Inovace tohoto kurzu byla v roce 2011/12 podpořena projektem CZ.2.17/3.1.00/33274 financovaným Evropským sociálním fondem a Magistrátem hl. m. Prahy.

OPP A

PRA HA PRA GUE PRA GA PRA G

EVROPSKÁ UNIE

Evropský sociální fond

Praha & EU: Investujeme do vaší budoucnosti

Department of Distributed and Dependable Systems

Embedded and Real-time Systems
Aperiodic Task Scheduling

http://d3s.mff.cuni.cz

Department of Distributed and Dependable Systems

D3S

Tomáš Bureš

<bures@d3s.mff.cuni.cz>

CHARLES UNIVERSITY IN PRAGUE

Faculty of Mathematics and Physics

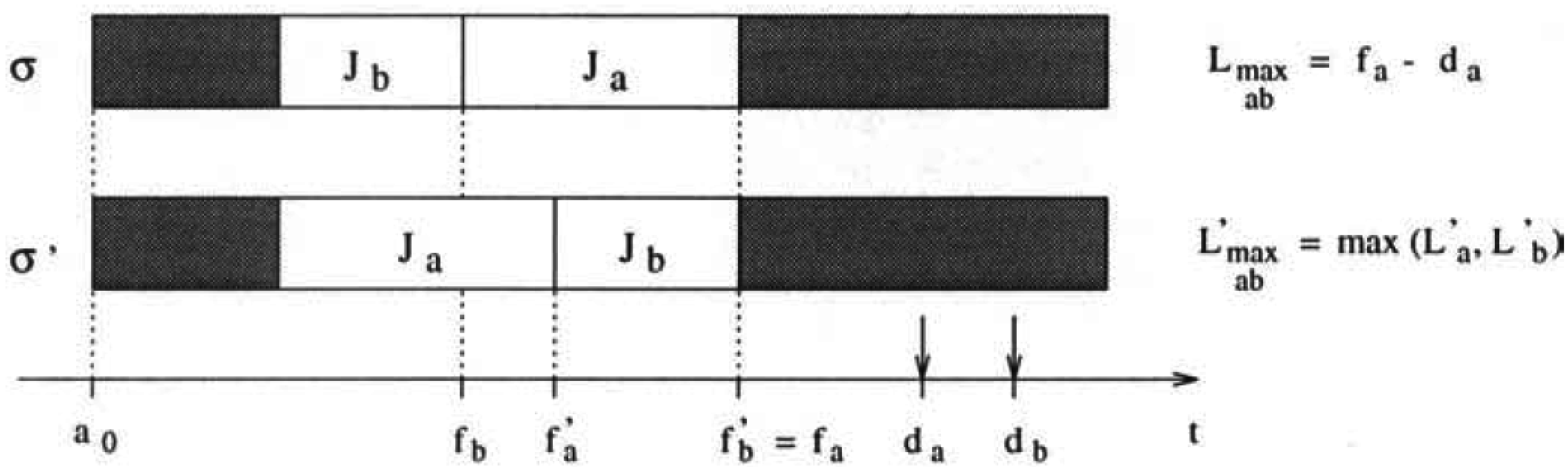
Earliest Due Date (EDD)

- Theorem (Jackson's rule)
  - Given a set of n independent tasks, any algorithm that executes the tasks in order of non-decreasing deadlines is optimal with respect to minimizing the maximum lateness

- Characteristics
  - uniprocessor
  - synchronous activation
  - minimizes maximum lateness
  - $O(n \log n)$

Earliest Due Date (EDD) – Proof of Optimality

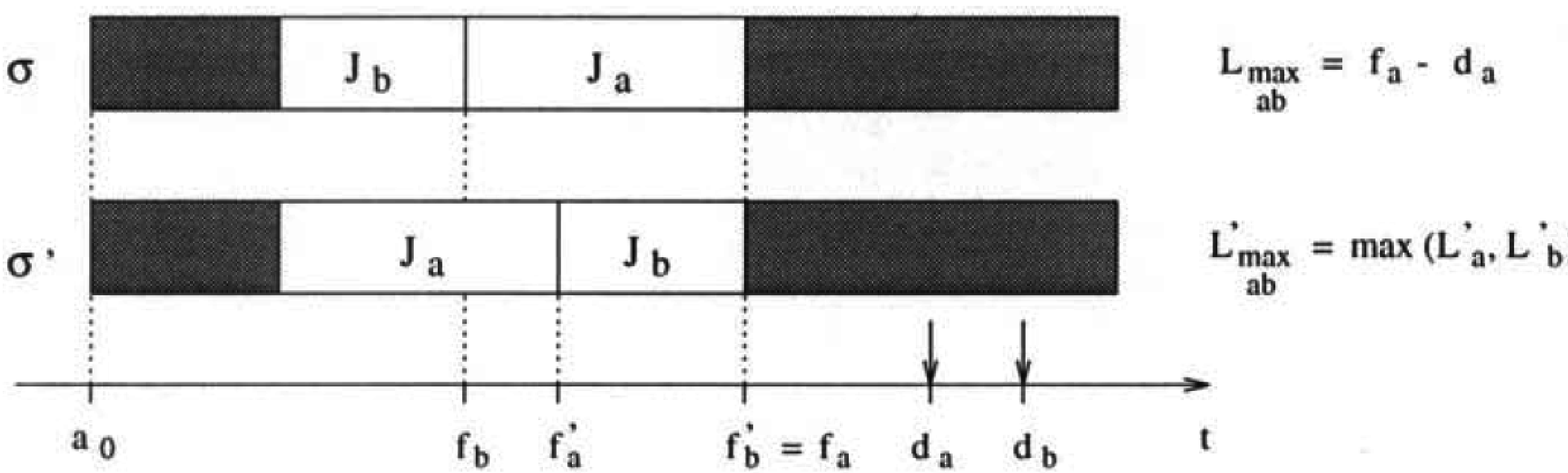
- Let σ be a schedule produced by any algorithm A. If A is different than EDD, than there exist two tasks J_a and J_b, with $d_a \leq d_b$, such that J_b immediately precedes J_a in σ. Now, let σ' be a schedule obtained from σ by exchanging J_a with J_b, so that J_a immediately precedes J_b in σ'.



- Interchanging position of J_a and J_b cannot increase the maximum lateness.

Earliest Due Date (EDD) – Proof of Optimality

- Two cases must be considered:
  If $L'_a \geq L'_b$, than $L'_{max}(\mathrm{a},\mathrm{b}) = \mathrm{f}'_\mathrm{a} - \mathrm{d}_\mathrm{a}$, and,
  since $f'_a < f_a$, we have $L'_{max}(a,b) < L_{max}(a,b)$.
  If $L'_a \leq L'_b$, than $L'_{max}(a,b) = f'_b - d_b = f_a - d_b$, and,
  since $d_a < d_b$, we have $L'_{max}(a,b) < L_{max}(a,b)$.



