PRO5802 - Programação de Produção Intermitente (2021)

Aula 3 – Teoria de Scheduling

Prof. Daniel de Oliveira Mota

Jobs and machines

- Data (assumed to be given)

m	number of machines
n	number of jobs
p_{ij}	processing time of job j at machine i
p_j	processing time of job j at when all machines are identical
r_j	release date of job j
d_j	due date of job j
w_j	weight of job j

Description of a Scheduling Problem

$\alpha \mid \beta \mid \gamma$

machine environment

constraints,
processing,
characteristics

objective (to be minimized)

Examples:

- Paper bag factory $FF3 \mid r_j \mid \Sigma\, w_j T_j$
- Gate assignment $P_m \mid r_j, M_j \mid \Sigma\, w_j T_j$
- Tasks in a CPU $l \mid r_j\ prmp \mid \Sigma\, w_j C_j$
- Traveling Salesman $l \mid s_{jk} \mid C_{max}$

Machine environment α

- Single machine and machines in parallel

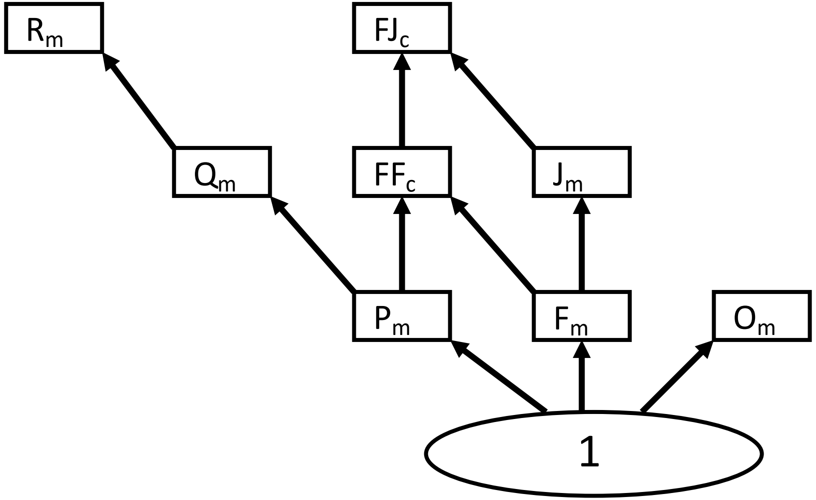
1	single machine
P_m	m identical machines in parallel
Q_m	m machines in parallel w/different speeds v_i
R_m	m unrelated machines in parallel

Machine environment α (2)

- Machines in series

F_m	flow shop: all jobs processed in the same order on the machines
FF_c	flexible flow shop: same as flow shop but with c stages of parallel machines
J_m	job shop: each job has its own routing
FJ_c	flexible job shop: same as job shop but with c stages of parallel machines
O_m	open shop: each job has to processed on all machines but no routing restrictions

Machine Environment



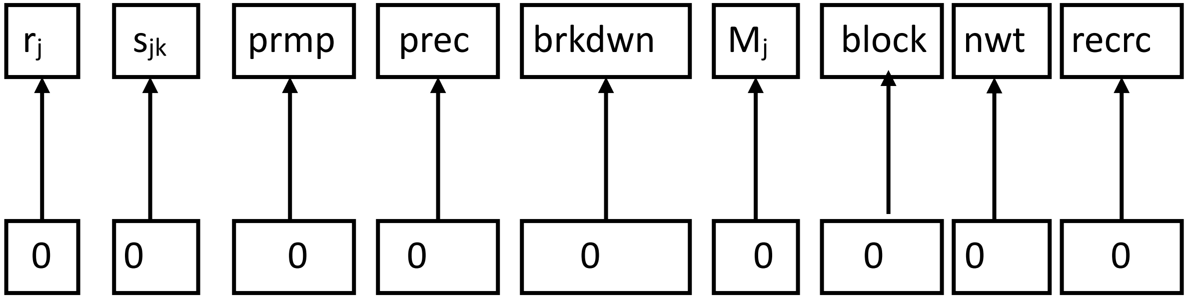
Processing characteristics and constraints β

	could be empty!
r_j	Release dates
s_{jk}	sequence dependent setup times
s_{ijk}	sequence and machine dependent setup times
prmp	preemption
prec	precedence constraints

