Chapter 2
Deterministic Models: Preliminaries

Dr. Purush Damodaran
Dept. Industrial & Systems Engineering

pdamodaran@niu.edu
815.753.3172

Assumptions

- Finite number of jobs and machines
  - n jobs and m machines
  - Jobs – j $\{ j = 1,\ldots,n \}$
  - Machines – i $\{ i = 1,\ldots,m \}$

Notation

- p_{ij} – processing time of job j on machine i
- r_j – release date of job j
- d_j – due date of job j
  - A penalty is incurred whenever a job is completed after its due date
- $\underline{d}_j$ – deadline of job j
  - Due date must be met
- w_j – weight of job j
  - Priority of job
  - Inventory holding cost
  - Amount of value added

Notation

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

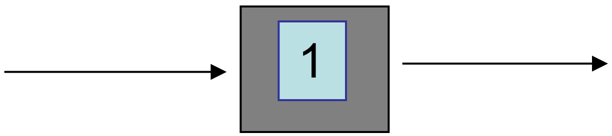
Machine environment

Objective

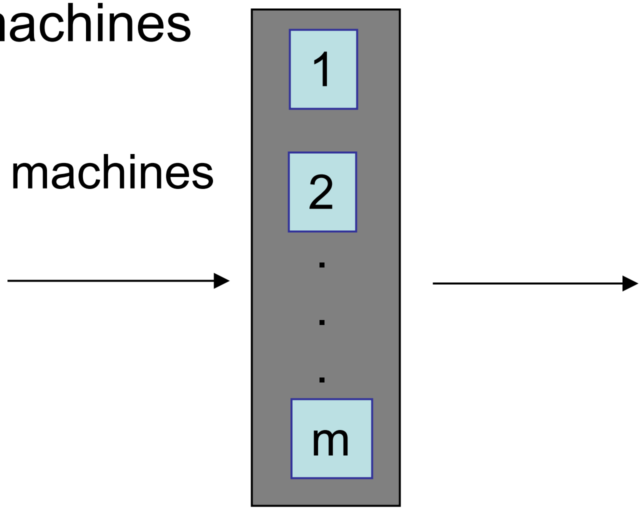
Processing characteristics and Constraints

Machine Environment (α)

- Single machine (*1*)

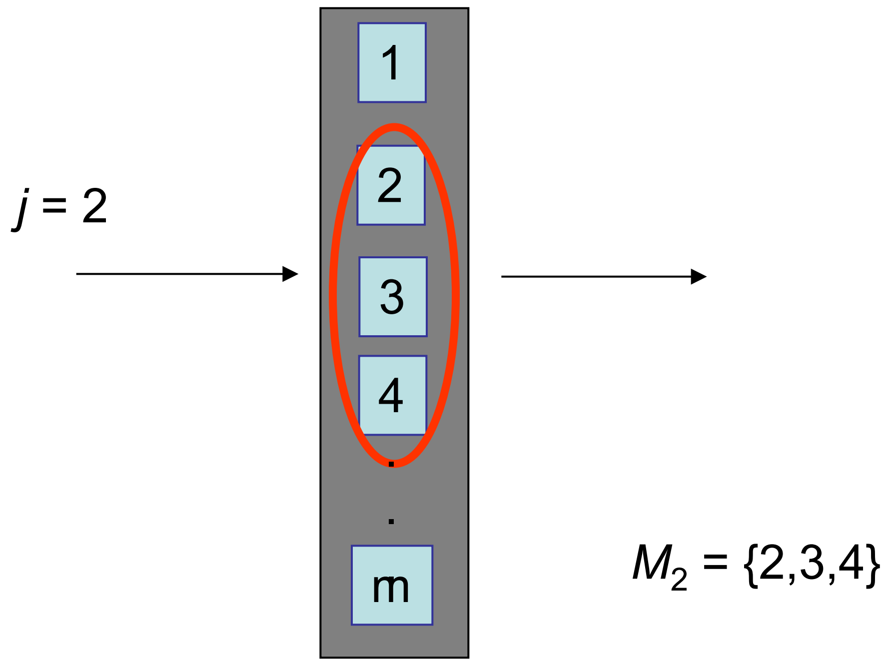


- *m identical* machines in parallel (*Pm*)
  - Job *j* is processed on anyone of the *m* machines
  - All the machines are identical
    - Processing time of job *j* is identical on all machines



Machine Environment (α)

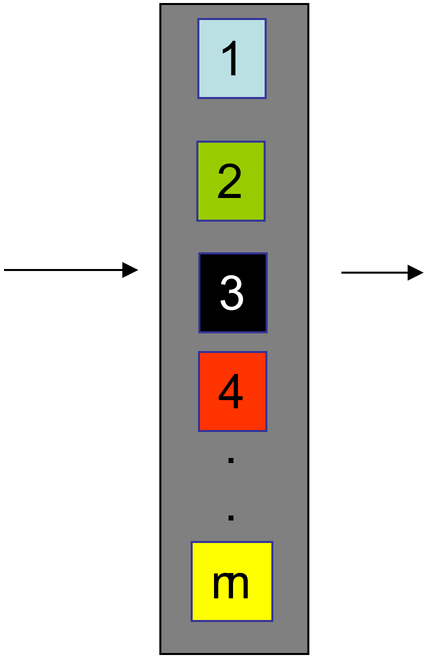
- *m* identical machines in parallel
  - Job *j* can be processed only on a subset of machines (M_j)
    - M_j is specified in the β field



$$Pm \mid M_j \mid \gamma$$

Machine Environment (α)

- *m* machines in parallel with different speeds (*Qm*)
  - Processing time of job *j* is p_j
  - Speed of machine *i* is v_i
    - Processing time of job *j* on machine *i* is $p_{ij} = p_j / v_i$
  - a.k.a uniform machines environment
  - When $v_i = 1$ for all *i*
    - Uniform machines ↔ identical machines



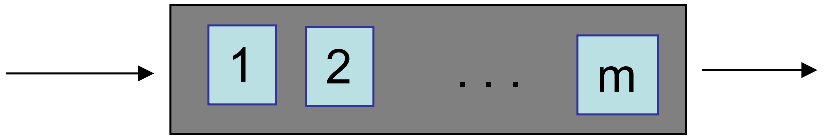
Machine Environment (α)

- *m* unrelated machines in parallel (*Rm*)
  - Speed of machine i for job j is v_{ij}
  - Processing time of job j on m/c i is $p_{ij} = p_j / v_{ij}$
  - When $v_{ij} = v_i$ for all i and j
    - Unrelated machines ↔ uniform machines

Machine Environment (α)

