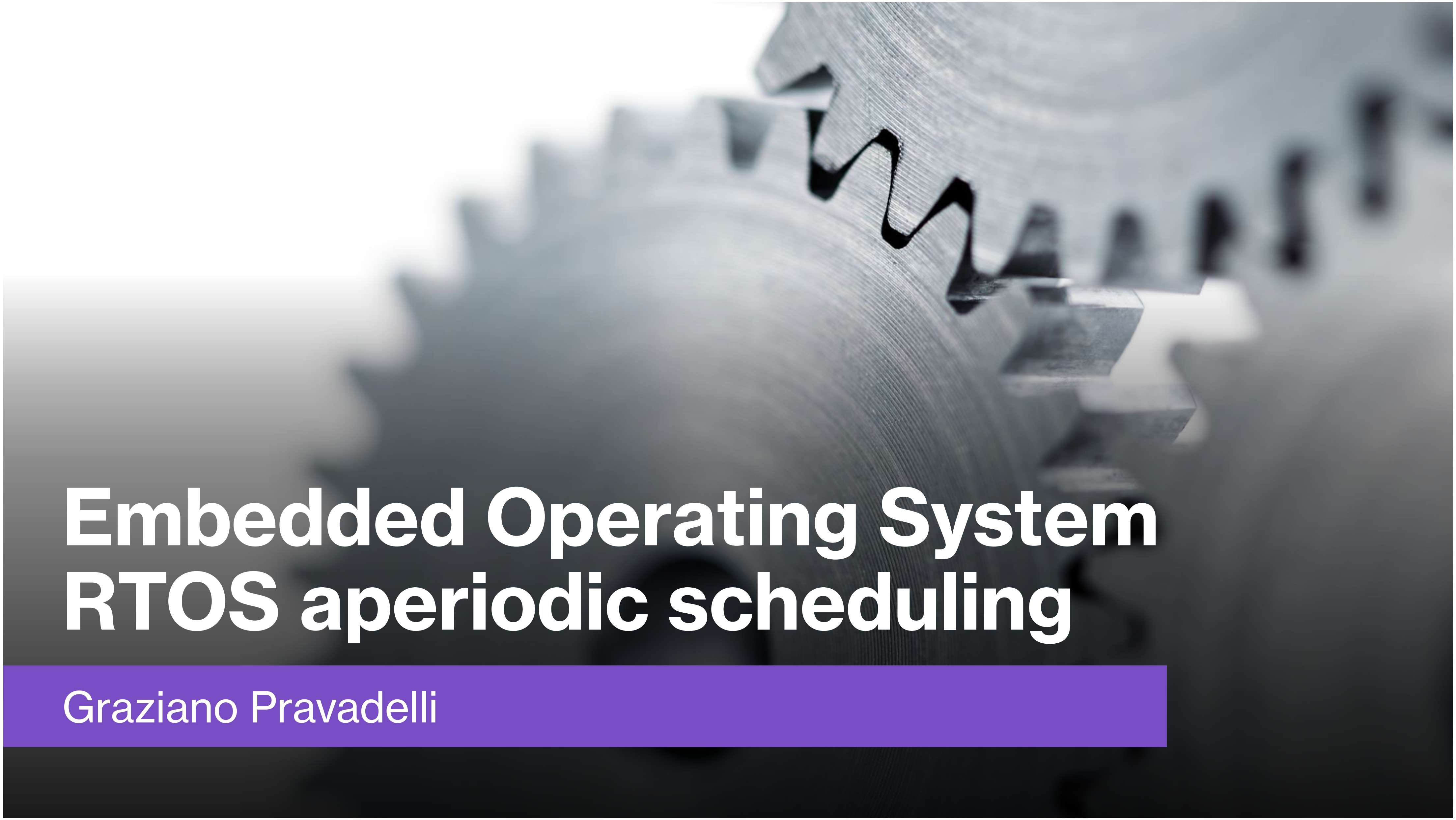


Embedded Operating System RTOS aperiodic scheduling

Graziano Pravadelli

Problem classification

To make the problem tractable a classification is necessary

Different algorithms solve different problems

Graham classification

Triple (α, β, γ)

- α = environment on which the task set has to be scheduled
  - typically number of processors
- β = tasks and resource characteristics
  - i.e., preemptive, precedence, synchronous activations etc.
- γ = cost function to be optimized
  - i.e., Max number of late tasks

