Implementation & Performance Analysis of Real Time Scheduling Algorithms for Three Industrial Embedded Applications (IEA)

Mr. Vishal Vora[1] and Dr. Ajay Somkuwar[2]

[1]Department of Electronics & Communication Engineering, A.I.T.S, Gujarat, India
vishal.s.vora@gmail.com
[2]Department of Electronics & Communication Engineering, M.A.N.I.T., Bhopal, India
asomkuwar@gmail.com

ABSTRACT

It is essential necessity of embedded system to generate response or output within deadline. Various Scheduling algorithms are used for this purpose to enhance performance, accuracy and security of embedded system applications. These algorithms are either fixed or dynamic type, which are designed for various constraints. Real time scheduling algorithms are mainly depending on type of task i.e. either periodic or aperiodic or sporadic. Some real time scheduling algorithms like Pre-emptive & Non Pre-emptive, Round Robin Scheduling, &Time Slicing are few of the favourites for various embedded applications. The type of task and accordingly type of scheduling algorithm used for various applications out of which 3 are discussed in this paper as case studies and they are Watch making, Conveyor belts and chair manufacturing..

KEYWORDS

Real Time Scheduling, Deadline, Preemptive & Non-preemptive Task, dynamic and static algorithms.

1. INTRODUCTION

In the physical world, the purpose of a real-time system is to have a physical effect within a chosen time-frame. The controlling system interacts with its environment based on information available about the environment. Any real time system controls a device or sensors which provide computer analysis data at regular intervals and then respond it by sending signals to actuators. On the other hand, irregular interval events also checked by sending special responding signal. All these systems are bound by time limit i.e. deadline. The performance of the real time system is decides by the matching of these deadlines. The more the miss the deadlines the less the performance of the system, and so to avoid that it must required real time scheduling for multitasking. Each task occurring in a real time system has some timing properties. To do so any real time system must have following parameters to understand and carried out.

Release time: This is the time required to any task for being ready. **Deadline**: this is the time by which task must be completed, after release time. **Minimum delay**: This is the minimum time required as latch up delay for getting task process after release time. **Maximum delay**: This is the maximum time required as latch up delay for getting task process after release time. **Worst case execution time**: This is the longest time required to complete any task successfully under critical and unfair system parameters. **Run time**: This is the time taken without interruption for completing the task, after released time. Although missing deadlines is not desirable in any real-time system and accordingly there are types of real time systems as explained below:

Hard Real Time System: In which deadline (Timing Constraints) is static and not changed in any circumstances. The output of this circuit is zero if it is not completed 100% successfully

Within deadline. Example: Car Manufacturing Plant, Watch/Cold Drink Manufacturing Plants.
Soft Real Time System: In which deadline (Timing Constraints) is dynamic and can be flexible so that task can be completed successfully. Example: ATM
Firm Real Time System: It is combination of both Hard & Soft Real Time systems

Meeting the timing constraints of the system, preventing simultaneous access to shared resources and devices, attaining a high degree of utilization while satisfying the timing constraints of the system however this is not a primary driver, reducing the cost of context Switches caused by preemption, Reducing the communication cost in real-time distributed systems; we should find the optimal way to decompose the real-time application into smaller portions in order to have the minimum communication cost between mutual portions.

2. Types of Tasks

2.1 Periodic/Aperiodic/Sporadic tasks

Periodic tasks execute at every known fixed time intervals. Normally, periodic tasks have constraints which indicate that instances of time constraints. While, aperiodic tasks execute at any random time constraints and would not have pre-defined timing sequence, the one it do have in periodic tasks. Sporadic tasks are combination of both periodic and Aperiodic, where in, the executing time is Aperiodic but the executing rate is periodic in nature, from the perspective of timing constraints. The time constraints are usually a deadline.

2.2 Pre-emptive/Non-pre-emptive tasks

It is necessary in some real-time scheduling algorithms that high priority tasks are execute first, and so current task is pre-empted, on the retro respect, it is also necessary that some tasks are never Pre-empted in executing sequence and consider as non-pre emptive task. This will decides by characteristic and nature of real time applications.

2.3 Fixed/Dynamic priority tasks

In some of real time systems, the priority of the tasks are decided either static (i.e. fixed in nature, which never changes once assigned) or dynamic (i.e. the priority can be changed at any time according to need of scheduler).