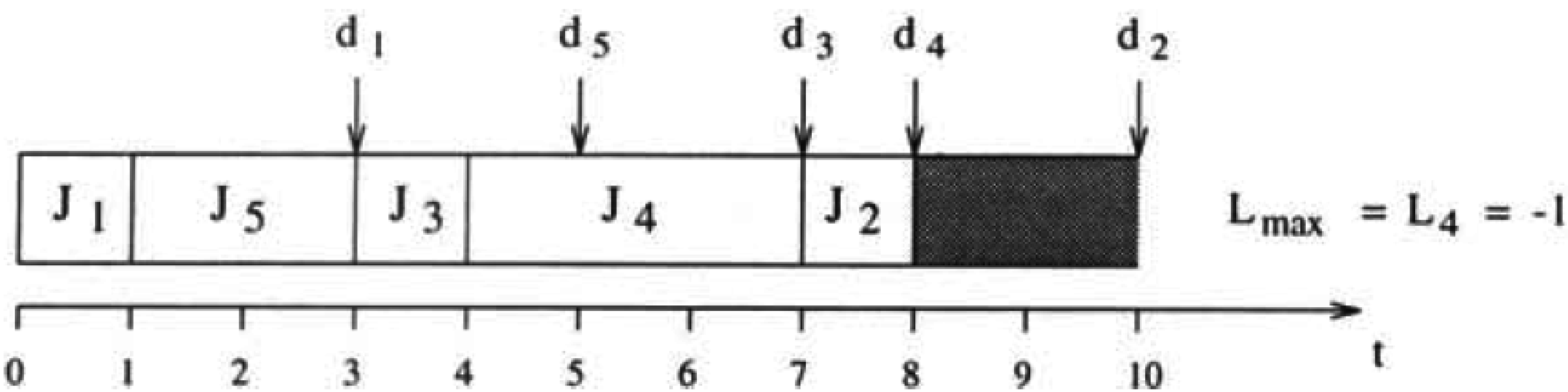
Earliest Due Date (EDD) – Proof of Optimality

- By a finite number of such transposition we get to the EDD schedule which must have maximum lateness less or equal to the original.

Earliest Due Date (EDD)

- Example of a feasible schedule

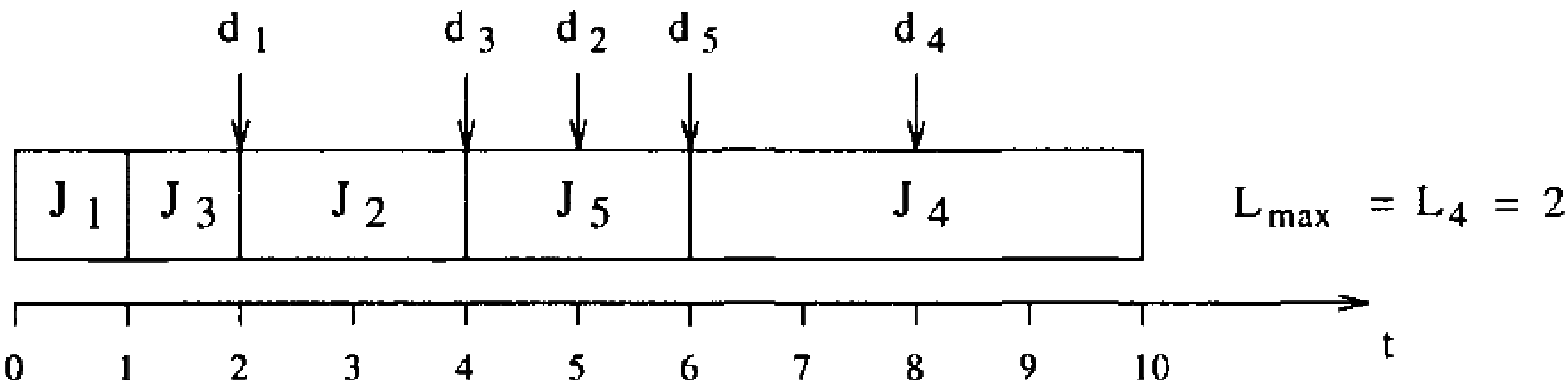
	J_1	J_2	J_3	J_4	J_5
C_i	1	1	1	3	2
d_i	3	10	7	8	5



Earliest Due Date (EDD)

- Example of an infeasible schedule

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



Earliest Due Date (EDD)

- Guarantee test (off-line)
  - Tasks $J_1, J_2, \dots, J_n$ ordered by increasing deadlines.

$$\forall i = 1, \dots, n \quad \sum_{k=1}^{i} C_k \le d_i$$

Department of Distributed and Dependable Systems

Earliest Deadline First (EDF)

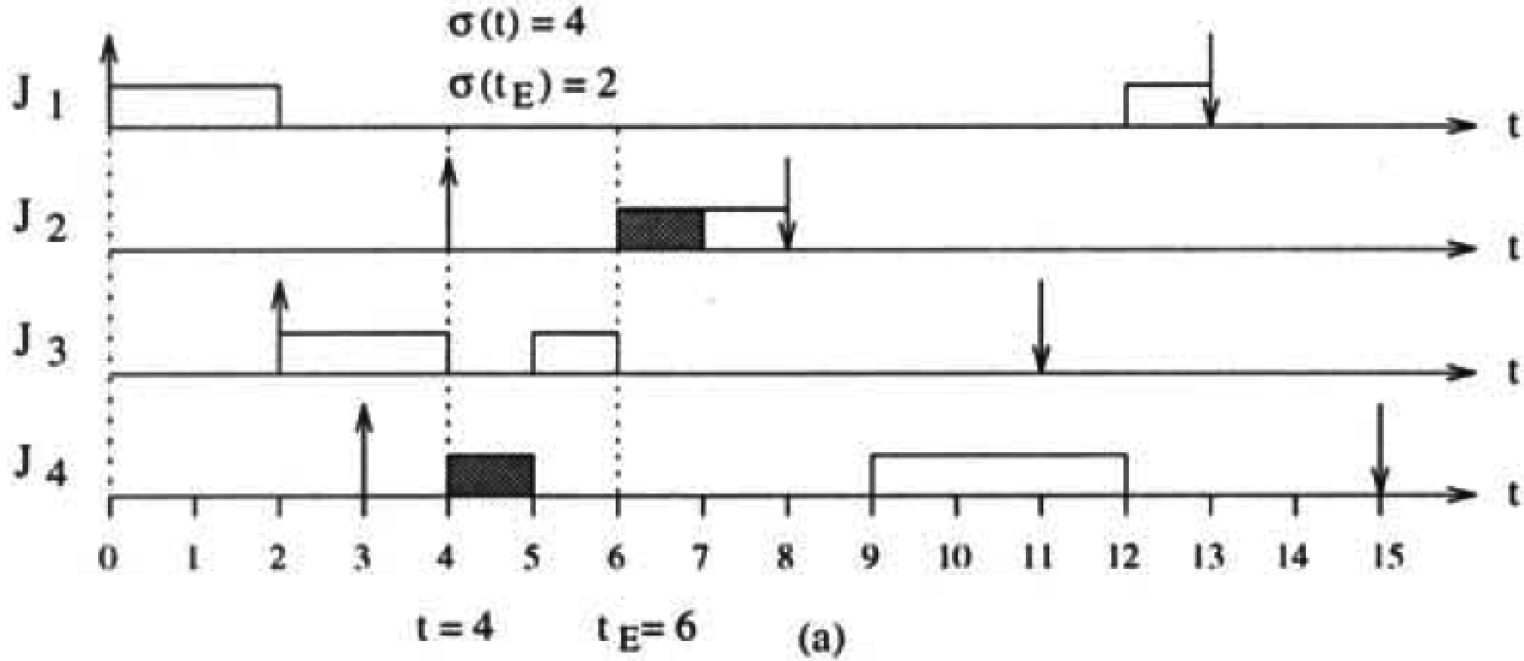
- Theorem (Horn)
  - Given a set of n independent tasks with arbitrary arrival times, any algorithm that at any instant executes the task with the earliest absolute deadline among all the ready tasks is optimal with respect to minimizing the maximum lateness
- Characteristics
  - uniprocessor
  - tasks may arrive at any time
  - preemptive
  - minimizes maximum lateness
  - $O(n \log n)$ or $O(n^2)$
    - depends on implementation of the ready queue

