ISSN:0975-9646

A New Approach to Schedule Precedence Constraint Tasks in Real Time Systems

Radhakrishna Naik
Dept. of Comp. Sci. & Engg
MIT, Aurangabad, India

R.R.Manthalkar
Dept. Of Ele. & Comm.
SGGSCOE,Nanded,India

Yogiraj P. Korde
Dept. of Info. Tech
SRESCOE,Kopergaon,India

Abstract--**In typical real time systems, tasks need to communicate so as to achieve effective resource utilization. Tasks should be scheduled considering their precedence constraints. Modified rate monotonic scheduling, earliest deadline scheduling algorithm and latest deadline first scheduling algorithm do well in precedence constraint tasks scheduling; however these algorithms do not take care about overall contribution of individual tasks in tasks network. This paper suggests novel idea which is considering both contribution of individual tasks and deadline. This algorithm is modification of performance contribution and deadline (PCD) algorithm. It is proved through analysis that, number of missing deadlines and context switching is less as compared to PCD. Important feature of this algorithm is that it supports both cyclic and acyclic process structure for scheduling.**

Keywords— **Performance Contribution and Deadline (PCD), Rate Monotonic (RM) and Earliest Deadline First (EDF)**

I. INTRODUCTION

In this paper analysis of architecture is done keeping in the view that individual messages between tasks have been failed.

In a typical real time systems, tasks interact directly or indirectly with each other. Tasks get interact in order to synchronize their execution by a message transmission or they may share the resources other than processor. These resources may be exclusive or shared. This creates precedence relationship among the tasks. Precedence relationship is known before execution. Here tasks are static and can be represented by a static precedence graph. If a task is not ready, but its output is necessary for the execution of next task then the next task has to wait for the execution irrespective of its priority. But however it is not possible to keep the processor idle for that time. Thus it is essential to consider task dependency at the time of scheduling.

In spite of the increased system complexity, real time applications are mainly configured acting on the task priorities based on single parameter, which usually express the importance of task [1].There are other system constraints like message communication, performability and reliability, which need to set into priority levels. Assigning priority with single parameter is not sufficient.

It is quite possible that in a task set, a task may be having earliest deadline but its performance contribution is low. Scheduling such tasks with highest priority does not carry any meaning. Majority of today's commercial operating systems schedule the tasks based on a single parameter. However recent research on flexible scheduling showed that a single parameter is not enough to express the entire application requirement.

In order to make the scheduling task more effective, an algorithm for scheduling communicating tasks is designed.This is a non pre-emptive precedence constraint, offline scheduling algorithm intended for uniprocessor architecture.

The objective of this algorithm is to decide that whether or not it is possible to schedule tasks under the given assignments such that all of their deadlines and precedence constraints can be met. This is in contrast to conventional methods which deal with either assignment or scheduling of tasks considering either period or deadline, alone but not both.

The algorithm deals with analysis of communication network and tries to find out performance contribution of each task with respect to each other. After identifying task contribution in terms of message communication, they are classified by considering both task contribution and deadlines.

Following are features of proposed algorithm:

- Tasks communicate with each other during the course of their execution to accomplish a common system goal.
- The tasks to be allocated are invoked periodically at fixed time intervals during the mission lifetime.
- Consequences of failure link on other task are calculated.
- Accordingly task ranking is decided.

II. RELATED WORK

Since the first results published in 1973 by Liu and Layland [1] on the Rate Monotonic (RM) and Earliest Deadline First (EDF) algorithms, a lot of progress has been made in the schedulability analysis of periodic task sets. Unfortunately all analysis is done based on schedulability, jitter, number of preemptions, runtime overhead, robustness during overloads and the transient overload etc. However today hardware technology is improved. Hardware resources are cheaper and speed of hardware has increased drastically. Therefore these issues are not important now.

A. Issues involved in precedence constrained task scheduling

Since the problem of assigning tasks subject to precedence constraints is generally NP hard [4], hence it is not possible to determine optimal schedule efficiency. Some form of approximation using heuristics was developed for this problem. For example CP/MISF

(Critical Path / Most Immediate Successors First), enumeration tree of task is generated and searched using a heuristic algorithm [5]. Chu and Leung [6] presented an optimal solution to the task assignment problem in the minimizing average task response time subject to certain timing constraints. Shen and Tsai[7], Ma et al. [8] and Sinclair[9] derived optimal task assignment to minimize the sum of task execution and communication costs with the branch and bound[10]. Considering embedded system's complexity growing day by day, task allocation algorithm has been suggested for task control [11] and turbo engine control [12].

Latest Deadline First Algorithm (LDF) suggested by L.Lawler[13] is the only algorithm suggested for uniprocessor precedence constrained task scheduling which is as below:

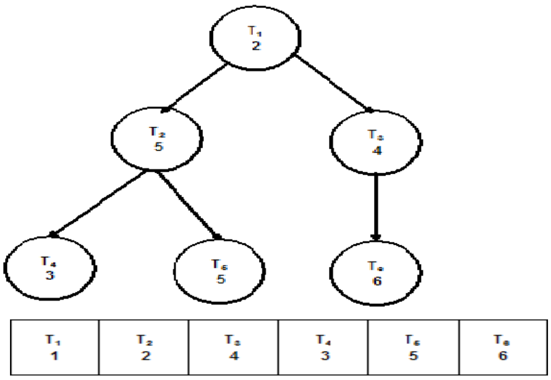


Fig. 1: Example of LDF scheduling algorithm and its Gantt chart

It constructs a schedule from tail to head using a stack:

1. Pick up a task from the current DAG that has the latest (Highest) deadline and does not precede any other tasks (a leaf!)
2. Remove the selected task from the DAG and put it to the stack. Repeat the two steps until the DAG contains no more tasks.
3. Select the last task to run first.

