Aggregation of EDF and ACO for Enhancing Real Time System Performance

Jashweeni Nandanwar
Department of Computer Science & Engineering
G.H.Raisoni College of Engineering, Nagpur, India

Urmila Shrawankar
Department of Computer Science & Engineering
G.H.Raisoni College of Engineering, Nagpur, India

ABSTRACT

Time constraint is the main factor in real time operating system and it affects the deadline of the process. To achieve deadline, proper scheduling algorithm is required to schedule the task. In this paper an Adaptive scheduling algorithm is developed which is the combination of Earliest Deadline First (EDF) and Ant Colony Optimization (ACO). The EDF algorithm places the process in a priority queue and executed using the deadline. The priority of the processes depends upon the deadline and handles the under loaded condition. The limitation of EDF algorithm is that it cannot handle the overloaded condition. The execution of ACO algorithm is based on the execution time. Process which contains the minimum execution time is executed first. The limitation of ACO algorithm is, it takes more time for execution than EDF. Therefore, to remove the limitation of both the algorithms an Adaptive scheduling algorithm is developed. It increases performance of the system and decreases the system failure. Also the percentage of missing deadline is reduced. The advantage of an Adaptive scheduling algorithm is, it handles over-loaded and under-loaded condition simultaneously.

The performance of an Adaptive scheduling algorithm is calculated in terms of Success Ratio that is the number of process scheduled and CPU Utilization. The result of execution time is compared with the EDF and ACO scheduling algorithm. The goal of an Adaptive scheduling algorithm is to show the switching between the scheduling algorithms and to decrease the system failure and increase the system performance.

Keywords

Real-Time Scheduling algorithm, Earliest Deadline First, Ant Colony Optimization, Load balancing, Adaptive Scheduling Algorithm.

1. INTRODUCTION

Real Time system completes its work and services on time so that the task or the process in system must complete within a specified time frame otherwise the failure will occur [1]. Consider the ATM network as an example of real time system, when the system becomes overloaded some transaction fails [2]. The time is measured using some internal measure of time such as clock ticks or instruction cycles [3]. The real-time systems maintain logical correctness it means if the users give the certain input to the system it will react to the input and generates the correct output. In addition, real-time systems must maintain temporal correctness – the output must be generated within the designated time frame. If a real-time system does not complete task in a definite time, it may cause the system which run the task.

Scheduling is the method used to allocate the resources between the various processes by the system and specify which job will receive next service by the resources. Scheduling algorithm is used to handle the real time system. Each of the algorithms provides best effort to increase the success ratio and decrease the CPU utilization as the performance of an Adaptive algorithm is measured using these two terms. EDF is most widely used algorithm for handling real-time traffic. It is the dynamic scheduling algorithm which places the task in a priority queue. Whenever the scheduler schedules the task it will search for the task closest to its deadline. The priority of the task is assign at the run time depending upon the deadline [4].

In overload situations, the performance of the system decreases when scheduler schedules the process using EDF algorithm. Researchers have proposed several methods to solve this problem. Previously proposed algorithms switched from EDF to another static algorithm after failure of more than two processes successively [5]. This will decrease the CPU utilization as well as the success ratio of the system. Recently some researcher proposed algorithm by combining both the static and dynamic algorithm together to make the recovery process faster [6].

An Adaptive algorithm removes the possibility of failure by combining EDF and ACO based scheduling algorithm. ACO algorithm is a branch of Swarm Intelligence. This algorithm handles the overloaded condition but the time required to execute the process is much more than the EDF algorithm so to remove this limitation, switching is performed between the EDF and ACO algorithm and the new algorithm is developed which is an Adaptive scheduling algorithm. In the rest of the paper the scheduling technique will be described as well as its advantages over other algorithm is described.

This paper makes the following contribution. Firstly it creates the process and executes using EDF algorithm. Secondly if the load of the processor is increased some process switched to ACO algorithm. Thirdly the algorithm calculates the Success Ratio, CPU Utilization and compares the execution time required by EDF and ACO algorithm with an Adaptive algorithm.

The paper is organized as follows. Section 2 provides the overview of scheduling techniques that are used for handling the under-loaded and over-loaded conditions. Section 3 discusses about an Adaptive scheduling algorithm which combine EDF and ACO algorithm. The designing steps of an Adaptive technique are discussed in section 4. Section 5 gives the working of an Adaptive scheduling algorithm. Section 6 gives the comparison of an Adaptive algorithm with the EDF and ACO algorithm and finally section 7 concludes the paper.

2. OVERVIEW OF SCHEDULING TECHNIQUES

Real time system always deals with the temporal parameters of the system. To handle the time factor it consists of hard and soft time constraint. Hard real-time system requires that deadlines must be met otherwise fatal error occurs such as traffic signals but in soft real-time system, missing an occasional deadline is undesirable, but nevertheless tolerable such as railway reservation [7]. Static and Priority driven are the techniques used by the scheduling algorithm to handle the under-loaded condition. Priority driven is classified as fixed priority and dynamic priority these techniques are used for managing the process in real time environment.