Processing characteristics and constraints β (2)

brkdwn	breakdowns
M_j	machine eligibility restrictions
prmu	permutation
block	blocking
nwt	no waiting
recrc	recirculation

Processing Restrictions and Constraints



Objectives γ

- Performance measures of individual jobs

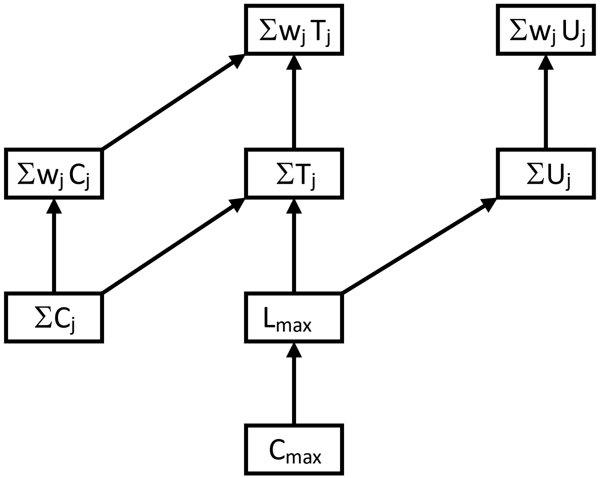
C_j	completion time of job j
L_j	lateness = $C_j - d_j$
T_j	tardiness = max(L_j, 0)
E_j	earliness = max(-L_j, 0)
U_j	unit penalty = 1 if $C_j > d_j$ and 0 otherwise
$h_j(C_j)$	h_j is a non-decreasing cost function

Objectives γ (2)

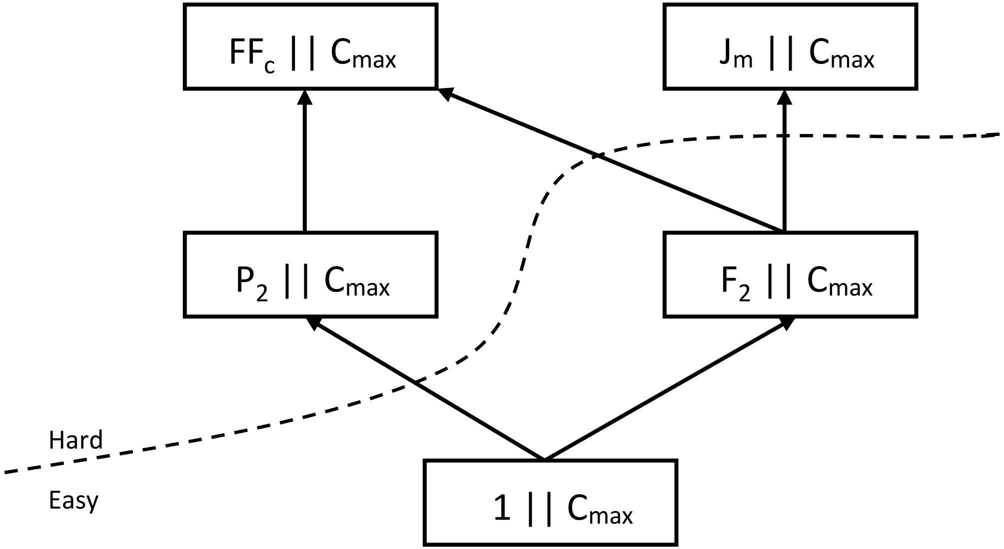
- Functions to be minimized

$C_{\max} = \max C_j$	makespan
$L_{\max} = \max L_j$	maximum lateness
$\Sigma w_j C_j$	total weighted completion time
$\Sigma w_j (1-e^{-rC_j})$	total weighted discounted C_j
$\Sigma w_j T_j$	total weighted tardiness
$\Sigma w_j U_j$	weighted number of tardy jobs
$\Sigma w_j' E_j + \Sigma w_j'' T_j$	total weighted earliness and tardiness

Objective Functions



Complexity of Makespan Problems



Classes of Schedules

- Nondelay (greedy) schedule
  - No machine is kept idle while a task is waiting for processing.

  An optimal schedule need not be nondelay!