The stack represents the order in which tasks should be scheduled. Following figure 1 shows example of LDF scheduling and its Gantt chart.

Although LDF is an optimal algorithm, it supports only acyclic graph. It does not support cyclic graph for analysis. It describes task priority only in terms of deadline.

B. Use of modeling in predictability analysis

Modeling plays a central role in system engineering. It was believed that modeling methodologies should be closely related to implementation methodologies for building correct real time systems, for supporting end to end constraints at step in the design task [14].

Modeling system in the large became an important research topic in both academia and industry [15]. A key issue in a modeling methodology was the issue of adequate operators to compose heterogeneous schedulers (e.g. synchronous, asynchronous, event triggered or time triggered). For this reason, some researchers proposed model based theories for computing scheduling policies [16]. Another challenge consists in adequately relating the functional and non functional requirements of the application software with the underlying execution platform. Following were two approaches to address these problems.

1. One relies on architecture description languages that provide means to relate software and hardware components e.g. meta-H [17].
2. The other is based on the formal verification of automata based models automatically generated from software and appropriately annotated with timing constraints [18, 19].

Thus analyzing above survey in detail, it is clear that MDE analysis can be used to design a scheduler specially when there is much complexity involved so that one can take corrective steps at design phase itself. As far as uniprocessor is concerned, only LDF scheduling algorithm is an optimal solution. However it supports only acyclic task structure. For scheduling cyclic task structure many heuristics have been suggested. These heuristics are application specific not applicable to all cases. Therefore it is essential to find out generalized solution which will fit for both acyclic as well as cyclic task structure.

Following algorithm is suggested to deal above discussed issues.

αPCD: Variance of PCD scheduling algorithm.

C. Performance Contribution Factor and Deadline (PCD) scheduling algorithm

PCD scheduling algorithm described in Radhakrishna Naik et al. [20] is as below

1. Various modules and taskes in the system to be developed are identified. Signals between taskes are also identified. Resources required for the system are identified.
2. System's behavior is presented using FDL in EVENT STUDIO.
3. Task interaction collaboration diagram is generated. It represents task communication diagram.
4. Simplified task interaction diagram is evaluated from task interaction collaboration diagram.
5. Precedence of taskes is identified and conditional probability is calculated. If a task is dependent on two or more taskes, then Bayes theorem for conditional probability is used to find out dependability of each task with others.
6. Accomplishment level for each task is assumed, based on criteria that how many incoming links are associated with each task.
7. Performance contribution factor (PCF) for each task is calculated.
8. Tasks are classified on PCF and relative deadline to four classes, class-I, class-II, class-III and class-IV. While scheduling tasks, highest priority is given to class with Highest PCF and quickest deadline.

III. αPCD: VARIANCE of PCD SCHEDULING ALGORITHM.

In αPCD, procedure of finding performance contribution of a task and classifying in various classes considering PCF and relative deadline is similar. Only difference is that the quantum of execution of mandatory portion is controlled by α parameter. This parameter is calculated as shown in equation 1. For calculation of this parameter highest priority task is being selected.

$$Alpha \Longrightarrow \left\lfloor \frac{D_1 - p_{1.}e}{k} \;|k\ is\ number\ of\ predecessor\ \&\&\ n \geq k \geq 0 \right\rfloor \quad (1)$$

Taskes are classified based on PCF and deadline as shown in the Table 1 and Table 2.

Table I: Classification of taskes for soft real time system based on *deadline*

Priority Levels	PCF(E_G)	Deadline(D)
Class-I	High	Low
Class-II	Low	Low
Class-III	High	High
Class-IV	Low	High

Table II: Classification of taskes for soft real time system based on Performance

Priority Levels	PCF(E_G)	Deadline(D)
Class-I	High	Low
Class-II	High	High
Class-III	Low	Low
Class-IV	Low	High

Scheduling policy is same as that of PCD algorithm.

A.System model of $\boldsymbol{\alpha}$ *PCD*

Consider a real time system consisting of n tasks. Any task P_i can have k_i different states: from complete failure to perfect functioning. The entire system has K different states as determined by the states of its taskes. Let $Y(t)$ is a multi state system (MSS) state at given instance of time t, where $Y(t)\in\{1,2,3....K\}$. Each state of task has its accomplishment level G_k, where $k \in \{1,2,3...K\}$ .The system output performance distribution(OPD) can be defined by two finite vectors, accomplishment level vector G_k and probability vector p_k such that, $p=\{p_k(t)\}=Pr\{G(t)=G_k\}$ where $0\leq k\leq K$. $G(t)$ is a random accomplishment level of a task of MSS and $Pr(x)$ is probability of event x.