Department of Distributed and Dependable Systems

Earliest Deadline First (EDF) – Proof of Optimality

- $\sigma(t)$ identifies the task executing in the slice $[t, t+1)$.
  $E(t)$ identifies the task that, at time t, has the earliest deadline
  $t_E(t)$ is the time ($\geq t$) at which the next slice of task $E(t)$ begins its execution in the current schedule.

- If $\sigma \neq \sigma_{EDF}$, then in σ there exists a time t such that $\sigma(t) \neq E(t)$.
  Interchanging the position of $\sigma(t)$ and $E(t)$ cannot increase the maximum lateness.
  If the schedule σ starts as time $t = 0$ and D is the latest deadline of the task set ($D = \max_i\{d_i\}$) then σ_{EDF} can be obtained from σ by at most D transactions.



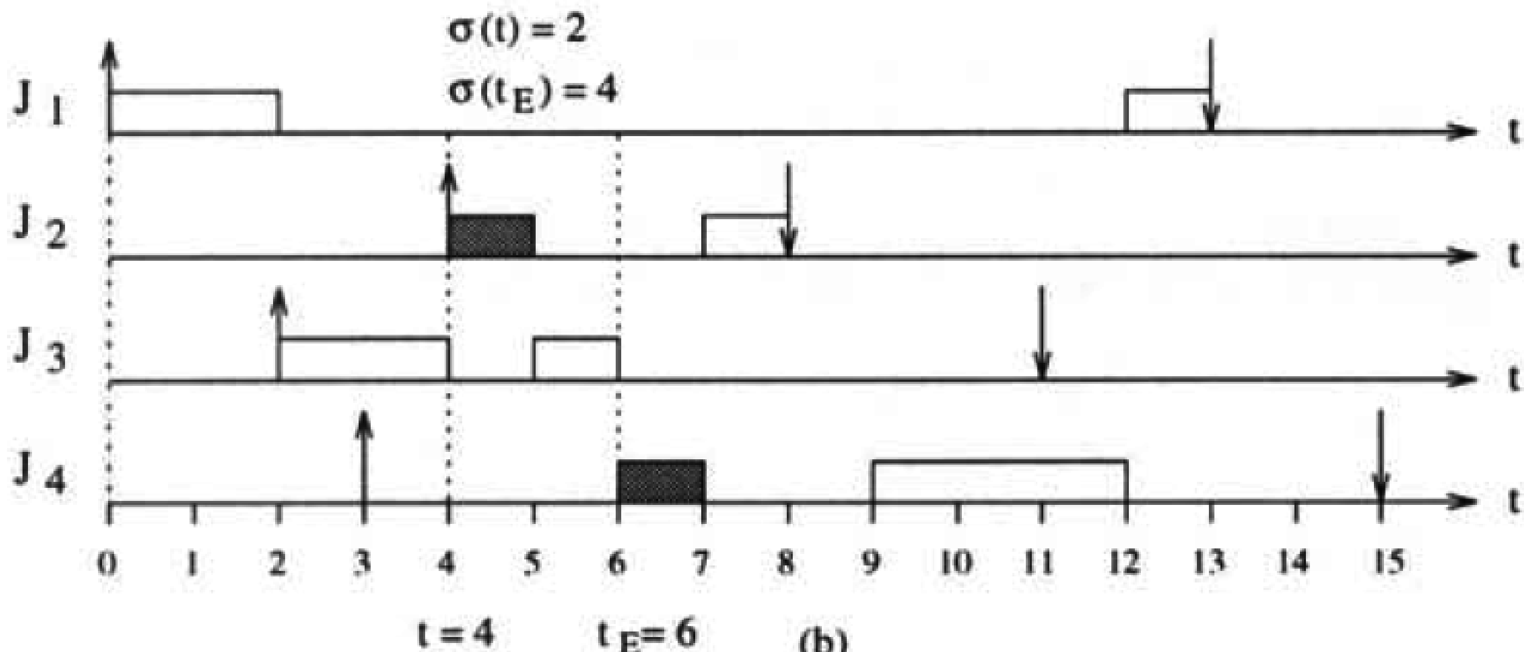


Figure from Buttazzo, G.:Hard RT Comp. Systems

Earliest Deadline First (EDF) – Proof of Optimality

- At any instant, each slice in σ can be either anticipated or postponed up to t_E. If a slice is anticipated, the feasibility of that task is obviously preserved.
  If a slice of J_i is postponed at t_E and σ is feasible, it must be $(t_E + 1) \leq d_E$, being d_e the earliest deadline. Since $d_E \leq d_i$ for any i, than we have $t_E + 1 \leq d_i$, which guarantees the schedulability of the slice postponed at t_E.