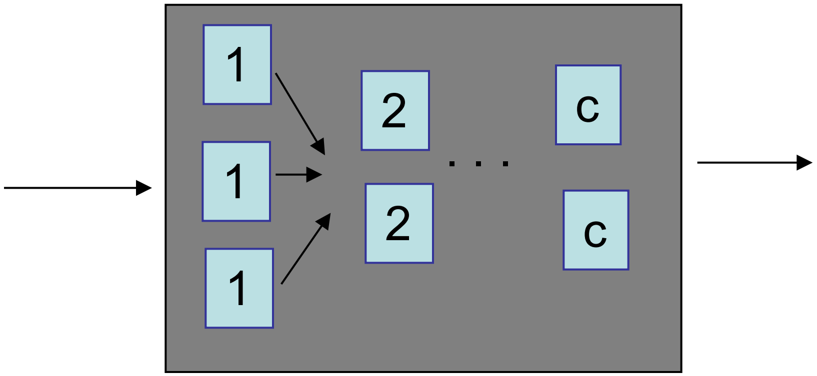
- Flow shop (*Fm*)
  - Process route 1→ 2 → … → *m*
  - *m* stages in this flow shop
  - Each stage has just one machine
  - Queues
    - FIFO – first in first out discipline
    - Permutation flow shop
      - *prmu* is included in the β field

FM | *prmu* | γ



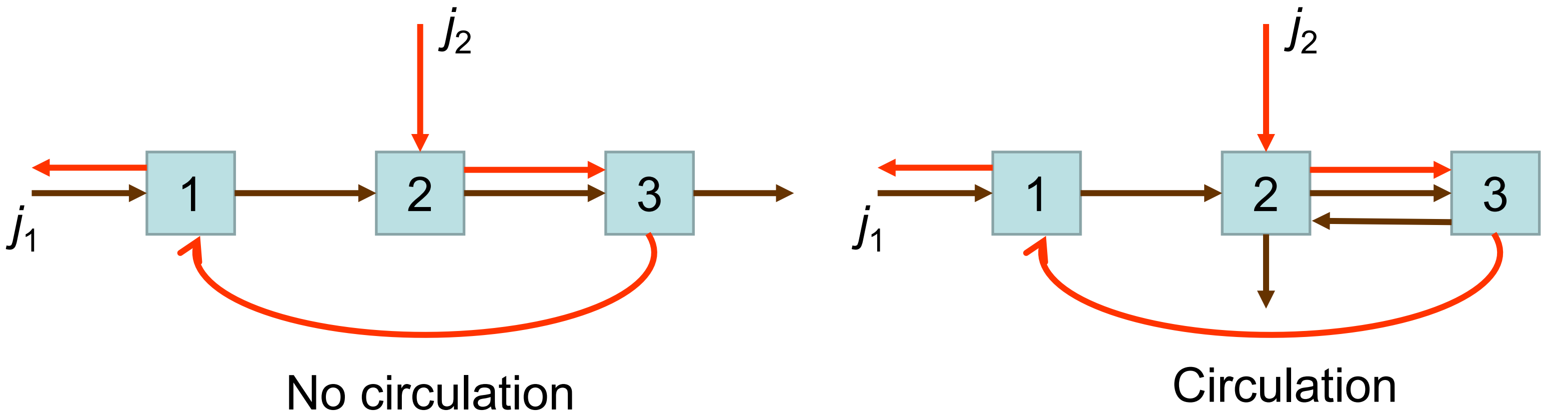
Machine Environment (α)

- Flexible flow shop (*FFc*)
  - Flow shop + parallel machine environment
    - *c* stages
    - Each stage has several identical parallel machines
    - Some stages can have just 1 machine



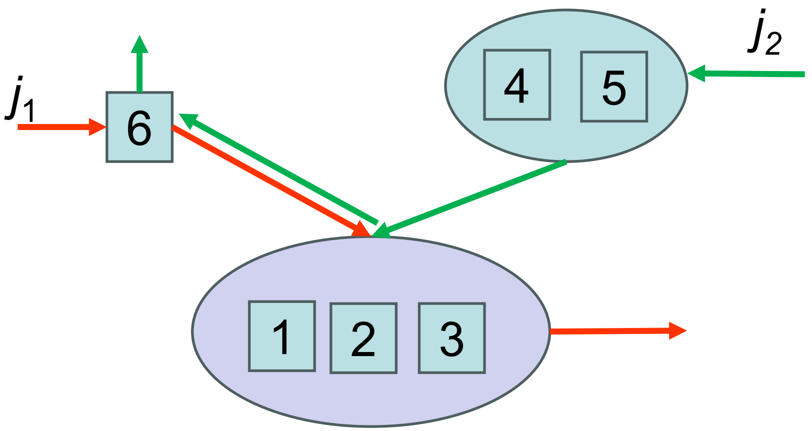
Machine Environment (α)

- Job shop (*Jm*)
  - *m* machines
  - Each job has its own route
    - No circulation – each machine visited once
    - Recirculation – some machines visited more than once
      - *recrc* is included in the β field



Machine Environment (α)

- Flexible Job Shop (*FJc*)
  - Job shop + parallel machine environment
  - *c* work centers
    - Each work center may have more than 1 machine
  - Each job has a predetermined process route
    - No circulation – each center visited once
    - Recirculation – some centers visited more than once
      - *recrc* is included in the β field



Machine Environment (α)

- Open shop (*Om*)
  - *m* machines
  - Each job processed on each one of the *m* machines
    - Some processing times can be zero
  - No restriction w.r.t routing of each job

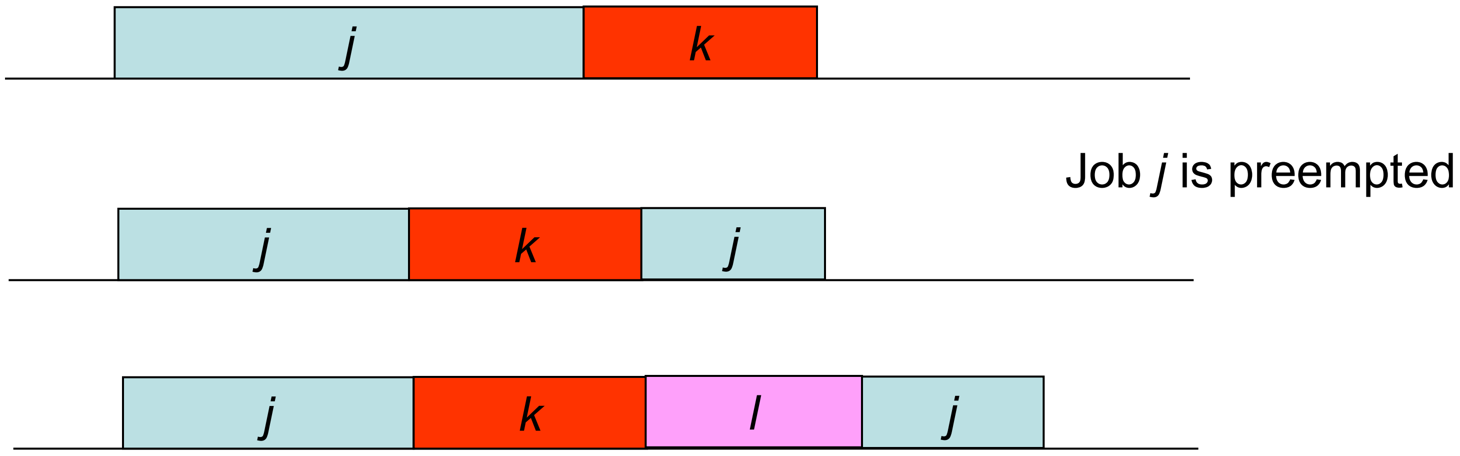
β Field

- Release dates (r_j)
  - The job can be processed only after its release date
  - If r_j is not mentioned, jobs can be scheduled anytime