2.4 Independent/Dependent tasks

In case of multitasking output of one task is related and drive another task. Therefore, execution of a task should be start after finishing the execution of the other task; this is the concept of dependency. The dependent tasks use shared memory or communicate data to transfer the information generated by one task and required by the other one. While we decide about scheduling of a real-time system containing some dependent tasks, we should consider the order of the starting and finishing time of the tasks.

3. Methods and Analysis

The scheduling problems considered in this paper are characterized by a set of tasks T= {τ1,τ2…. τn} and a set of processors (machines) π= { π1, π2…. π n} on which the tasks are to be processed. Besides processors, tasks may require certain additional resource during their execution. Scheduling, generally speaking, means the assignment of processors from S and resources from 9 to task from in order to complete all tasks under certain imposed constraints. Considering very simple scenario of real time where in each processor will complete one task at time T, so does this time T have following real time properties: Release time Rj; if the ready times are the same for all tasks from T, then Rj=0 is assumed for all j. Completion time Cj Deadline D; usually, penalty functions are defined in accordance with deadlines. Priority Wj

Precedence constrains among tasks τi < τj means that the processing of τi must be complete before τj can be started. In other words, set T is partially ordered by a precedence relation. Task T is called depended if it is restricted by two tasks simultaneously. Otherwise, the tasks are called independent. Depending on the type of application we are confronted with, different performance measures or optimality criteria are used to evaluate schedules. In real-time applications, performance measures are used that take lateness or tardiness of tasks into account. A schedule for which the value of a particular performance measure is at its minimum will be called optimal.

3.1 Scheduling for the Simple Model

The nature and no. of tasks will decides by the characteristic and type of real time applications and the priority of tasks will decide by scheduling algorithm used for real time system.

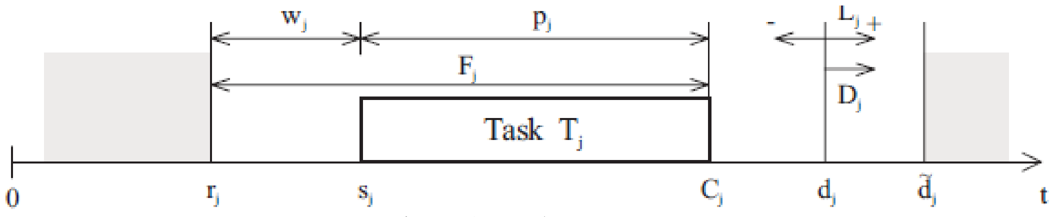


Figure 1. Task parameters

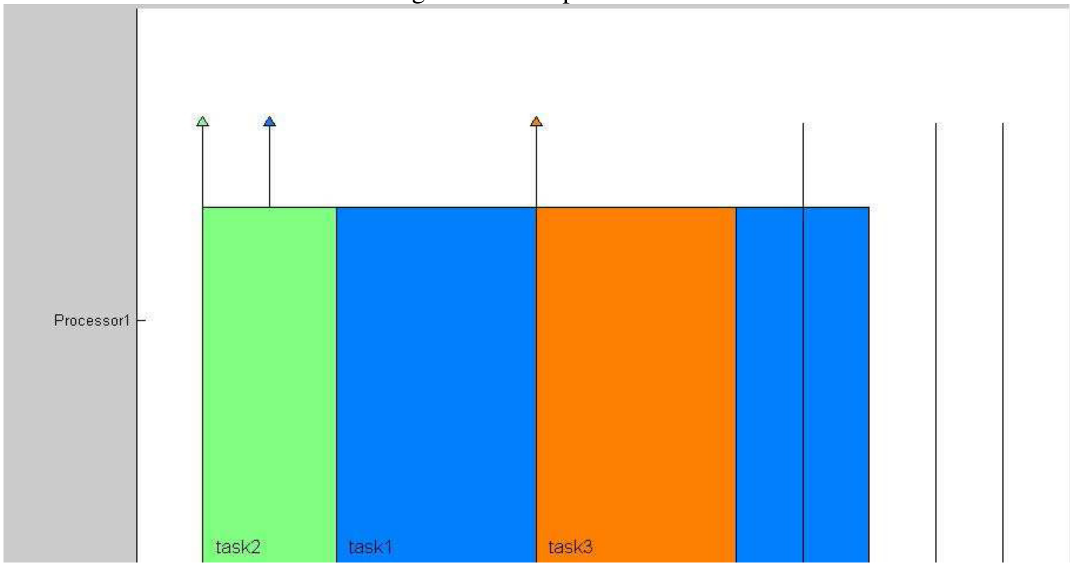


Figure 2. Task Priority schedule by processor

In some cases, the priorities allotted to tasks can be used to solve such problems; for this paper we had taken a case of 3 simple tasks as shown in figure 2 and the analysis is shown following real time results.