System use the scheduling techniques to schedule the process so rate monolithic and deadline monolithic scheduling algorithm is used in which the priority of the task is constant all the time user can not change the priority. Dynamic Priority consists of EDF and LST algorithms, the priority of the task may changed during its execution which give good result when the job is preemptive and the processor is not overloaded. But the system performance decreases when system is slightly overloaded. The priority of the process change dynamically in EDF algorithm and an Adaptive algorithm combined EDF and ACO algorithm.

3. AN ADAPTIVE SCHEDULING ALGORITHM

An Adaptive scheduling algorithm is the combination of EDF and ACO algorithm [8]. The limitation of the EDF algorithm is that it cannot handle the overloaded condition. When the overload condition occurs, EDF algorithm fails to execute. The ACO algorithm handle the overload condition but the execution time required is more than the EDF algorithm. This is the limitation of ant colony optimization, so to overcome the limitation of both the algorithm the new algorithm is developed which is called as an Adaptive scheduling algorithm and is described in the next section. Every scheduling algorithm gives a schedule for a set of task consider job J_{ij} which starts its execution after its ready time r_{ij} and complete before its deadline d such a schedule is called as a feasibly schedule, and the schedule produced is said to be feasible [9].

3.1 Earliest Deadline First

Liu and Layland [10] described the Rate- Monotonic (RM) and the Earliest Deadline First (EDF) scheduling policies. EDF algorithm is the dynamic scheduling algorithm which depends upon the deadline of the task. The task with the nearest deadline gives the highest priority [11].

EDF algorithm schedules the process and to create that process some parameter is required that is Start time, Execution Time, Deadline of the process, Release time and the Load of each process. After creating the process the process are store in a queue and the priority of that process depend upon the deadline [10]. "Figure 1" shows the working of EDF algorithm.

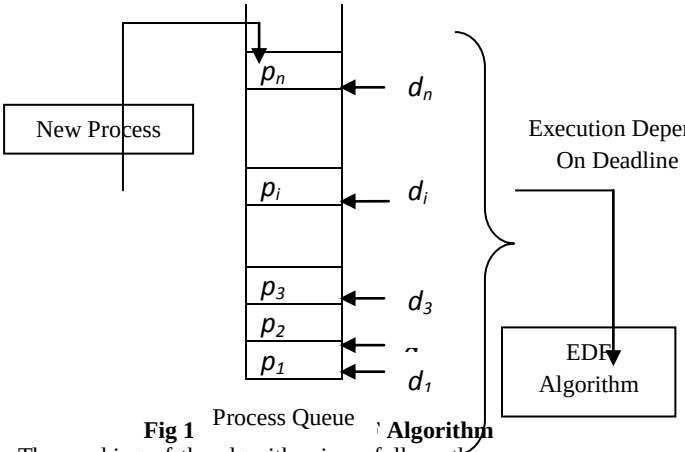

Fig 1 **Algorithm**

The working of the algorithm is as follow: the processes are added one by one in a process queue or EDF queue. Let $P = (p_1, p_2, p_3 \ldots\ldots\ldots p_n)$ denote a set of process $P_i = (r_i, e_i, d_i)$ is characterized by its release time r_i, execution time e_i and deadline of the process d_i. The process is added in the system and the process with highest priority is switched to EDF algorithm and the switching depends upon deadline of the process.

The periodic task $\tau_i = (c_i, p_i)$ is characterized by two parameters: an execution time c_i and a period p_i. The utilization of periodic task τ_i is defined as [12]

$$u_i = \frac{c_i}{p_i} \qquad (1)$$

A task can be feasibly scheduled using EDF algorithm if the total utilization of a task set τ is

$$U(\tau) = \sum_{i=1}^{n} u_i \qquad (2)$$

When the process is underloaded, the EDF algorithm executes all the process and CPU usage for that process is also minimum but when the process is overloaded some process fail to execute. This is the drawback of EDF algorithm. To overcome this drawback an Adaptive algorithm combines the ant colony optimization algorithm.

3.2 Ant Colony Optimization

ACO was proposed by Dorigo and Gambardella in the early 1990s and by now has been successfully applied to various combinatorial optimization problems [13], [14]. This algorithm is based on the behavior of real ant, each ant constructs a path and one or more ants concurrently active at the same time.

In ACO algorithm, each ant is called as a node and each of them will start their journey from different node. To apply scheduling in ACO each node is considered as a task and probability of each node depend upon the pheromone value τ and heuristic value η and it is calculated as [15]

$$p_i(t) = \frac{(\tau_i(t))^{\alpha} * (\eta\eta_i(t))^{\beta}}{\sum_{l \in R_1}(\tau_l(t))^{\alpha} * (\eta\eta_l(t))^{\beta}} \qquad (3)$$

Where,

$p_i(t)$ is the probability of i^{th} node at time t.

τ_i is pheromone on i^{th} node at time t.

η_i is a heuristic value of i^{th} node at time t and is determined by,

$$\eta_i = \frac{k}{D_i - t}$$

Here, t is current time k is constant and D_i is absolute deadline of i^{th} node.

α and β are constants which decide importance of τ and η .

