CS528
Task Scheduling

A Sahu
Dept of CSE, IIT Guwahati

Scheduling Problems

Ref: “Scheduling Algorithm” Book by P. Brucker

Google “Scheduling Algorithm Brucker pdf” to get a PDF copy of the Book
Soft copy uploaded to Course Website

Parallel Machine Problems

- **P:** We have jobs j as before and m **identical machines** $M_1, \ldots, M_m$.
- The processing time for j is the same on each machine.
- One has to assign the jobs to the machines and to schedule them on the assigned machines.
- This problem corresponds to an RCPSP with $r = 1$, $R_1 = m$, and $r_{j1} = 1$ for all jobs j.

Parallel Machine Problems

- **Q:** The machines are called **uniform** if $p_{jk} = p_j/r_k$.
- **R**: For **unrelated machines** the processing time p_{jk} depends on the machine M_k on which j is processed.
- ***MPM:*** *In a problem with* ***multi-purpose machines*** *a set of machines μ_j is is associated with each job j indicating that j can be processed on one machine in μ_j only.*

Parallel Machines

Ti	P1	P2	P3	P4
T1	10	10	10	10
T2	12	12	12	12
T3	16	16	16	16
T4	20	20	20	20

P: Identical

Ti	P1	P2	P3	P4
T1	10	15	20	25
T2	12	18	24	30
T3	16	24	32	40
T4	20	30	40	50

Q: Uniform : with speed difference (S_1=1, S_2=2/3, S_3=1/2, S_4=2/5

Ti	P1	P2	P3	P4
T1	10	8	12	2
T2	12	28	25	13
T3	16	4	32	14
T4	20	38	42	22

R: Unrelated : heterogeneous

Classification of Scheduling Problems

Classes of scheduling problems can be specified in terms of the three-field classification

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

where

- α specifies the **machine environment**,
- β specifies the **job characteristics**, and
- γ describes the **objective function(s).**

Machine Environment : α

Symbol	Meaning
1	Single Machine
P	Parallel Identical Machine
Q	Uniform Machine
R	Unrelated Machine
MPM	*Multipurpose Machine*
J	*Job Shop*
F	*Flow Shop*

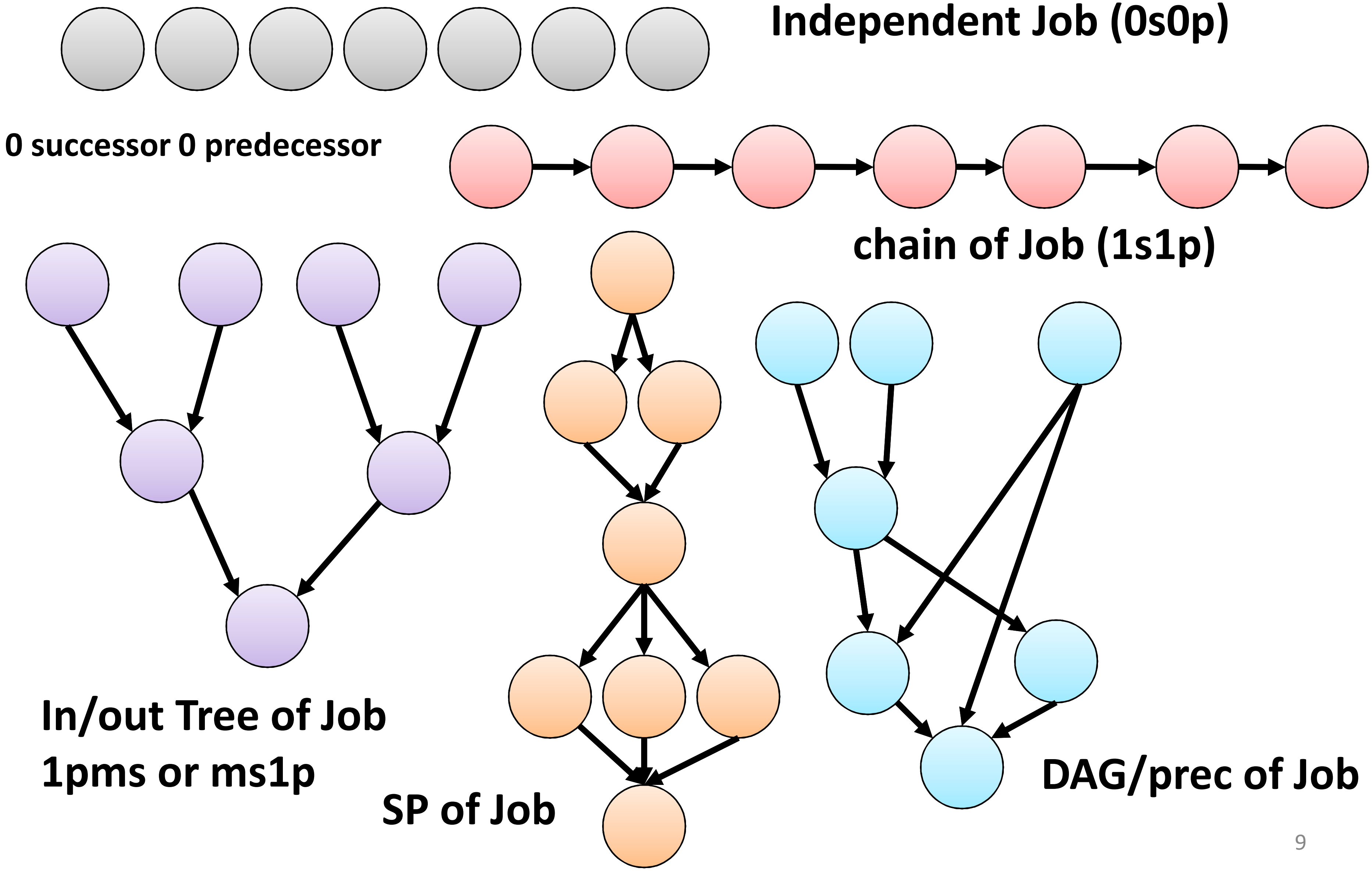
If the number of machines is fixed to m we write

