40	9	29	55
	5	3	2	1	4	3	4	3	2	1

Crossover Point

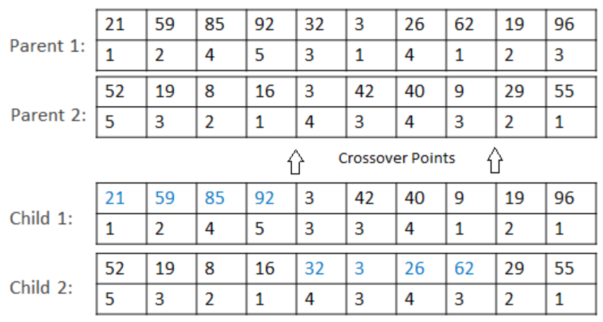
Child 1:	21	59	85	92	3	42	40	9	29	5
	1	2	4	5	3	3	4	1	2	3
Child 2:	52	19	8	16	32	3	26	62	19	96
	5	3	2	1	4	3	4	3	2	1

Fig 3: Single point crossover

4.2.1 *Double-Point Crossover*

Fig 4. Implements double point crossover. This type of crossover includes two crossover point. The entire chromosome from the first crossover point to the second of the chromosome is replaced by another string selected from another chromosome taken form the same position

.



Parent 1:	21	59	85	92	32	3	26	62	19	96
	1	2	4	5	3	1	4	1	2	3
Parent 2:	52	19	8	16	3	42	40	9	29	55
	5	3	2	1	4	3	4	3	2	1

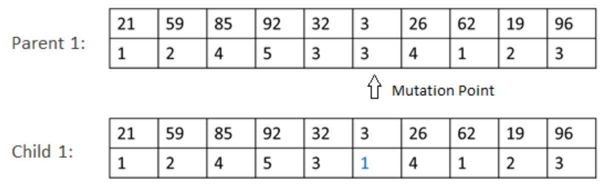
Crossover Points

Child 1:	21	59	85	92	3	42	40	9	19	96
	1	2	4	5	3	3	4	1	2	1
Child 2:	52	19	8	16	32	3	26	62	29	55
	5	3	2	1	4	3	4	3	2	1

Fig 4: Double point crossover

4.3 Mutation

This operator randomly selects a point and flips the data in the individual. Fig 5. Implemnts a mutation where a the value of the mutation point is changed.



Parent 1:	21	59	85	92	32	3	26	62	19	96
	1	2	4	5	3	3	4	1	2	3

Mutation Point

Child 1:	21	59	85	92	32	3	26	62	19	96
	1	2	4	5	3	1	4	1	2	3

Fig 5: Mutation

These operators are used to modify the chosen solution in order to select a most appropriate offspring to pass to the succeeding generation. GA if used on problems that are too large may have excessive complexity. GA allows parallel processing to be performed when finding solutions to larger and complex problems since they work on population of individuals rather than a single solution. GA is an iterative process consisting of individuals that are encoded in the population string known as chromosomes, encoding a possible solution in a given problem space. This space, denoted as search space consists of all possible solutions to the problem. At every generation the individuals are decoded and fitness of an individual in the population is calculated according to the fitness function set for the problem. The goal of the fitness function is to find the shortest possible schedule. Crossover operator fuses the information within pairs of selected individual to generate new individuals. To prevent premature convergence the mutation operator is used and mutation rate is taken very small (typically between 0.1 and 1) [11].

5. RESULT AND DISCUSSIONS

Various researchers have implemented a scheduling algorithm incorporating GA and produced better results than heuristic algorithms. Fig 6. Shows the task feasibility implementing their scheduling algorithm. The percentage of task scheduled was 90% and above [6].

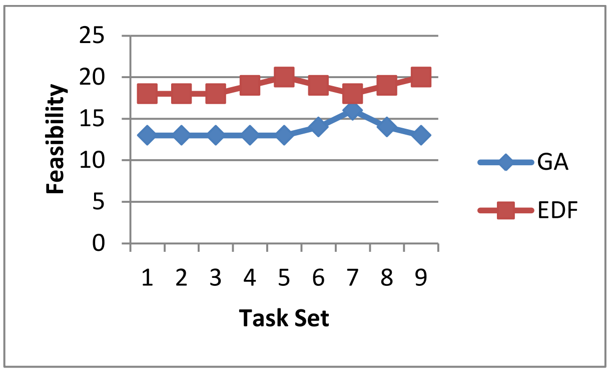


Fig 6: Task feasibility of EDF and GA

Fig 7. Shows the performance comparision of Genetic Algorithm and Node Duplication Genetic Algorithm done by Hadis Heidari and Adolah Chalchale [4]. The graph clearly shows that their improvisation of genetic algorithm for scheduling in multiprocessor system have produced better results than conventional GA.