The ACO algorithm is applied to the process with the same parameter as the EDF algorithm. The process is switched from EDF to ACO when the process is loaded and the process count is more. After switching to ACO, process executed depends upon the execution time the process with less execution time is executed first. The advantages of ant colony optimization is it indirectly communication between ants using pheromone variables. It also handles the overloaded condition. The behavior of the ant is successfully applied to several optimization problems [16].

4. DESIGNING STEPS OF AN ADAPTIVE ALGORITHM

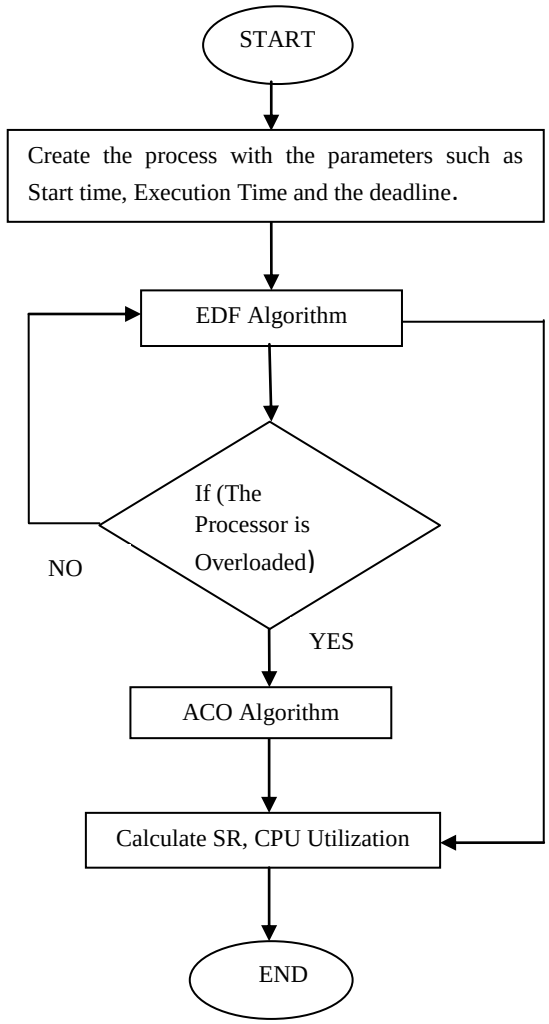

Fig 2: Flowchart of an Adaptive Scheduling Algorithm

1. Create the process with parameters such as Start time, Execution Time, Deadline, Release Time and Load of process.
2. Switch the process from the system to the EDF algorithm.
3. The EDF algorithm executes depending on the deadline of the process. The process with the nearest deadline is executed with the highest priority.
4. For switching to ACO algorithm, it checks the three conditions such as the Process Count, Priority of the process and the Load of each process. If the load of the process is more than the given CPU load, then the process switches from EDF to ACO algorithm.
5. In ACO it will calculate the total time required for all the process and the process which contains minimum execution time is executed first. In short execution depends upon the execution time.
6. Performance of an Adaptive scheduling algorithm is measured in terms of the Success Ratio, CPU Utilization and it also calculates the total number of process which are executed using EDF and the ACO algorithm with the process missed the deadline.
7. Finally the result is calculated in terms of graph and tables.

"Figure 2" shows the working of an Adaptive scheduling algorithm.

5. WORKING OF AN ADAPTIVE ALGORITHM

Scheduling algorithm is necessary when the new task arrives or the running task completes. An Adaptive algorithm improves the load balancing technique [17]. An Adaptive algorithm works as follows:

- During underloaded condition, the algorithm uses EDF algorithm and priority of the job is decided dynamically depending on its deadline [18].
- During overloaded condition, the algorithm switch to ACO algorithm calculates the total execution time and minimum time required for each process and the process which contain the minimum execution time is executed first [19, 20]. Following are the steps to create and schedule the process:

5.1 Total number of process running on system

Process is the instance of a program running on a Computer. Each process gives the complete description of the process with Process ID (PID), Total CPU usage for each process, priority of the process, thread required to run each process and the total CPU time. Priority is the main parameter for each process user can change the priority from idle to normal, normal to high and from high to low.

5.2 Total number of tasks and the working of EDF algorithm

A task is like a process or thread in an operating system. If the user wants to add the process then he has to specify the basic parameter such as

i) Total number of process.

ii) Start time is automatically defined by the system when the process is entering into the system.

iii) Deadline of the process when the process completes its execution.

iv) Execution time of the process (time required to execute the process).

v) Release time of process when the process is ready for execution.

vi) Threshold of CPU loads for each process as well as the type of the algorithm to run the process (using EDF, ACO or Adaptive scheduling algorithm).

Depending upon the parameter process is created. "Table 1" shows the process creation. By using the above parameter one

by one process is created. Let the process P_1 is created $P_1 = (d_1, r_1, e_1)$

Where, d_1 is the deadline of process P_1

e_1 is the total time required for process P_1

r_1 is the release time of P_1 .