The MSS behavior is characterized by its evolution in the space of states. To characterize numerically this evolution task, one has to determine the MSS reliability indices. In order to define the MSS ability to perform its task, a function $f(G,W)$ is represented which defines the desired relation between the MSS random accomplishment level G and expected accomplishment level W[138].

$$f(G,W) = G - W \quad (2)$$

When $f(G,W) < 0$ then it is assumed that task has completely failed its execution. MSS availability $A(t)$ is the probability that task's execution will reach its desired accomplishment level at a given instance of time.

$$f(G,W) \geq 0 \qquad t>0 \quad (3)$$

Availability of task state at expected accomplishment level W is given by

$$A(W) = \sum_{f(G_k,W)\geq 0} p_k \quad (4)$$

Where p_k steady state probability of MSS state k. The resulting sum is is taken only for the states satisfying condition $f(G,W) \geq 0$.

A real time application is composed of number of tasks and can be represented by a task interaction diagram.

$$PID = (M,E) \quad (5)$$

Where M is set of tasks $(m \in M)$ and E is the set of edges $(e \in E)$.

The edges in the task interaction diagram correspond to the communication messages, m_{ij} associated with each task represents messages coming from task p_i to p_j. The source task is called parent or predecessor and destination task is called child or successor. A task can not start execution before it gathers all the messages from its predecessor. If any of the messages is lost from predecessor taskes to successor task accomplishment level of G of the successor is going to reduce proportionally. Hence in a given system, many taskes can perform at different accomplishment level.

In such case overall accomplishment level of system is also going to vary. In this way system and its taskes can have an arbitrary finite number of states. The system is termed as multistate system and its performance factor can be calculated as

$$E_G = \sum_{k=0}^{k} p_k G_k \quad (6)$$

Here in this framework, use of design tool EVENT STUDIO 2.5 has been done to design the system and generate inter task communication diagrams.

Figure 2 shows simplified task interaction diagram which is derived from

Figure 3 task interaction diagram generated by the EVENT STUDIO.

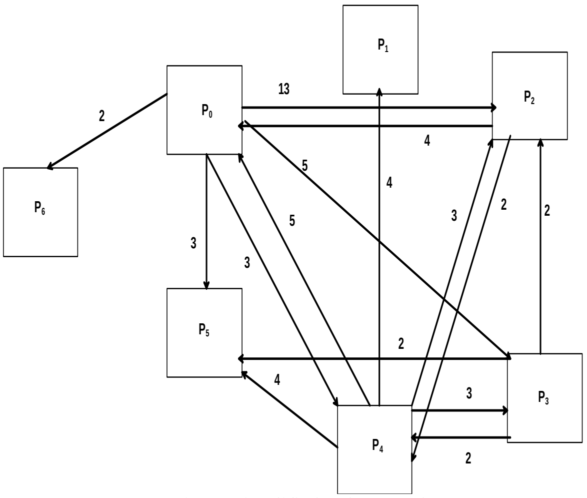


Fig. 2: Simplified task network

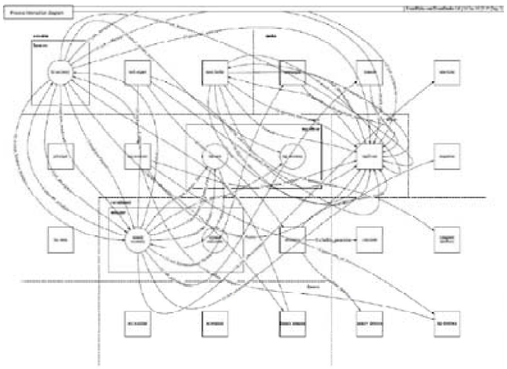


Fig. 3 Process inretaction diagram generated by event studio

L_{in} Total number of incoming links to a task.

L_t Total links associated with that task.

Now from figure 2, task P_6 is dependent on P_0 .Then P_6 is calculated as

$$p(P_6) = p(P_0)p(P_6|P_0) \quad (7)$$

$$p(P_0) = \frac{L_{in}}{L_t} \quad (8)$$

However here dependability of tasks is considered. Thus the conditional probability is calculated by the following equation.

$$p(P_6 \Big| P_0) = \frac{n(P_6 \cap P_0)}{n(P_0)} \quad (9)$$

If one task precedes another, then equation 7 is true. However, there are some tasks in task network which are dependent on two or more tasks. In the above example, P_4 is dependent on P_0, P3 and P_2 .Thus Bayes theorem is used to evaluate conditional probability as shown in equation 10.

$$p(P_4) = p(P_0)(P_4|P_0) + p(P_3)(P_4|P_3) + p(P_2)(P_4|P_2) \quad (10)$$

There is various possible relative accomplishment levels that characterize the performance of each task in the task network which depends on number of incoming links associated with that task. Various values of G_k are assumed depending upon $G_k L_{in}$.

Substituting values of p_k and G_k in equation 6, various MSS performance contribution factor E_G can be calculated. Scheduling policy for soft real time system is based on two parameters as shown in Table 3 and for hard real time system is as shown in Table 4. It is quite possible that values calculated will not be precise coming in one category.

B.Classification of taskes on PCF and deadline

P set of taskes.

C: set of classes.

pc: PCF of each task.

D: Relative deadline of each task.

n: Total number of tasks.

k: Total number of classes.