TABLE I
TASK PRIORITIES WITH PREEMPTION

Task	Priority	Computation Time (in Seconds)	Time Period/ Processor (in seconds)
τ1	1	2	7
τ2	2	4	16
τ3	3	7	31

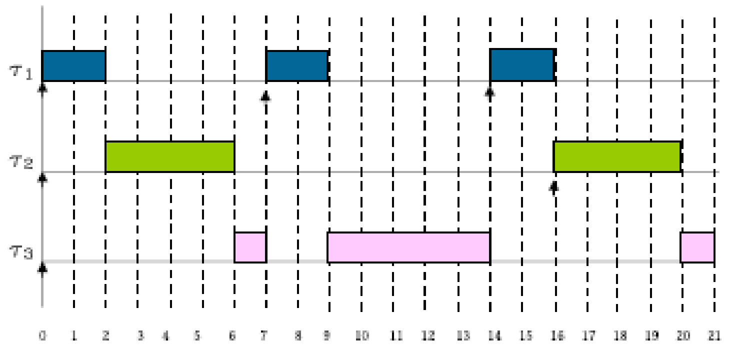


Figure 3. Priorities with Pre-emption

Above timing waveforms are providing visual display of various three tasks for each and every timing interval. Under which the program is executed. The system called having highest performance if it did not miss any deadline. Let us study the above concepts on the Rate-Monotonic algorithm (RM) and Earliest Deadline First algorithm (EDF).Both the RM and EDF algorithms are optimal real-time Scheduling algorithms. An optimal real-time scheduling algorithm is one which may fail to meet a deadline only if no other scheduling algorithm can meet the deadline. The following Assumptions are made for both the RM and EDF algorithms. (a) No task has any non-preempt able section and the cost of preemption is negligible (b) processing requirements are to be consider as negligible and (c) all tasks are consider to do not have any precedence constraint and so independent in nature.

4. SCHEDULING ALGORITHMS OF REAL-TIME SYSTEMS

The goals for real-time scheduling are completing tasks within specific time constraints and Preventing from simultaneous access to shared resources and devices. Although system resource utilization is of interest, it is not a primary driver. In fact, predictability and temporal correctness are the principal concerns. The algorithms used, or proposed for use; in real-time scheduling vary from relatively simple to extremely complex. The topic of real-time scheduling algorithms can be studied for either uni processor or multiprocessor systems. We first study uni processor real-time scheduling algorithms. Various on line scheduling algorithms are either of static or dynamic type and are discussed as follow:

4.1 Static-priority based algorithms

Static-priority based algorithms are relatively simple to implement but lack flexibility. They are arguably the most common in practice and have a fairly complete theory. They work well with fixed periodic tasks but do not handle aperiodic tasks particularly well, although there are some methods to adapt the algorithms so that they can also effectively handle aperiodic tasks. Static priority-based scheduling algorithms have two disadvantages, which have received a significant amount of study. Their low processor utilization and poor handling of aperiodic and soft-deadline tasks have prompted researchers to search for ways to combat these deficiencies many real-time systems have the characteristic in which the order of task execution is known a priori and each task must complete before another task can start. These systems can be scheduled non-preemptively. This scheduling technique, which is called non preemptive static-priority based algorithms, avoids the overhead associated with multiple context switches per task. This property improves processor utilization. Additionally, tasks are guaranteed of meeting execution deadlines

4.2 Dynamic-priority based algorithms

These types of algorithms are required more on line resources as they provide extreme level of flexibility at run time. There are two subsets of dynamic algorithms: planning based and best effort. This is very suitable for soft real time systems from the timing constraints perceptive. This is often accomplished by utilization of spare processor capacity to service soft and aperiodic tasks.