Process P_1 is created and the load of process P_1 is calculated as

$$Load\ (l) = \sum_{i=1}^{m} \frac{e_i}{q_i} \qquad (4)$$

Where,

e_i is the execution time required for each process.

q_i depends upon the value of period p_i and the deadline d_i

After creating the process user has to specify the algorithm type weather he wants to use EDF, ACO or an Adaptive algorithm. Suppose user select the EDF algorithm then the process entered into the EDF queue. In this way process is created "Table 2" show queue structure of EDF algorithm.

All the created processes are moved to the EDF queue. When the processes are entered into the queue then the system gives the specific Process ID (PID) to each process. If the user added the ten processes then the system allocate the process name such as P1, P2, P3……etc. When the process is created that time is the start time of that process. Suppose process P1 is created at 10:14, so the start time is 10:14:26. And for same process user give the deadline as 3 minute then the deadline for system is 10:17:26 then depending upon the release time the status of the process is changed from waiting to running that is the process is releases for execution. In "Table 2" process P1, P3, P5, P6, P10 is of minimum release time so the status of this process changed to running and the priority is normal but when the process is nearest to deadline then the process is executed using EDF algorithm. This algorithm checks the load of each process. If the calculated load is greater than the CPU load then the process fails to execute. "Table 3" shows the execution of process using EDF algorithm.

Table 1. Process creation

Process Count	Start Time	Deadline in Min	Execution Time in Min	Release time in Sec	CPU load
1	10:14:26	10:17:26	2	1	≥ 10 %

Table 2. EDF Queue

Process ID	Process Name	Start Time	Deadline	Execution Time	Release time in Sec	Status	Priority	CPU Load
1667	P1	10:14:26	10:17:26	3	1	Waiting	Normal	9.192341
1668	P2	10:15:10	10:16:60	1	2	Waiting	Normal	5.68743
1669	P3	10:16:57	10:17:19	1	1	Waiting	Normal	11.22071
1670	P4	10:17:44	10:20:55	2	2	Waiting	Normal	14.55657
1671	P5	10:18:23	10:22:45	4	1	Waiting	Normal	14.55657
1672	P6	10:19:56	10:23:34	3	1	Waiting	Normal	7.89763
1673	P7	10:20:34	10:23:12	2	4	Waiting	Normal	11.22071
1674	P8	10:21:22	10:22:23	1	3	Waiting	Normal	12.34563
1675	P9	10:22:46	10:26:33	2	2	Waiting	Normal	13.34562
1676	P10	10:23:06	10:24:29	1	1	Waiting	Normal	10.00071

Table 3. Execution of EDF algorithm

Process ID	Process Name	Start Time	Deadline	Execution Time	Release time in Sec	Status	Priority	CPU Load	Algorithm Used
1667	P1	10:14:26	10:17:26	3	1	Execution Start	High	9.192341	EDF
1668	P2	10:15:10	10:16:60	1	2	Running	Normal	5.68743	-
1669	P3	10:16:57	10:17:19	1	1	Execution Start	High	11.22071	EDF
1670	P4	10:17:44	10:20:55	2	2	Running	Normal	14.55657	-
1671	P5	10:18:23	10:22:45	4	1	Execution Start	High	14.55657	Failed To Execute
1672	P6	10:19:56	10:23:34	3	1	Execution Start	High	7.89763	EDF
1673	P7	10:20:34	10:23:12	2	4	Running	Normal	11.22071	-
1674	P8	10:21:22	10:22:23	1	3	Running	Normal	12.34563	-
1675	P9	10:22:46	10:26:33	2	2	Running	Normal	13.34562	-
1676	P10	10:23:06	10:24:29	1	1	Execution Start	High	10.00071	EDF

Process P1, P3, P5, P6, P10 are started to execute because all the process have the nearest deadline but the load of process P5 is greater than the CPU load so process P5 fails to execute and this is the limitation of EDF algorithm. The rest of the process is also executed depending on the load.

5.3 Working of ACO Algorithm

Secondly user wants to apply the ACO algorithm for the same process which is created. "Table 4" shows the queue structure of ACO algorithm 'n' number of process are added total 10 process are added.

After Adding the process queue structure is formed. Same procedure is follow for execution of ACO algorithm. In ACO algorithm, the algorithm checks the release time as well as the execution time of each process.

Consider the process P4 which contains the release time 1 and the minimum execution time than other processes then the process P4 is executed first and likewise the other processes are executed. If the other process contains the same release time i.e. 1 then the algorithm checks the execution time of individual process. "Table 5" shows the execution of ACO algorithm.

The remaining process such as P2, P7, P8, P9 and P10 are also released depends upon the release time. These processes also executed in the same way as the other processes. But out of these 5 processes some process is failed to execute because of the time this is the limitation of the ACO algorithm.

5.4 Process Execution Using an Adaptive Scheduling Algorithm

An Adaptive algorithm combines the EDF and ACO algorithm user has to specify the algorithm type which is applied for the created process. "Table 6" shows the queue structure of an Adaptive algorithm.