$P = \{P_1, P_{2,} P_3, P_4 \ldots\ldots P_n\}$

$C = \{C_1, C_{2,} C_3, C_4 \ldots\ldots C_n\}$

$$\{P_k \Rightarrow P_k \in C_1 | P_k C_P \wedge C_1 C_C, pc = High \wedge D_k = Low, n \geq k > 0\}$$

else

$$\{P_k \Rightarrow P_k \in C_2 | P_k C_P \wedge C_2 C_C, pc = High \wedge D_k = High, n \geq k > 0\}$$

else

$$\{P_k \Rightarrow P_k \in C_3 | P_k C_P \wedge C_3 C_C, pc = Low \wedge D_k = Low, n \geq k > 0\}$$

else

$$\{P_k \Rightarrow P_k \in C_4 | P_k C_P \wedge C_4 C_C, pc = Low \wedge D_k = High, n \geq k > 0\} \quad (12)$$

Table III: Classification of tasks for soft real time system

Priority Levels	PCF(E_G)	Deadline(D)
Class-I	High	Low
Class-II	High	High
Class-III	Low	Low
Class-IV	Low	High

Table IV: Classification of tasks for hard real time system

Priority Levels	PCF(E_G)	Deadline(D)
Class-I	High	Low
Class-II	Low	Low
Class-III	High	High
Class-IV	Low	High

C.Assumptions:

A1: Tasks are divided into mandatory and optional portion.

A2: Data required for transmission as to successor are tasked by mandatory portion.

A3: For scheduling dependent tasks, if predecessors are from class-I, then its mandatory as well as optional portion is executed. If predecessors are from other than class-I, tasks are scheduled only for mandatory portion.

A4: The semantics assumed is that one instance of all tasks should be executed for every period. This scheduler is of non pre-emptive type.

A5: It is assumed that, a task should get all the messages to complete its goal. If any of the messages is failed, then it is assumed that its accomplishment level reduces accordingly.

D.Suggested scheduling policy

```
For index1 = class1 to class4
 Index 2 =task1 to MAXtaskINDEX1
            num=calculate_preceded_task(index2);
              for index3=0 to num
if (preceded[index3].mandatory=0
then
        execute(preceded[index3].mandatory);
        setflag (preceded[index3].mandatory);
              endif
        endfor
              if(task[index2].mandatory=0 then
      execute                (task[index2].mandatory);
        setflag(task[index2].mandatory);
              endif
```

```
                if (task[index2].optional=0 then
        execute (task[index2].optional);
                setflag(task[index2].optional);
                endif
        endfor
endfor
```

However it has been observed that in PCD algorithm number of context switching and number of missing of deadline are high. Therefore another scheduling strategy suggested is αPCD.

E.Performance evaluation of algorithms

In order to evaluate performance of designed scheduling algorithms, simulation is done for many case studies with LDF, PCD and αPCD. A sample case study 1 for acyclic structure and case 2 for cyclic task structure is illustrated as below.

Case study 1: Following case study elaborates comparative performance of LDF, PCD and αPCD. The task scheduling attributes and its LDF scheduling is shown in the figure 6.

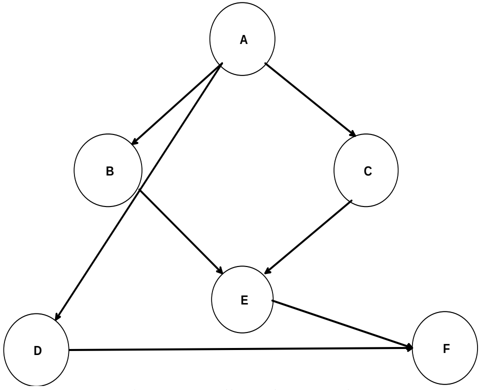
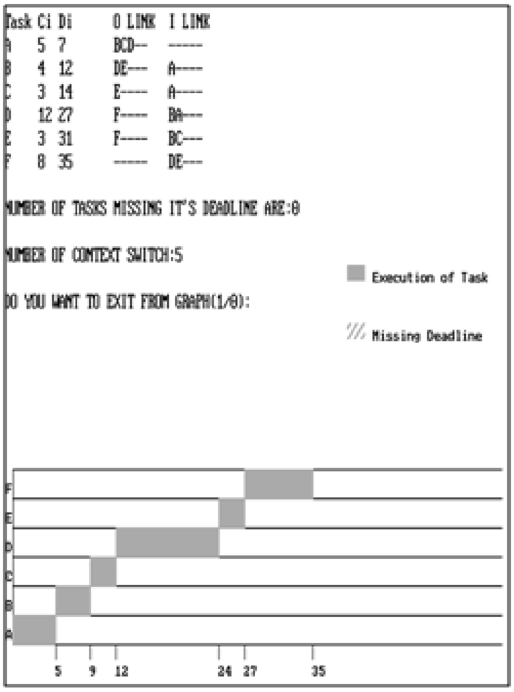


Fig. 4: Acyclic task network



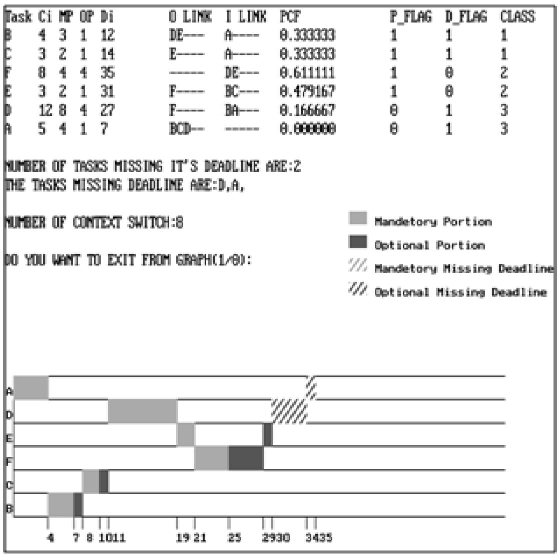
```
Task Ci Di      O LINK  I LINK
A    5  7       BCD--   -----
B    4  12      DE---   A----
C    3  14      E----   A----
D    12 27      F----   BA---
E    3  31      F----   BC---
F    8  35      -----   DE---

NUMBER OF TASKS MISSING IT'S DEADLINE ARE:0

NUMBER OF CONTEXT SWITCH:5

DO YOU WANT TO EXIT FROM GRAPH(1/0):
```