Pm, Qm, Rm, MPMm, Jm, Fm, Om.

Job Characteristics : β

Symbol	meaning
pmtn	preemption
r_j	release times
d_j	deadlines
$p_j = 1$ or $p_j = p$ or $p_j \in \{1,2\}$	restricted processing times
prec	arbitrary precedence constraints
intree (outtree)	intree (or outtree) precedence
chains	chain precedence
series-parallel	*a series-parallel precedence graph*

Job Precedence Examples



Objective Functions : γ

Two types of objective functions are most common:

- **bottleneck objective functions** $\max \{f_j(C_j) \mid j= 1, \dots , n\}$, and
- **sum objective functions** $\Sigma f_j(C_j) = f_1(C_1) + f_2(C_2) + \dots \dots + f_n(C_n)$.

C_j is completion time of task j

Objective Functions : γ

- $\mathbf{C_{max}}$ and $\mathbf{L_{max}}$ symbolize the bottleneck objective
  - $\mathbf{C_{max}}$ objective functions with $f_j(C_j) = C_j$ (makespan)
  - $\mathbf{L_{max}}$ objective functions $f_j(C_j) = C_j - d_j$ (maximum Lateness)

- Common sum objective functions are:
  - $\Sigma\, \mathbf{C_j}$ (mean flow-time)
  - $\Sigma\, \omega_j\, \mathbf{C_j}$ (weighted flow-time)

Objective Functions : γ

- Σ $\mathbf{U_j}$ (number of late jobs) and Σ ω_j $\mathbf{U_j}$ (weighted number of late jobs) where $U_j = 1$ if $C_j > d_j$ and $U_j = 0$ otherwise.

- Σ $\mathbf{T_j}$ (sum of tardiness) and Σ ω_j $\mathbf{T_j}$ (weighted sum of tardiness/lateness) where the tardiness of job j is given by

$$T_j = \max \{ 0, C_j - d_j \}.$$

Examples of Scheduling Problem

- $1 \mid prec; p_j = 1 \mid \Sigma\, \omega_j C_j$
- $P2 \mid \mid C_{max}$
- $P \mid p_j = 1; r_j \mid \Sigma\, \omega_j U_j$
- $R2 \mid chains; pmtn \mid C_{max}$
- $R \mid n = 3 \mid C_{max}$
- $P \mid p_{ij} = 1; outtree; r_j \mid \Sigma\, C_j$
- $Q \mid p_j = 1 \mid \Sigma\, T_j$

Scheduling
of
Independent Tasks

Polynomial algorithms

- A problem is called polynomially solvable if it can be solved by a polynomial algorithm.