4.3 Planning Based Algorithms

They are guarantee that if a task is accepted for execution, the task and all previous tasks accepted by the algorithm will meet their time constraints The planning based algorithms attempt to improve the response and performance of a system to aperiodic and soft real-time tasks while continuing to guarantee meeting the deadlines of the hard real-time tasks The general model for these types of algorithms is a system where all periodic tasks have hard deadlines equal to the end of their period, their period is constant, and their worst case. Execution times are constant. All aperiodic tasks are assumed to have no deadlines and their arrival or ready times are unknown. Planning based algorithms tend to be quite flexible in servicing aperiodic tasks while still maintaining the completion guarantees for hard-deadline tasks. Most of the algorithms also provide a form of guarantee for aperiodic tasks. They reject a task for execution if they cannot guarantee its on-time completion. Most of the planning based algorithms can provide higher processor utilization than static priority-based algorithm while still guaranteeing on-time completion of accepted tasks.

4.4 Multiprocessor Scheduling Algorithms

Some advanced and complex real time applications do require more than one processor to complete set of tasks (multitasking) successfully. The following assumptions may be made to design a multiprocessor scheduling algorithm:

4.4.1 Job preemption is permitted

Here, a job executing on a processor may be preempted prior to completing execution, and its execution may be resumed later. We may assume that there is no penalty associated with Such preemption.

4.4.2 Job migration is permitted

Here, a job that has been preempted on a particular processor may resume execution on a different processor. Once again, we may assume that there is no penalty associated with such Migration.

4.4.3 Job parallelism is forbidden

Here, each job at least run on any one processor simultaneously, and not keeping any processor idle in nature, in process, all processors consume same computing power for keeping it identical for scheduler.

4.5 Partitioning Scheduling Algorithms

These technique partition the same processor equally among all available tasks. Tasks are not allowed to migrate; hence the multiprocessor scheduling problem is transformed to many uni processor scheduling problems Constraints of Real-Time Systems Many industrial applications with real-time demands are composed of tasks of various types and constraints. Arrival patterns and importance, for example, determine whether tasks are periodic, aperiodic, or sporadic, and soft, firm, or hard. The same holds for the various constraints on tasks. In addition to basic temporal constraints, such as periods, start-times, deadlines, and synchronization demands such as precedence, or mutual exclusion, a system has to fulfill complex application demands which cannot be expressed directly with basic constraints

4.5.1 Scheduling of Sporadic Tasks

Sporadic Tasks are released irregularly, often in response to some event in the operating environment. While sporadic tasks do not have periods associated with them, there must be some maximum rate at which they can be released. That is, we must have some minimum interval time between the releases of successive iterations of sporadic tasks. Some approaches to deal with sporadic tasks are outlined as follows the first method is to simply consider sporadic tasks as periodic tasks with a period equal to their minimum inter arrival time. The other approach is to define a fictitious periodic task of highest priority and of some chosen fictitious execution period. During the time that this task is scheduled to run on the processor, the processor is available to run any sporadic tasks that may be awaiting service. Outside this time, the processor attends to the periodic tasks. This method is the simplest approach for the problem. The Deferred Server is another approach, which wastes less bandwidth. Here, whenever the processor is scheduled to run sporadic tasks and finds no such tasks awaiting service, it starts executing the periodic tasks in order of priority. However, if a sporadic task arrives, it preempts the periodic task and can occupy a total time up to the time allotted for sporadic tasks.

4.5.2 Scheduling of Aperiodic Tasks

Real-time scheduling algorithms that deal with a combination of mixed sets of periodic real Time tasks and aperiodic tasks have been studied extensively. The objective is to reduce the average response time of aperiodic requests without compromising the deadlines of the periodic tasks. Several approaches for servicing aperiodic requests are discussed as follows. The Polling Server executes as a high-priority periodic task, and every cycle checks if an event needs to be processed. If not, it goes to sleep until its next cycle and its reserved execution time for that cycle is lost, even if an aperiodic event arrives only a short time after. This results in poor aperiodic response time. Polling consists of creating a periodic task for servicing aperiodic requests. At regular intervals, the polling task is started and services any pending aperiodic requests. We apply the following method in order to achieve an effective release time: If a job has no predecessors; its effective release time is its release time. If it has predecessors, its effective release time is the maximum of its release time and the effective release times of its predecessors. An effective deadline can be found as follows. If a job has no successors, its Effective deadline is its deadline. Priority Inversion In a preemptive priority based real-time system, sometimes tasks may need to access resources that cannot be shared. For example, a task may be writing to a block in memory. Until this is completed, no other task can access that block, either for reading or for writing. The method of ensuring exclusive access is to guard the critical sections with binary semaphores.