Problem classification - Examples

$1 \mid prec \mid L_{max}$

- uniprocessor machine
- task set with precedence constraints
- minimize maximum lateness

$2 \mid sync \mid \sum Late_i$

- two processors
- tasks with synchronous arrival time
- minimize number of late tasks

Typical scheduling space

Task activation times

- Synchronous activations ($a_i=0$, $\forall i$)
- Asynchronous activations ($\exists i$, s.t. $a_i \neq 0$)

Task relations

- With/without precedence relations

Preemption

- With/without preemption

Aperiodic task scheduling algorithms

Without precedence constraints

- Jackson's algorithm
- Horn's algorithm

Jackson's algorithm

To solve 1 | sync | L_{max}

Tasks are independent

- no precedence relations
- no shared resources

Preemption useless because of synch arrivals

- Task set:
  $J = \{J_i \ (C_i, D_i) \mid i = 1 \ldots n\}$
  - Computation time C_i
  - Deadline D_i

- **Principle:**
  Earliest Due Date (EDD)

- Complexity
  - Sorting n values (O (n*log n))

Jackson's algorithm – Example (1)

- 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

Draw the schedule

Jackson's algorithm – Example (2)

- An unfeasible schedule even if EDD minimized L_{max}

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

Draw the schedule

Jackson's algorithm – Optimality

Given a set of n independent tasks, any algorithm that executes the tasks in order of non-decreasing deadlines is optimal with respect to minimize the maximum lateness

Does it mean that EDD always succeed?

EDD can not guarantee feasible schedule	EDD only guarantees that if a feasible schedule exists it will find it

Jackson's algorithm – Guarantee

To guarantee the set J can be feasibly scheduled by EDD?

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

by considering tasks listed with increasing deadlines

Horn's algorithm

To solve 1 | preem | L_{max}

- Asynchronous arrival
- Tasks are independent
  - no precedence relations
  - no shared resources
- Preemption is necessary

- Task set:
  $J = \{J_i \ (A_i, C_i, D_i) \mid i = 1 \ldots n\}$
  - Arrival time A_i
  - Computation time C_i
  - Deadline D_i
- **Principle:**
  Earliest Deadline First (EDF)
- Complexity
  - O(n) per task
    - inserting a newly arriving task into an ordered list properly
  - n tasks => total complexity $O(n^2)$

Horn' s algorithm – Example (1)

- A feasible 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

Draw the schedule

Horn's algorithm – Optimality

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

Like EDD, EDF can not guarantee feasible schedule

Horn’s algorithm – Guarantee

To guarantee the set J can be feasibly scheduled by EDF?

$$\forall i = 1, \ldots, n \quad \sum_{k=1}^{i} c_k(t) \leq d$$

by considering tasks listed with increasing deadlines, and $c_i(t)$ being the remaining WCET of J_i

Horn's algorithm: without preemption?