Example

$1 \mid \mid \Sigma\, \omega_j C_j$ can be solved by

Scheduling the jobs in an ordering of non-increasing ω_j/p_j - values.

Complexity: $O(n \log n)$

Polynomial algorithms for $1 \mid \mid \Sigma C_j$

Example

$1 \mid \mid \Sigma\, C_j$ can be solved by

Scheduling the jobs in an ordering of non-increasing $1/p_j$ - values. == > SJF

$C_i = Q_i + P_i$: Waiting time + Processing time

(SJF is optimal)

Complexity: O(n log n)

Polynomial algorithms : $P|p_i=1|Cmax$

- A problem is called polynomially solvable if it can be solved by a polynomial algorithm.

Example

$P|p_i=1|Cmax$ can be solved by

Scheduling the jobs in phase wise, P jobs in one phase, require ceil(n/P) phases.

Complexity: O(n)

$P2||C_{max}$

- n tasks, 2 processors
- ET: t_1, t_2, t_3,…., t_n
- **Subset Sum problem : 1+e APPROX**
  - **Ref: CLR Book Chapter 37 Section 4**
- Divide the tasks in two sets such that
  - Difference of Sum of ETs of both the set is minimized
  - Min (Max(Sum(Set_1), Sum(Set_2)))

$P_m || C_{max}$

- n tasks, m processors
- ET: t_1, t_2, t_3,…., t_n
- **m-Subset Sum problem**
- **INDEP(m) Problem: NPC in strong sense**
- Divide the tasks in m sets such that
  - Difference of Sum of ETs of all the set is minimized**: does not exceed a value K**
  - Min (Max(Sum(Set_1), Sum(Set_2), …Sum(Set_m)))

$P_m | pmtn | C_{max}$

- n tasks, m processors, infinite pre-emption allowed
- ET: $t_1, t_2, t_3, \ldots, t_n$
- Divide all the work among all the cores equally
  - Avg=($\Sigma\, t_i$)/m work to each cores
  - If $\max(t_i)>Avg$, $C_{max}=\max(t_i)$, Task need to execute serially
- Handle the boundary cases

$Q_m | pmtn | C_{max}$

- n tasks, m uniform processors, infinite pre-emption allowed
- Longer task executed on high speed processor till it execution is long enough as compared to others
  - Sort the tasks based on LPT
  - Allocate long task to higher speed processors one by one
  - When execution time of longer task is no longer long as compared to other then co-execute

$Q_m \mid pmtn \mid C_{max}$

Algorithm level

```
t := 0;
WHILE there exist jobs with positive level {
        Assign(t);
    t1 := min{s > t | a job completes at time s};
    t2 := min{s > t | there are jobs i, j with pi(t) > pj(t)
              and pi(s) = pj(s) at time s};
    t := min{t1, t2}
}
```

$t := 0;$

WHILE there exist jobs with positive level {

Assign(t);

$t_1 := \min\{s > t \mid \text{a job completes at time } s\};$

$t_2 := \min\{s > t \mid \text{there are jobs } i, j \text{ with } p_i(t) > p_j(t) \text{ and } p_i(s) = p_j(s) \text{ at time } s\};$

$t := \min\{t_1, t_2\}$

}

$Q_m \mid pmtn \mid C_{max}$

Assign (t)

$J := \{i \mid pi(t) > 0\}$;

$M := \{M1, . . ., Mm\}$;

WHILE $J = \emptyset$ and $M = \emptyset$ {

Find the set $I \subseteq J$ of jobs with highest level;

