Real-time Systems

Week 5:
Aperiodic real time scheduling

Ngo Lam Trung

Dept. of Computer Engineering

Contents

- Notation of scheduling algorithms
- Aperiodic scheduling algorithms
  - Jackson's algorithm
  - Horn's algorithm
  - Bratley's algorithm
- Scheduling with precedence constraints

Notation of scheduling algorithms

- A scheduling algorithm involves in a scheduling problem with
  - A class of task set,
  - An optimality criterion, and
  - A machine environment.

- Systematic notation of scheduling algorithms:
  - $\alpha \mid \beta \mid \gamma$
  - α: machine environment
    - Uniprocessor, multiprocessor, distributed architecture etc
  - β: tasks & resource characteristics
    - preemptive, independent, precedence constrained, synchronous activations …
  - γ: optimality criterion

Notation of scheduling algorithms

- Examples
  - $1|prec|L_{max}$
    - Minimizes the maximum lateness of a task set with precedence constraints on a uniprocessor machine.
  - $3|no\_preem|\Sigma f_i$
    - Minimizes the sum of finishing times of a non-preemptive task set on a 3 processor machine.
  - $2|sync|\Sigma Late_i$
    - Minimizes the number of late tasks of a task set with synchronous arrival times on a 2 processor machine.

Jackson's algorithm (1|*sync*|L_{max})

- A uniprocessor scheduling algorithm of a task set with synchronous arrival times
  - All tasks have the same arrival time

- For synchronous arrival times, the scheduling problem only considers execution times & deadlines
  - Task set notation:

$$\mathcal{J} = \{J_i(\underline{C_i}, \underline{D_i}),\ \ i = 1, \ldots, n\}$$

Jackson's rule

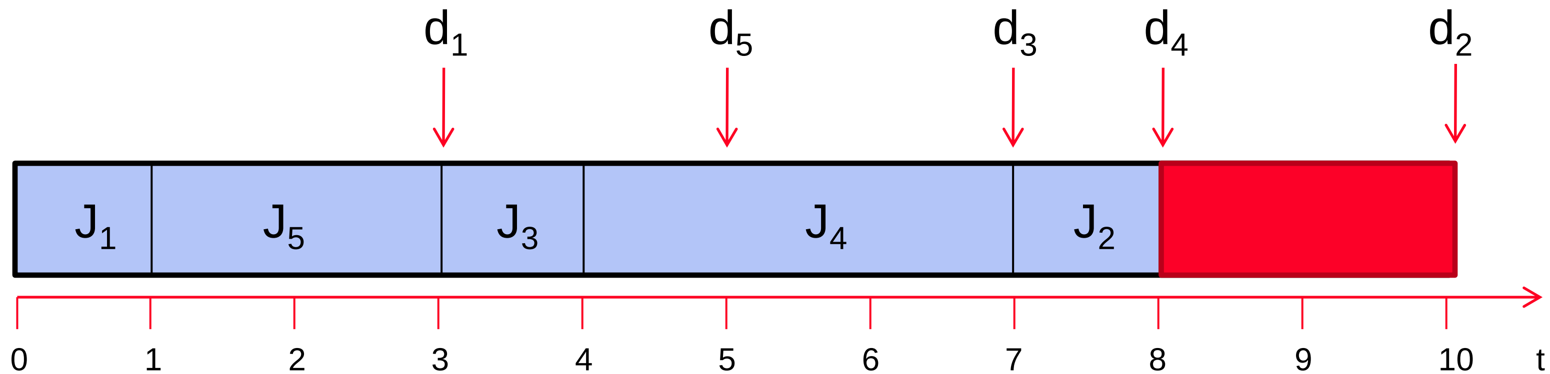
- **Theorem 3.1** (Jackson's rule)
  - Given a set of *n* independent tasks (i.e. no resource & precedence constraints), any algorithm that executes the tasks in order of non-decreasing deadline is optimal with respect to minimizing the maximum lateness.

➡ Earliest Due Date (EDD)

