Aperiodic Task Scheduling

Radek Pelánek

Tento projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem České republiky.

EVROPSKÁ UNIE

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

OP Vzdělávání pro konkurenceschopnost

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Preemptive Scheduling: The Problem

- 1 processor
- arbitrary arrival times of tasks
- preemption
- performance measure: maximum lateness
- no resources, no precedence constraints

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

1. Solve the example (manually).
2. Try to find out a scheduling algorithm.

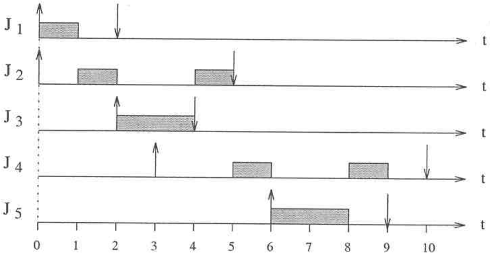
Earliest Deadline First Algorithm

EDF

At any instant execute the task with the earliest absolute deadline among all the ready tasks.

Example – 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



Optimality of EDF

Theorem (Horn)

Given a set of n independent tasks with arbitrary arrival times, the EDF algorithm is optimal with respect to minimazing the maximum lateness.

Proof of Horn's Theorem

Basic idea of the proof:

- let σ be optimal schedule; we transform it into EDF schedule σ_{EDF} without increasing maximum lateness
- schedule is divided into time slices of 1 unit
- transformation: interchange 1 appropriate time slice

Proof of Horn's Theorem (cont.)

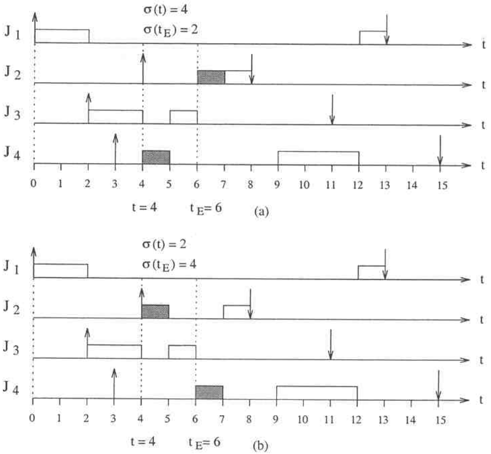


Figure 3.4 Proof of the optimality of the EDF algorithm. **a.** schedule σ at time $t = 4$. **b.** new schedule obtained after a transposition.

Guarantee Test

- is the set of tasks schedulable?
- sort tasks by increasing deadlines
- synchronous activation ($a_i = 0$) – schedulability guaranteed by conditions:

$$\forall i : \Sigma_{k=1}^{i} C_k \leq d_i$$

- asynchronous activation: "dynamical version" of the previous test

EDF is Fine ...

EDF:

- is optimal
- works on-line
- easy to implement
- simple guarantee test

… but Not a Silver Bullet

- EDF is not optimal with more than 1 processor
- Try to find a specific set of tasks:
    - the set is schedulable on 2 processor
    - EDF schedule misses some deadline

EDF with More Processors

2 Processors

	J_1	J_2	J_3
a_i	0	0	0
C_i	1	1	5
d_i	1	2	5

Schedulable, but EDF schedule misses deadline for J_3.

Least Slack Time Algorithm

LST

At any instant execute the task with the least slack time (that is $d_i - C_i$) among all the ready tasks.

LST is also optimal.

Example

Find a set of task such that EDF and LST produce different schedules.

Non-preemptive Scheduling: The Problem

- 1 processor
- arbitrary arrival times of tasks
- preemption not allowed
- performance measure: maximum lateness
- no resources, no precedence constraints

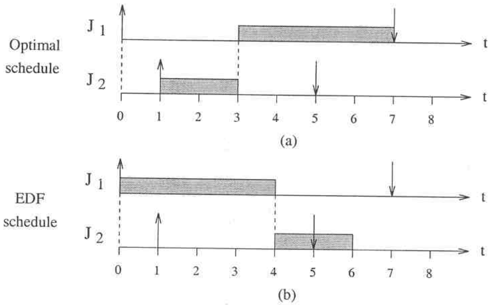
Non-optimality of EDF

EDF is not optimal for non-preemptive scheduling.

Find a set of task such that EDF does not produce optimal schedule.

