Real Time Systems 2

Scheduling with precedence constraints

Lecture 3

Scheduling with precedence constraints

- The problem of finding an optimal schedule for a set of tasks with precedence relations is in general *NP*–hard.
- However, optimal algorithms that solve the problem in polynomial time can be found under particular assumptions on the tasks

Scheduling with precedence constraints

LATEST DEADLINE FIRST

- In 1973, Lawler presented an optimal algorithm that minimizes the maximum lateness ($L_i = f_i - d_i$) of a set of tasks with precedence relations and simultaneous arrival times.

- The algorithm is called ***Latest Deadline First*** **(LDF)** and can be executed in polynomial time with respect to the number of tasks in the set.

Scheduling with precedence constraints

LATEST DEADLINE FIRST (LDF)

- Given a set *J* of *n* tasks and a directed acyclic graph (DAG) describing their precedence relations, LDF builds the scheduling queue from tail to head: among the tasks without successors or whose successors have been all selected, LDF selects the task with the latest deadline to be scheduled last.

- This procedure is repeated until all tasks in the set are selected.

Scheduling with precedence constraints

- LATEST DEADLINE FIRST

At run time, tasks are extracted from the head of the queue, so that the first task inserted in the queue will be executed last, whereas the last task inserted in the queue will be executed first.

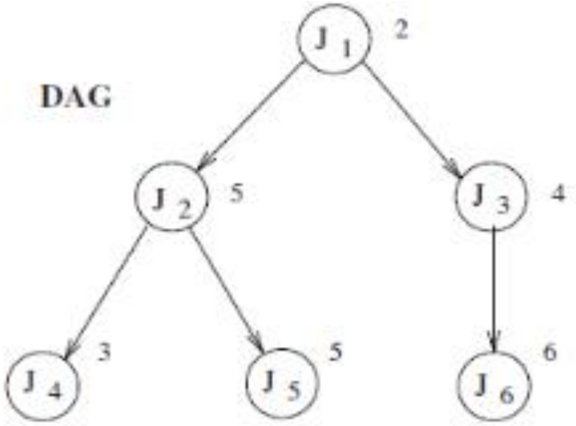
LATEST DEADLINE FIRST

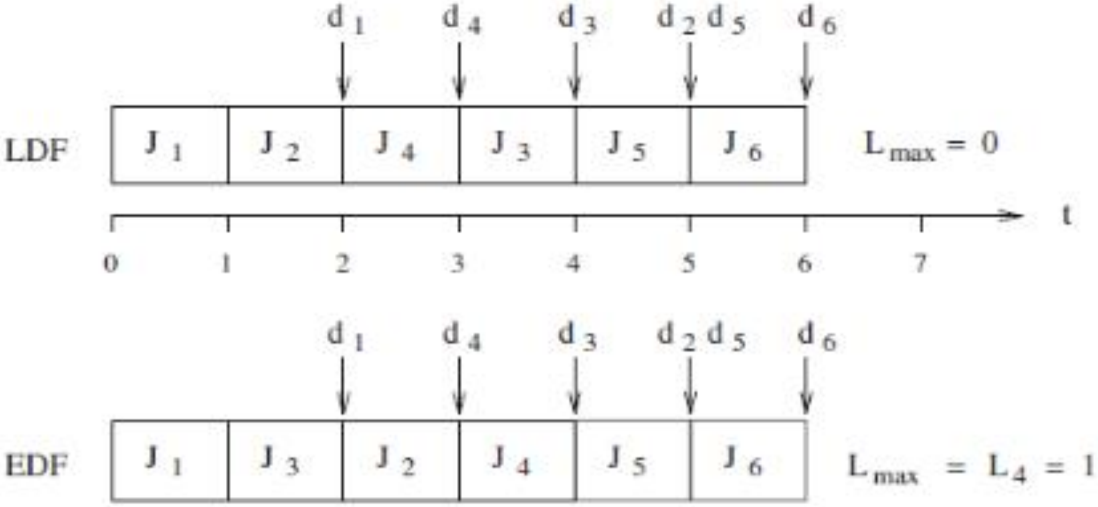
- Consider the example depicted in the following Figure, which shows the parameters of six tasks together with their precedence graph.
- The numbers beside each node of the graph indicate task deadlines. Figure also shows the schedule produced by EDF to highlight the differences between the two approaches.
- The EDF schedule is constructed by selecting the task with the earliest deadline among the current eligible tasks.
- Notice that EDF is not optimal under precedence constraints, since it achieves a greater *Lmax* with respect to LDF.