- Sequence dependent setup times (s_{jk})
  - Setup time is required between adjacent jobs *j* and *k*
  - s_{0j} – setup time if job *j* is the first job scheduled
  - s_{k0} – cleanup time if job *k* is the last job scheduled
  - s_{ijk} – setup time on machine *i* for jobs *j* and *k*
  - If s_{jk} is not mentioned, setup time is zero

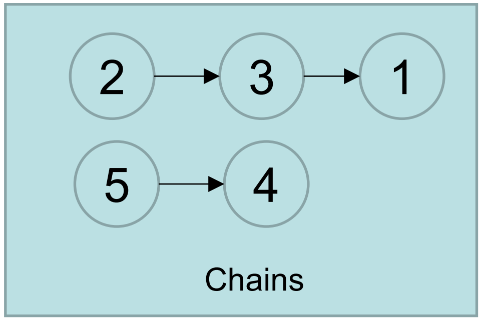
β Field

- Preemptions (*prmp*)
  - Processing of a job can be interrupted (preempted)
  - If *prmp* is not included, preemption is not allowed



β Field

- Precedence constraints (*prec*)
  - Typical in single or parallel machine environment
  - One or more jobs have to be completed before start processing another job
  - Special cases:
    - Chains – at most one successor and one predecessor
    - Intree – at most one successor
    - Outtree – at most one predecessor
  - If *prec* is not included, no precedence constraints



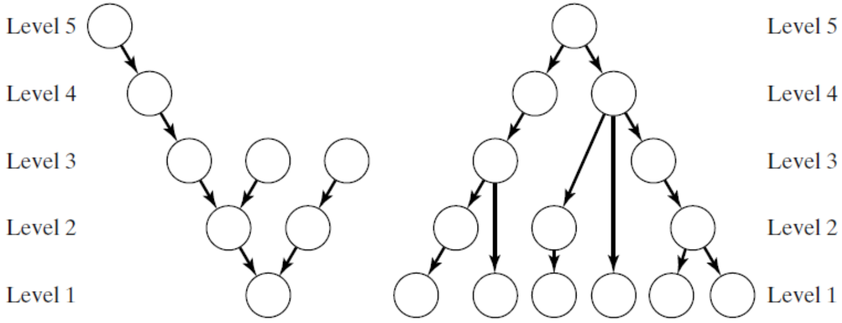


Fig. 5.3 Intree and outtree

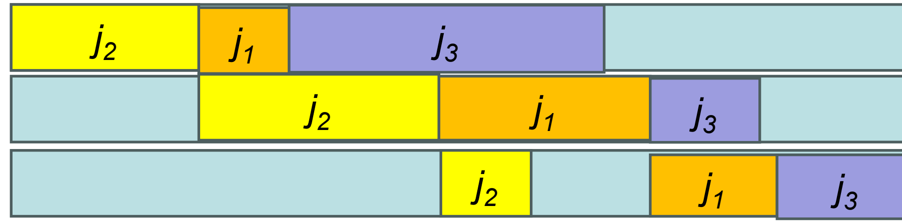
β Field

- Breakdowns (*brkdwn*)
  - Machine breaks down

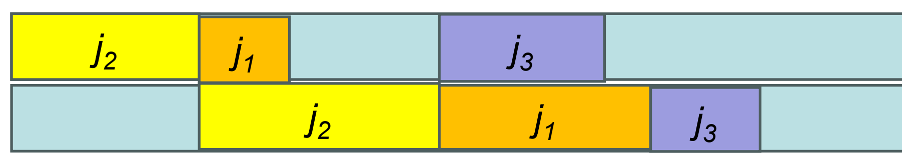
- Machine eligibility restrictions (M_j)
  - Used in parallel machine environment
  - M_j denotes the set of machines capable of processing job j
  - If M_j is not mentioned, all the m machines are capable of processing the job j

β Field

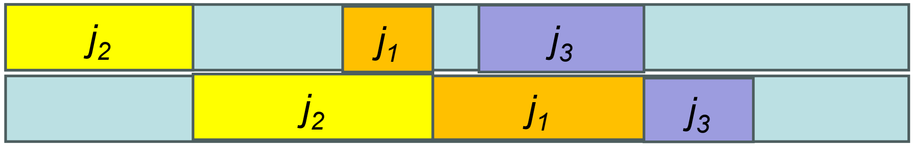
- Permutation (*prmu*)
  - Flow shop environment
  - Queue discipline – FIFO
  - The order (or permutation) in which the jobs are processed are same on all the machines



- Blocking (*block*)
  - Flow shop environment
  - Limited buffer between two successive machines
  - When the buffer is full, the upstream machine is not allowed to release a completed job

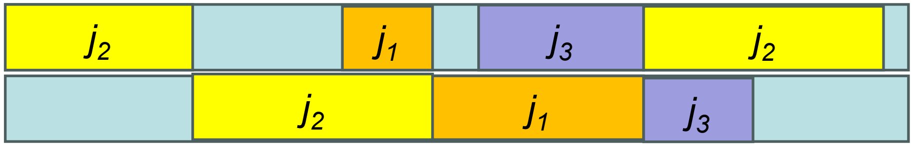


β Field



- No-wait (*nwt*)
  - Flow shop environment
  - Jobs are not allowed to wait between two successive machines
  - FIFO discipline

- Recirculation (*recrc*)
  - Job shop or flexible job shop
  - A job may visit a machine more than once



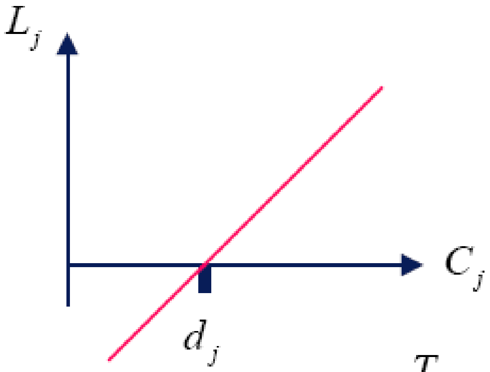
β Field

- Other entries
  - Self explanatory
  - $p_j = p$
  - $d_j = d$
  - …

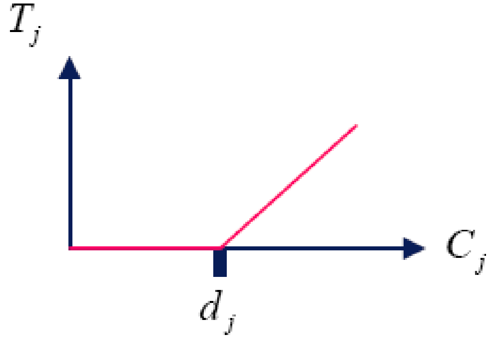
Objective

- Completion time of job j (C_j)
  - Completion time of job j on the last machine
  - Completion time of job j on machine i is C_{ij}