Fig. 5: LDF scheduling of case study1



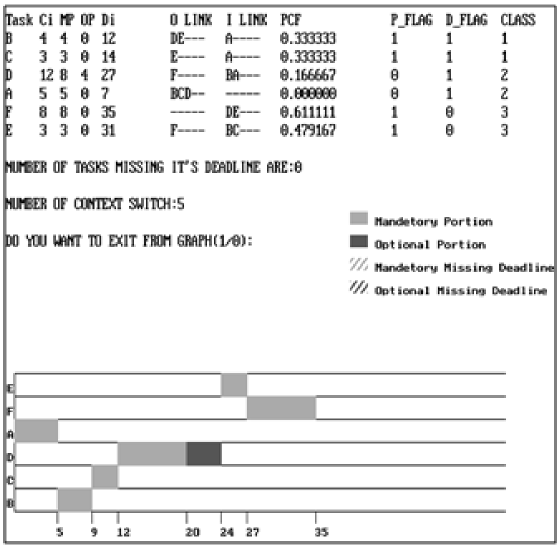
```
Task Ci MP OP Di        O LINK  I LINK  PCF             P_FLAG  D_FLAG  CLASS
B    4  3  1  12        DE---   A----   0.333333        1       1       1
C    3  2  1  14        E----   A----   0.333333        1       1       1
F    8  4  4  35        -----   DE---   0.611111        1       0       2
E    3  2  1  31        F----   BC---   0.479167        1       0       2
D    12 8  4  27        F----   BA---   0.166667        0       1       3
A    5  4  1  7         BCD--   -----   0.000000        0       1       3

NUMBER OF TASKS MISSING IT'S DEADLINE ARE:2
THE TASKS MISSING DEADLINE ARE:D,A,

NUMBER OF CONTEXT SWITCH:8

DO YOU WANT TO EXIT FROM GRAPH(1/0):
```

Fig. 6: PCD scheduling of case study1



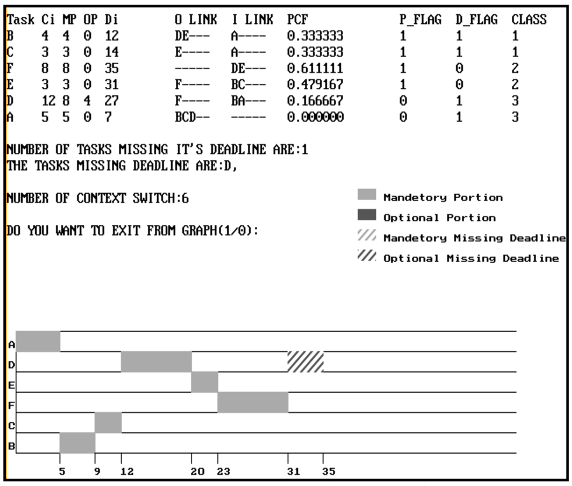
```
Task Ci MP OP Di        O LINK  I LINK  PCF             P_FLAG  D_FLAG  CLASS
B    4  4  0  12        DE---   A----   0.333333        1       1       1
C    3  3  0  14        E----   A----   0.333333        1       1       1
D    12 8  4  27        F----   BA---   0.166667        0       1       2
A    5  5  0  7         BCD--   -----   0.000000        0       1       2
F    8  8  0  35        -----   DE---   0.611111        1       0       3
E    3  3  0  31        F----   BC---   0.479167        1       0       3

NUMBER OF TASKS MISSING IT'S DEADLINE ARE:0

NUMBER OF CONTEXT SWITCH:5

DO YOU WANT TO EXIT FROM GRAPH(1/0):
```

Fig, 7: αPCD scheduling with deadline of case study1



```
Task Ci MP OP Di        O LINK  I LINK  PCF             P_FLAG  D_FLAG  CLASS
B    4  4  0  12        DE---   A----   0.333333        1       1       1
C    3  3  0  14        E----   A----   0.333333        1       1       1
F    8  8  0  35        -----   DE---   0.611111        1       0       2
E    3  3  0  31        F----   BC---   0.479167        1       0       2
D    12 8  4  27        F----   BA---   0.166667        0       1       3
A    5  5  0  7         BCD--   -----   0.000000        0       1       3

NUMBER OF TASKS MISSING IT'S DEADLINE ARE:1
THE TASKS MISSING DEADLINE ARE:D,

NUMBER OF CONTEXT SWITCH:6

DO YOU WANT TO EXIT FROM GRAPH(1/0):
```

Fig. 8: αPCD scheduling with performance of case study1

Performance of LDF scheduling algorithm is shown in the figure 5, where it does not miss any deadline. The number of context switching observed is 5. Considering performance of PCD scheduling algorithm as shown in the figure 6, the number of tasks missing their deadlines are 2 and context switching is 8 which is very high. The peculiarity of PCD scheduling algorithm is that it is considering two parameters, PCF and deadline and it is also scheduling both cyclic as well as acyclic tasks. On the contrary LDF is considering only deadline as a parameter of priority and supporting only acyclic task structure. PCD is more complex than LDF.

