

# **CS528**

# **Scheduling of Dependent Tasks**

A Sahu

Dept of CSE, IIT Guwahati

# Outline

- $P_m \mid p_j, \text{pmtn} \mid C_{\max}$  : Linear time solution
- $Q_m \mid p_j, \text{pmtn} \mid C_{\max}$  : Poly time solution
- $Q_m \mid \text{ptmn} \mid \sum C_j$  Optimal Solution
- $P_m \mid p_j \mid C_{\max}$ 
  - **ILP Solution : Exponential**
  - 2 Approx,  $2-1/m$  approx.
  - LPT :  $3/2$  and  $4/3$  Approx
- $P_m \mid p_j=1 \mid \sum w_j U_j$  Optimal Solution
- $P_m \mid p_j \mid \sum U_j$  NPC, Heuristic and Counter example
- $P_m \mid \text{pmtn}, p_j \mid \sum U_j$  **in NPC**
- $P_m \mid \text{prec}, p_j = 1 \mid C_{\max}$  **in NPC**
  - 2 Approx

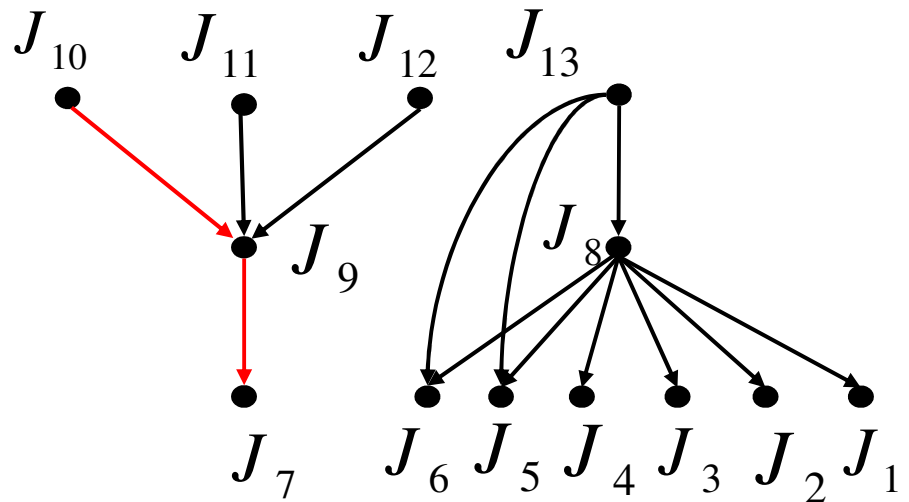
# **Scheduling of Dependent Tasks**

# Precedence constraints (*prec*)

Before certain jobs are allowed to start processing, one or more jobs first have to be completed.

## Definition

- Successor
- Predecessor
- Immediate successor
- Immediate predecessor
- Transitive Reduction



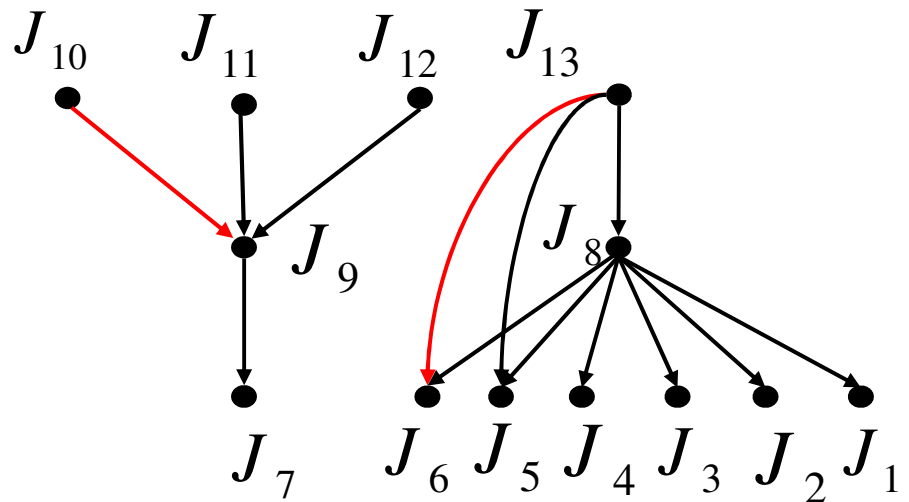
$$p(J_i) = 1$$

# Precedence constraints (*prec*)

One or more job have to be completed before another job is allowed to start processing.