- Lateness of job j (L_j)
  - $L_j = C_j - d_j$
    - > 0, when late
    - < 0, when early

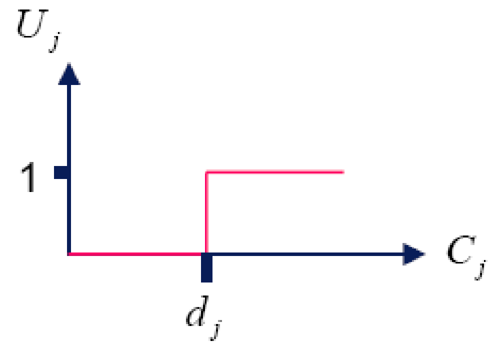


- Tardiness of job j (T_j)
  - $T_j = \max(C_j - d_j, 0) = \max(L_j, 0)$



Objective

- Unit penalty of job *j* (*U_j*)
  - $U_j = 1$, if $C_j > d_j$; 0, otherwise
  - Number of tardy jobs



- Due date related objectives
  - Lateness, tardiness, unit penalty

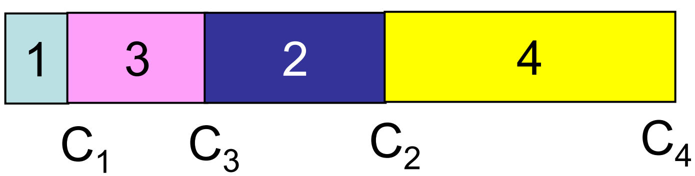
Objective

- Makespan (C_{max})
  - $C_{max} = \max(C_1, C_2, \ldots, C_n)$
  - Completion time of the last job to leave the system
  - Minimizing makespan will improve machine utilization

- Maximum lateness (L_{max})
  - $L_{max} = \max(L_1, L_2, \ldots, L_n)$
  - Worst violation of due dates

Objective

- Total completion time ($\Sigma_j C_j$)
  - A.k.a. flow time
- Total weighted completion time ($\Sigma_j w_j C_j$)
  - Weighted flow time
  - Total holding or inventory cost



$$C_{max} = C_4$$

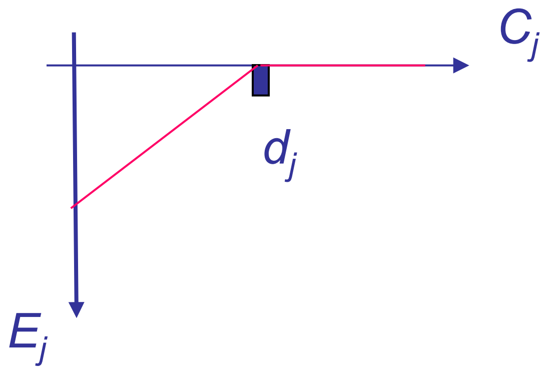
$$\Sigma_j C_j = C_1 + C_2 + C_3 + C_4$$

$$\Sigma_j w_j C_j = w_1C_1 + w_2C_2 + w_4C_3 + w_4C_4$$

Objective

- Total weighted tardiness ($\Sigma_j w_j T_j$)
- Weighted number of tardy jobs ($\Sigma_j w_j U_j$)

- Regular performance measures
  - Non-decreasing in $C_1, C_2, \ldots, C_n$
  - Consider earliness of job *j*
    - $E_j = \max(d_j - C_j, 0)$
    - Non-increasing in $C_1, C_2, \ldots, C_n$
    - Not a regular measure



Objective

- Other non-regular performance measures

- Total earliness + total tardiness
  - $\Sigma_j E_j + \Sigma_j T_j$

- Total weighted earliness + total weighted tardiness
  - $\Sigma_j w'_j E_j + \Sigma_j w''_j T_j$

Examples for Notation

- FFc | r_j | $\Sigma_j\, w_j T_j$
  - Flexible flow shop
  - Jobs have release times and due dates
  - Objective: minimize total weighted tardiness

- 1 | r_j, *prmp* | $\Sigma_j\, w_j C_j$
  - One machine
  - Jobs have release dates
  - Preemption is allowed
  - Objective: minimize total weighted completion times

Examples for Notation

- $1 \mid s_{jk} \mid C_{max}$
  - One machine
  - Sequence dependent setup times
  - Objective: minimize makespan

- Other practical applications may not be represented with this notation

Classes of Schedules

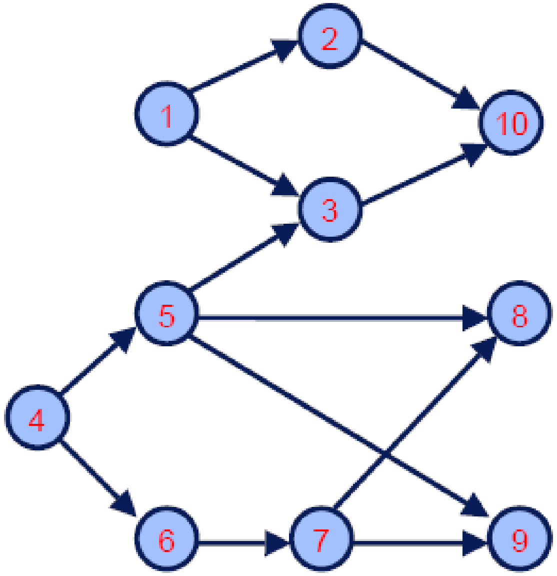
- Sequence, schedule and scheduling policy
  - Sequence – permutation or order in which the jobs are processed
  - Schedule – allocation of jobs in a more complex setting of machines. Preemptions of jobs by other jobs that are released at later points in time
  - Scheduling policy – only in stochastic setting

Class of Schedules

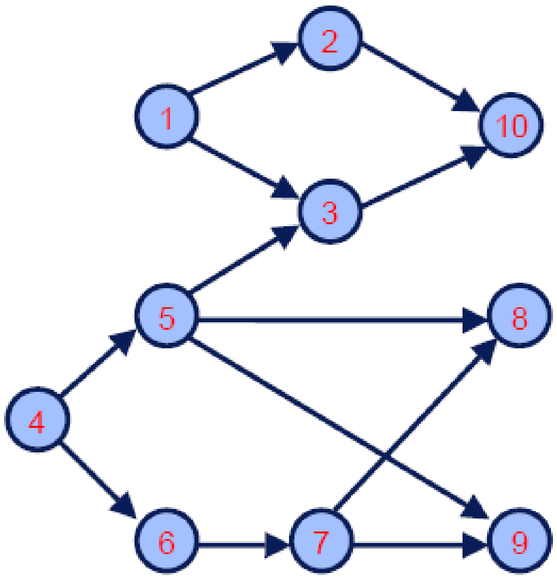
- Non-delay schedule
  - No machine is kept idle when a task is waiting for processing
  - Prohibits unforced idleness