5. CASE STUDIES

5.1 Watchmaker

Consider a case study, where total 5 watches are required to have list of repairs and that to be schedule such that it would complete within assigned time interval and in this case it is 24 hours.

List of repairs:

- Perform battery replacement of Watch-1 at exactly 14:00 Hours.
- Watch 2 is missing hand and has to be ready at 12:00 Hours.

- Clockwork of Watch-3 should repair by 16:00 Hours.
- Watch-4 seal ring repair and need to reseal it before 15:00 Hours.
- Watch- 5 has repaired bad battery and broken light. Make Ready for the customer before 13:00 Hours.
  Batteries will be delivered to you at 9:00 Hours, seal ring at 11:00 Hours and the hand for watch number 2 at 10:00 Hours.

Time of repairs:

- Battery replacement: 1 hour
- Replace missing hand: 2 hours
- Fix the clockwork: 2 hours
- Seal the case: 1 hour
- Repair of light: 1 hour

Solution of the case study is shown in five steps:

1. Definitions of no. of tasks
   t1=task('Watch1',1,9,inf,14);
   t2=task('Watch2',2,10,inf,12);
   t3=task('Watch3',2,8,inf,16);
   t4=task('Watch4',1,11,inf,15);
   t5=task('Watch5',2,9,inf,13);
2. Decide one task object. T=taskset([t1 t2 t3 t4 t5]);
3. Our goal is to assign the tasks on one processor, in order, which meets the required due dates of all tasks if possible. prob=problem('1|pmtn,rj|Lmax');
4. The design of 2 set of parameters, where in the first one is the taskset we have defined above, and the second one is the type of problem written in Graham-Blaziewitz notation TS=horn(T,prob) Solving time: 0.046875s
5. Visualize the final schedule by standard plot function, plot(TS,'proc',0);

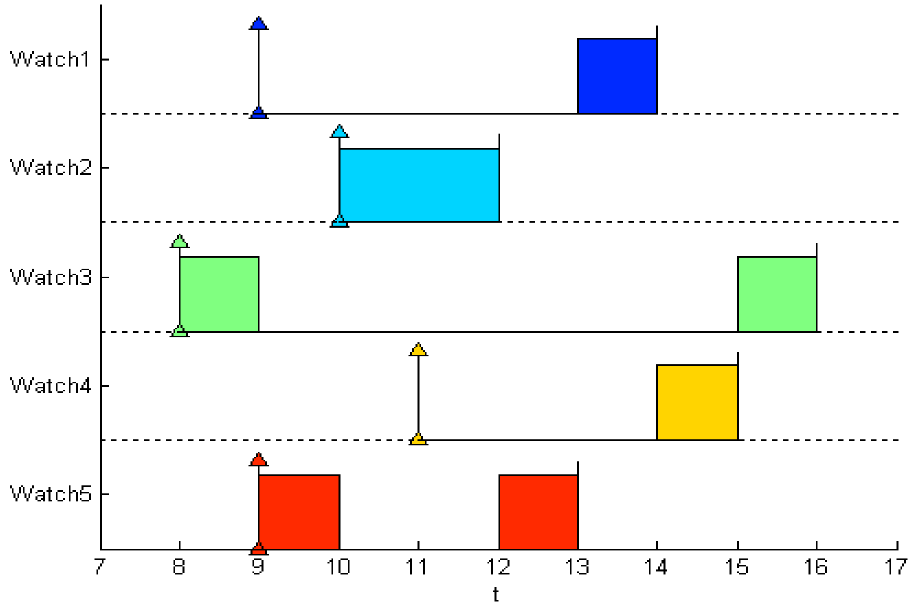


Figure 4. Scheduling by processor for watch making batches in MATLAB

5.2 Conveyor Belts

Second case study is taken for list scheduling algorithm, where in goods are taken by two conveyor belts in a factory and their material carried from one place to another with minimal

Time constrains. Transported articles represent five kinds of construction material and two conveyors belts as processors are available. Below table shows the assignment of this problem.