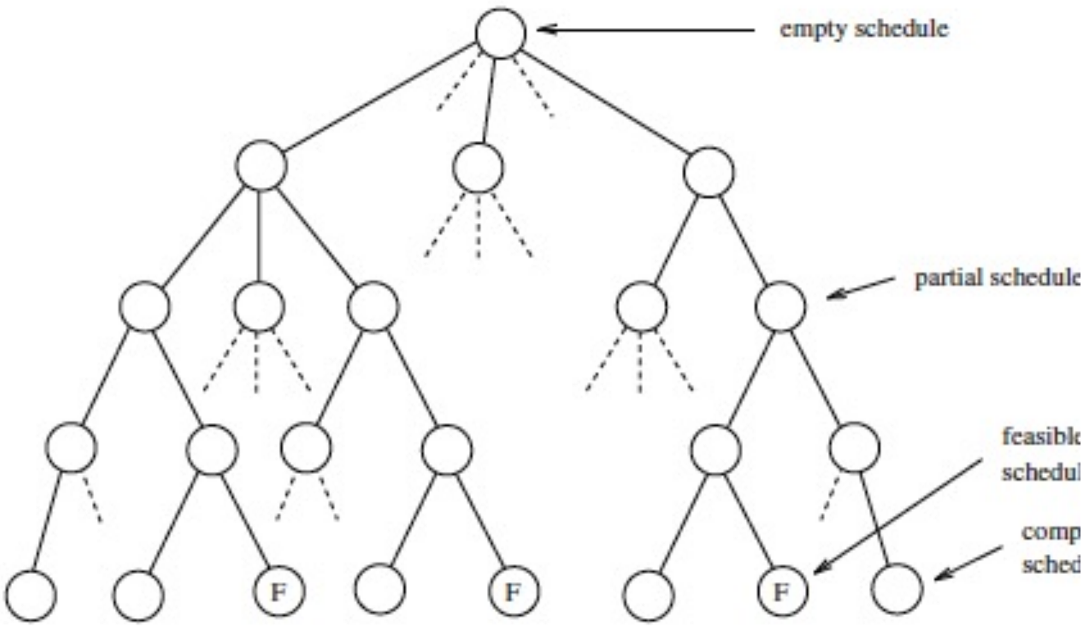
- **No longer optimal!**

	J_1	J_2
a_i	0	1
C_i	4	2
d_i	7	5

Draw EDF schedule without preemption

EDF no pre-emption is unfeasible
Can we do better without preemption?

Asynchronous arrivals and no preemption



When preemption is not allowed and tasks can have arbitrary arrivals

The problem of minimizing the maximum lateness and the problem of finding a feasible schedule become NP-hard

Aperiodic task scheduling algorithms

With precedence constraints

In general a NP-hard problem

- For special cases polynomial time algorithms possible

Two schemes

- Latest Deadline First (LDF)
- Modified EDF

LDF algorithm [Lawler]

To solve 1 | (prec, sync) | L_{max}

n tasks with

a DAG for their precedence relations

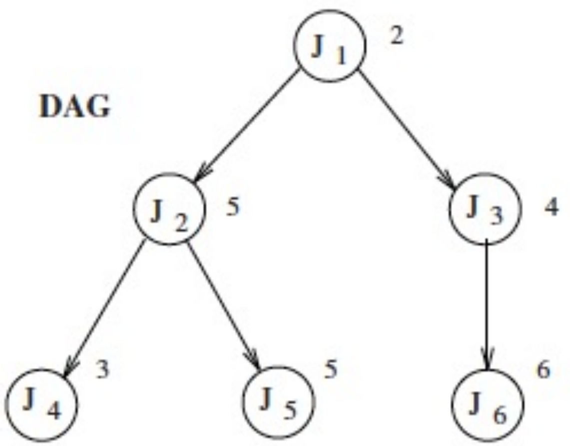
simultaneous arrival times

Optimal

- **Principle**
  LDF builds the scheduling queue from tail to head
  - Among the tasks without successors or with all successors already selected, LDF selects the one with latest deadline to be scheduled last
- Complexity: $O(n^2)$
  - For each job, the precedence graph has to be visited

LDF algorithm – Example

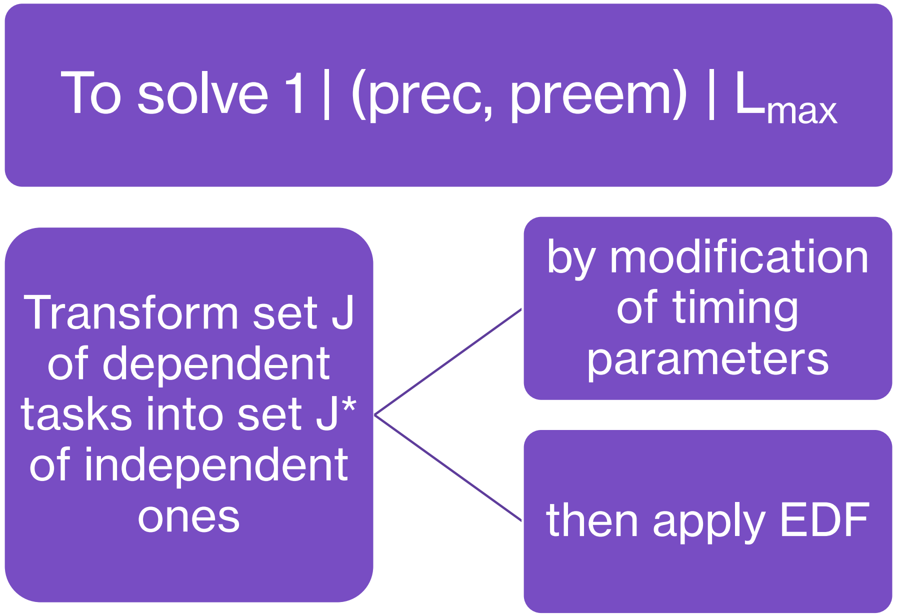
	J_1	J_2	J_3	J_4	J_5	J_6
C_i	1	1	1	1	1	1
d_i	2	5	4	3	5	6