Example: $P_2 \mid \mid C_{max}$

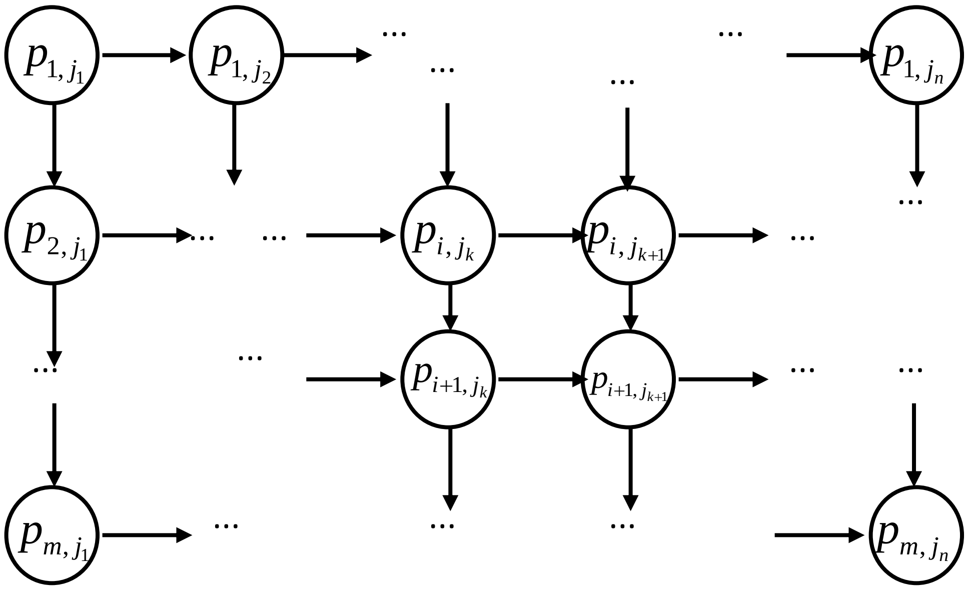
jobs	1	2	3	4	5	6	7	8	9	10
p_j	8	7	7	2	3	2	2	8	8	15

Como funciona na prática?

Flow Shops

- Each job must follow the same route.
  - There is a sequence of machines.
- There may be limited buffer space between neighboring machines.
  - The job must sometimes remain in the previous machine: **Blocking**.
- The main objective in flow shop scheduling is the makespan.
  - It is related to utilization of the machines.
- If the First-come-first-served principle is in effect, then jobs cannot pass each other.
  - **Permutation flow shop**