Earliest Deadline First

- Example of EDF schedule

	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

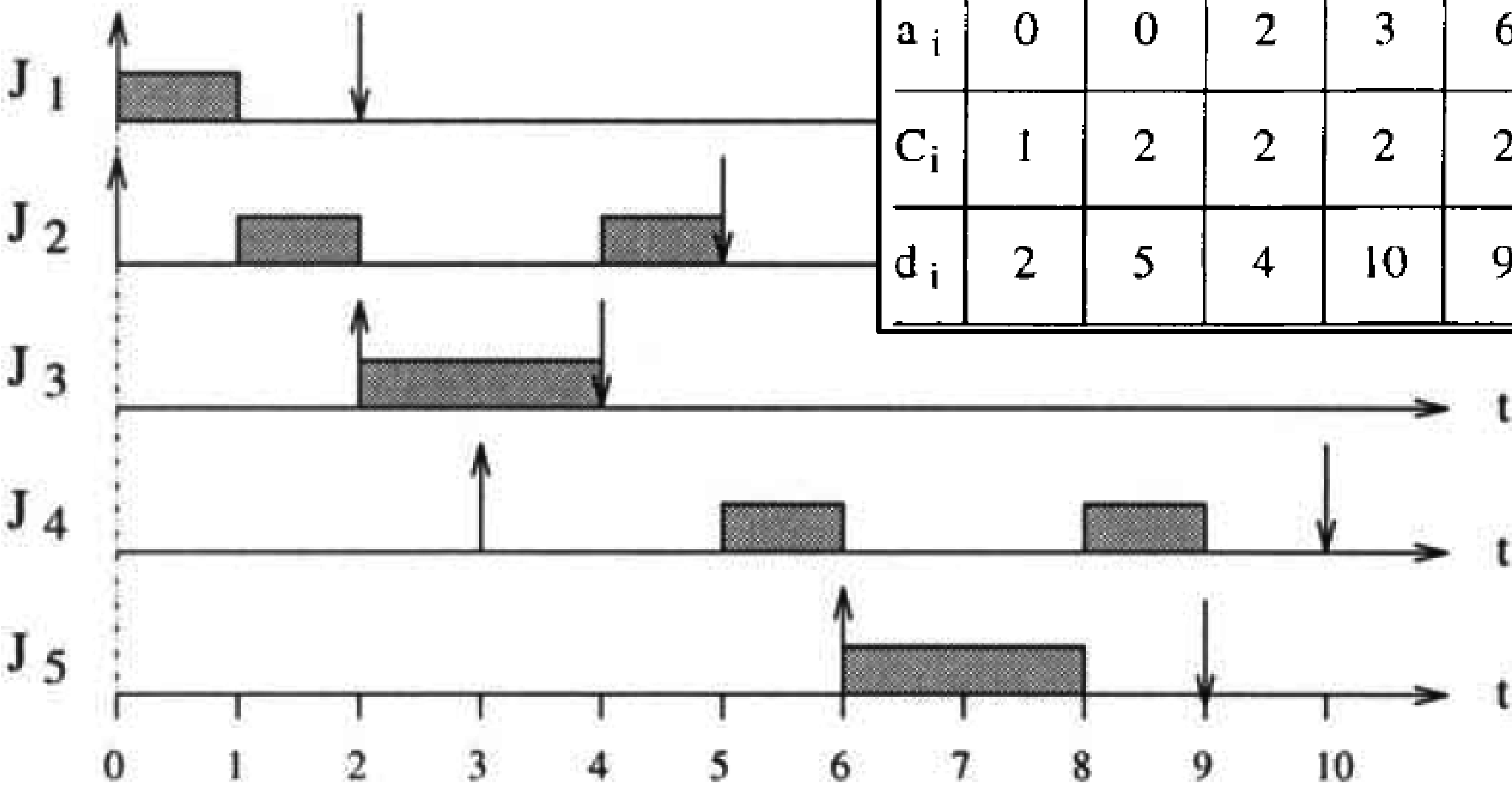


Figure from Buttazzo, G.:Hard RT Comp. Systems

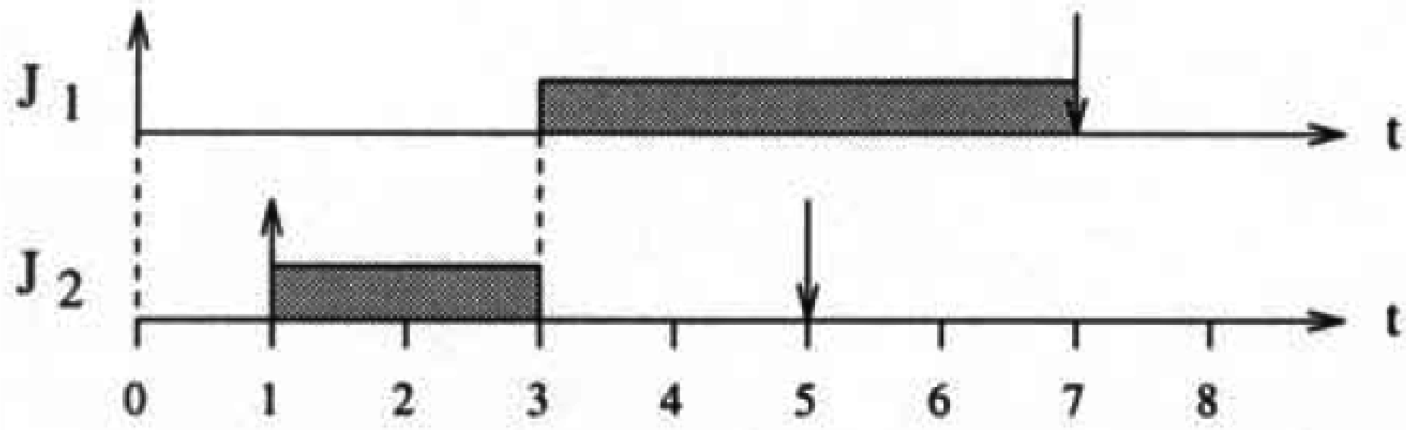
Earliest Deadline First (EDF)

- Guarantee test (on-line)
  - Tasks $J_1, J_2, \dots, J_n$ ordered by increasing deadlines.
  - $c_i(t)$ is the remaining worst-case execution time of task J_i.

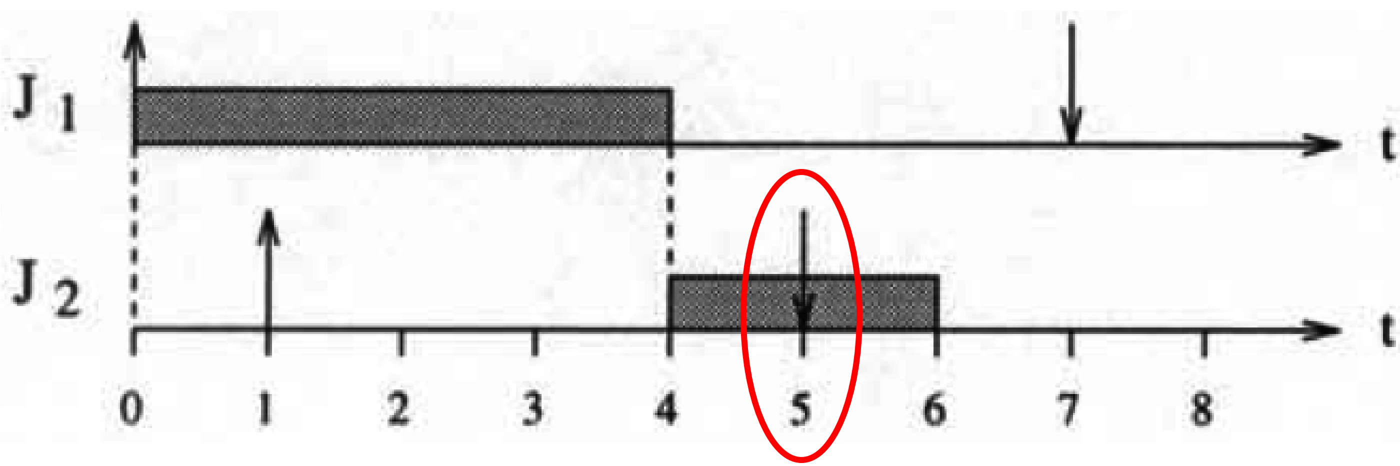
$$\forall i = 1, \dots, n: \sum_{k=1}^{i} c_k(t) \le d_i$$