Examples

- Example 1

	J_1	J_2	J_3	J_4	J_5
C_i	1	1	1	3	2
d_i	3	10	7	8	5

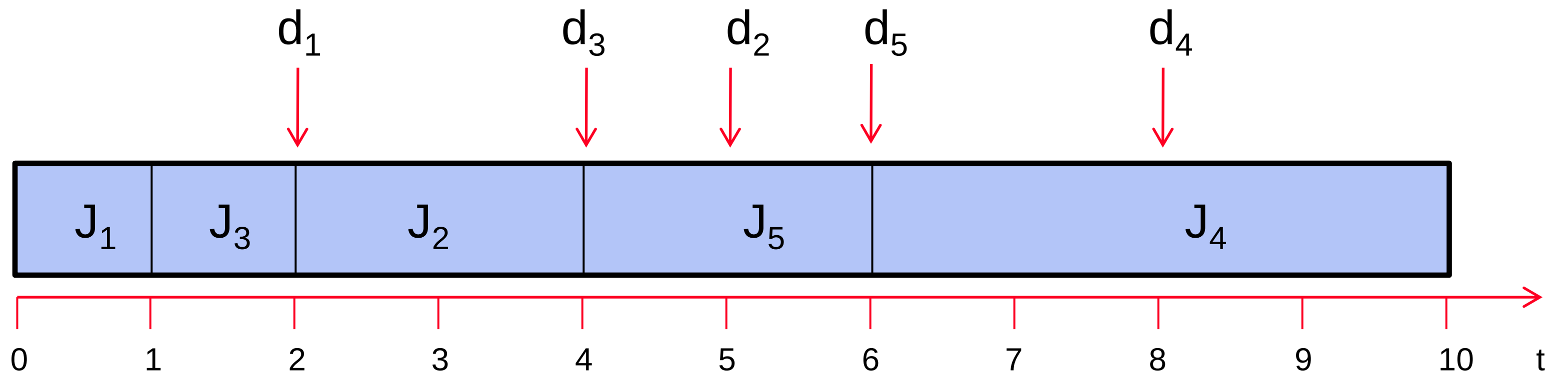


a feasible schedule produced by Jackson's algorithm

Examples

- Example 2

	J_1	J_2	J_3	J_4	J_5
C_i	1	2	1	4	2
d_i	2	5	4	8	6

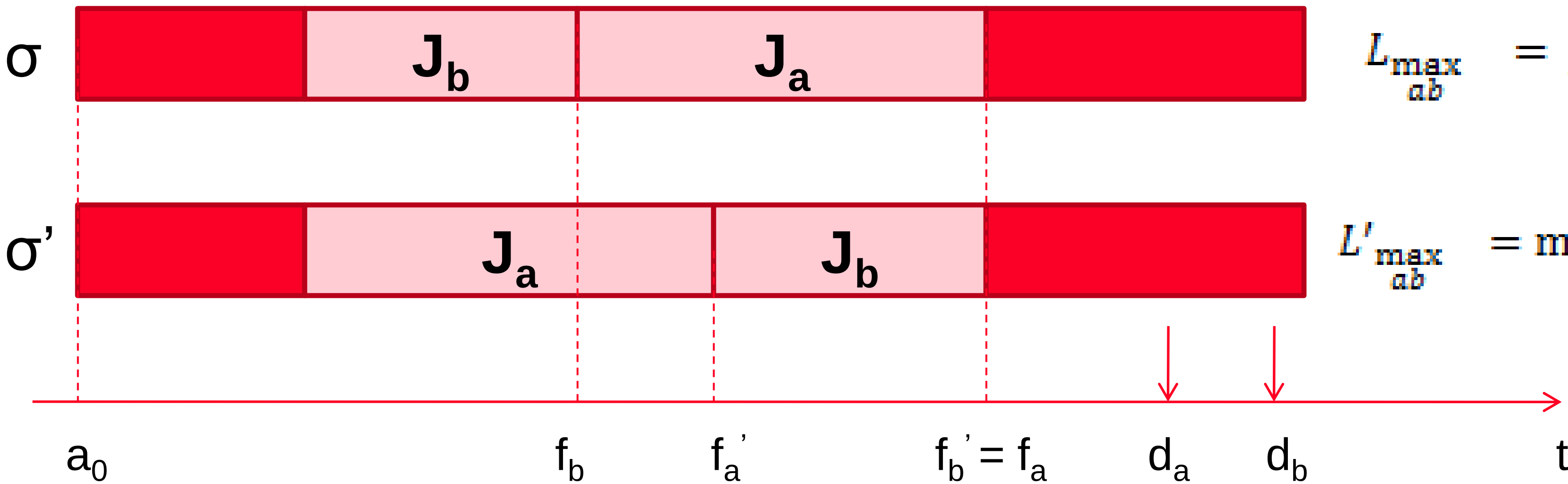


an infeasible schedule produced by Jackson's algorithm

Proof: Jackson's

- Jackson's theorem can be proved by a simple interchange argument.
- Let σ be a schedule produced by algorithm A.
- if A is different than EDD, then there exist 2 tasks J_a and J_b with $d_a < d_b$ such that J_b immediately precedes J_a in σ
- Let σ' be a schedule obtained from σ by interchanging J_a and J_b, so that J_a immediately precedes J_b in σ'
- We'll show $L_{max}(ab) > L'_{max}(ab)$

Proof: Jackson's theorem



$$L_{\max_{ab}} = f_a - d_a$$

$$L'_{\max_{ab}} = \max\left(L'_a, L'_b\right)$$

$$if\ (L'_a \geq L'_b)\ then\ L'_{\max_{ab}} = f'_a - d_a < f_a - d_a$$