Now all the process are entered into an Adaptive queue with the status of each process is waiting and depend upon the release time the process status change to the running. Load of each process is calculated if the calculated load is greater than the given CPU load that is the threshold value then the process is switch to ACO algorithm. In "Table 1" the CPU load is ≥ 10 for all the process and in "Table 6" process P1, P3, P4, P5 and P6 contain the load < 10 or load ≥ 10 so the five processes start their execution because of their release time is earlier from five process two processes are executed using EDF algorithm and the remaining process is switch to ACO algorithm depending upon the following condition:

i) Load of process is greater than the given CPU load.
ii) The priority of the process is high it means the deadline of the process is nearer.
iii) The process count is more.

"Table 7" shows the execution of an Adaptive algorithm and the remaining five process are executed when these process are release for execution it depend upon their release time. The switching depends upon the load, if the load change then the algorithm also changes. An Adaptive algorithm removes the drawback of EDF and ACO algorithm. An Adaptive algorithm schedules more process than the EDF and ACO algorithm.

Table 4. ACO queue

Process ID	Process Name	Start Time	Deadline	Execution Time	Release time in Sec	Status	Priority	CPU Load
1237	P1	10:14:26	10:18:26	4	1	Running	Normal	9.192341
1238	P2	10:15:10	10:18:60	3	2	Waiting	Normal	5.68743
1239	P3	10:16:57	10:19:19	3	1	Running	Normal	11.22071
1240	P4	10:17:44	10:18:55	1	1	Running	Normal	14.55657
1241	P5	10:18:23	10:20:45	2	1	Running	Normal	14.55657
1242	P6	10:19:56	10:24:34	5	1	Running	Normal	7.89763
1243	P7	10:20:34	10:22:12	2	4	Waiting	Normal	11.22071
1244	P8	10:21:22	10:22:23	1	3	Waiting	Normal	12.34563
1245	P9	10:22:46	10:24:33	1	2	Waiting	Normal	13.34562
1246	P10	10:23:06	10:25:29	2	2	Waiting	Normal	10.00071

Table 5. Execution of ACO algorithm

Process ID	Process Name	Start Time	Deadline	Execution Time	Release time in Sec	Status	Priority	CPU Load	Algorithm Used
1237	P1	10:14:26	10:18:26	4	1	Execution Start	High	9.192341	ACO
1238	P2	10:15:10	10:18:60	3	2	Running	Normal	5.68743	-
1239	P3	10:16:57	10:19:19	3	1	Execution Start	High	11.22071	ACO
1240	P4	10:17:44	10:18:55	1	1	Execution Start	High	30.55657	ACO
1241	P5	10:18:23	10:20:45	2	1	Execution Start	High	14.55657	ACO
1242	P6	10:19:56	10:24:34	5	1	Execution Start	High	7.89763	ACO
1243	P7	10:20:34	10:22:12	2	4	Waiting	Normal	11.22071	-
1244	P8	10:21:22	10:22:23	1	3	Waiting	Normal	12.34563	-
1245	P9	10:22:46	10:24:33	1	2	Running	Normal	13.34562	-
1246	P10	10:23:06	10:25:29	2	2	Running	Normal	10.00071	-

Table 6. An Adaptive Algorithm Queue

Process ID	Process Name	Start Time	Deadline	Execution Time	Release time in Sec	Status	Priority	CPU Load
1237	P1	10:14:26	10:18:26	4	1	Running	Normal	9.192341
1238	P2	10:15:10	10:18:60	3	2	Waiting	Normal	5.68743
1239	P3	10:16:57	10:19:19	3	1	Running	Normal	11.22071
1240	P4	10:17:44	10:18:55	1	1	Running	Normal	14.55657
1241	P5	10:18:23	10:20:45	2	1	Running	Normal	14.55657
1242	P6	10:19:56	10:24:34	5	1	Running	Normal	7.89763
1243	P7	10:20:34	10:22:12	2	4	Waiting	Normal	11.22071
1244	P8	10:21:22	10:22:23	1	3	Waiting	Normal	12.34563
1245	P9	10:22:46	10:24:33	1	2	Waiting	Normal	13.34562
1246	P10	10:23:06	10:25:29	2	2	Waiting	Normal	10.00071

Table 7. Execution of an Adaptive Scheduling Algorithm

Process ID	Process Name	Start Time	Deadline	Execution Time	Release time in Sec	Status	Priority	CPU Load	Algorithm Used
1237	P1	10:14:26	10:18:26	4	1	Execution Start	High	9.192341	EDF
1238	P2	10:15:10	10:18:60	3	2	Waiting	Normal	5.68743	-
1239	P3	10:16:57	10:19:19	3	1	Execution Start	High	11.22071	ACO
1240	P4	10:17:44	10:18:55	1	1	Execution Start	High	14.55657	ACO
1241	P5	10:18:23	10:20:45	2	1	Execution Start	High	14.55657	ACO
1242	P6	10:19:56	10:24:34	5	1	Execution Start	High	7.89763	EDF
1243	P7	10:20:34	10:22:12	2	4	Waiting	Normal	11.22071	-
1244	P8	10:21:22	10:22:23	1	3	Waiting	Normal	12.34563	-
1245	P9	10:22:46	10:24:33	1	2	Waiting	Normal	13.34562	-
1246	P10	10:23:06	10:25:29	2	2	Waiting	Normal	10.00071	-

