Real-Time Programming

Lecture 9

Farhang Nemati

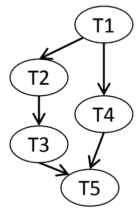
Spring 2016

Repetition

- Aperiodic Scheduling

Same Arrival Times	Different Arrival Times	
	Preemptive	Non-preemptive
•EDD	•EDF •LST	•Bratley •Spring

- Directed Acyclic Graph (DAG)



Scheduling with Precedence Constraints

- With same arrival times
- With arbitrary arrival times

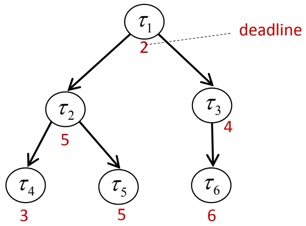
Scheduling with Precedence Constraints; Latest Deadline First (LDF)

- With same arrival times
- Algorithm:
  - Adding a task to a schedule queue:
    - Select from **tail to head**
    - Tasks without successors or those whose successors are selected, select the task with the latest deadline
  - To schedule then will be the reverse order of the queue; the task from the end of the queue will run first.
- Optimal

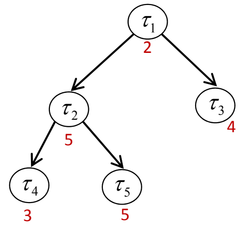
LDF; Example

- Schedule the following task set with LDF

	e_i	d_i
τ_1	1	2
τ_2	1	5
τ_3	1	4
τ_4	1	3
τ_5	1	5
τ_6	1	6

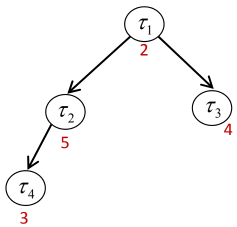


LDF; Example



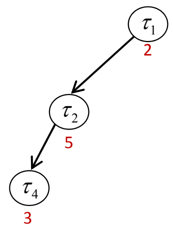
queue: τ_6

LDF; Example



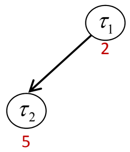
queue : τ_6 τ_5

LDF; Example



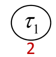
queue : τ_6 τ_5 τ_3

LDF; Example



queue : τ_6 τ_5 τ_3 τ_4

LDF; Example



queue : τ_6 τ_5 τ_3 τ_4 τ_2

LDF; Example

	e_i	d_i
τ_1	1	2
τ_2	1	5
τ_3	1	4
τ_4	1	3
τ_5	1	5
τ_6	1	6

queue : τ_6 τ_5 τ_3 τ_4 τ_2 τ_1 → **Schedule** : τ_1 τ_2 τ_4 τ_3 τ_5 τ_6 **It's feasible!**

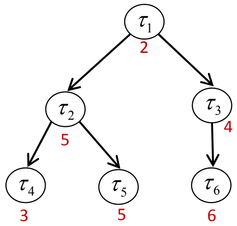
Scheduling with Precedence Constraints; with Same Arrival Times

- EDF
- Algorithm: Adding a task to a schedule queue:
  - Select from **root**
  - Among tasks that are without predecessor or their predecessors are already selected, select the task with the earliest deadline

EDF; Example

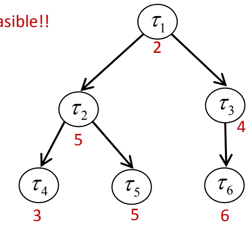
- Schedule the following task set with EDF

	e_i	d_i
τ_1	1	2
τ_2	1	5
τ_3	1	4
τ_4	1	3
τ_5	1	5
τ_6	1	6



EDF; Example

Schedule: τ_1 τ_3 τ_2 τ_4 τ_5 τ_6 NOT Feasible!!



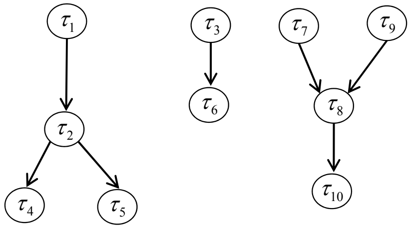
EDF with Precedence Constraints

- EDF with precedence constraints with same or arbitrary arrival times is **NOT** optimal!
- What to do?
  - Bratley's Algorithm
  - Spring Algorithm
- There is a better way

Scheduling with Precedence Constraints; with Arbitrary Arrival Times

- **EDF*** Algorithm
- Transform the dependent set of tasks into an independent task set by:
  - Modifying Arrival Times
  - Modifying Deadlines