Draw LDF schedule

Draw EDF schedule

Modified EDF [Chetto]

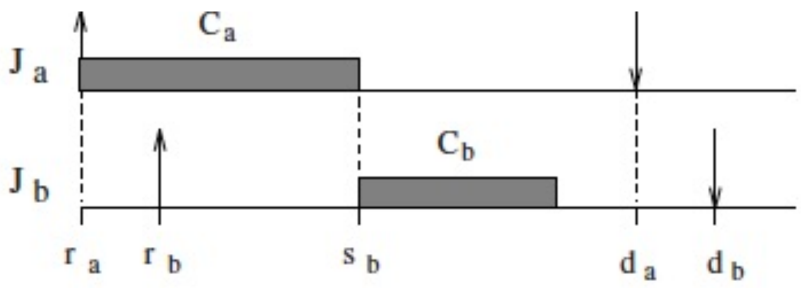


- **Modification**
  - Change arrival times and deadlines such that each task
    - Cannot start before its predecessors
    - Cannot preempt their successors (other tasks, however, may be preempted)
- The transformation ensures
  - J* schedulable ⇔ J schedulable
  - prec constraints satisfied

Modified EDF - Arrival times changes

- Try to postpone arrival time
- Given $J_a \rightarrow J_b$, the following two conditions must be satisfied:
  - $s_b \geq r_b$ — J_b cannot start earlier than its arrival time
  - $s_b \geq r_a + c_a$ — J_b cannot start earlier than minimum finish time of J_a

- **New release time for J_b**
  $r^*_b = \max(r_b, r_a + c_a)$

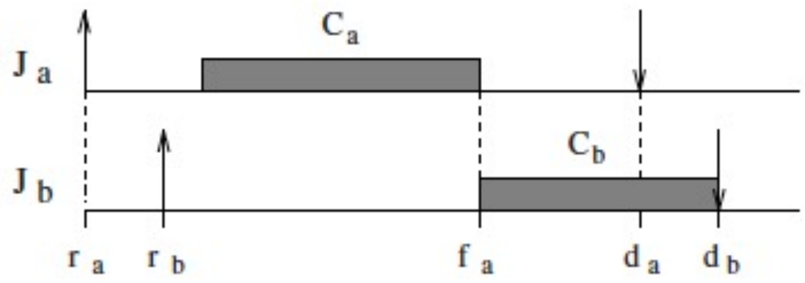


Complexity: $O(n^2)$

Modified EDF – Deadline changes

- Try to anticipate the deadline
- Given $J_a \rightarrow J_b$, the following two conditions must be satisfied:
  - $f_a \leq d_a$ J_a must finish before its deadline
  - $f_a \leq d_b - c_b$ J_a must finish before latest start time of b

- **New deadline for J_a**
  $d^*_a = \min(d_a, d_b - C_b)$

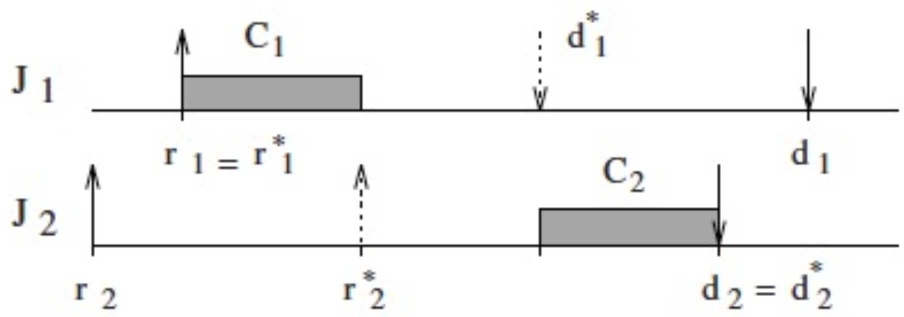


EDF with precedence constraints - example

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

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

In origin arrival time of J2 is before arrival time of J1



Graph is removed but the order of execution preserves the graph precedencies

Aperiodic task scheduling: summary

	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