In order to remove these problems and improve the performance of PCD, modified scheduling strategy αPCD was developed. It has got two flavors. αPCD with deadline where performance is same like LDF as shown in the figure 8. Another flavor of αPCD is αPCD with performance, as shown in the figure 9. Here priority is given to performance due to which number of missing deadline is 1 and context switching is 6. Here although one task is missing its deadline, only its optional part of that task is missing its deadline. Comparative performance of LDF, PCD and αPCD is illustrated in Table 5. Comparative missing of deadlines performance of LDF, PCD and αPCD for acyclic task structure of two case studies is as shown in the figure 10. Similarly context switching performance for the same is elaborated in the figure 11.

Table V: Comparative performance of LDF, PCD and αPCD for acyclic tasks

Case Study No.	Framework		Context Switching	Number of tasks/taskes missed deadline
I	LDF		6	1
	PCD		9	3
	Alpha-PCD	Deadline	9	2
		Performability	9	3
II	LDF		5	0
	PCD		8	2
	Alpha-PCD	Deadline	5	0
		Performability	6	1

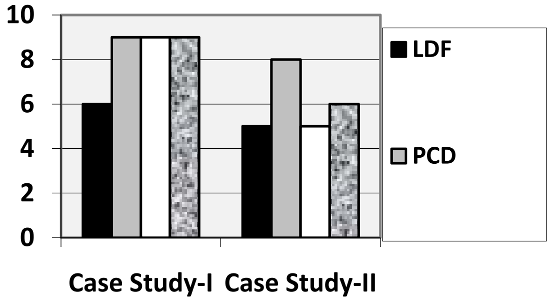


Fig. 9: Context switching performance of LDF, PCD and αPCD for acyclic tasks/taskes

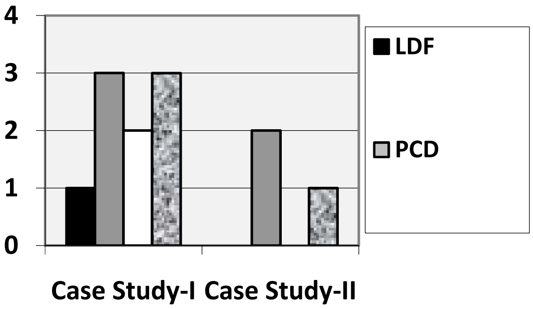


Fig. 10: Deadline missing performance of LDF, PCD and αPCD for acyclic tasks

Case study 2 is intended for cyclic tasks where performance of PCD, αPCD with deadline and αPCD with performance algorithm is evaluated as LDF does not support cyclic tasks.

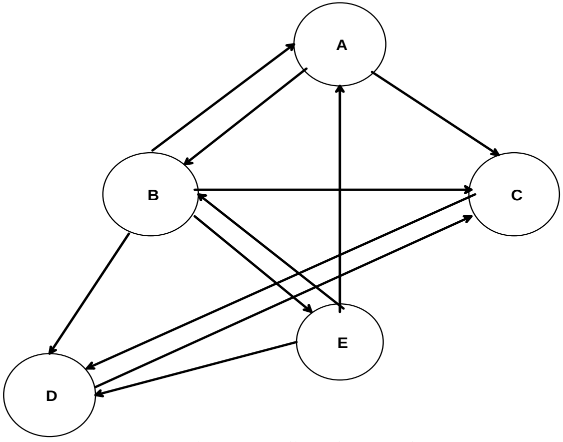
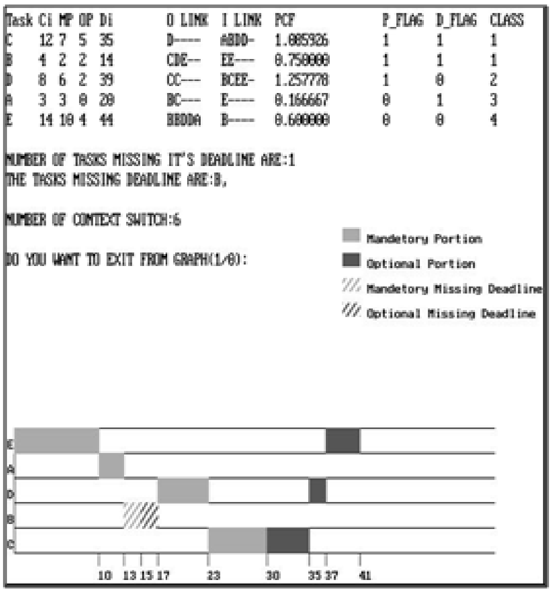