Scheduling with precedence constraints

LATEST DEADLINE FIRST

	J_1	J_2	J_3	J_4	J_5	J_6
C_i	1	1	1	1	1	1
d_i	2	5	4	3	5	6





LATEST DEADLINE FIRST

- Our goal is to minimize the maximum lateness.
- Lets calculate lateness

Job	Lateness Li=fi-di
J6	6-6=0
J5	5-5=0
J4	3-3=0
J3	4-4=0
J2	2-5=-3
J1	1-2=-1

See Maximum lateness =0 , means all of them are scheduled feasibly.

LATEST DEADLINE FIRST

- But if we use normal EDF (notice the example figure)

Job	Lateness Li=fi-di
J6	6-6=0
J5	5-5=0
J4	4-3=1
J3	2-4=-2
J2	3-5=-2
J1	1-2=-1

See Maximum lateness =1 for J4 means it can not meet deadline

Scheduling with precedence constraints

EDF WITH PRECEDENCE CONSTRAINTS

The problem of scheduling a set of *n* tasks with precedence constraints and dynamic activations can be solved in polynomial time complexity only if tasks are preemptable.

In 1990, Chetto, Silly, and Bouchentouf [CSB90] presented an algorithm that solves this problem in elegant fashion

Scheduling with precedence constraints

EDF WITH PRECEDENCE CONSTRAINTS

The basic idea of their approach is to transform a set J of dependent tasks into a set $J*$ of independent tasks by an adequate modification of timing parameters.

Then, tasks are scheduled by the Earliest Deadline First (EDF) algorithm. The transformation algorithm ensures that J is schedulable and the precedence constraints are obeyed if and only if $J*$ is schedulable.

Scheduling with precedence constraints

EDF WITH PRECEDENCE CONSTRAINTS

Basically, all release times and deadlines are modified so that each task cannot start before its predecessors and cannot preempt their successors.

EDF WITH PRECEDENCE CONSTRAINTS

- **MODIFICATION OF THE RELEASE TIMES**
- The rule for modifying tasks' release times is based on the following observation.
- Given two tasks *Ja* and *Jb*, such that $Ja \rightarrow Jb$ (that is, *Ja* is an immediate predecessor of *Jb*), then in any valid schedule that meets precedence constraints the following conditions must be satisfied :

- $sb \geq rb$ (that is, *Jb* must start the execution not earlier than its release time);
- $sb \geq ra + Ca$ (that is, *Jb* must start the execution not earlier than the minimum finishing time of *Ja*).

EDF WITH PRECEDENCE CONSTRAINTS

- Therefore, the release time *rb* of *Jb* can be replaced by the maximum between *rb* and ($ra + Ca$) without changing the problem. Let $r* b$ be the new release time of *Jb*.
- Then, $r*b = \max(rb, ra + Ca)$.

- The algorithm that modifies the release times:
- 1. For any initial node of the precedence graph, set $r*_i = ri$.
- 2. Select a task *Ji* such that its release time has not been modified but the release times of all immediate predecessors *Jh* have been modified. If no such task exists, exit.
- 3. Set $r*_i = \max[ri, \max(r* h + Ch : Jh \rightarrow Ji)]$.
- 4. Return to step 2.