## Definition

- Successor
- Predecessor
- Immediate successor
- Immediate predecessor
- Transitive Reduction



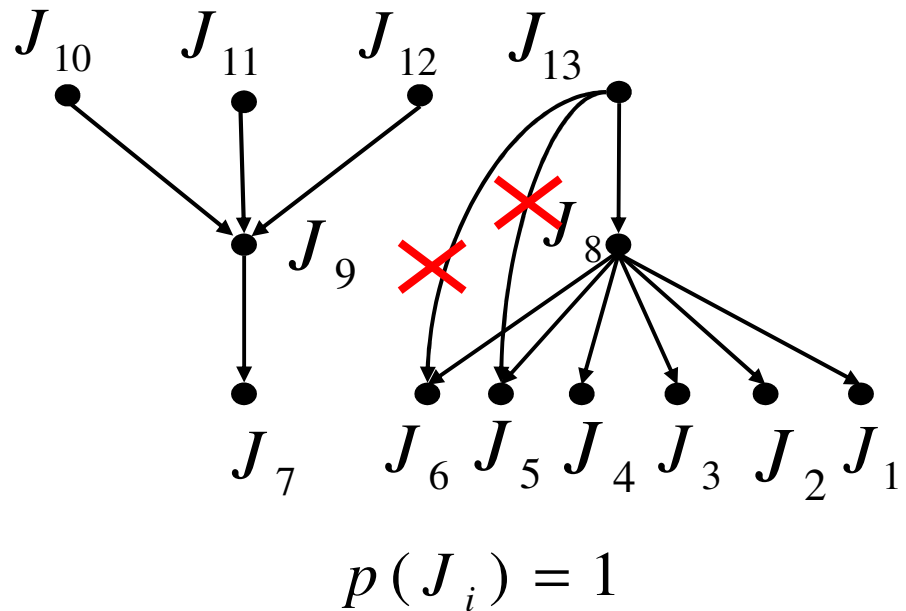
$$p(J_i) = 1$$

# Precedence constraints (*prec*)

One or more job have to be completed before another job is allowed to start processing. *Prec : Arbitrary acyclic graph*