Non-Preemptive Scheduling

Optimal schedule



EDF schedule



	J_1	J_2
a_i	0	1
C_i	4	2
d_i	7	5

Department of Distributed and Dependable Systems

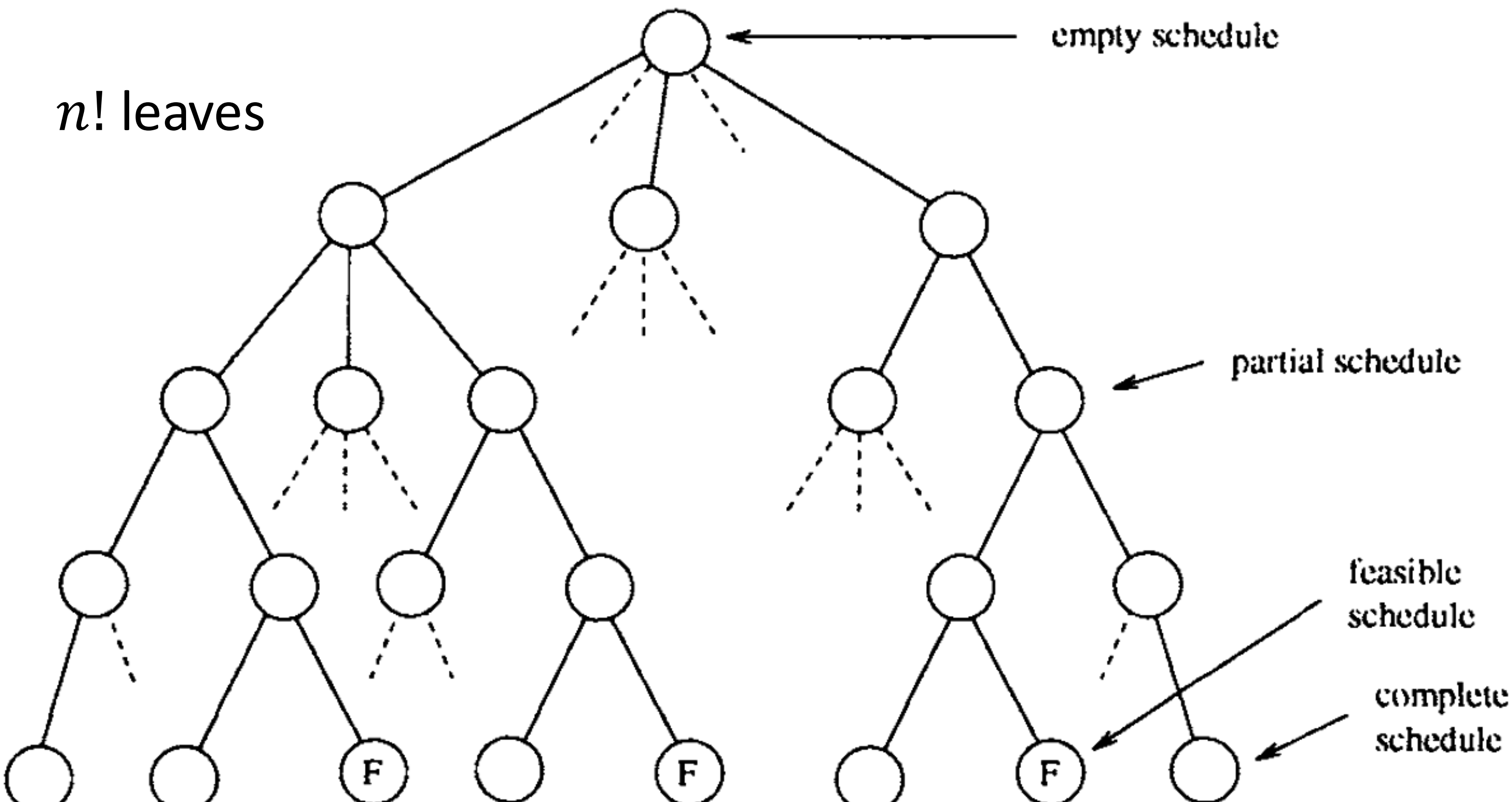
Non-Preemeptive Scheduling

- The optimal schedule is based on waiting one tick at the beginning
  - This requires the algorithm to know a priori arrival times
  - Thus no on-line algorithm can produce the optimal schedule

Non-Preemptive Scheduling

- The problem of finding a feasible schedule is NP hard
- Treated as off-line with tree search algorithms.