Scheduling with Precedence Constraints; with Arbitrary Arrival Times



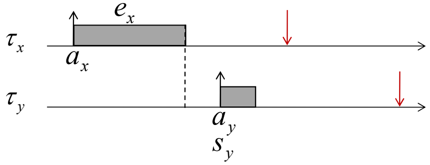
EDF*; Modification of Arrival Times

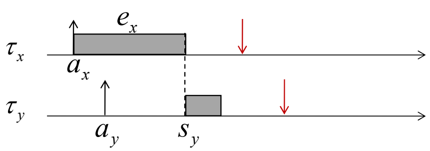
- Given two tasks τ_x and τ_y where $\tau_x \rightarrow \tau_y$ meaning that τ_y is an immediate successor of τ_x
  - The start time of τ_y (denoted by s_y) can not be earlier than its arrival time
  - τ_x should have enough time to finish before τ_y can start; s_y can not be earlier than minimum finishing time needed for τ_x

$$s_y \geq a_y$$

$$s_y \geq a_x + e_x$$

EDF*; Modification of Arrival Times





EDF*; Modification of Arrival Times

- The new arrival of task τ_y (start time) denoted by a^*_y is calculated as follows:

$$a^*_y = \max(a_y\ ,\ a_x + e_x)$$

EDF*; Modification of Arrival Times

- The algorithm for modifying all arrival times
  1. For each initial node in DAG set the arrival time

     $$a^*_i = a_i$$

  2. Select a task τ_j whose arrival time is not yet modified but the arrival times of all its immediate predecessors have been modified. If such task does not exist, exit the algorithm
  3. Modify the arrival time of τ_j

     $$a^*_j = \max(a_j \ , \ \max(a^*_h + e_h : \tau_h \rightarrow \tau_j)$$

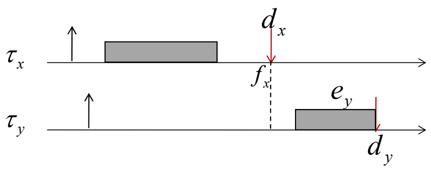
  4. Continue from step 2

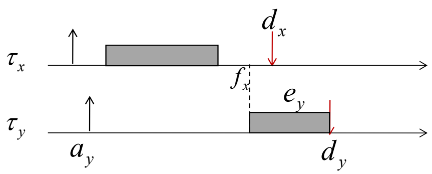
EDF*; Modification of Deadlines

- Given two tasks τ_x and τ_y where $\tau_x \rightarrow \tau_y$ meaning that τ_y is an immediate successor of τ_x
  - The finishing time of τ_x can not be later than its deadline
  - τ_y should have enough time to finish after τ_x is finished; f_x can not finish later than the latest time that τ_y can start

$$f_x \leq d_x$$
$$f_x \leq d_y - e_y$$

EDF*; Modification of Arrival Times





EDF*; Modification of Deadlines

- The new deadline of task τ_x (start time) denoted by d^*_x is calculated as follows:

$$d^*_x = \min(d_x \ , \ d_y - e_y \ )$$

EDF*; Modification of Deadlines

- The algorithm for modifying all deadlines
  1. For each terminal node in DAG set the deadline

     $$d^*_i = d_i$$

  2. Select a task τ_j whose deadline is not yet modified but the deadlines of all its immediate successors have been modified. If such task does not exist, exit the algorithm
  3. Modify the deadline of τ_j

     $$d^*_j = \min(d_j\ ,\ \min(d^*_k - e_k : \tau_j \to \tau_k))$$

  4. Continue from step 2

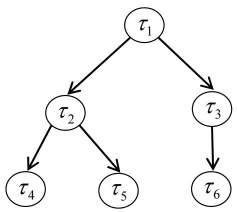
EDF*

- After modifying all arrival times and deadlines the task set has been transformed into a task set where there is no precedence constraints anymore, i.e., the tasks with the new arrival times and deadlines are independent.
- Now EDF can be used with the transformed task set

Modify Arrival Times and Deadlines; Example

- Schedule the following task set with EDF

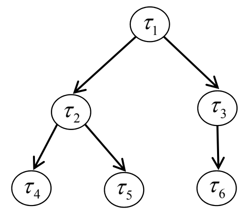
	a_i	e_i	d_i
τ_1	0	1	2
τ_2	1	1	5
τ_3	0	1	4
τ_4	2	1	3
τ_5	1	1	5
τ_6	0	1	6



Modify Arrival Times and Deadlines; Example

- Schedule the following task set with EDF

	a_i	e_i	d_i	a^*_i	d^*_i
τ_1	0	1	2	0	**1**
τ_2	1	1	5	1	**2**
τ_3	0	1	4	**1**	4
τ_4	2	1	3	2	3
τ_5	1	1	5	**2**	5
τ_6	0	1	6	**2**	6



Aperiodic Scheduling Algorithms; Summary

	Same Arrival Times	Different Arrival Times	
		Preemptive	Non-preemptive
Independent	•EDD	•EDF •LST	•Bratley •Spring
Precedence Constraints	•LDF	•EDF*	•Spring