- Example: P2 | *prec* | C_{max}

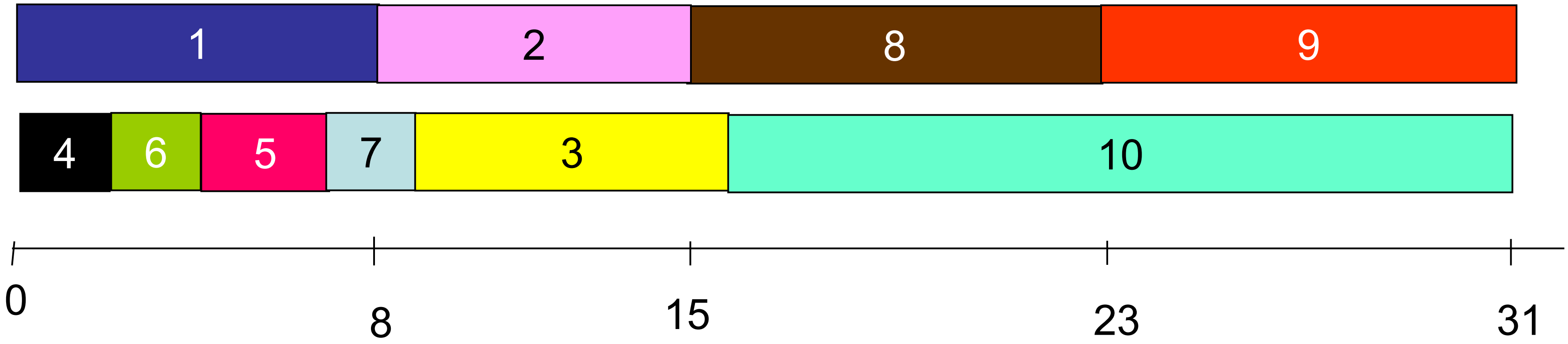
jobs	1	2	3	4	5	6	7	8	9	10
p_j	8	7	7	2	3	2	2	8	8	15