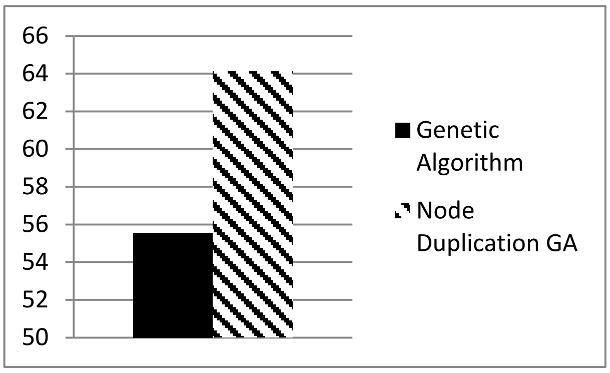


Fig 7. Performance comparision of Genetic Algorithm and Node Duplication Genetic Algorithm

Fig. 8 shows the performance analysis of Task List Processor List Chromosome(TLPLC) and Bipartite Genetic Algorithm(GA) [7]. The problems was done for 15 task. The graph clearly shows that the improbvised algorithm produced better results than the BGA.

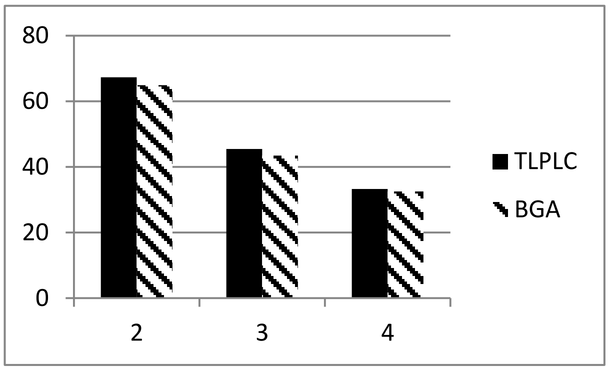


Fig 8: Performance analysis of TLPLC and BGA

Fig 9. Shows the success ratio obtained by Gheni Ahmed Ali for chromosome size = 5 in a system with 5 processor implementing single point and double point crossover on their GA based scheduling algorithm [11]. The graph clearly shows that using double point crossover better results were obtained.

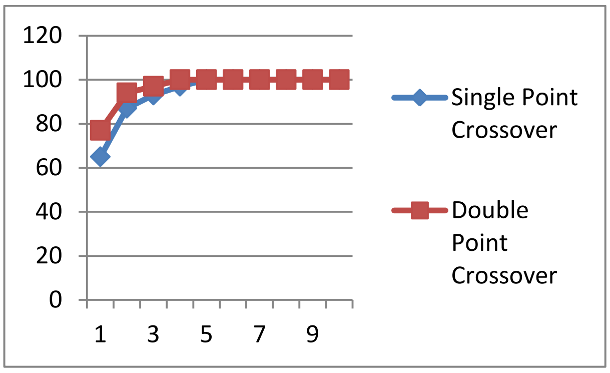


Fig 9: Success ratio for chromosome size = 5 in system with 5 processor

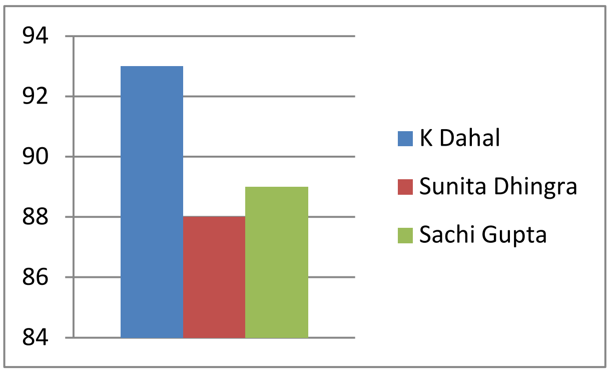


Fig 10: Success rate of algorithms applying GA

Fig. 10 comprises of the success rate achieved by the algorithms for 100 task. Here we can clearly see that GA combined with the heuristic 'EDF' produced better results than GA applied by Sachi Gupta and Sunita Dhingra.