$$if\ (L'_a \leq L'_b)\ then\ L'_{\max_{ab}} = f'_b - d_b < f_a - d_a$$

Guarantee

- Guarantee test:
  - All tasks can complete before their deadlines
  - The worst-case finishing time f_i is less than or equal to its deadline d_i

$$\forall i = 1, \ldots, n \;\; f_i \leq d_i.$$

  - Suppose a set of task J_1, J_2, J_3 … J_n is listed by increasingly deadline, thus in the worst-case finishing time f_i

$$f_i = \sum_{k=1}^{i} C_i$$

If $d_1 < d_2 < \ldots < d_n$, the feasibility condition under EDD is given by checking n conditions

$$\forall i = 1, \ldots, n \;\; \sum_{k=1}^{i} C_k \leq d_i.$$

- What if the task set is not synchronous?

→ Horn's algorithm (1|*preem*|L_{max})

Horn's theorem

Theorem 3.2

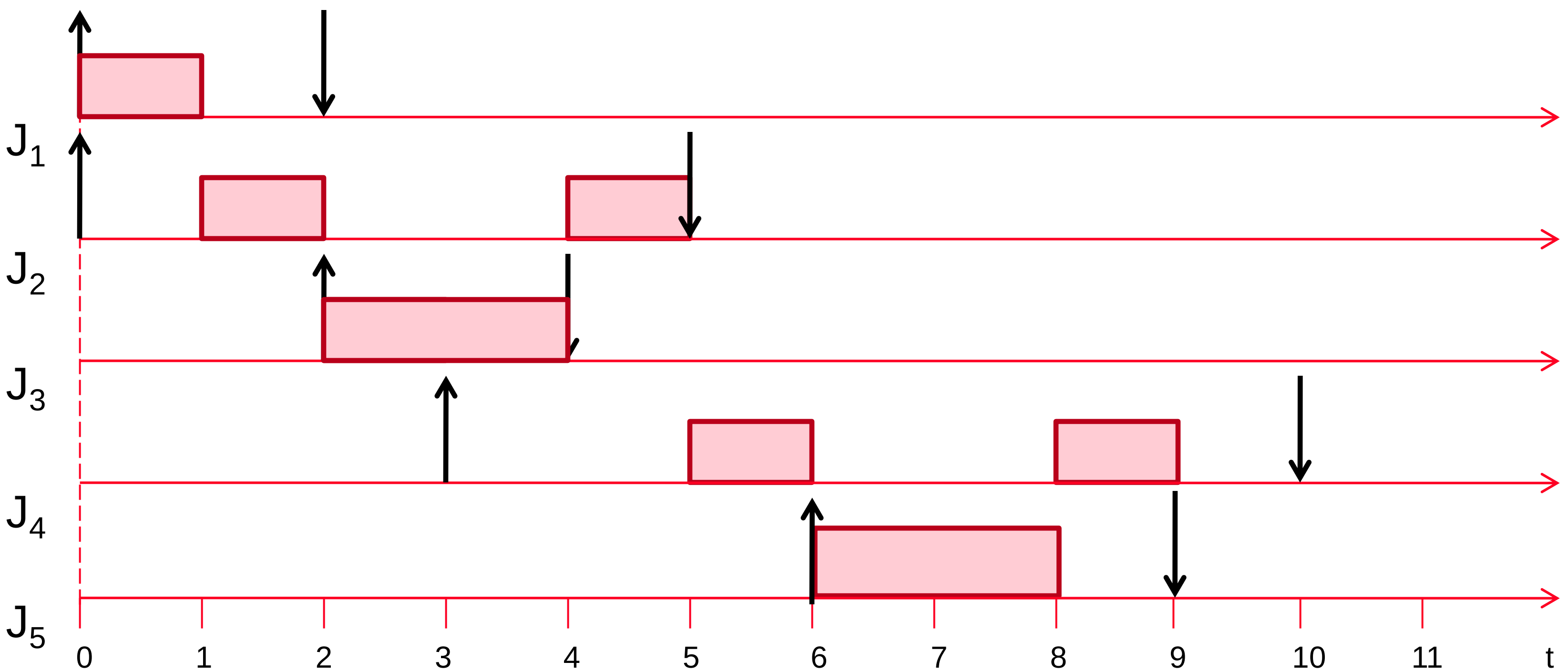
- Given a set of n independent tasks with arbitrary arrival times, any algorithm that at any instant executes the task with the earliest absolute deadline among all the ready tasks is optimal with respect to minimizing the maximum lateness.