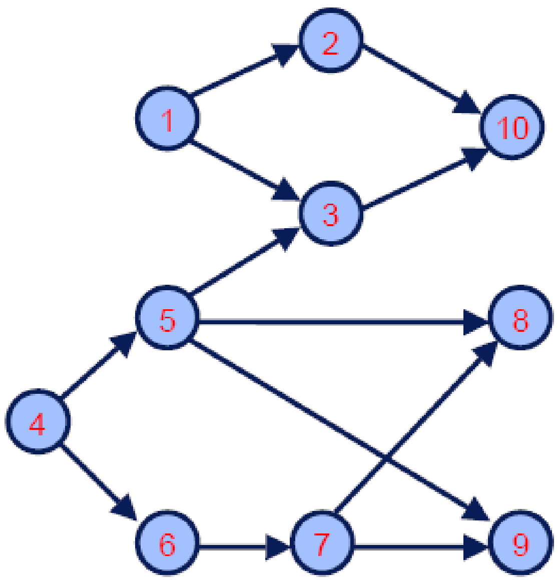
Non-delay Schedule Example



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



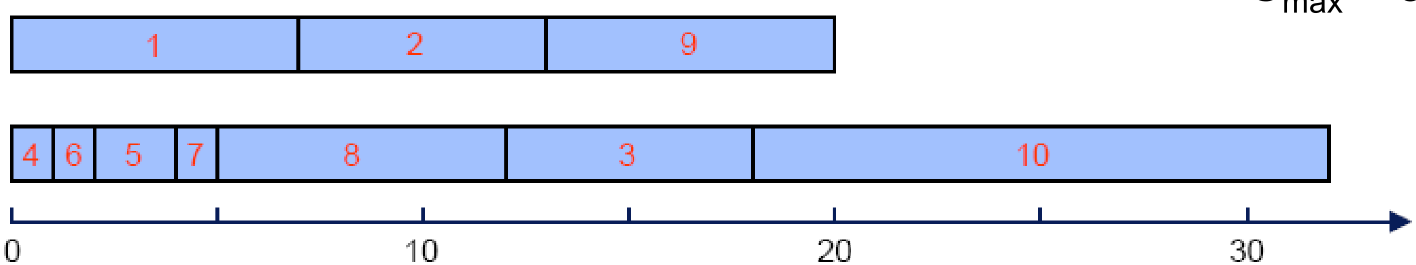
Non-delay Schedule Example



Processing time reduced by 1 unit

jobs	1	2	3	4	5	6	7	8	9	10
p_j	7	6	6	1	2	1	1	7	7	14

$C_{max} = 32$

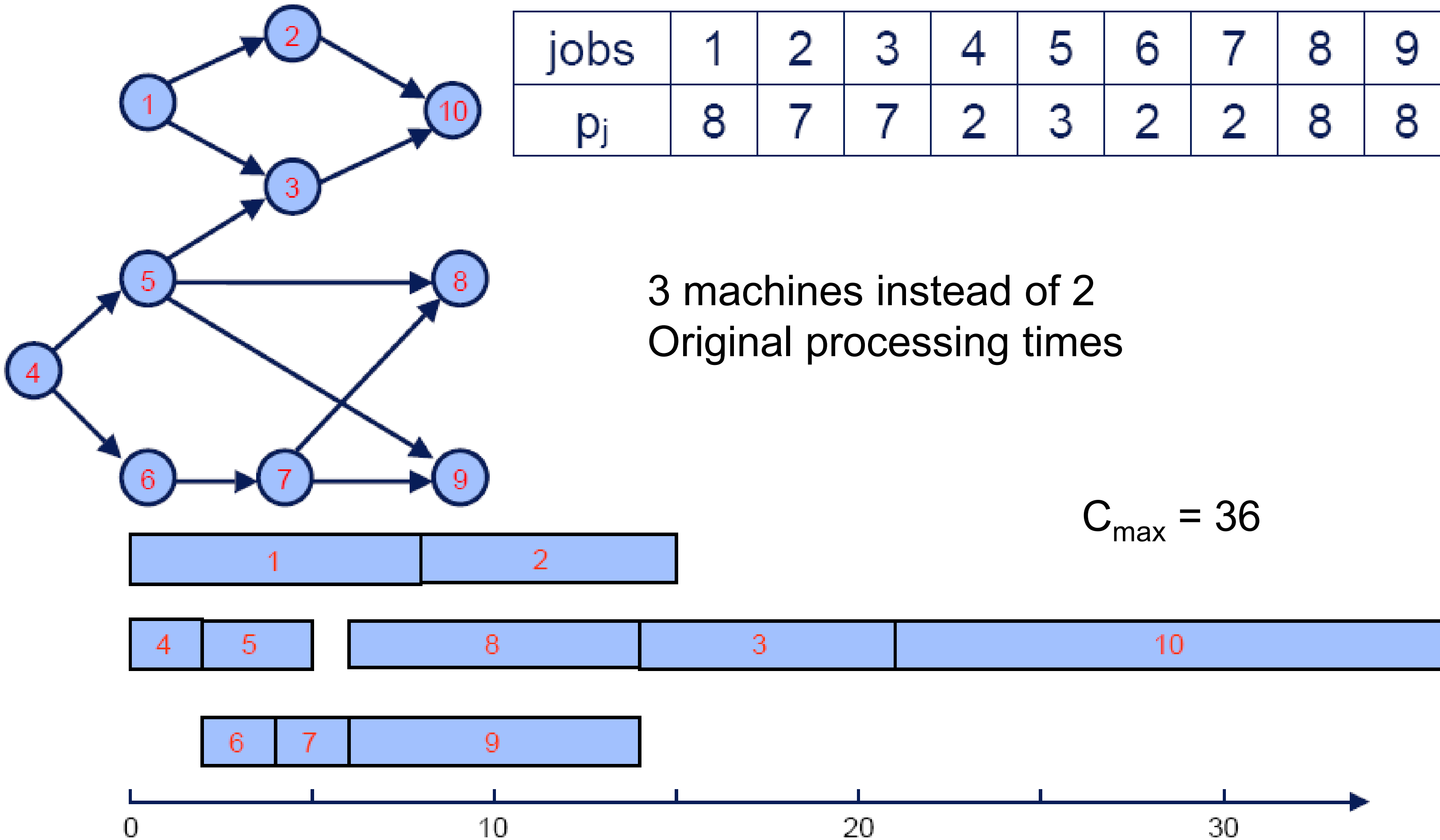


Non-delay Schedule Example

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

3 machines instead of 2
Original processing times

$C_{max} = 36$



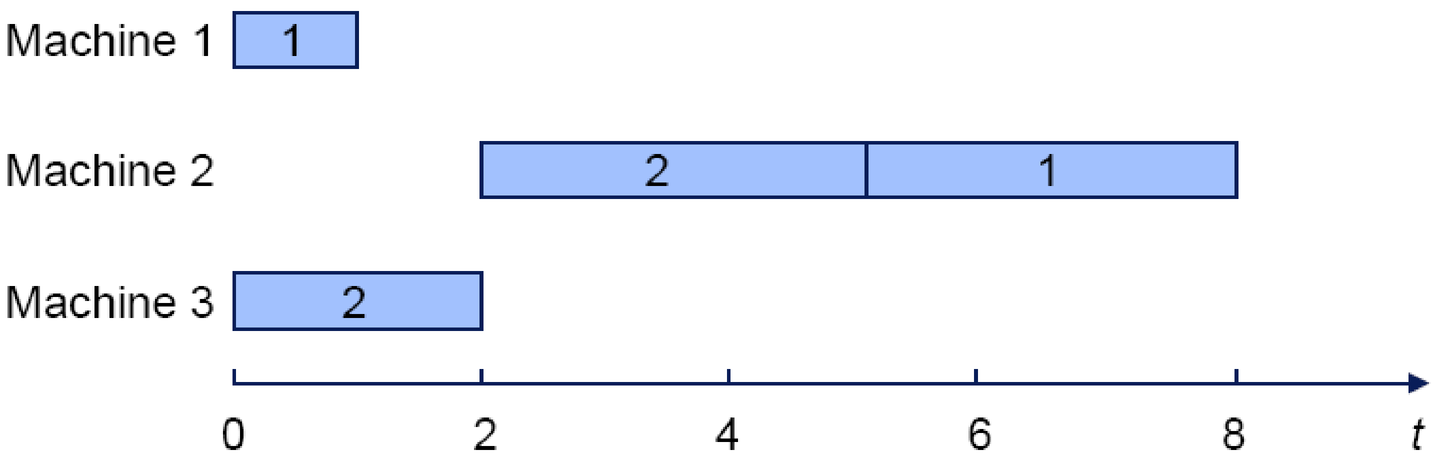
Non-delay schedules are not always the best

Active Schedule

- A feasible schedule is active
  - If it is not possible to construct another schedule by changing the order of processing on the machines and
    - Have at least one operation finishing earlier, and
    - No operation finishing later

Active Schedule Example

- J3 || C_{max}
  - Job 1 needs 1 time unit on m/c 1 and 3 time units on m/c 2
  - Job 2 needs 2 time units on m/c 3 and 3 time units on m/c 2
  - Last operation is on m/c 2



Reversing the sequence of the two jobs on m/c2 postpones the processing of job 2. So the above schedule is an active schedule

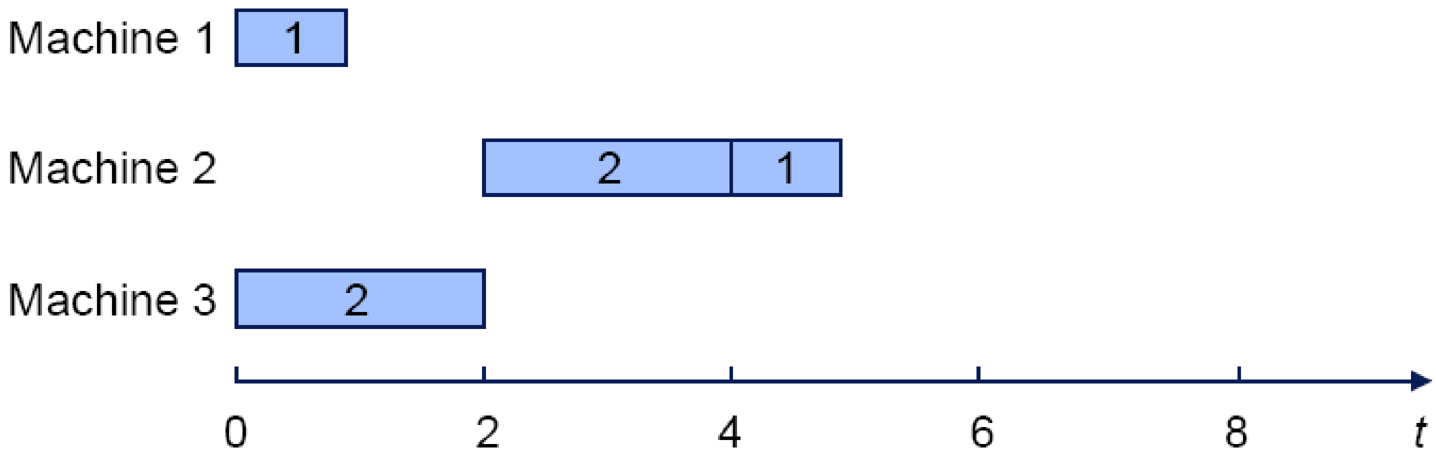
The above schedule is not a non-delay schedule.