5.5 Success Ratio and CPU Utilization

In real time system, deadline achieving is the most important factor. For that most appropriate performance metric is the Success Ratio and it is defined as [21],

$$Sucess\ Ratio = \frac{Number\ of\ process\ schedule}{Total\ number\ of\ process} \qquad (5)$$

It is same as the number of process scheduled. So the "Figure 3" shows that the Success Ratio and CPU utilization by using an Adaptive scheduling algorithm when system is overloaded. CPU utilization is the total amount of work handle by the CPU. CPU utilization depends upon the task or processes. It is shown in equation (2). When adding a new task in the system then the CPU Utilization is high. "Figure 4" shows the total CPU utilization and the total physical memory required by each task [22]. When the process is created the CPU usage and the memory utilization is changed. "Table 8" gives the total description of each process depend on the load.

Table 8. CPU usage with total memory used when load is varied

Load	Process	Thread	Handles	CPU Usage	Memory Used
10	70	908	22315	2.34	35.09 MB
15	66	879	21619	3.00	39.33MB
20	62	820	21559	4.56	40.11MB
25	57	789	21499	5.89	43.33MB
30	51	756	21434	6.99	44.55MB
35	47	715	20994	10.77	46.33MB

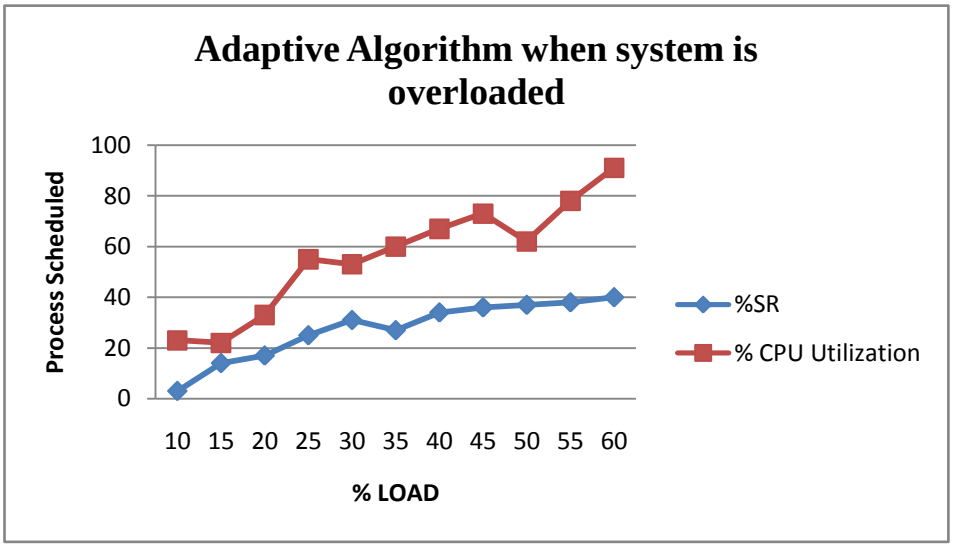

Fig 3: Success Ratio and CPU Utilization

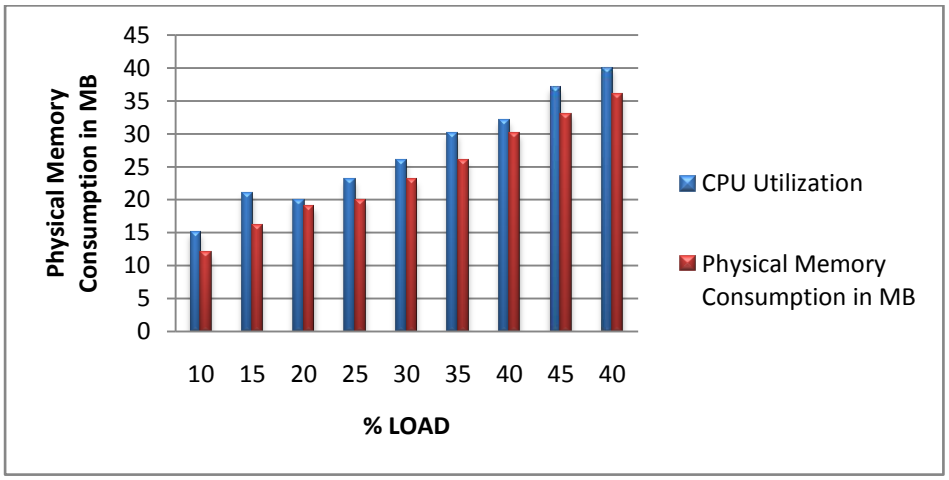

Fig 4: CPU Utilization and Physical Memory Consumption

6. COMPARISON OF SCHEDULING ALGORITHMS

"Figure 5" shows the execution time required for EDF, ACO and an Adaptive algorithm and "Figure 6" shows number of process schedule by the same algorithm. When the load of processor is 5 and total 10 processes are added then the time required for execution by EDF algorithm is less than the ACO and an Adaptive algorithm [23]. It means EDF is better for execution but it cannot handle the overloaded condition. "Figure 6"show the comparison of number of process schedule by the EDF, ACO and an Adaptive algorithm. Graph shows that if the user gives 5 processes and at that time processor load is 5% then the EDF algorithm schedule the 2 out of 5 processes and rest of three processes failed to execute.