→ Earliest deadline first (EDF)

Time complexity of EDF: $O(n^2)$

Example

	J_1	J_2	J_3	J_4	J_5
a_i	0	0	2	3	6
C_i	1	2	2	2	2
D_i	2	5	4	10	9

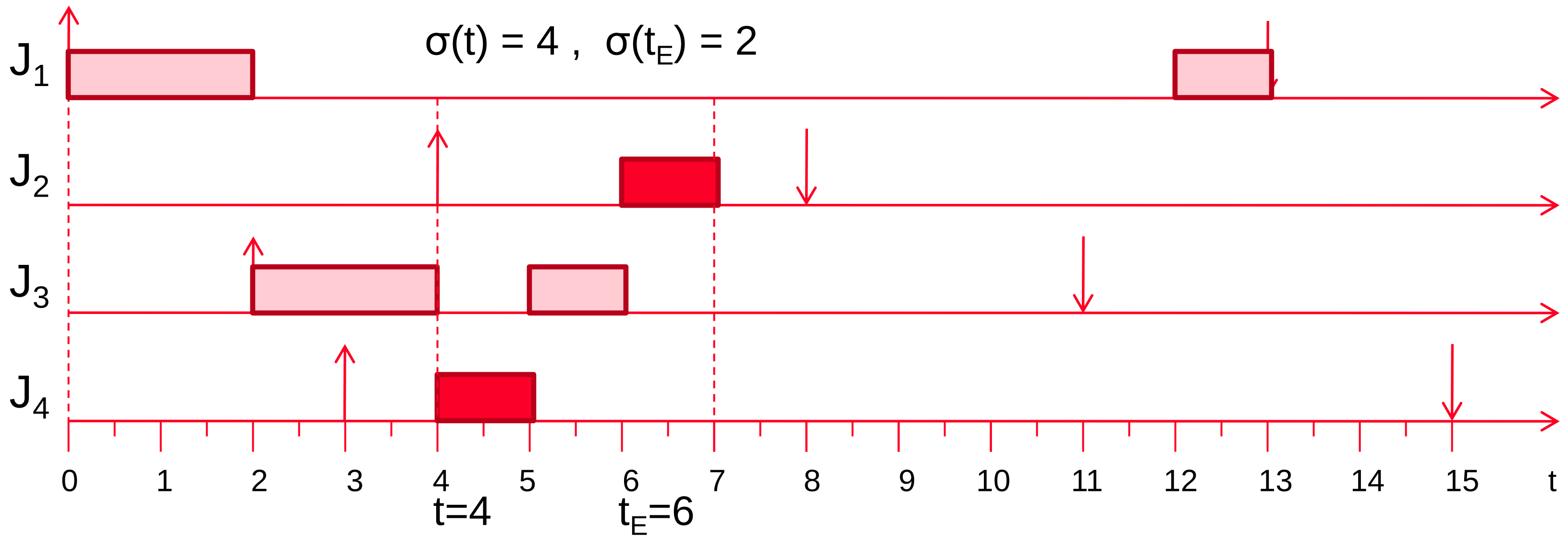


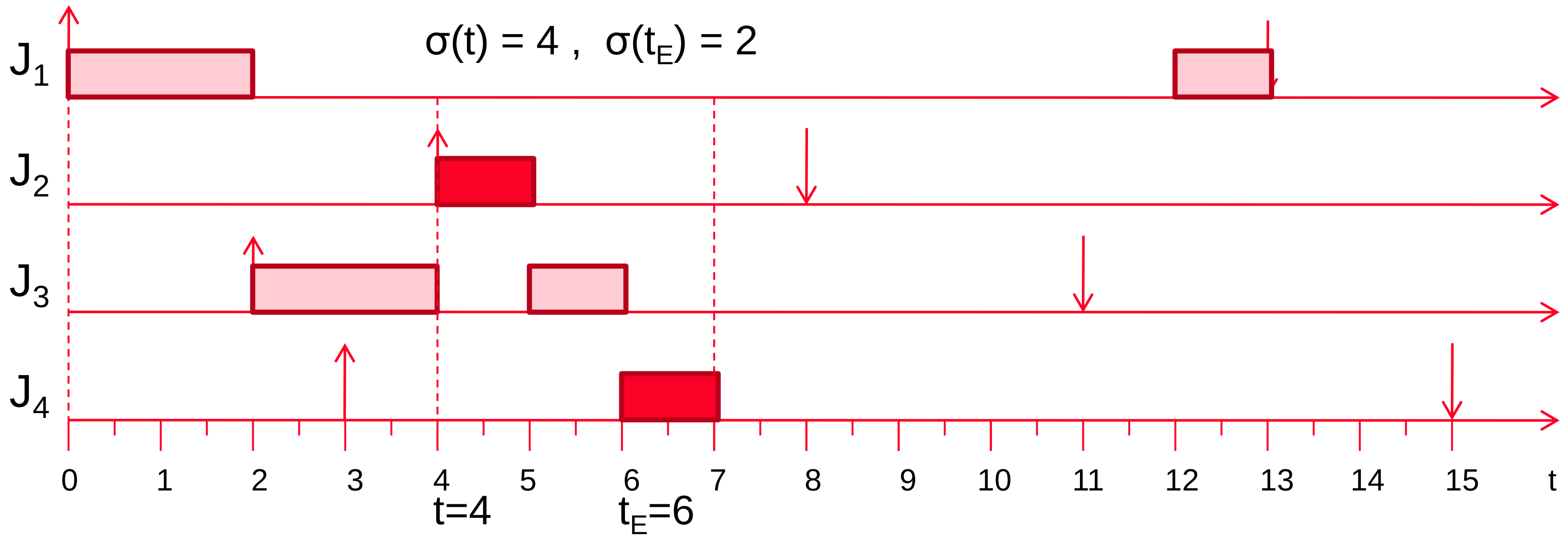
EDF optimality

- σ: a schedule generated by algorithm *A*
- σ_{EDF}: a schedule generated by EDF (not equal to σ)

- $\sigma(t)$:
  - the task executing in the time slice [t,t+1) in σ
- $E(t)$:
  - the ready task with the earliest deadline at time *t*
- $t_E(t)$:
  - the time ($\geq$t) at which the next slice of task E(t) begins its execution in the current schedule

EDF optimality





- $L_{max} = \max(f2-d2, f4-d4)$
- $L'_{max} = \max(f'2-d2, f'4-d4)$
- $d2<d4$
- $f'2 < f2$
- In the worst case when σ(t) are the time slice that each task finishes
  - $f2 = f'4$
- $f'2-d2 < f2-d2$
- $f'4-d4 = f2-d4 < f2-d2$
- ➔ $L'_{max} < L_{max}$

Guarantee

- Guarantee test has to be done dynamically, whenever a new task enters the system.
- Given new task $_{\text{Jnew}}$ arrives at *t*.
- The new task set (including J_{new}) is listed with increasing deadline
- Guarantee test:
  - The worst-case finishing time f_i is less than or equal to its deadline d_i
  - C_i(t): remaining worst-case execution time of task J_i at time t

$$\forall i = 1, \ldots, n \; \sum_{k=1}^{i} C_k(t) \leq d_i.$$

Non-preemptive scheduling

- Scheduling problem of finding a feasible schedule of non-preemptive tasks
  - NP-hard
- When arrival times are known a priori, the scheduling problems are treated by branch-and-bound algorithms.
  - Bratley's algorithm
  - Spring kernel

Bratley's algorithm (1|*no_preem*|*feasible*)