Semi-Active Schedule

- A feasible schedule is semi-active if no operation can be completed earlier without changing the order of processing on any one of the machines

Semi-Active Schedule Example

- J3 || C_{max}
  - Job 1 needs 1 time unit on m/c 1 and 1 time units on m/c 2
  - Job 2 needs 2 time units on m/c 3 and 2 time units on m/c 2
  - Last operation is on m/c 2



This is a semi-active schedule, but not active. Job 1 can be processed on machine 2 without delaying the job 2

Class of Schedules

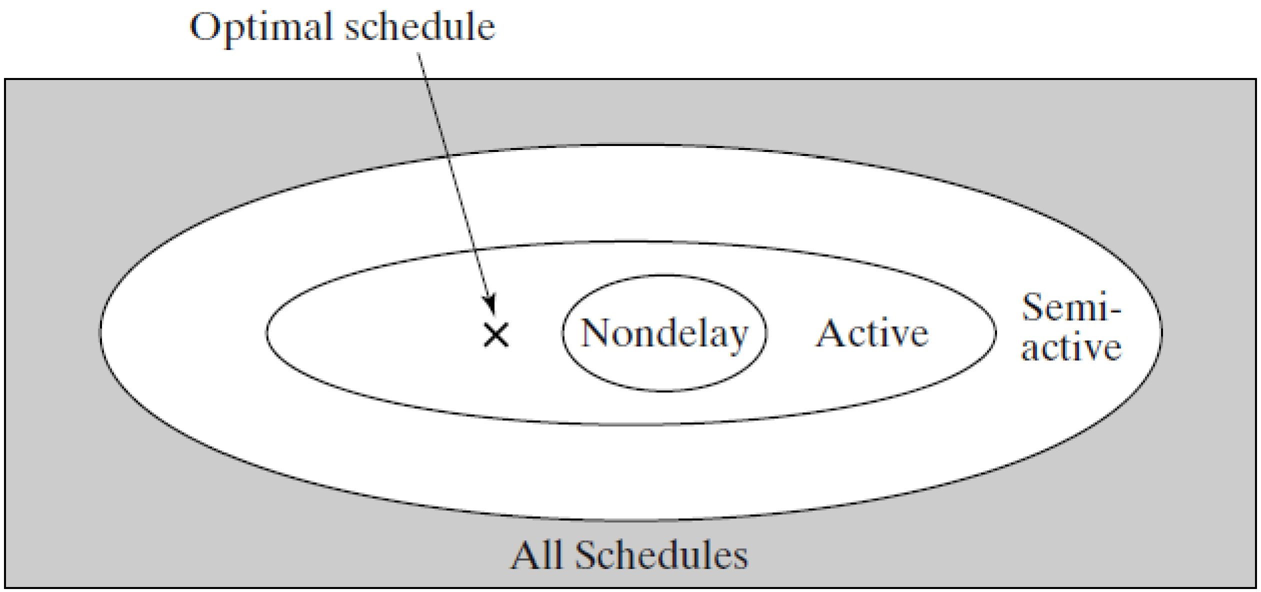


Fig. 2.6 Venn diagram of classes of nonpreemptive schedules for job shops

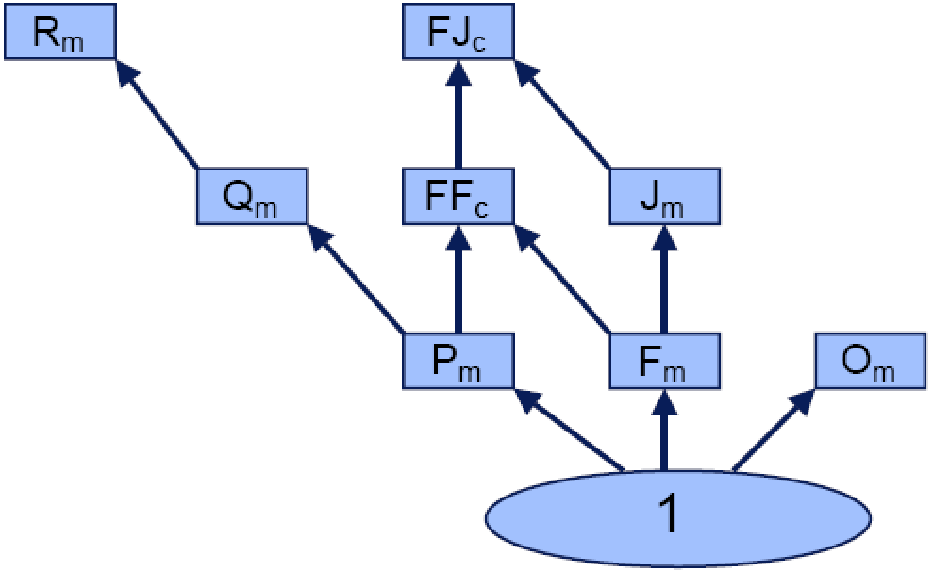
Complexity Hierarchy

- Some scheduling problems are special cases of other problems

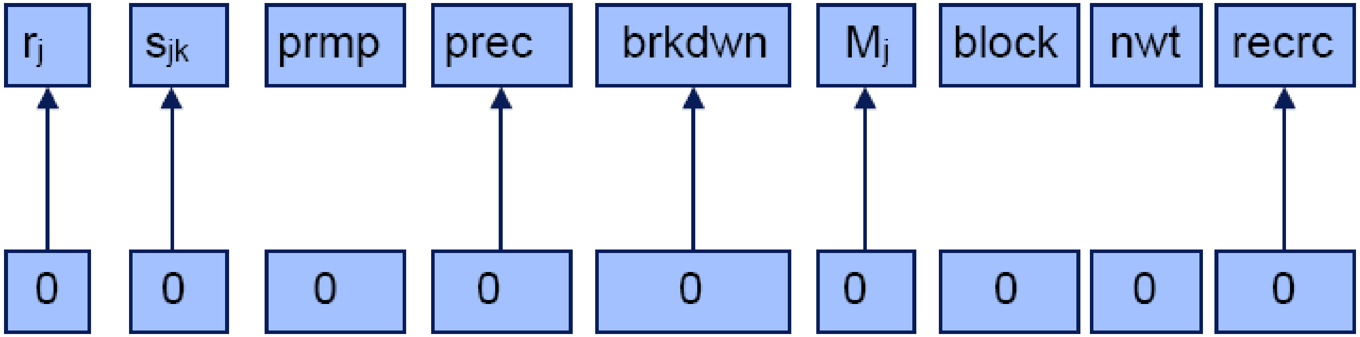
- $1 \mid\mid \Sigma_j C_j$ is a special case of $1 \mid\mid \Sigma_j w_j C_j$
  - When all $w_j = 1$
  - $1 \mid\mid \Sigma_j C_j$ reduces to $1 \mid\mid \Sigma_j w_j C_j$
  - $1 \mid\mid \Sigma_j C_j \;\; \alpha \;\; 1 \mid\mid \Sigma_j w_j C_j$

- $1 \mid\mid \Sigma_j C_j \;\alpha\; 1 \mid\mid \Sigma_j w_j C_j \;\alpha\; Pm \mid\mid \Sigma_j w_j C_j \;\alpha\; Qm \mid prec \mid \Sigma_j w_j C_j$