At the same time if the user applied ACO algorithm then 4 processes execute and one process is failed to execute because of the execution time. If an Adaptive algorithm is applied then all the processes are executed and all the process deadline is meet. An Adaptive algorithm is very useful when future workload of the system is unpredictable.

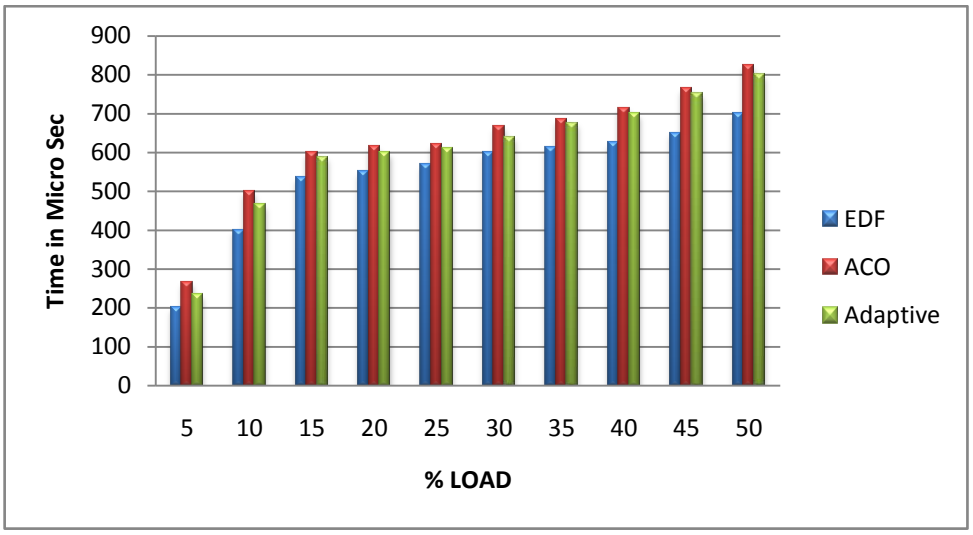

Fig 5: Comparison of Execution Time

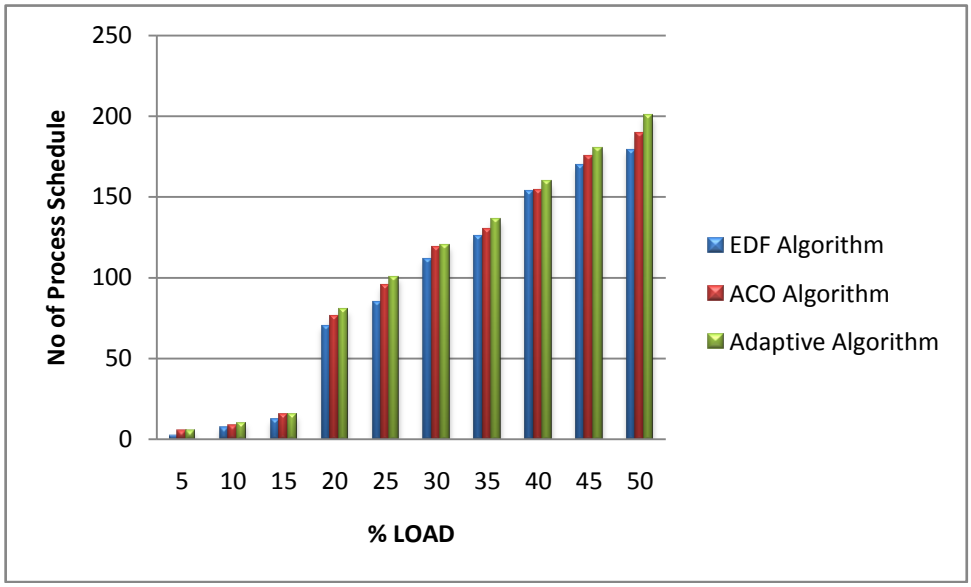

Fig 6: Comparison of Number of Process Schedule

7. CONCLUSION

An Adaptive Algorithm is a dynamic scheduling algorithm and it is beneficial for single processor real-time operating systems. The algorithm is useful when future workload of the system is unpredictable. An Adaptive Algorithm combines the EDF and ACO algorithm. An Adaptive Algorithm schedules the process on single processor when it is preemptive. The advantage of the algorithm is that it automatically switches between the EDF and ACO algorithm and overcome the limitation of both the algorithms (EDF and ACO) algorithms. An Adaptive algorithm requires less time for execution as compared to EDF and ACO and the algorithm gives the result when the system is overloaded. Memory usage of the system is increased when the load of the process is increased.

8. REFERENCES

[1] M.Kaladevi and Dr.S.Sathiyabama "A Comparative Study of Scheduling Algorithms for Real Time Task" International Journal of Advances in Science and Technology, Vol. 1, No. 4, 2010.