- Finds a feasible schedule of non-preemptive tasks

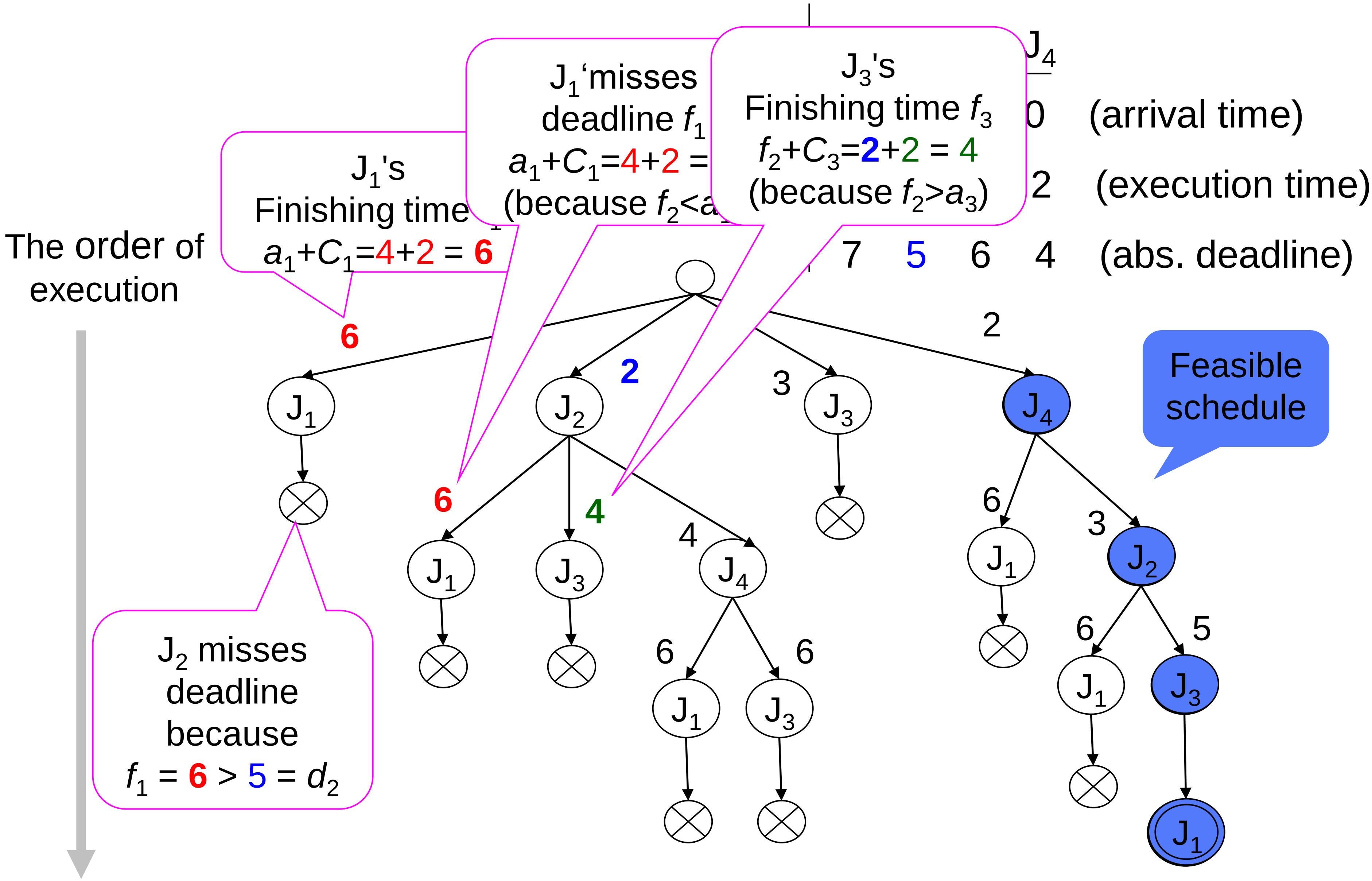
- Start with empty schedule

- Present all possible schedules with a tree-structure

- Calculate finishing time of each task considering the finishing time of the predecessor, the arrival time, & the execution time

- Finds a path of a feasible schedule

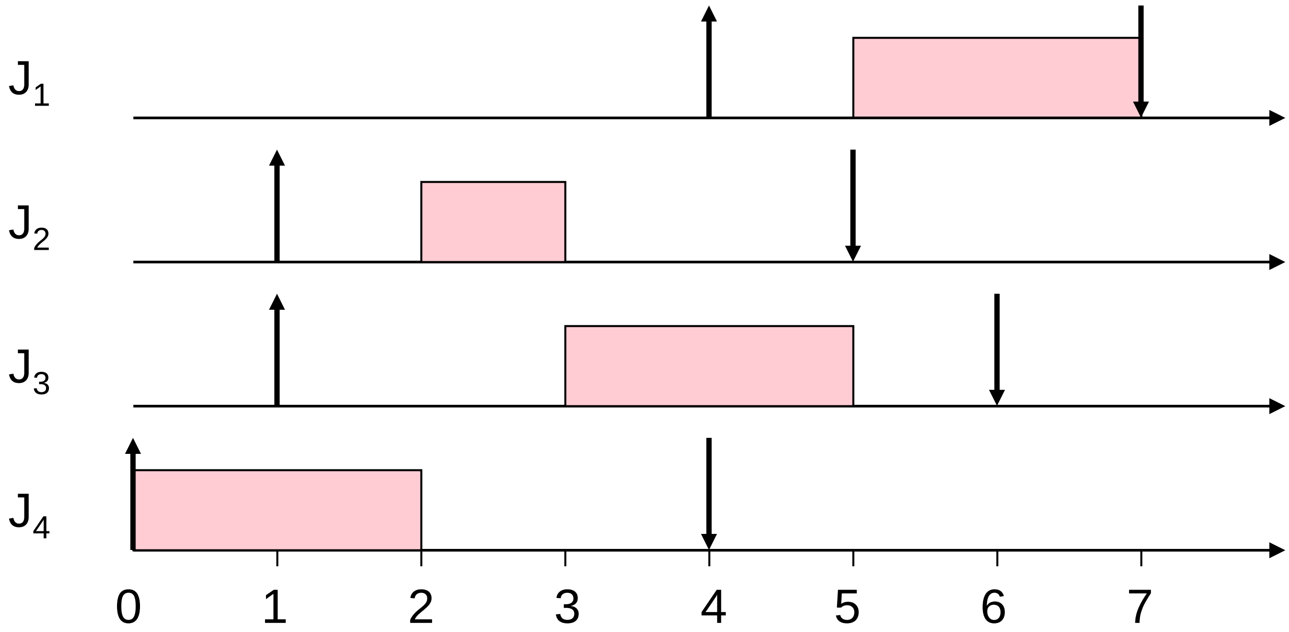
Bratley's algorithm (1|no_preem|feasible)



Bratley's algorithm (1|*no_preem*|*feasible*)

- Found a feasible schedule:
  - $J_4 \rightarrow J_2 \rightarrow J_3 \rightarrow J_1$

	J_1	J_2	J_3	J_4
a_i	4	1	1	0
C_i	2	1	2	2
d_i	7	5	6	4



Scheduling with precedence constraints

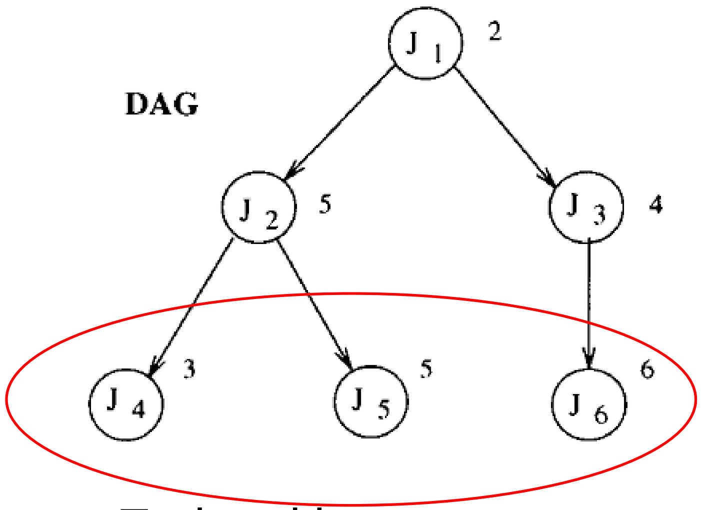
- NP-hard problem
- Applying assumptions
  - Synchronous activation: LDF algorithm
  - Pre-emptive tasks set: EDF with precedence constraints

Latest Deadline First (LDF) *(1|prec, sync|L_{max})*

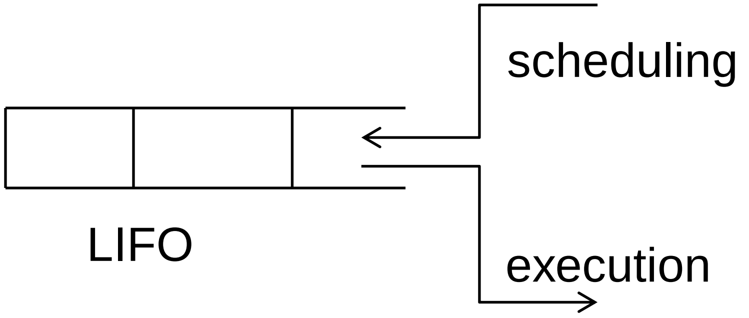
- Given a task set J and a directed acyclic graph representing its precedence constraints