## Definition

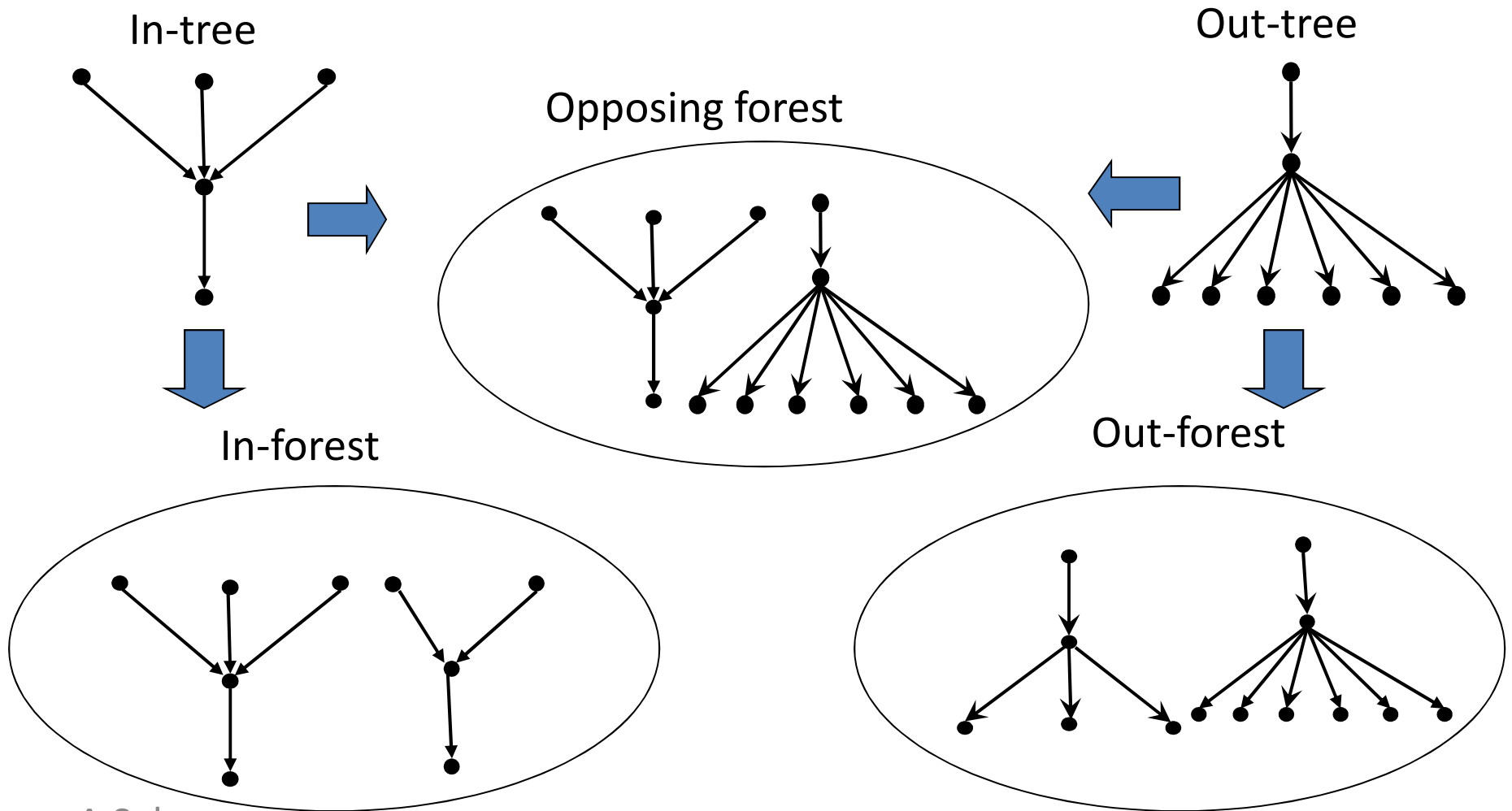
- Successor
- Predecessor
- Immediate successor
- Immediate predecessor
- **Transitive Reduction**



# Special precedence constraints

- In-tree (Out-tree)
- In-forest (Out-forest)
- Opposing forest
- *Interval orders*
- *Series-parallel orders*
- *Level orders*