[2] A.F.M.Suaib Akhter, Mahmudur Rahman Khan,Shariful Islam " Overload Avoidance Algorithm for Real-Time Distributed System" IJCSN International Journal of Computer Science and Network Security, Vol. 12 no.9,September 2012.

[3] Marko Bertogna and Sanjoy Baruah "Limited Preemption EDF Scheduling of Sporadic Task Systems" IEEE Transactions on Industrial Informatics, Vol. 6, no. 4, November 2010.

[4] Fengxiang Zhang and Alan Burns "Schedulability Analysis for Real-Time Systems with EDF Scheduling" IEEE Transactions on Computers, Vol. 58, no. 9, September 2009.

[5] Lalatendu Behera Durga Prasad Mohapatra "Schedulability Analysis of Task Scheduling in Multiprocessor Real-Time Systems Using EDF Algorithm" 2012 International Conference on Computer Communication and Informatics Coimbatore, INDIA

[6] Shuhui Li, Shangping Re, Yue Yu, Xing Wang, Li Wang, and Gang Quan, " Profit and Penalty Aware Scheduling for Real-Time Online Services" IEEE Transactions on Industrial Informatics, Vol. 8, no. 1, February 2012.

[7] Sachin R. Sakhare and Dr. M.S. Ali "Genetic Algorithm Based Adaptive Scheduling Algorithm for Real Time Operating Systems" International Journal of Embedded Systems and Applications (IJESA) Vol.2, No.3, September 2012.

[8] Jashweeni Nandanwar, Urmila Shrawankar "An Adaptive Real Time Task Scheduler" IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 6, No 1, November 2012.

[9] Ching-Chih Han, Member and Kwei-Jay Lin, "Distance constraint scheduling and its application to real time system" IEEE Transactions On Computers, Vol. 45, no. 7, July 1996.

[10] C. Liu and J. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," J. ACM, vol. 20, pp. 46–61, 1973.

[11] Giorgio C. Buttazzo, Marko Bertogna and Gang Yao "Limited Preemptive Scheduling for Real-Time Systems" IEEE Transactions On Industrial Informatics, Vol. 9, no. 1, February 2013.

[12] Xuefeng Piao, Sangchul Han, Heeheon Kim, Minkyu Park, Yookun Cho "Predictability of Earliest Deadline Zero Laxity Algorithm for Multiprocessor Real-Time Systems" Proceedings of the Ninth IEEE International

[13] Symposium on Object and Component-Oriented Real-Time Distributed Computing 2006

[14] W.N. Chen and J. Zhang, "An Ant Colony Optimization Approach to a Grid Workflow Scheduling Problem with Various QoS Requirements," IEEE Trans. System, Man, and Cybernetics- Part C, vol. 39, no. 1, pp. 29-43, Jan. 2009.

[15] W.N. Chen, J. Zhang, H.S.H. Chung, R.Z. Huang, and O. Liu, "Optimizing Discounted Cash Flows in Project Scheduling—An Ant Colony Optimization Approach," IEEE Trans. Systems, Man, and Cybernetics-Part C, vol. 40, no. 1, pp. 64-77, Jan. 2010.

[16] Ketan Kotecha and Apurva Shah "Scheduling Algorithm for Real-Time Operating Systems using ACO" 2010 International Conference on Computational Intelligence and Communication Networks.

[17] Yuren Zhou, Xinsheng Lai, Yuanxiang Li, and Wenyong Dong "Ant Colony Optimization with Combining Gaussian Eliminations for Matrix Multiplication" IEEE Transactions On Cybernetics, Vol. 43, no. 1, February 2013.

[18] Michael A. Palis "The Granularity Metric for Fine-Grain Real-Time Scheduling" IEEE Transactions on Computers, Vol. 54, no. 12, December 2005.

[19] Ya-Shu Chen, Han Chiang Liao, and Ting-Hao Tsai " Online Real-Time Task Scheduling in Heterogeneous Multicore System-on-a-Chip" IEEE Transactions On Parallel And Distributed Systems, Vol. 24, no. 1, January 2013.

[20] Joon-Woo Lee and Ju-Jang Lee "Ant-Colony-Based Scheduling Algorithm for Energy-Efficient Coverage of WSN" IEEE Sensors Journal, Vol. 12, no. 10, October 2012.

[21] Wei-Neng Chen, Member, IEEE, and Jun Zhang, Senior Member, IEEE "Ant Colony Optimization for Software Project Scheduling and Staffing with an Event-Based Scheduler" IEEE Transactions on Software Engineering, Vol. 39, no. 1, January 2013

[22] Krithi Ramamritham, John A Stankovik and Perng Fei Shiah, "Efficient scheduling algorithms for real-time multiprocessor systems", IEEE Transaction on Parallel and Distributed Systems, vol. 1(2), April 1990.

[23] Silviu S. Craciunas, Christoph M. Kirsch Ana Sokolova "Response Time versus Utilization in Scheduler Overhead Accounting" 2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium.Bowman, M., Debray, S. K., and Peterson, L. L. 1993. Reasoning about naming systems.