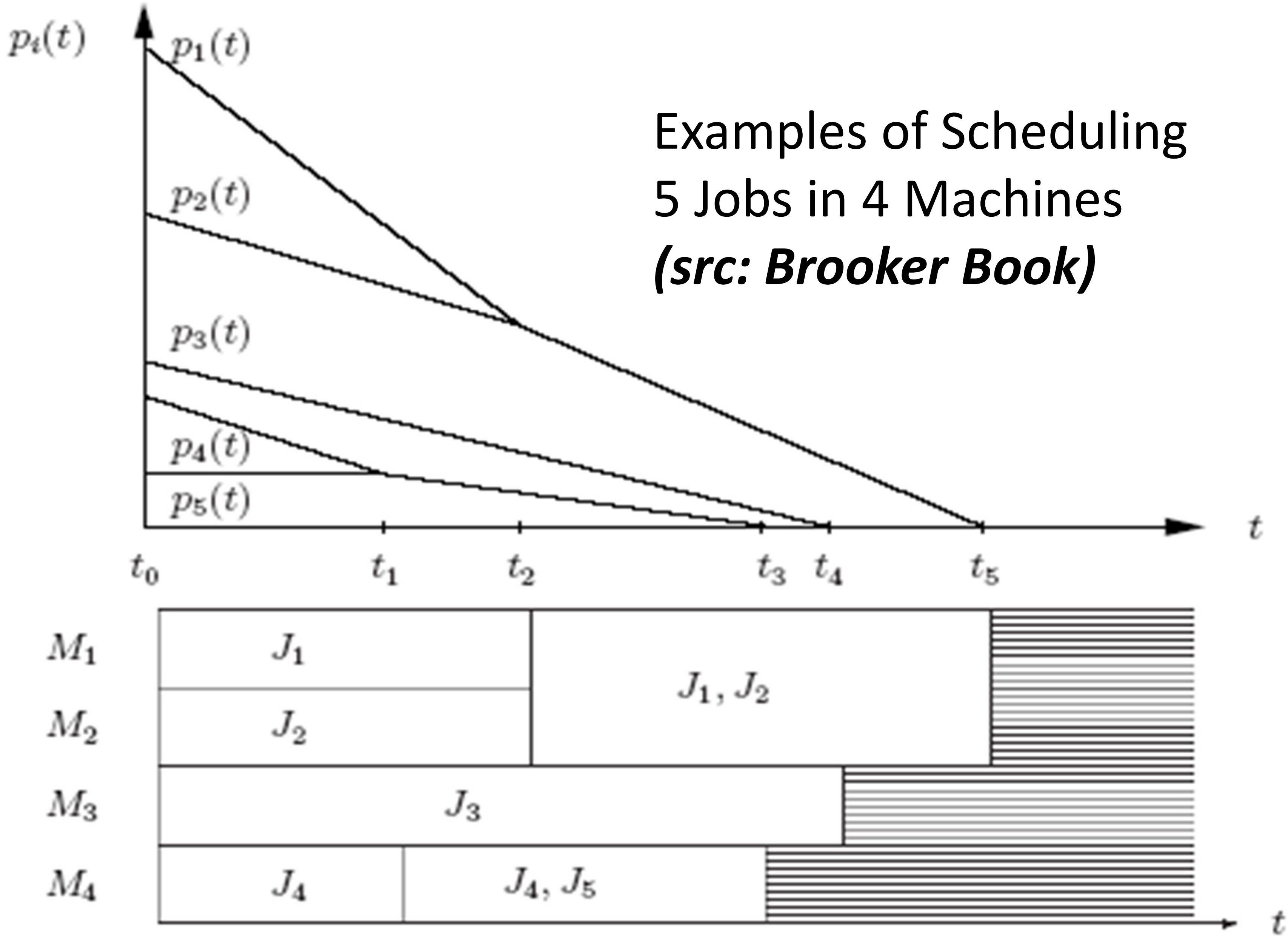
$r := \min\{|M|, |I|\}$;

Assign jobs in I to be processed jointly on the r fastest machines in M;

$J := J \backslash I$;

Eliminate the r fastest machines in M from M

$Q_m \mid pmtn \mid C_{max}$



Examples of Scheduling
5 Jobs in 4 Machines
(src: Brooker Book)

$Q|ptmn|\Sigma C_j$

- **LPT on High speed is good to optimize Σe_j the sum of task execution time but not ΣC_j**
- **Modified version of SPT** (shortest remaining time) rule. **As ΣC_j include waiting time of all the tasks**
- Order the tasks according to non-decreasing processing time.
- Schedule task 1 on available highest speed machine up to time $t_1=p_1/s_1$.
- Schedule 2nd task on M2 for t_1 time and then on M_1 from time t1 to time $t_2 \geq t_1$ until it is completed and same process continues

$Q|ptmn|\Sigma C_j$

- Example m=3, s1=3, s2=2, s3=1 and n=4, p1=10, p2=8, p3=8, p4=3
- SRT Job J_4 get scheduled on M1 with speed s1 for 1 time unit. Job 3 get scheduled on M_2 upto time 1 and then shifted to M1. Gant chat is given bellow with ΣC_i =14