Non-optimality of EDF

	J_1	J_2
a_i	0	1
C_i	4	2
d_i	7	5



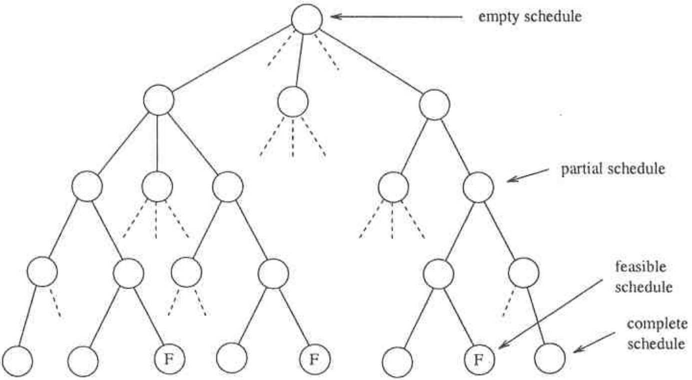
EDF and Non-idle Schedules

- non-idle algorithm - does not permit the processor to be idle when there are active jobs
- restriction to non-idle algorithms $\Rightarrow$ EDF is optimal

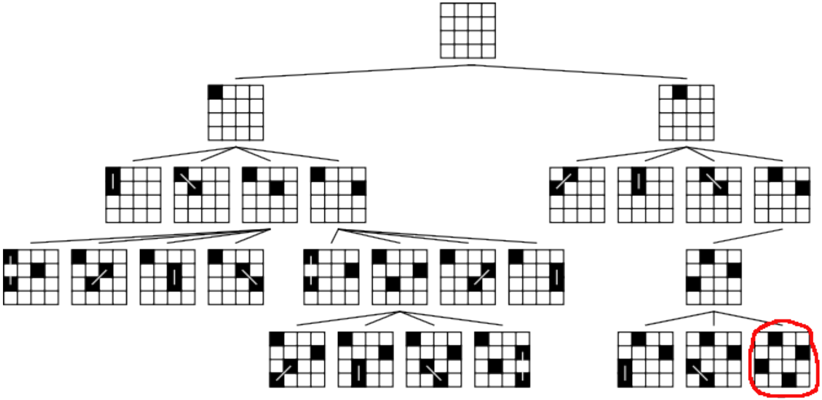
Optimal Scheduling

- no on-line algorithm can generate optimal schedule (example above: time 0, stay idle or start J_1?)
- off-line: non-preemptive scheduling is NP-complete
- branch-and-bound (backtracking) algorithms, worst case complexity $O(n \cdot n!)$

Search Tree


empty schedule
partial schedule
feasible schedule
complete schedule
F
F
F

Reminder: Backtracking, n Queen Problem



Pruning

- branch is abandoned when:
  - the addition of any node to the current path causes a missed deadline
  - a feasible schedule is found at the current path
- size of the search tree is exponential in the worst case
- significant pruning in the average case

Pruning: Example

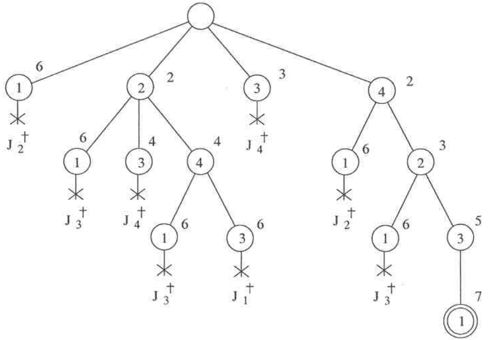
	J_1	J_2	J_3	J_4
a_i	4	1	1	0
C_i	2	1	2	2
d_i	7	5	6	4

Number in the node = scheduled task

Number outside the node = finishing time

$J_i^{\dagger}$ = task that misses its deadline

$\circledcirc$ = feasible schedule



Example

	J_1	J_2	J_3	J_4
a_i	0	4	2	6
C_i	6	2	4	2
d_i	18	8	9	10

Draw the search tree (with pruning).

Scheduling with Precedence Constraints

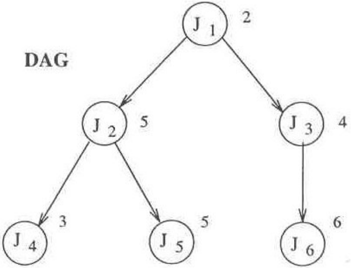
- generally NP-complete
- we consider two special cases, polynomial algorithms
- remark on heuristical approach

Precedence Constraints: Problem 1

- 1 processor
- precedence constraints
- synchronous activation ($\forall i : a_i = 0$)
- (preemption does not matter)
- performance measure: maximum lateness

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



1. solve the example (manually)
2. construct the EDF schedule
3. try to find out an optimal scheduling algorithm

Latest Deadline First Algorithm

- one possible solution: *latest deadline first* (LDF)
- note: different interpretations
  - EDF algorithm = *earliest deadline* job is scheduled to run *first*
  - LDF algorithm = *latest deadline* job is put into schedule *first*

