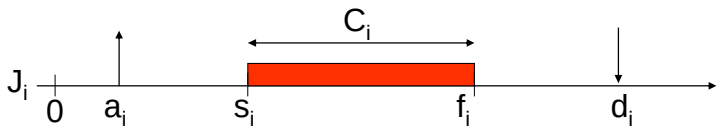
Embedded Systems

Bernd Finkbeiner
Calogero Zarba
Moritz Hahn

Sommersemester 2007

A-periodic scheduling



- Given:
  - A set of non-periodic tasks $\{J_1, \ldots, J_n\}$ with
    - arrival times a_i, deadlines d_i, computation times C_i
    - precedence constraints
    - resource constraints
  - Class of scheduling algorithm:
    - Preemptive, non-preemptive
    - Off-line / on-line
    - Optimal / heuristic
    - One processor / multi-processor
    - …
  - Cost function:
    - Minimize maximum lateness (soft RT)
    - Minimize maximum number of late tasks (feasibility! – hard RT)
- Find:
  Optimal / good schedule according to given cost function

1 | sync | L_{max}

EDD – Earliest Deadline Due

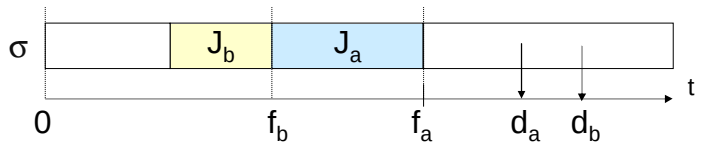
- Lemma:
  If arrival times are synchronous, then preemption does not help, i.e. if there is a preemptive schedule with maximum lateness L_{max}, then there is also a non-preemptive schedule with maximum lateness L_{max}.
- Proof:
  - Consider a preemptive schedule with maximum lateness L_{max}.
  - If there is a task which is preempted, then choose the last point t of preemption. Let J_l be the task preempted in the schedule. Reshuffle the schedule by postponing all of J_l's runtime allocated immediately before t s.t. that I happens immediately before the time t' of resumption of J_l, thus removing the preemption at t. This will not change lateness of J_l and will at most reduce lateness of all other tasks, as those are unaffected or shuffled forward.
  - Repeat this reshuffling until there is no further preemption.

EDD

- Theorem (Jackson ’55):
  Given a set of n independent tasks with synchronous arrival times, any algorithm that executes the tasks in order of non-decreasing deadlines is optimal with respect to minimizing the maximum lateness.
- Remark: Minimizing maximum lateness includes finding a feasible schedule, if there exists one. The reverse is not necessarily true.

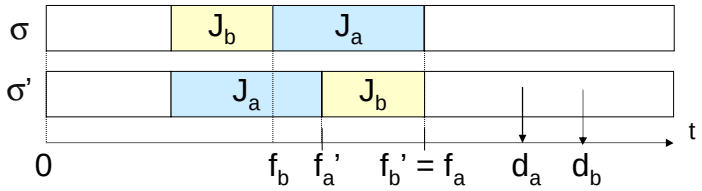
EDD

- Proof of Theorem:
  - By simple interchange argument. We construct from an arbitrary schedule an EDD schedule without increasing L_{max}.
  - Let σ be a schedule produced by any algorithm A.
    According to our lemma we can assume w.l.o.g. that σ is non-preemptive.
  - Assume that σ does not follow EDD.
  - Then there exist two tasks J_a and J_b, with $d_a < d_b$, such that J_b immediately precedes J_a in σ.



EDD

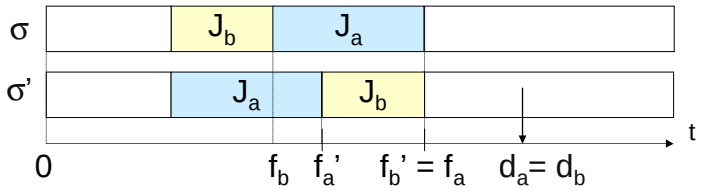
- Let σ' be a schedule obtained from σ by exchanging J_a with J_b, so that J_a immediately precedes J_b in σ'.



- Maximum lateness L'_{max} of σ' is not larger than maximum lateness L_{max} of σ, as $f_a - d_a > f_b' - d_b$.
- Repeat the reshuffling until EDD order is achieved.

EDD

- The repeated reshuffling transforms σ into an EDD schedule without increasing maximum lateness.
- Different EDD algorithms may produce schedules that permute tasks with the same deadline; all such permutations have the same maximum lateness.



$f_b' - d_b = f_a - d_a$

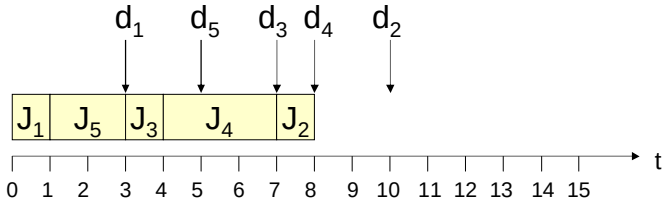
- Complexity of EDD scheduling:
  - Sorting n tasks by increasing deadlines

  ⇒ O(n log n)

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

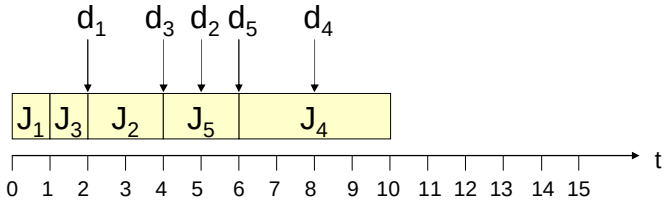


$L_{max} = L_4 = -1$, schedule feasible

EDD - 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



$L_{max} = L_4 = 2$, infeasible schedule
(but schedule with minimum L_{max})