6. CONCLUSION

This paper discusses some of the scheduling algorithm used to solve multiprocessor scheduling problems. Generally traditional algorithms such as Utilization Balancing Algorithm tried to decrease the make span by considering only low utilized processors for scheduling with the help of EDF. Next fit Algorithm created an environment of having many uniprocessor systems. The make span was further decreased with the help of genetic algorithms. Many algorithms incorporated genetic algorithms to generate an optimal solution. Studies showed that genetic algorithm converge to an optimal solution very slow thus combining it with other heuristics often improves the result. Many improvements have been made since the introduction of evolutionary algorithm and further improvements are also expected to be made. Genetic algorithms combined with heuristics produce better results and thus further improvements may be possible on how cleverly the algorithms are used with other heuristics to produce better results. However, the classical genetic algorithm still suffers from the lack of generalization in generating valid schedules due its crossover and mutation operators. In addition to this, the time consuming genetic operations used in most of these approaches for updating the schedules fails to meet the deadline constraint of the real time tasks. The problem of multiprocessor real-time scheduling can be addressed much efficiently by employing the quantum inspired genetic algorithm with some well-known existing heuristics. The research scholars are currently engaged in this direction.

7. REFERENCES

[1] Mall, R. 2007 Real- Time Systems . Pearson Education

[2] George, D.I., Amalarethinam, A., Josphin, M. 2015. Dynamic Task Scheduling Methods in Heterogeneous Systems- A Survey. International Journal of Computer Applications (0975 – 8887) Volume 110 – No. 6.

[3] Bohler, M., Moore, F., Pan, Y. 1999. Improved Multiprocessor Task Scheduling using Genetic Algorithms. Proceeding of the Twelfth international FLAIRS Conference.

[4] Heidari, H., Chalechale, A. 2012. Scheduling in Multiprocessor System using Genetic Algorithm. International Journal of Advanced Science and Technology ,Vol.43.

[5] Roy, P., Alam, U.M., and Das, N. 2012. Heuristic based Task Scheduling in Multiprocessor Systems with Genetic Algorithm by choosing the eligible processor. International Journal of Distributed and Parallel Systems (IJDPS) Vol.3, No.4.

[6] Dahal, K., Hossain, A., Varghese, B.,Abraham, A., Xhafa, F., Daradoumis, A. 2008. "Scheduling in Multiprocessor System Using Genetic Algorithms. Proc.IEEE Computer Information System and Industrial Management Applications, 7, pp.281-286

[7] Mostafa, R.M., Medhat, H., Awadalla, A. 2011. "Hybrid Algorithm for Multiprocessor Task Scheduling". IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 2, May ISSN (Online): 1694-0814

[8] Dhingra, S., Gupta, S.B., Biswas, R. 2014. Genetic Algorithm Parameters Optimization for Bi-Criteria Multiprocessor Task Scheduling Using Design of Experiments. World Academy of Science, Engineering and Technology. International Journal of Computer,Control,Quantum and Information Engineering Vol: 8, No: 4,

[9] Gupta, S., Agarwal, G., Kumar, V. 2013. An Efficient and Robust Genetic Algorithm for Multiprocessor Task Scheduling, International Journal of Computer Theory and Engineering. Vol: 5, No: 2.

[10] Cheng, S.C., Huang, Y.M. 2004. Dynamic real-time scheduling for multi-processor tasks using genetic algorithm.Computer Software and Applications Conference, COMPSAC ,pp 154-161.

[11] Ali, G.A. 2008. Dynamic Task Scheduling in Multiprocessor Real Time Systems Using Genetic Algorithms. Iraq Academic Scientific Journal (IASJ), ISSN: 16816870 Issue**:** 23 Pages**:** 46-65

[12] Laboudi, Z., Chikhi, S. 2012. Comparison of Genetic Algorithm and Quantum Genetic Algorithm. The International Arab Journal of Information Technology, Vol. 9, No. 3, May