EDF WITH PRECEDENCE CONSTRAINTS

- **MODIFICATION OF THE DEADLINES**
- The rule for modifying tasks' deadlines is based on the following observation.
- Given two tasks *Ja* and *Jb*, such that $Ja \rightarrow Jb$ (that is, *Ja* is an immediate predecessor of *Jb*), then in any feasible schedule that meets the precedence constraints the following conditions must be satisfied :
- $fa \leq da$ (that is, *Ja* must finish the execution within its deadline);
- $fa \leq db - Cb$ (that is, *Ja* must finish the execution not later than the maximum start time of *Jb*).

EDF WITH PRECEDENCE CONSTRAINTS

- Therefore, the deadline da of Ja can be replaced by the minimum between da and $(db - Cb)$ without changing the problem. Let $d{*}\, a$ be the new deadline of Ja. Then,
- $d{*}a = \min(da, db - Cb).$

EDF WITH PRECEDENCE CONSTRAINTS

- The algorithm that modifies the deadlines:
- 1. For any terminal node of the precedence graph, set $d^*_i = d_i$.
- 2. Select a task J_i such that its deadline has not been modified but the deadlines of all immediate successors J_k have been modified. If no such task exists, exit.
- 3. Set $d^*_i = \min[d_i, \min(d^*_k - C_k : J_i \rightarrow J_k)]$.
- 4. Return to step 2.

EDF WITH PRECEDENCE CONSTRAINTS

- Given seven tasks, A, B, C, D, E, F, and G, construct the precedence graph
- from the following precedence relations:
- $A \rightarrow C$
- $B \rightarrow C$ $B \rightarrow D$
- $C \rightarrow E$ $C \rightarrow F$
- $D \rightarrow F$ $D \rightarrow G$
- Then, assuming that all tasks arrive at time $r = 0$, have deadline D = 25, and
- computation times (Ci) 2, 3, 3, 5, 1, 2, 5, respectively.
- Modify their arrival times and deadlines to schedule them by EDF

Scheduling with precedence constraints

EDF WITH PRECEDENCE CONSTRAINTS

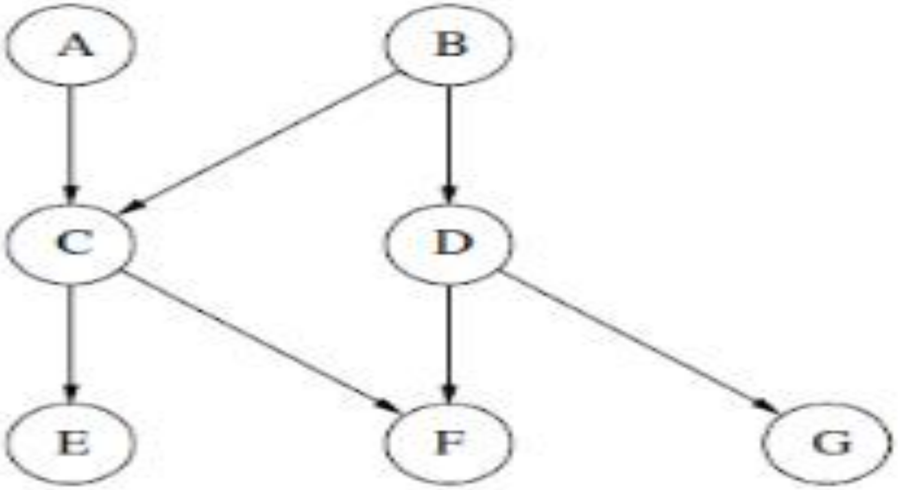


Figure 13.3 Precedence graph for Exercise 3.5.

	C_i	r_i	$r*_i$	d_i	$d*_i$
A	2	0	0	25	20
B	3	0	0	25	15
C	3	0	3	25	23
D	5	0	3	25	20
E	1	0	6	25	25
F	2	0	8	25	25
G	5	0	8	25	25

HW:

- Given DAG assuming that all tasks arrive at time $r = 0$, deadline D = 26, the number beside each node is its computation times (Ci).

- Modify their arrival times and deadlines to schedule them by EDF.
- Draw the Gantt chart for LDF.
- Calculate the lateness of two approaches.