Fig..11: Cyclic task network



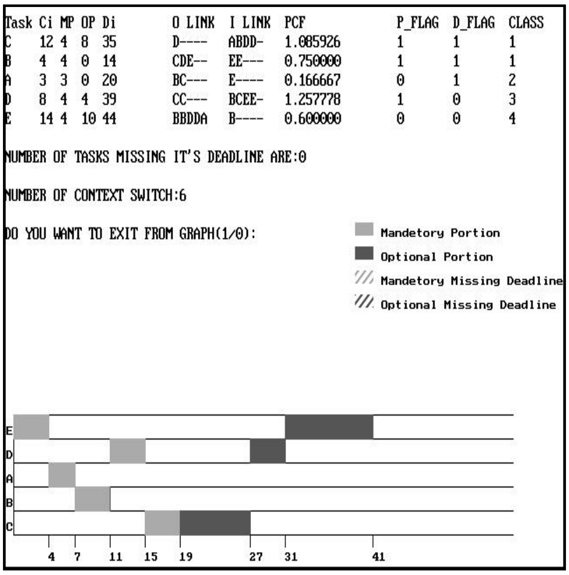
```
Task Ci MP OP Di       O LINK  I LINK  PCF           P_FLAG  D_FLAG  CLASS
C    12 7  5  35       D----   ABDD-   1.085926      1       1       1
B    4  2  2  14       CDE--   EE---   0.750000      1       1       1
D    8  6  2  39       CC---   BCEE-   1.257778      1       0       2
A    3  3  0  20       BC---   E----   0.166667      0       1       3
E    14 10 4  44       BBDDA   B----   0.600000      0       0       4

NUMBER OF TASKS MISSING IT'S DEADLINE ARE:1
THE TASKS MISSING DEADLINE ARE:B,

NUMBER OF CONTEXT SWITCH:6

DO YOU WANT TO EXIT FROM GRAPH(1/0):
```

Fig. 12: PCD scheduling of case study2



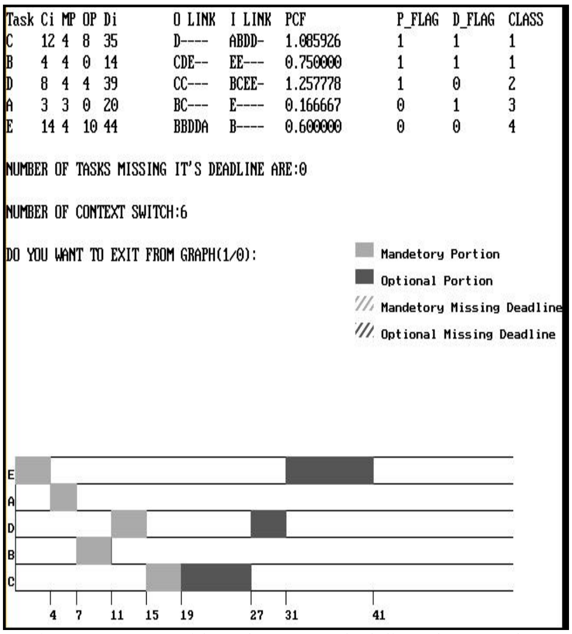
```
Task Ci MP OP Di        O LINK  I LINK  PCF           P_FLAG  D_FLAG  CLASS
C    12 4  8  35        D----   ABDD-   1.085926      1       1       1
B    4  4  0  14        CDE--   EE---   0.750000      1       1       1
A    3  3  0  20        BC---   E----   0.166667      0       1       2
D    8  4  4  39        CC---   BCEE-   1.257778      1       0       3
E    14 4  10 44        BBDDA   B----   0.600000      0       0       4

NUMBER OF TASKS MISSING IT'S DEADLINE ARE:0

NUMBER OF CONTEXT SWITCH:6

DO YOU WANT TO EXIT FROM GRAPH(1/0):
```

Fig. 13: αPCD with deadline scheduling of case study2



```
Task Ci MP OP Di        O LINK  I LINK  PCF           P_FLAG  D_FLAG  CLASS
C    12 4  8  35        D----   ABDD-   1.085926      1       1       1
B    4  4  0  14        CDE--   EE---   0.750000      1       1       1
D    8  4  4  39        CC---   BCEE-   1.257778      1       0       2
A    3  3  0  20        BC---   E----   0.166667      0       1       3
E    14 4  10 44        BBDDA   B----   0.600000      0       0       4

NUMBER OF TASKS MISSING IT'S DEADLINE ARE:0

NUMBER OF CONTEXT SWITCH:6

DO YOU WANT TO EXIT FROM GRAPH(1/0):
```

Figure 14: αPCD with performance scheduling of case study2

For given cyclic task structure as shown in figure 12, performance of PCD scheduling algorithm as shown in figure 13, number of tasks missing their deadline is 1 and context switching is 6. Modified scheduling strategy αPCD with deadline algorithm's performance for the same task structure is shown in the figure 14 where number of tasks missing their deadline are 0 and context switching is 6. αPCD with performance algorithm is as shown in the figure 15. Here no task is missing its deadline and context switching is also 6.

Comparative performance for various case studies for cyclic structure is illustrated in Table 6.Performance of PCD and αPCD is elaborated in figure 16 and figure 17.

Table VI: Comparative performance of PCD and αPCD for cyclic task / tasks

Case Study No.	Framework		Context Switches	Number of tasks missed deadline
III	PCD		7	1
	Alpha-PCD	Deadline	7	0
		Performability	7	0
IV	PCD		6	1
	Alpha-PCD	Deadline	6	0
		Performability	6	0