n! leaves



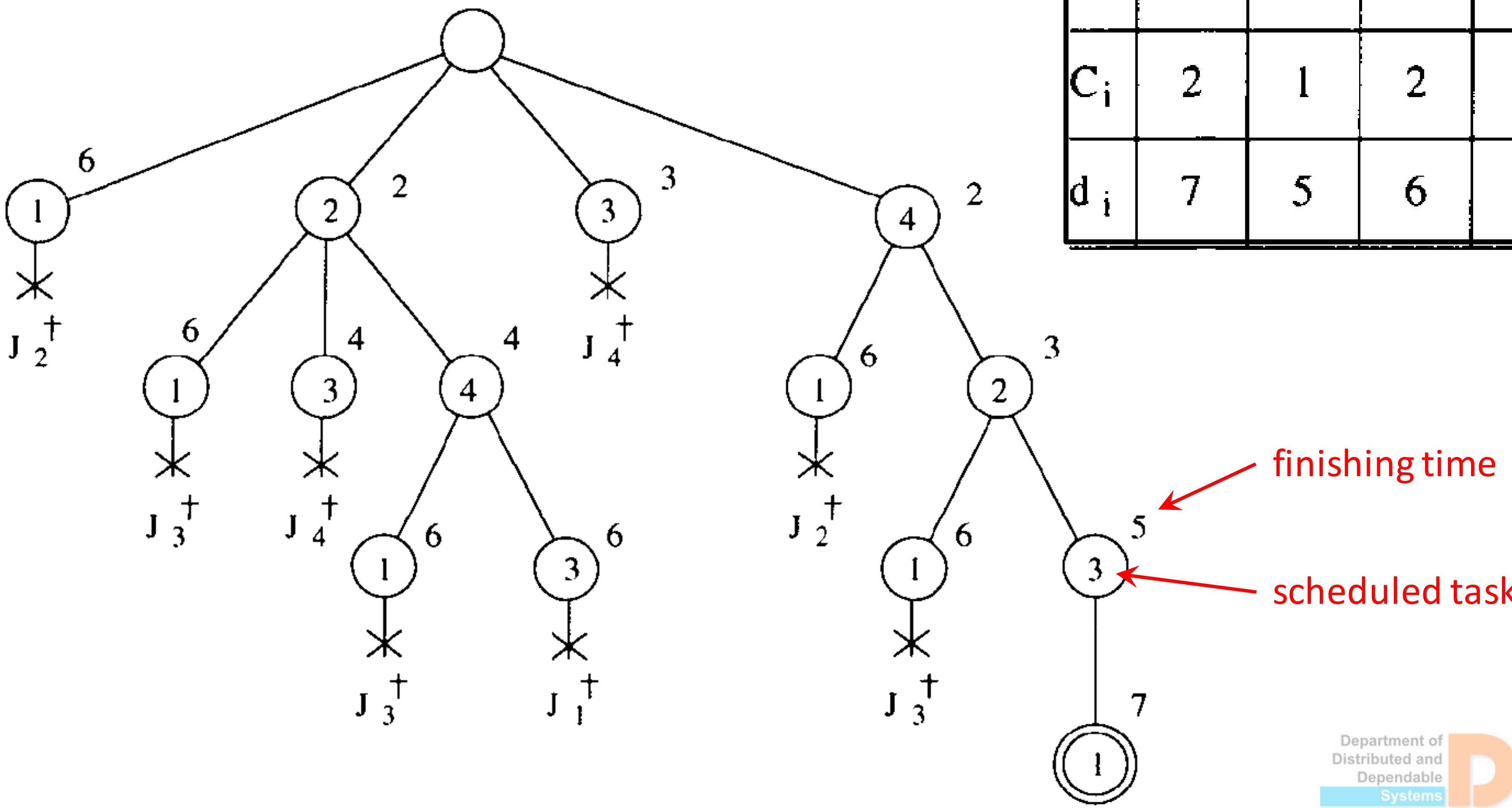
Bratley's Algorithm

- Reduces average complexity by pruning techniques
  - Do not expand unless the partial schedule is found to be strongly feasible
  - A partial schedule strongly feasible if after adding any of the remaining nodes remains feasible
- Characteristics
  - uniprocessor
  - non-preemptive
  - minimizes maximum lateness

Department of Distributed and Dependable Systems

Bratley's Algorithm

	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



Spring Algorithm

- Similar to Bratley's algorithm with heuristic function used to guide the search
  - At each step, the algorithm selects the task that minimizes the heuristic function
- If no backtracking
  - Complexity – $O(n^2)$
  - The algorithm is not optimal – if it does not find a feasible schedule, it doesn't mean that such a schedule doesn't exist

Spring Algorithm

- Example of heuristic functions:
- H = a First Come First Served
- H = C Shortest Job First (SJF)
- H = d Earliest Deadline First (EDF)
- H = d + wC EDF + SJF
- etc.

Spring Algorithm

- Handling precedence constraints – eligibility
  - $E_i = 1$ if all ancestors in precedence graph are completed;
    $E_i = \infty$ otherwise

- Heuristic functions
  - $H = E_i d_i$

Scheduling with Precedence Constraints

- Generally NP-hard
- Optimal algorithms working in polynomial time exist under particular assumptions on the tasks

Latest Deadline First

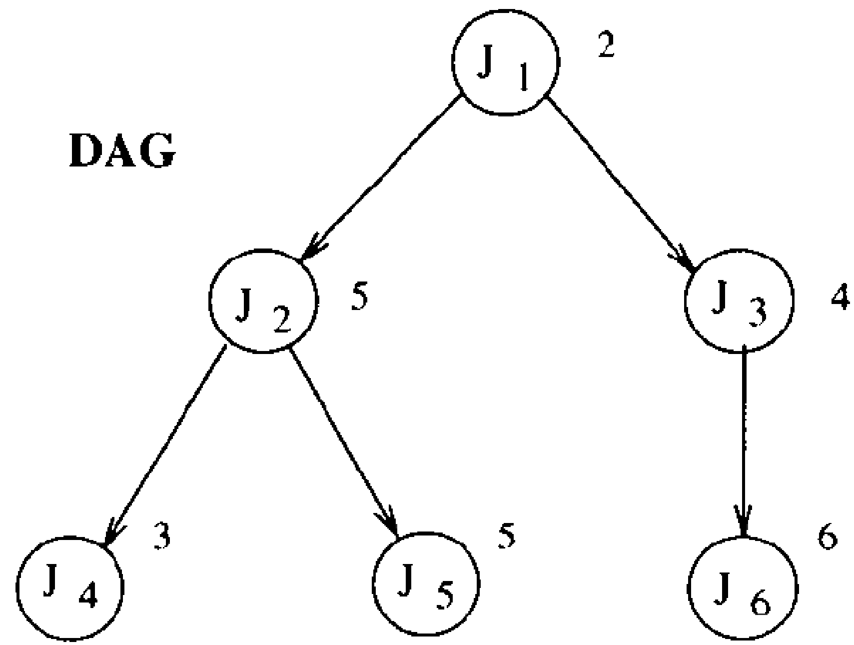
- Characteristics
  - uniprocessor
  - synchronous activation
  - minimizes maximum lateness

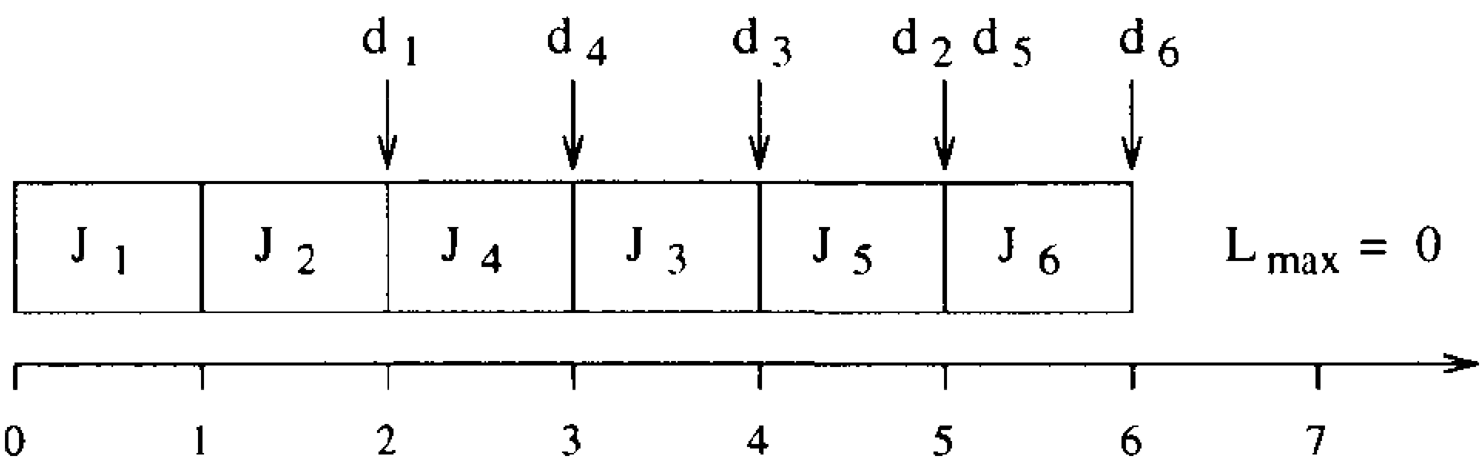
- Builds the scheduling queue from tail to head: among the tasks without successors or whose successors have been all selected, LDF selects the task with the latest deadline to be scheduled last.