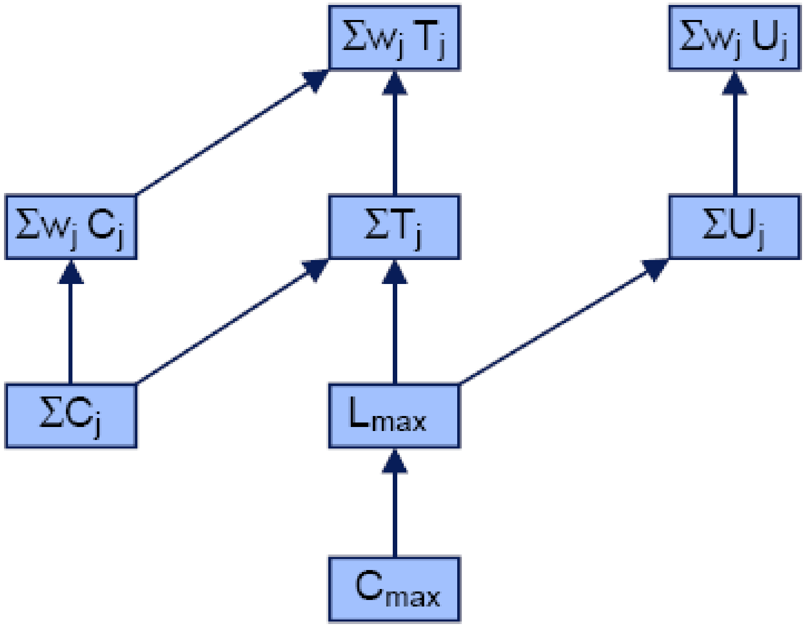
Machine Environments



Processing Restrictions and Constraints



Objective Functions



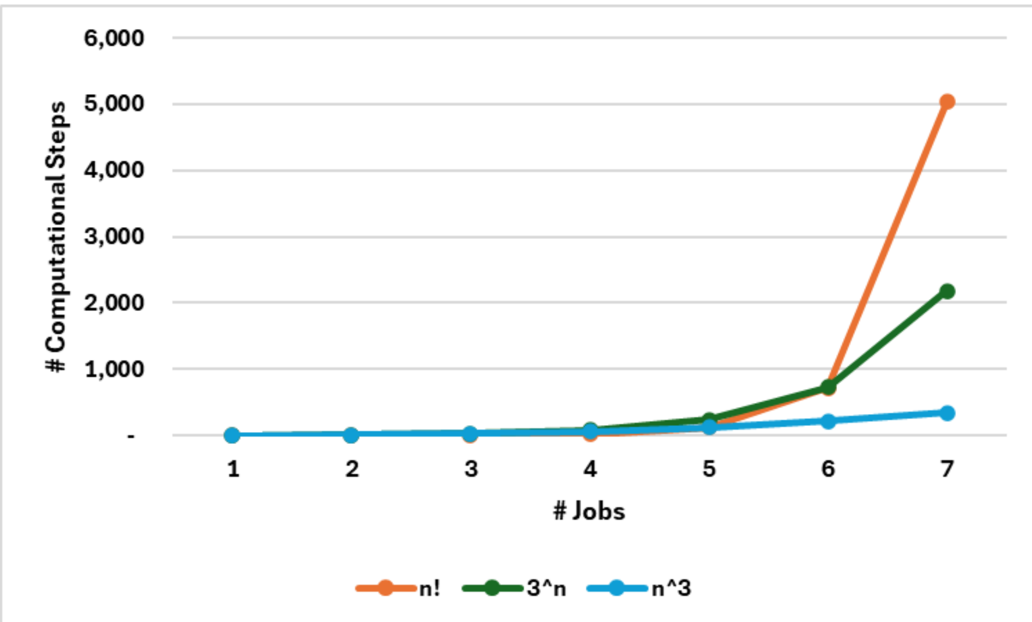
Time Complexity of Algorithms

- Number of computational steps required to find an optimal solution
- Problem: P2 || C_{max}
- Instance: 2 m/c's, 5 jobs, p = {2, 3, 5, 5, 8}
- encode: 2, 5, 2, 3, 5, 5, 8 (7 elements)
  - Binary: 10, 101, 10, 11, 101, 101, 1000
  - Unary: 11, 11111, 11, 111, 11111, 11111, 11111111
  - Unary encoding longer than binary
  - Size depends on number of jobs and processing times
- Size of an instance ~ number of jobs
- Computational steps: multiplication, comparison and any data manipulation

Time Complexity of Algorithms

- $1500 + 100n^2 + 5n^3 = O(n^3)$
- $O(n^3)$ polynomial time algorithm
- $O(3^n)$ and $O(n!)$ is not polynomial

n	n!	3^n	n^3
1	1	3	1
2	2	9	8
3	6	27	27
4	24	81	64
5	120	243	125
6	720	729	216
7	5,040	2,187	343
8	40,320	6,561	512
9	362,880	19,683	729
10	3,628,800	59,049	1,000
11	39,916,800	177,147	1,331
12	479,001,600	531,441	1,728



Time Complexity of Algorithms

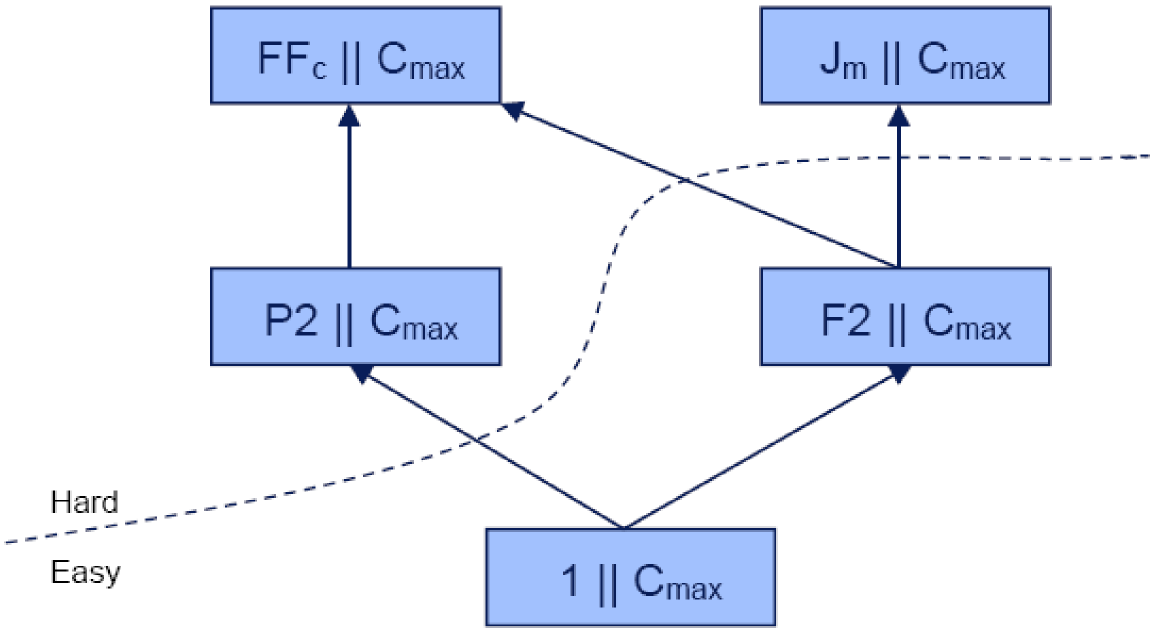
Easy (polynomial time complexity)

NP-hard in the ordinary sense (pseudo polynomial time complexity)

NP-hard in the strong sense

The problem cannot be optimally solved by an algorithm with pseudo polynomial time complexity

Complexity of Makespan Problems



Complexity of Maximum Lateness Problems