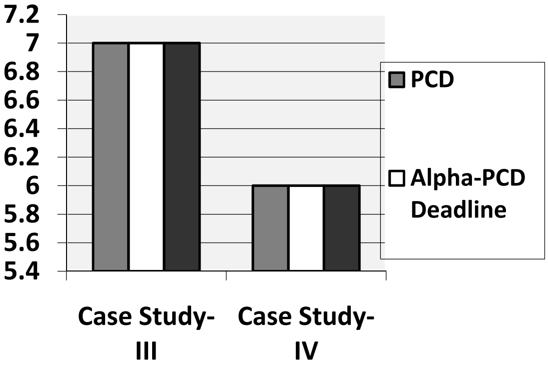


Fig. 15: Context switching performance of PCD and αPCD for cyclic tasks/taskes

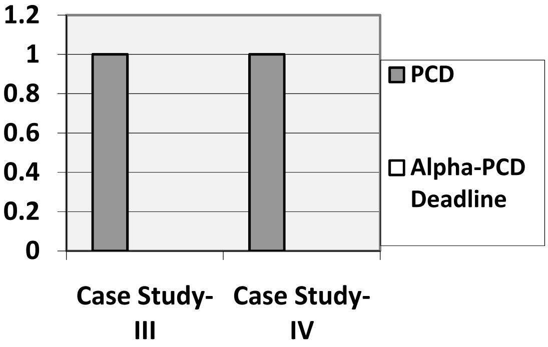


Fig. 16: Missing of deadline performance of PCD and αPCD for cyclic tasks

IV. CONCUSION

Novel scheduling algorithms are designed using MDE analysis. The important aspects of these algorithms are

- These are non preemptive offline precedence constraint task scheduling algorithms for uniprocessor architecture.
- Two parameters are taken into account to decide pseudo deadline for scheduling of tasks i.e. PCF and relative deadline.
- These support both cyclic as well as acyclic task structure.
- It gives optimal solution for around 75 % cyclic as well as acyclic structures.
- This can be used as a tool by newly entrant real time designer to view possible scheduling of taskes at design phase itself so that he has a lot

of scope to adjust period of invocation and deadline of taskes.

- This avoids further consequences and saves considerable cost and time of development of real time application

REFERENCES

[1] Liu C. and Layland J.W., "Scheduling algorithms for multiprogramming in a hard real-time environment," *Journal of ACM*, 20(1): pp.46–61, 1973.

[2] E.G. Coffman, "Computer and Job-Shop Scheduling Theory," *New York: John Wiley & Sons*, 1976.

[3] J.K. Lenstra and A.H.G.R. Kan, "Complexity of Scheduling Under Precedence Constraints," *Operations Research*, 26(1): pp. 23- 35, Jan. 1978.

[4] E.L. Lawler, "Deterministic and Stochastic Scheduling, "Recent Developments in Deterministic Sequencing and Scheduling: A Survey," *The Netherlands: Reidel*, *Dordrecht*, pp. 35-74, 1982.

[5] H. Kasahara and S. Narita, "Practical Multiprocessor Scheduling Algorithms for Efficient Parallel Processing," *IEEE Trans. Computers*,33(1):pp. 1023-1029, Nov. 1984.

[6] W.W. Chu and K. Leung, "Module Replication and Assignment for Real-Time Distributed Processing Systems," *Proc. IEEE*,75(5), pp. 547-562, May 1987.

[7] C.C. Shen and W.H. Tsai, "A Graph Matching Approach to Optimal Task Assignment in Distributed Computing Systems Using a Minimax Criterion, " *IEEE Trans. Computers*, 34(3), pp. 197- 203, Mar. 1985.

[8] P.Y.R. Ma et al., "A Task Allocation Model for Distributed Computing Systems," *IEEE Trans. Computers*,31(1): pp. 41-47, Jan. 1982.

[9] J.B. Sinclair, "Efficient Computation of Optimal Assignments for Distributed Tasks," *J. Parallel and Distributed Computing*, 4: pp. 342-362, 1987.

[10] W.H. Kohler and K. Steiglitz, "Computer and Job-Shop Scheduling Theory: Enumerative and Iterative Computational Approach," *John Wiley & Sons*, pp. 229-287, 1976.

[11] M. Alfano. A. Di-Stefano, L. Lo-Bello, O. Mirabella, and J.H. Stewman, "An Expert System for Planning Real-Time Distributed Task Allocation," *Proc. Florida AI Research Symposium*, Key West, Fla.,May 1996.

[12] P. Altenbernd, C. Ditze, P. Laplante, and W. Halang, "Allocation of Periodic Real-Time Tasks," *Proc. 20th IFAC/IFIP Workshop*, Fort Lauderdale, Fla., Nov. 1995.

[13] E.L.Lawler, "Optimal sequencing of a single machine subject to precedence constraints," *JSTOR management science*, 19(5), pp.544-546 ,1973.

[14] J.Sitakis, "Modeling real time systems-challenges and work directions," *proceedings of the international Conference on Embedded Software (EMSOFT 2001),* Springer, LNOS 2211, 2001.

[15] K.Alstisen, G.Goessler and J.Sitakis, "Scheduler modeling based on the controller synthesis paradigm," *Journal of real time system*, special issue on control approaches to Real Time computing.23:pp.55-84, 2002.

[16] P.Binns and S.Vesta, "Formalizing software architecture for embedded system," *proceeding of the first International Conference on embedded software(EMSOFT 2001),* Tohae city, Springer, 2001.

[17] V.Bertin, E.Closse, M.Poize atel, "Taxys = Esterel t Kronos, a tool for verifying real time properties of embedded system," *Proceedings the conference on Decision and control*, Dec-2001.

[18] J.Sitakis, S.Tripakis, S.Yovine, "Building models of real time system from application software," *Proceeding of the IEEE ,special issues on modeling and design of embedded* ,pp.100-111, Jan. 2003.

[19] Hong Pham, "Handbook of reliability engineering," *Springer publication*, pp.60-68, 2003.