Latest Deadline First

- Example:

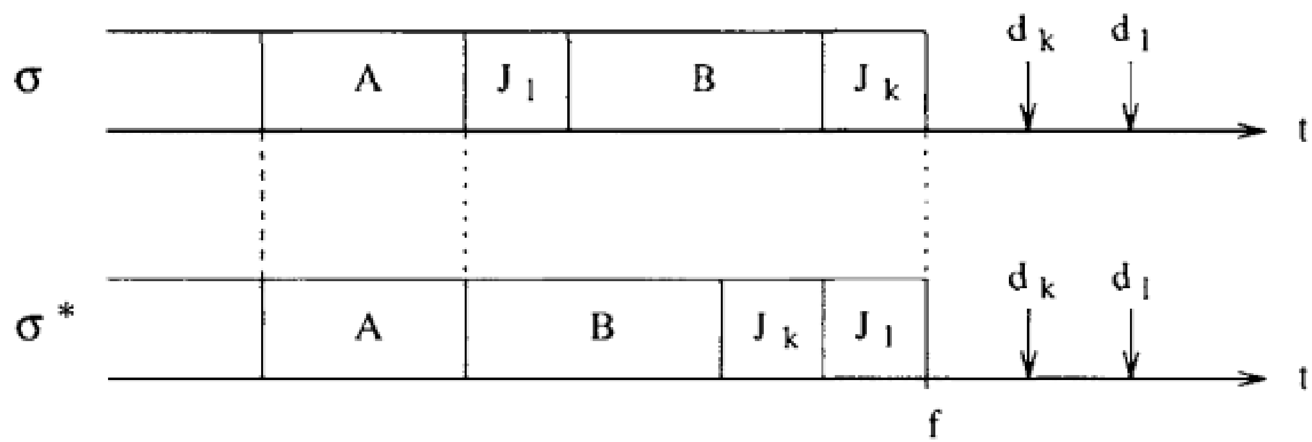
	J_1	J_2	J_3	J_4	J_5	J_6
C_i	1	1	1	1	1	1
d_i	2	5	4	3	5	6





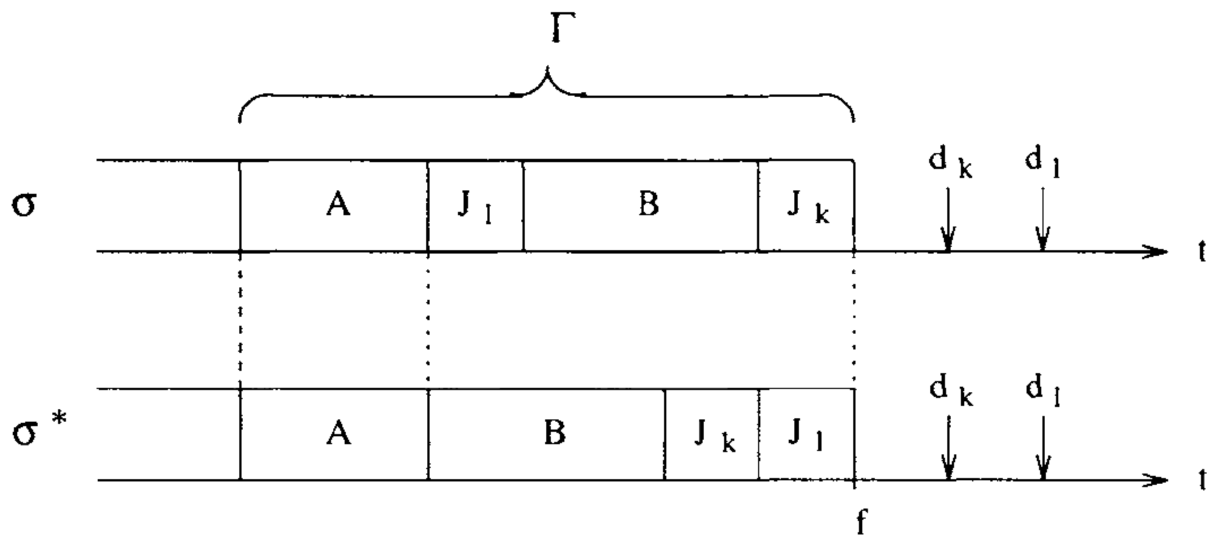
Latest Deadline First – Proof of Optimality

- $\mathcal{J}$ – complete set of tasks to be scheduled
  $\Gamma \subseteq \mathcal{J}$ – subset of tasks without successors
  J_l - task in Γ with the latest deadline d_l
  σ – a schedule not following EDL with the last task J_k
- $\Gamma = A \cup \{J_l\} \cup B \cup \{J_k\}$
- σ^* is a schedule obtained from σ by moving task J_l to the end of the queue and shifting all other tasks to the left.



Latest Deadline First – Proof of Optimality

- Such transformation does not increase the maximum lateness of tasks in Γ: $L^*_{max}(\Gamma) = \max[L^*_{max}(A), \max[L^*_{max}(B), L^*_k, L^*_l]]$
- $L^*_{max}(A) = L_{max}(A) \le L_{max}(\Gamma)$ since A is not moved;
  $L^*_{max}(B) = L_{max}(B) \le L_{max}(\Gamma)$ since B starts earlier in σ^*;
  $L^*_k \le L_k \le L_{max}(\Gamma)$ since task J_k starts earlier in σ^*;
  $L^*_l = f - d_l \le f - d_k \le L_{max}(\Gamma)$ since $d_k \le d_l$.



