CS528
Task Scheduling
(Part II)

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 will be uploaded to MS Team

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*

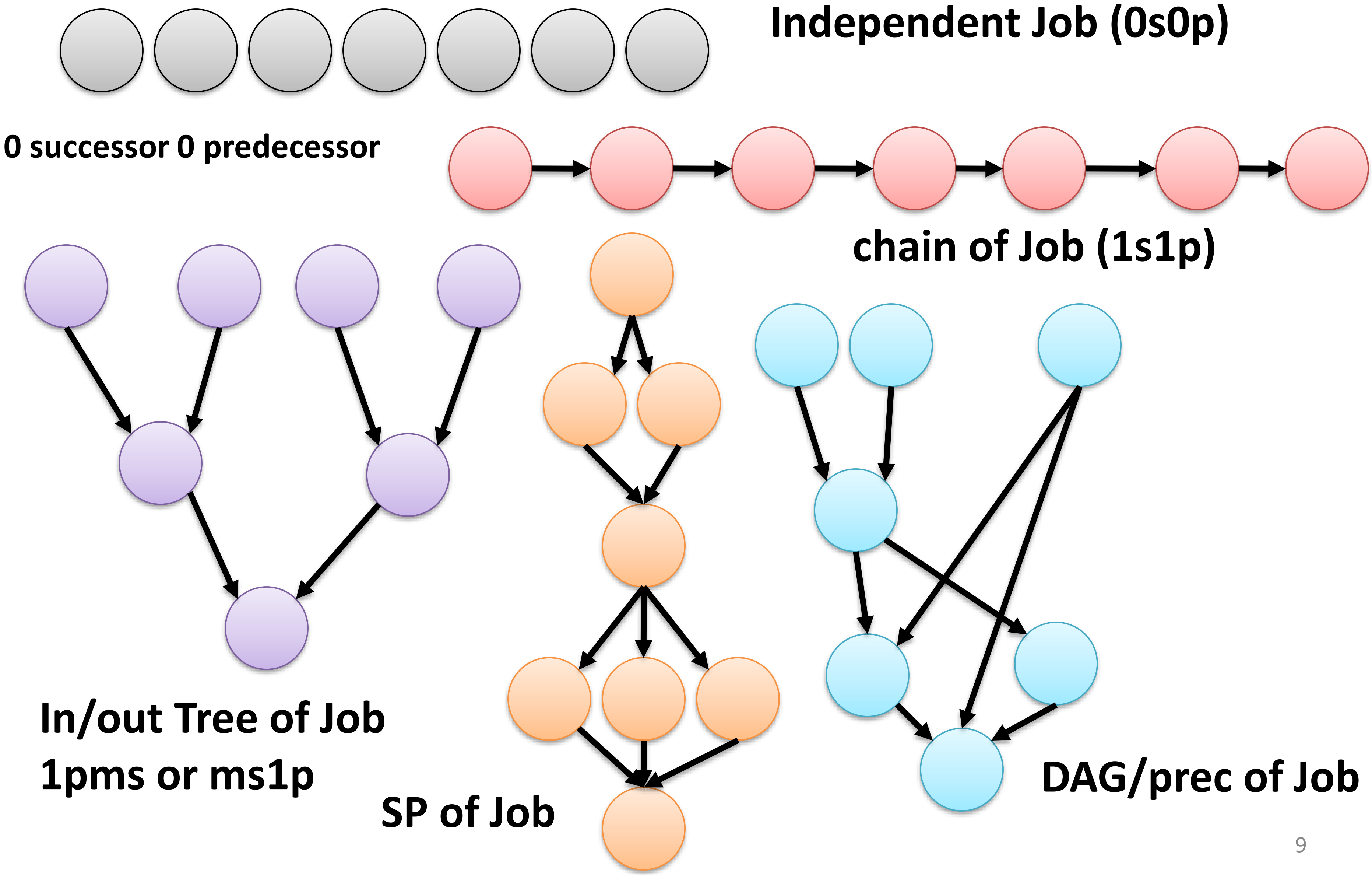
If the number of machines is fixed to m we write

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

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

- $\boldsymbol{\Sigma}$ **U_j** (number of late jobs) and $\boldsymbol{\Sigma\ \omega_j}$ **U_j** (weighted number of late jobs) where $U_j = 1$ if $C_j > d_j$ and $U_j = 0$ otherwise.

- $\boldsymbol{\Sigma}$ **T_j** (sum of tardiness) and $\boldsymbol{\Sigma\ \omega_j}$ **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

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

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 : $P|p_i=1|C_{max}$

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

Example

$P|p_i=1|C_{max}$ 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)))