Directed Graph for $F_m|prmu|C_{max}$

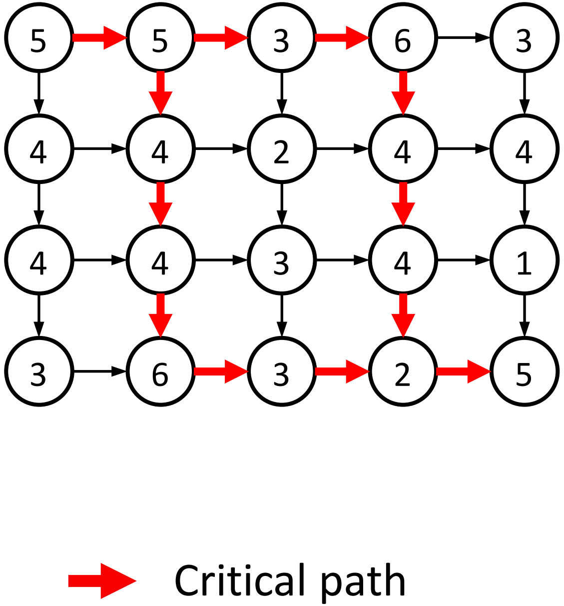


Example F4|prmu|C_{max}

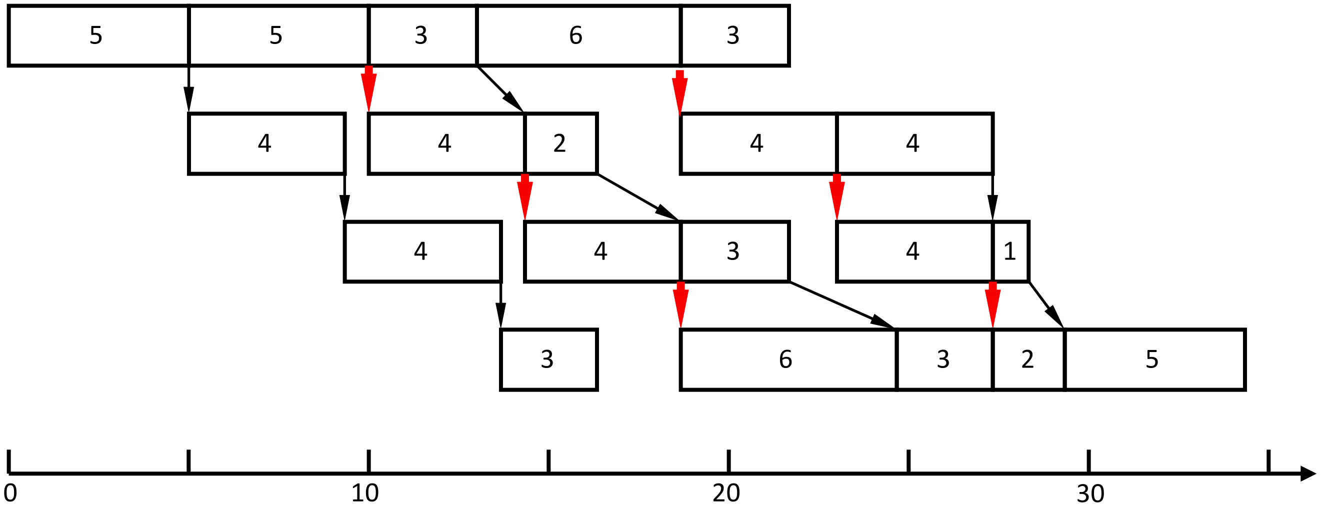
5 jobs on 4 machines with the following processing times

jobs	j_1	j_2	j_3	j_4	j_5
p_{1,j_k}	5	5	3	6	3
p_{2,j_k}	4	4	2	4	4
p_{3,j_k}	4	4	3	4	1
p_{4,j_k}	3	6	3	2	5

Directed Graph in the Example



Gantt Chart in the Example



Slope Heuristic

- Slope index A_j for job j

$$A_j = -\sum_{i=1}^{m}(m-(2i-1))p_{ij}$$

- Sequencing of jobs in decreasing order of the slope index
- Consider 5 jobs on 4 machines with the following processing times

jobs	j_1	j_2	j_3	j_4	j_5
p_{1,j_k}	5	2	3	6	3
p_{2,j_k}	1	4	3	4	4
p_{3,j_k}	4	4	2	4	4
p_{4,j_k}	3	6	3	5	5

A_1 = – (3 x 5) – (1 x 4) + (1 x 4) + (3 x 3) = – 6

A_2 = – (3 x 5) – (1 x 4) + (1 x 4) + (3 x 6) = +3

A_3 = – (3 x 3) – (1 x 2) + (1 x 3) + (3 x 3) = +1

A_4 = – (3 x 6) – (1 x 4) + (1 x 4) + (3 x 2) = – 12

A_5 = – (3 x 3) – (1 x 4) + (1 x 1) + (3 x 5) = +3

Sequences 2,5,3,1,4 and 5,2,3,1,4 are optimal and the makespan is 32.

Classification of Scheduling Problems

5 job single machine exercise

minimize $\Sigma w_j C_j$

where C_j is the completion time of job j,

p_j is the processing time of job j,

w_j is the weight (priority) of job j

j	p_j	w_j
1	4	3
2	1	1
3	3	10
4	10	15
5	2	4

Regular objective functions

- Regular objective functions
  - non-decreasing in $C_1, \ldots, C_n$
  - most objective functions considered in this class are regular
- Non-regular objective functions
  - Example: $\Sigma w_j' E_j + \Sigma w_j'' T_j$
  - Much harder to solve!

Discussion on complexity

What does complexity mean?