- Scheduling algorithm

  - Select the subtask J' of J with no successor
  - In J', select the task j with the latest deadline
  - Put j into a LIFO queue
  - Repeat until all tasks are put into the LIFO queue
  - The tasks in the queue will be pop out for execution

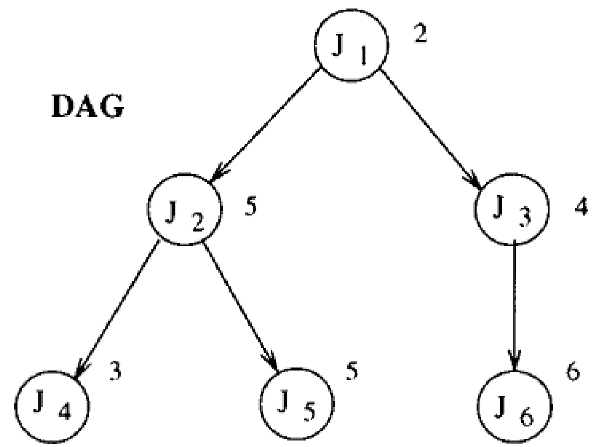


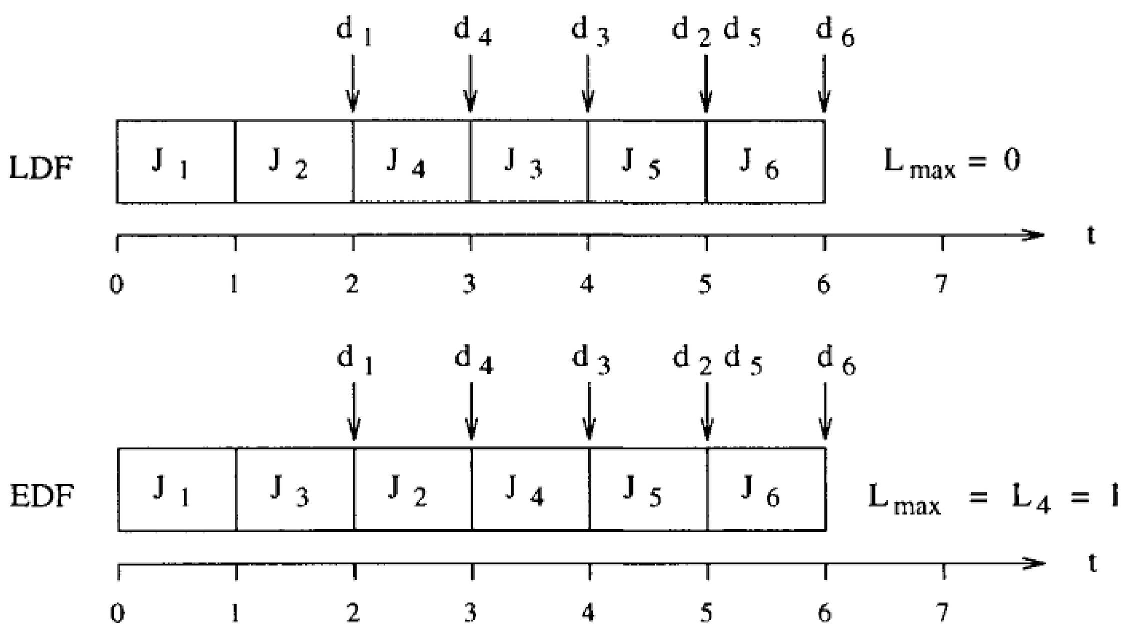
Tasks with no successor



Example

	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





➔ LDF achieves smaller L_{max}

➔ EDF is not optimal under precedence constraints

LDF optimality

- Let J be the complete set of tasks to be scheduled
- $\Gamma \in J$ be the subset of tasks without successors
- J_i be the task in Γ with the latest deadline d_i.
- J_k be the last scheduled task
- If the schedule σ does not follow the LDF rule then

$$d_k < d_i$$

- Partition Γ into four subsets

$$\Gamma = A \cup \{J_i\} \cup B \cup \{Jk\}$$

- We'll show moving J_i to the end of the schedule will not increase L_{max}, thus show LDF's optimality

LDF optimality

- Partition Γ with original schedule σ

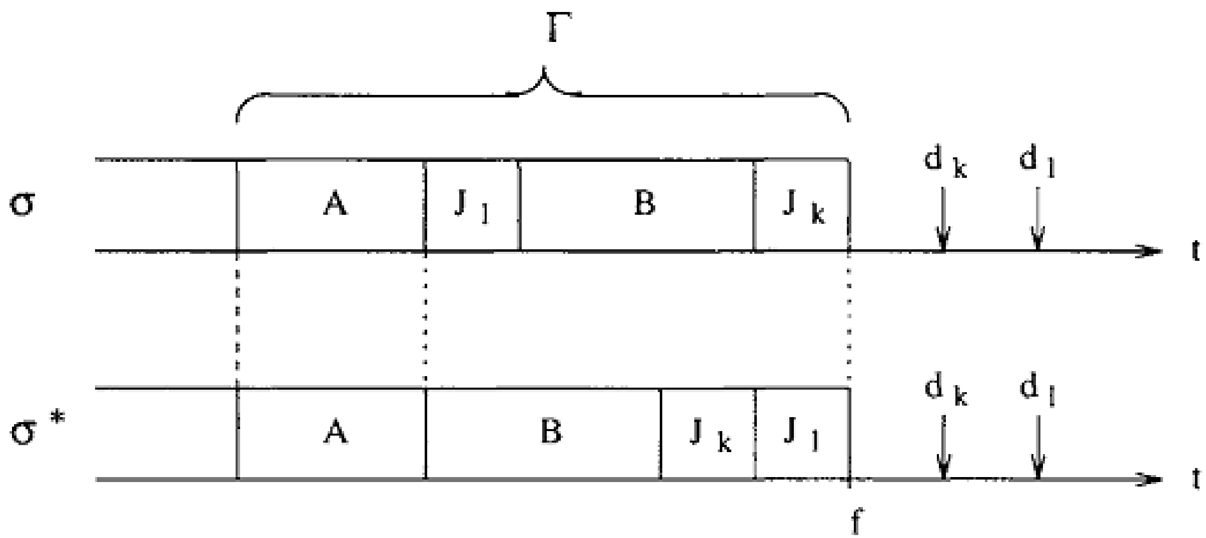
$$\Gamma = A \cup \{J_i\} \cup B \cup \{J_k\}$$