Solution of the case study is shown in five steps:

1. Design Taskset as T = taskset([40 50 30 50 20]);
2. Finalize task parameters as T.Name = {'sand','grit','wood','bricks','cement'};
3. Define the problem, which will be solved. p = problem('P|prec|Cmax');
4. Perform list scheduling for defined no. of processors as TS = listsch(T,p,2)
5. Visualize the final schedule by standard plot function, plot(TS)

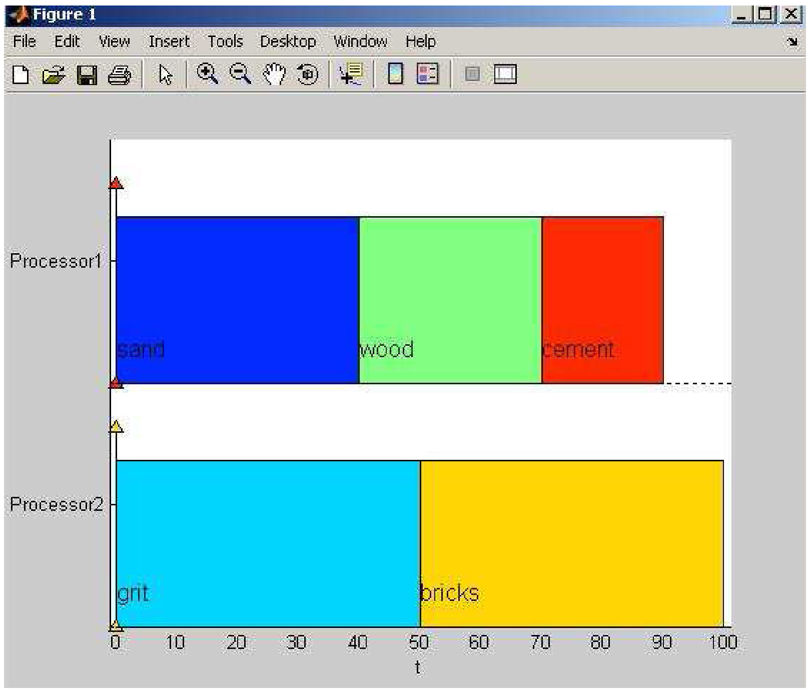


Figure 5. Scheduling by processor for Conveyor Belt Case Study in MATLAB

5.3 Chair Manufacturing

In this case study, we will discuss chair manufacturing application, where in two factory works require to assemble chair, after manufactured its various parts like legs, seat and backrest independently, within assigned time intervals. Below diagram shows the mentioned problem by graph representation.

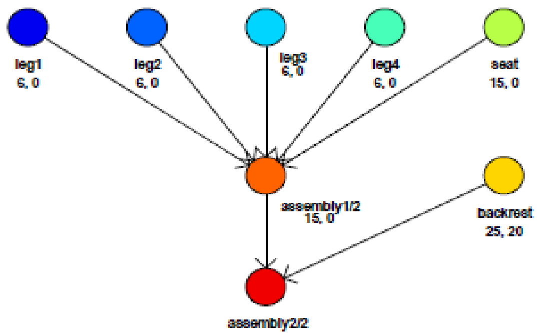


Figure 6. Graph Representation of Chair Manufacturing

After its solution its output response can be visualized as per below output screen of by MATLAB Program:

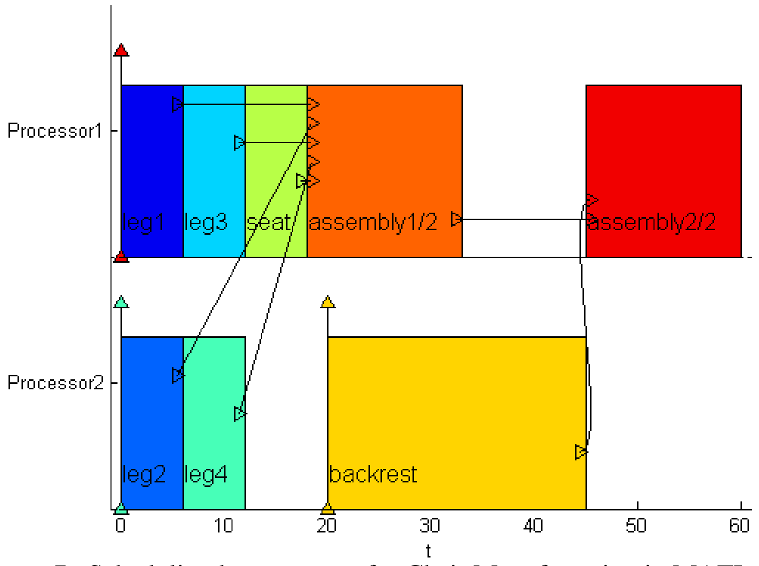


Figure 7. Scheduling by processor for Chair Manufacturing in MATLAB

3. CONCLUSIONS

Deadline is the major factor by which the performance of any real time system is measured and judge. Deadline Failure happens when a system cannot satisfy any one real time properties like the release time, the deadline and the execution time. The deadline may be hard, soft or firm real time depending on the type of system. Tasks may have precedence constraints. A task may be periodic, aperiodic, or sporadic. The schedule may be preemptive or non-preemptive. Less critical tasks must be allowed to be preempted by higher critical ones when it is necessary to meet deadlines. For the real-time systems in which tasks arrive extensively we have to use more than one processor to guarantee that tasks are feasibly scheduled. Therefore, the number of available processors is another parameter to consider. The available processors may be identical, uniform or unrelated.

ACKNOWLEDGEMENTS

The authors would like to acknowledge math works incorporation, Management and R &D Team of Orpat Pvt. Ltd, India for providing required infrastructure for this work.

REFERENCES

[1] Network Flows, Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin, Prentice Hall, February 18, 1993, 864, 013617549X..

[2] Mixed Integer Quadratic Program (MIQP) solver for Matlab, Alberto Bemporad and Domenico Mignone, Automatic Control Laboratory, ETH Zentrum, Zurich, Switzerland, 2004.

[3] Sequencing unit-time jobs with treelike precedence on m processors to minimize maximum lateness, P.J. Brucker, Proc. IX International Symposium on Mathematical Programming, Budapest, 1976.

[4] Scheduling subject to resource constrains: classification and complexity, J. B lazewicz, J. K. Lenstra, and A. H. Rinnooy Kan, Ann. Discrete Math, 11-24, 1933.

[5] Scheduling with Start Time Related Deadlines, P. Sucha and Z. Hanzalek, IEEE Conference on Computer Aided Control Systems Design, September, 2004.

[6] TORSCHE Scheduling Toolbox for Matlab, P. Sucha, M. Kutil, M. Sojka, and Z. Hanzalek, IEEE International Symposium on Computer-Aided Control Systems Design (CACSD'06), Munich, Germany, 2006.

[7] Scheduling Computer and Manufacturing Process, J. B la˙zewicz, K. H. Ecker, E. Pesch, G. Schmidt, and J. W¸eglarz, Springer, 2001, 3-540-41931-4.

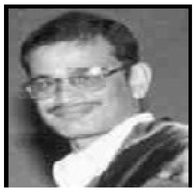
[8] Scheduling subject to resource constrains: classification and complexity, J. B la˙zewicz, J. K. Lenstra, and A. H. Rinnooy Kan, Ann. Discrete Math, 11-24, 1933.

[9] Scheduling with Start Time Related Deadlines, P. ˇS°ucha and Z. Hanz´alek, IEEE Conference on Computer Aided Control Systems Design, September 2004.

[10] Scheduling, Michael Pinedo, Prentice Hall, 2002, 586, 0-13-028138-7.

Authors

Mr. V. S. Vora is currently a research Scholar at Dr. K.N. Modi University, Newai, at Rajasthan, India. He is working as an Assistant Professor in Department of Electronics & Communication Engineering at Atmiya Institute of Technology & Science, Rajkot, at Gujarat, India. He has published 4 text books on Embedded Systems for 3 universities with Techmax Publications, Pune, India. He has published more than 20 research papers of national & international repute.



Dr. Ajay K. Somkuwar carried out his research from Indian Institute of Technology New Delhi and awarded PhD In 2003. Presently He is working as Professor in the department of Electronics & Communication Engineering at M.A.N.I.T. Bhopal, India. His research area includes signal processing, biomedical engineering & embedded systems. He has published more than 50 research papers of national & international repute. He has visited Italy, Germany for his research presentation.