- Complexity is an indication of how much <u>computation</u> is required to solve a problem
- Significance of the complexity of a scheduling problem
  - Does an efficient algorithm for solving the problem exist?
  - Worst case analysis

Problems and instances

- A <u>problem</u> is the generic description of a problem
- An <u>instance</u> refers to a problem with a given set of data
- The <u>size</u> of an instance refers to the length of the data string necessary to specify the instance (on a computer)
  - In this class the size of an instance is usually measured in the number of jobs n

The classes $\mathcal{P}$ and $\mathcal{NP}$

- Class $\mathcal{P}$
  - The class $\mathcal{P}$ contains all decision problems for which there exists a Turing machine algorithm that leads to the right yes-no answer in a number of steps bounded by a polynomial in the length of the encoding
- Class $\mathcal{NP}$
  - The class $\mathcal{NP}$ contains all decision problems for which the correct answer, <u>given a proper clue</u>, can be verified by a Turing machine in a number of steps bounded by a polynomial in the length of the encoding

Problem reduction

- Problem P <u>polynomially reduces</u> to problem P’ if a polynomial time algorithm for P’ implies a polynomial time algorithm for problem P
  - Denoted P $\propto$ P’
  - P’ is at least as hard as P

The classes $\mathcal{NP}$- hard and $\mathcal{NP}$- complete

- Class $\mathcal{NP}$- hard
  - A problem P is called $\mathcal{NP}$- hard if the entire class $\mathcal{NP}$ polynomially reduces to problem P
  - Problem P is at least as hard as all the problems in $\mathcal{NP}$
- Class $\mathcal{NP}$- complete (not in the textbook)
  - A problem P is called $\mathcal{NP}$- complete if it is both in classes $\mathcal{NP}$ and $\mathcal{NP}$–hard

Pseudopolynomial algorithms

- Polynomial time algorithms exist for some $\mathcal{NP}$-hard problems under the appropriate encoding of the problem data
- Such problems are referred to as <u>$\mathcal{NP}$-hard in the ordinary sense</u> and the algorithms are called <u>pseudopolynomial</u>
- Problem P is called <u>strongly $\mathcal{NP}$-hard</u> if a pseudopolynomial algorithm for it does not exist

Some $\mathcal{NP}$-hard problems

- $\mathcal{NP}$-hard in the ordinary sense
  - PARTITION
- Strongly $\mathcal{NP}$-hard
  - SATISFIABILITY
  - 3-PARTITION
  - HAMILTIONIAN CIRCUIT
  - CLIQUE

https://news.mit.edu/2009/explainer-pnp

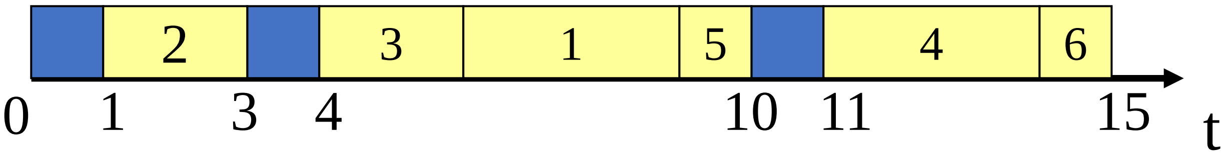
http://www.claymath.org/sites/default/files/pvsnp.pdf

- Mãos à obra!!!

Example 2 ($1|batch|\Sigma w_i C_i$)

i	1	2	3	4	5	6
p_i	3	2	2	3	1	1
w_i	1	2	1	1	4	4

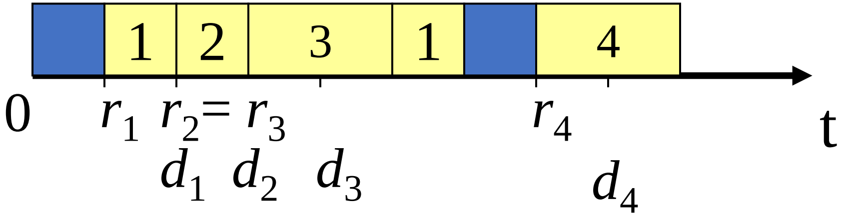
$s = 1$



$$\Sigma w_i C_i = 2 \cdot 3 + (1+4+4) \cdot 10 + (1+4) \cdot 15 = 171$$

Example 3 $(1|r_i; pmtn|L_{max})$

i	1	2	3	4
p_i	2	1	2	2
r_i	1	2	2	7
d_i	2	3	4	8



$L_{max} = 4$

Example 4 ($J3 \mid p_{ij}=1 \mid C_{max}$)

$i \backslash j$	1	2	3	4
1	M_1	M_3	M_2	M_1
2	M_2	M_3		
3	M_3	M_1		
4	M_1	M_3	M_1	
5	M_3	M_1	M_2	M_3