$$L_{max}(\Gamma) \geq f - dk$$

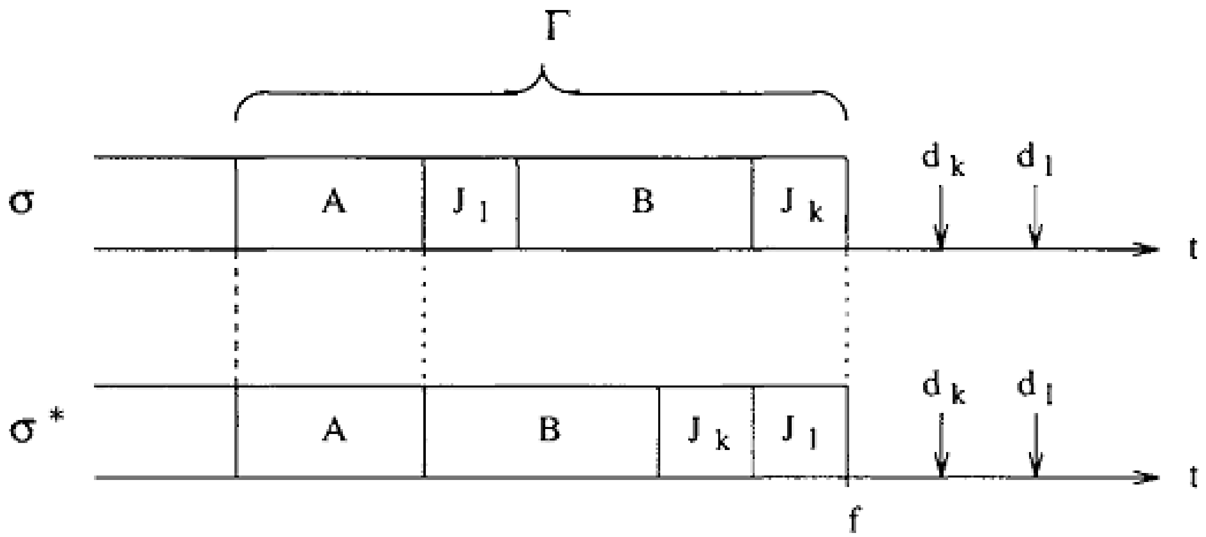
- Schedule σ^* obtained by moving J_i to the end of the schedule

$$\Gamma = A \cup B \cup \{Jk\} \cup \{J_i\}$$

$$L^*_{max}(\Gamma) = \max[L^*_{max}(A),\ L^*_{max}(B),\ L^*_k,\ L^*_l].$$



LDF optimality



$L_{max}(\Gamma) \geq f - dk$

$$L^*_{max}(\Gamma) = \max[L^*_{max}(A),\ L^*_{max}(B),\ L^*_k,\ L^*_l].$$

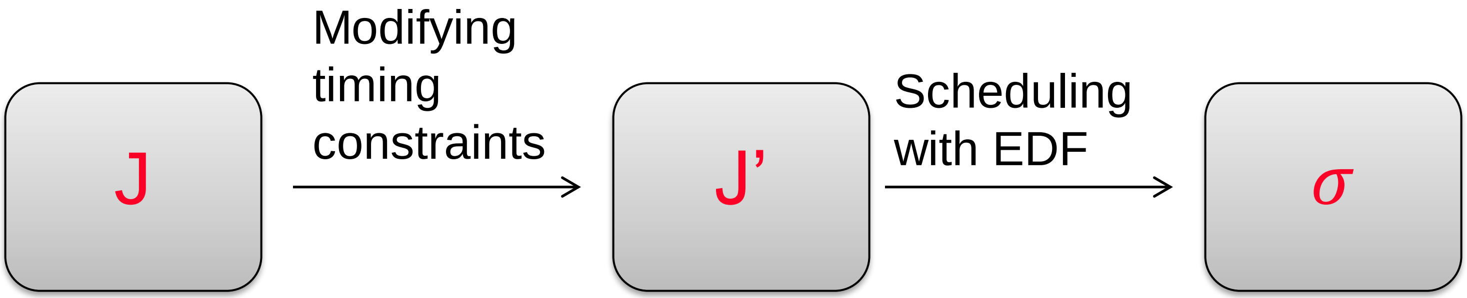
- $L^*_{max}(A) = L_{max}(A) \leq L_{max}(\Gamma)$ because A is not moved;
- $L^*_{max}(B) \leq L_{max}(B) \leq L_{max}(\Gamma)$ because B starts earlier in σ^*;
- $L^*_k \leq L_k \leq L_{max}(\Gamma)$ because task J_k starts earlier in σ^*;
- $L^*_l = f - d_l \leq f - d_k \leq L_{max}(\Gamma)$ because $d_k \leq d_l$.