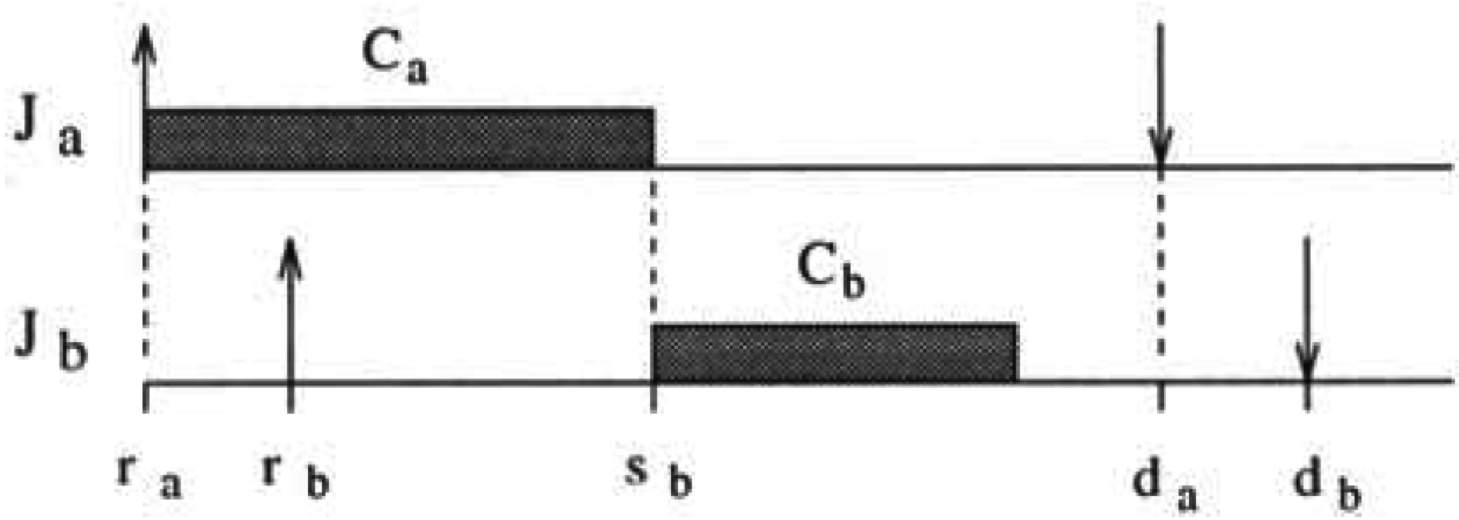
EDF with Precedence Constraints

- Characteristics
  - uniprocessor
  - preemptive
  - minimizes maximum lateness

- A task set with dependent tasks is transformed into a task set with independent tasks by an adequate modification of timing parameters. Then, tasks are scheduled by EDF.
  - The transformation ensures that the dependent tasks are schedulable if and only if the independent tasks are schedulable.
  - Release times and deadlines are modified so that each task cannot start before its predecessors and cannot preempt their successors.

EDF with Precedence Constraints

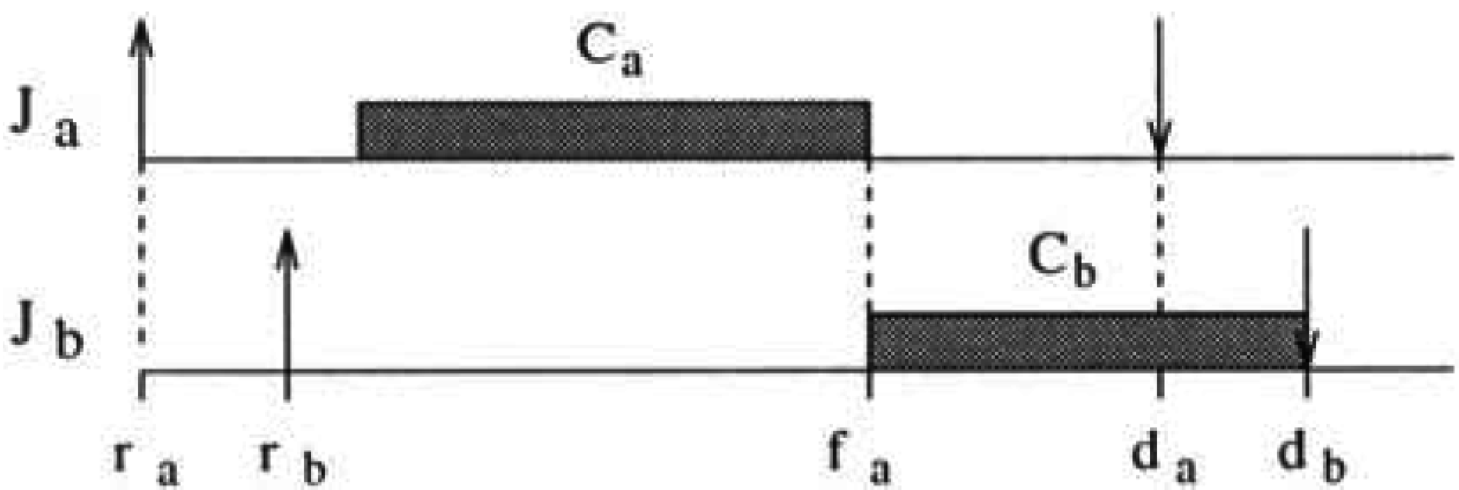
- Modification of release times ($J_a \rightarrow J_b$)
  - $s_b \geq r_b$ (that is, J_b must start execution not earlier than its release time);
  - $s_b \geq r_a + C_a$ (that is, J_b must start execution not earlier than minimum finishing time of J_a).



Department of Distributed and Dependable Systems

EDF with Precedence Constraints

- Modification of deadlines ($J_a \rightarrow J_b$)
  - $f_a \leq d_a$ (that is, J_a must finish execution within its deadline);
  - $f_a \leq d_b - C_b$ (that is, J_a must finish execution not later than the maximum start time of J_b).

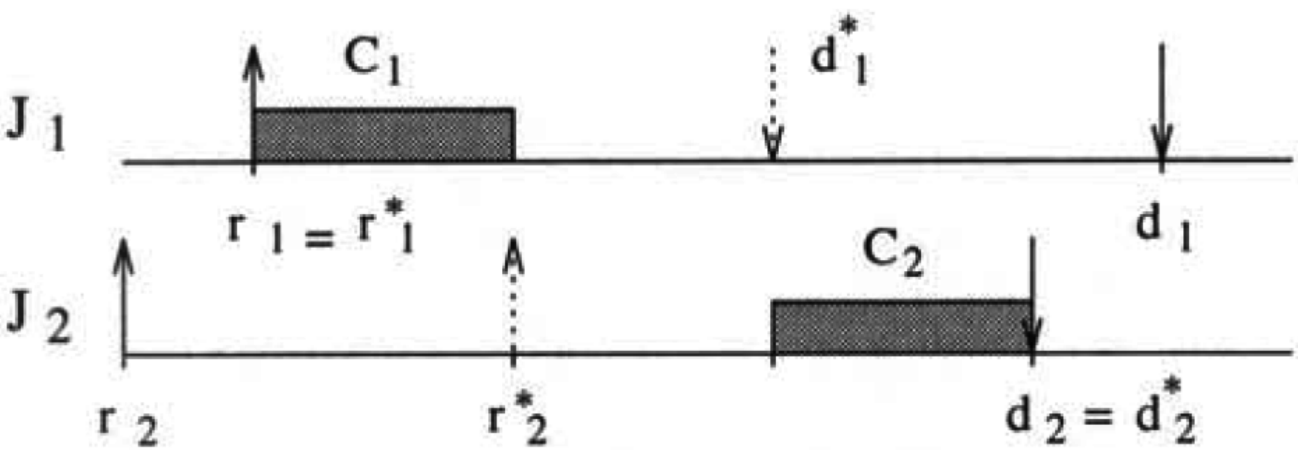


EDF with Precedence Constraints

- Example

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

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



Comparison of Algorithms

	sync. activation	preemptive async. activation	non-preemptive async. activation
independent	**EDD** (Jackson '55) *O(n logn)* 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