Latest Deadline First Algorithm

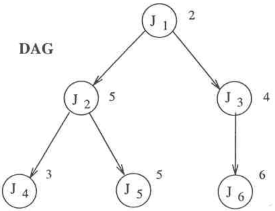
LDF

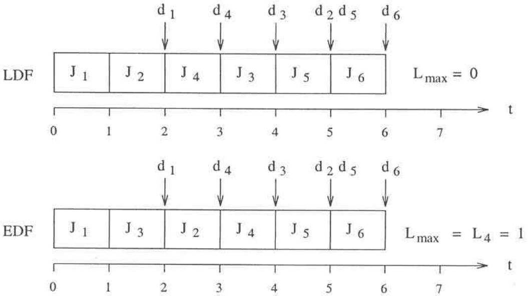
- among tasks without successors select the task with the latest deadline
- remove this task from the precedence graph and put it into a stack
- repeat until all tasks are in the stack
- the stack represents the order in which tasks should be scheduled

LDF is optimal.

LDF: 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





Precedence Constraints: Problem 2

- 1 processor
- precedence constraints
- arbitrary arrival times of tasks
- preemption
- performance measure: maximum lateness

Example

Precedence constraints:

$$A \to C; A \to D; B \to D, B \to E; E \to D; C \to F; D \to F$$

	A	B	C	D	E	F
a	0	2	5	4	1	2
C	3	2	2	3	1	3
d	8	8	13	10	5	14

Basic Idea

1. transform set J of dependent tasks into set $J*$ of independent tasks
2. apply EDF to set $J*$

transformation done by modification of arrival times and deadline times

Modification of Arrival Times

If $J_Y \rightarrow J_X$ then:

- $s_X \geq a_X$
  J_X cannot start earlier than its activation time
- $s_X \geq a_Y + C_Y$
  J_X cannot start earliear than the minimum finishing time of J_Y

Modification of Arrival Times (cont.)

new arrival time:

$$a_X* = max(a_X, max(a_Y + C_Y, J_Y \rightarrow J_X))$$

modified arrival time must be computed in the correct order (given by precedence constraints)

Modification of Deadlines

If $J_X \rightarrow J_Y$ then:

- $f_X \leq d_X$
  J_X must finish within its deadline
- $f_X \leq d_Y - C_Y$
  J_X must finish not later than the maximum starting time of J_Y

Modification of Deadlines (cont.)

new deadline:

$$d_X* = min(d_X, min(d_Y - C_Y, J_X \to J_Y))$$

modified deadline times must be computed in the correct order (given by precedence constraints)

Optimality

Theorem

There exists feasible schedule for the modified task set $J*$ under EDF if and only if the original task set is schedulable.

Example

Precedence constraints:

$A \to C; A \to D; B \to D, B \to E; E \to D; C \to F; D \to F$

	A	B	C	D	E	F
a	0	2	5	4	1	2
C	3	2	2	3	1	3
d	8	8	13	10	5	14
$a*$	0	2	5	5	4	8
$d*$	7	4	11	10	5	14

Example

- precedence constraints: $A \rightarrow C, B \rightarrow C, C \rightarrow E, D \rightarrow F, B \rightarrow D, C \rightarrow F, D \rightarrow G$
- all tasks: $a_i = 0, D_i = 25$, computation times: 2, 3, 3, 5, 1, 2, 5
- compute modified arrival times and deadlines
- compute schedule according to EDF

Heuristical search

- more complicated problems (e.g., non-preemptive, precedence constraints) NP-complete
- brute-force search (with pruning)
- heuristical search
    - only some schedules are considered
    - no guarantee of optimality
- note: general computer science approach

Heuristical search

- constructs partial schedules (like brute-force search)
- considers only one (or few) tasks for extension of a schedule
- selection based on heuristic function H

Heuristic function

$$
\begin{aligned}
H(i) &= a_i & &\text{first come first serve}\\
H(i) &= C_i & &\text{shortest job first}\\
H(i) &= d_i & &\text{earliest deadline first}
\end{aligned}
$$

- more complicated parameters (taking into account precedence relations, resources, ...)
- weighted combinations of different parameters

Overview of Problems and Algorithms

	sync. act.	preemptive async. act.	non-preemptive async. act.
independent	EDF, $O(n \log n)$	EDF, LST, $O(n^2)$	tree search, $O(n \cdot n!)$
precedence	LDF, $O(n^2)$	modified EDF, $O(n^2)$	heuristic search

Summary

- aperiodic task scheduling (arbitrary arrival times)
- 1 processor, no resources
- precedence constraints: yes/no
- preemption: yes/no
- performance measure: maximum lateness
- algorithms: earliest deadline first (EDF), least slack time (LST), branch and bound, latest deadline first (LDF), transformations