➔ $L^*_{max}(\Gamma) \leq L_{max}(\Gamma)$

EDF with precedence constraints

- Scheduling problem: $(1|prec, preem|L_{max})$
- Difficulty: EDF is not optimal with precedence constraints
- Provided task $J_a, J_b \in J$ that $J_a \rightarrow Jb$(immediate predecessor)

➔Starting time of J_b and deadline of J_a will be modified to replace precedence constraint

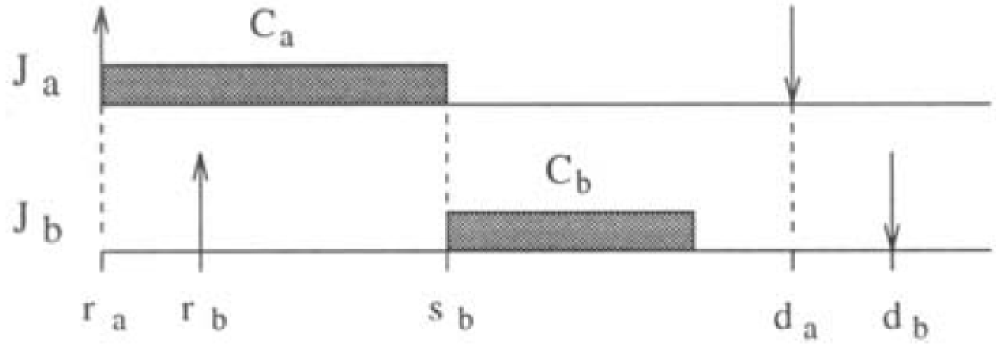


Modifying timing constraints

- Starting time

$s_b \geq r_b$	(that is, J_b must start the execution not earlier than its release time);
$s_b \geq r_a + C_a$	(that is, J_b must start the execution not earlier than the minimum finishing time of J_a).

$$\begin{cases} s_b \geq r_b \\ s_b \geq r_a + C_a \end{cases}$$



➔ Modifying release time of J_b

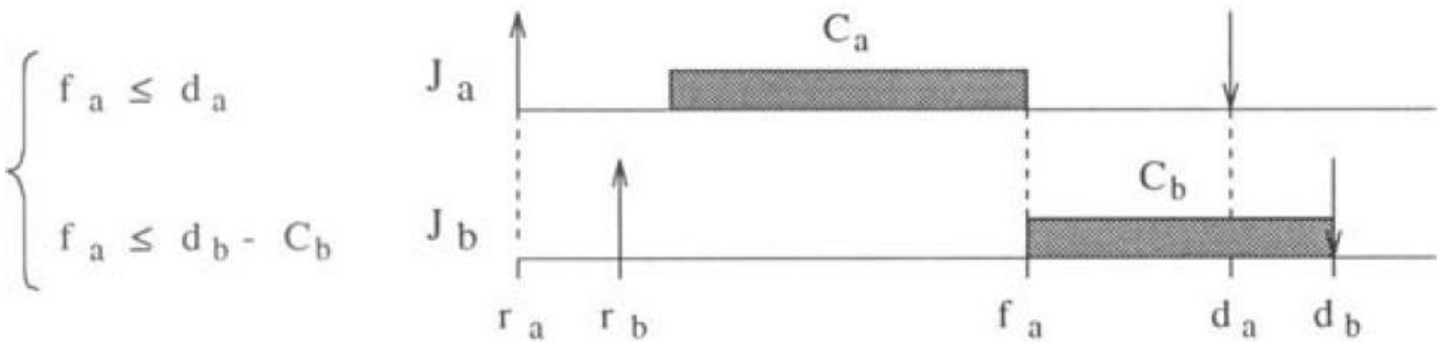
$$r_b^* = \max(r_b, r_a + C_a).$$

Modifying timing constraints

- Deadline

$f_a \leq d_a$ (that is, J_a must finish the execution within its deadline);

$f_a \leq d_b - C_b$ (that is, J_a must finish the execution not later than the maximum start time of J_b).



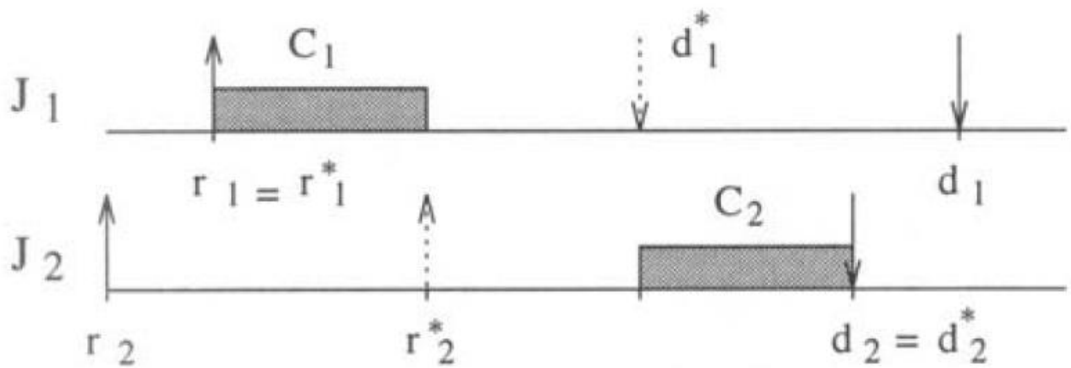
➔ Modifying deadline of J_a

$$d_a^* = \min(d_a, d_b - C_b)$$

Proof of optimality

$$\begin{cases} r^*_1 = r_1 \\ r^*_2 = r_1 + C_1 \end{cases}$$

$$\begin{cases} d^*_1 = d_2 - C_2 \\ d^*_2 = d_2 \end{cases}$$



- New timing constraints preserve precedence constraints and original deadline constraints

➔ Schedulability of task set is preserved

Example

- Given seven tasks A, B, C, D, E, F, G with precedence relations

$$A \rightarrow C$$

$$B \rightarrow C \qquad B \rightarrow D$$

$$C \rightarrow E \qquad C \rightarrow F$$

$$D \rightarrow F \qquad D \rightarrow G$$

All tasks arrive at time t = 0, have deadline D = 20, and computation time 2, 3, 3, 5, 1, 2, 5, respectively.

Modify their arrival times and deadlines to schedule them by EDF.

$$r_b^* = \max(r_b, r_a + C_a).$$

$$d_a^* = \min(d_a, d_b - C_b)$$

Summary

	sync. activation	preemptive async. activation	non-preemptive async. activation
independent	**EDD** (Jackson '55) $O(n \log n)$ Optimal	**EDF** (Horn '74) $O(n^2)$ Optimal	***Tree search*** (Bratley '71) $O(n\ n!)$ Optimal
precedence constraints	**LDF** (Lawler '73) $O(n^2)$ Optimal	**EDF *** (Chetto et al. '90) $O(n^2)$ Optimal	**Spring** (Stankovic & Ramamritham '87) $O(n^2)$ Heuristic

Figure 3.17 Scheduling algorithms for aperiodic tasks.