# Special precedence constraints





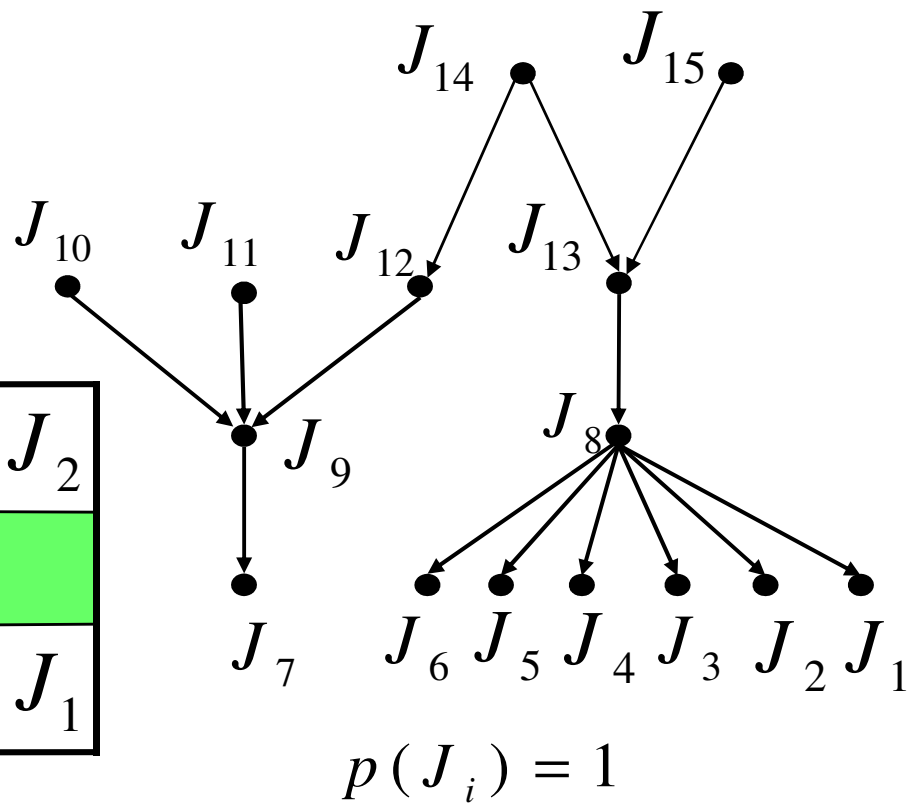
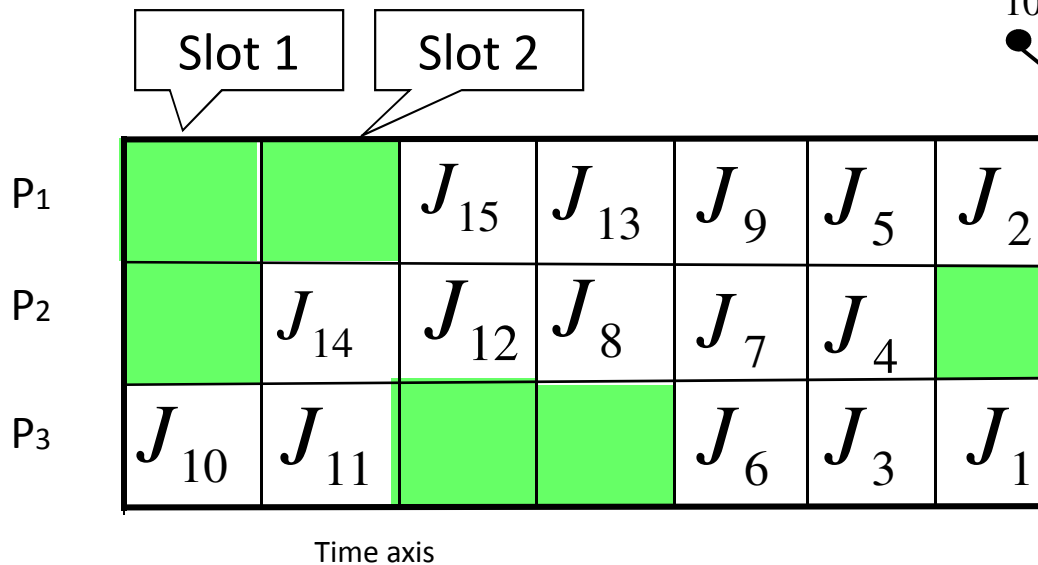
# $P_m \mid \text{prec}, p_j = 1 \mid C_{\max} (m \geq 1)$

- Processor Environment
  - $m$  identical processors are in the system.
- Job characteristics
  - Precedence constraints are given by a precedence graph;
  - Preemption is not allowed;
  - The release time of all the jobs is 0.
- Objective function
  - $C_{\max}$  : the time the last job finishes execution.
  - If  $c_j$  denotes the finishing time of  $J_j$  in a schedule  $S$ ,

$$C_{\max} = \max_{1 \leq j \leq n} c_j$$

# Gantt Chart

A Gantt chart indicates the time each job spends in execution, as well as the processor on which it executes *of some Schedule*



$$P_m \mid \text{prec}, p_j = 1 \mid C_{\max}$$

## Theorem 1

$P_m \mid \text{prec}, p_j = 1 \mid C_{\max}$  is NP-complete.

1. Ullman (1976)

$$3\text{SAT} \leq P_m \mid \text{prec}, p_j = 1 \mid C_{\max}$$

2. Lenstra and Rinnooy Kan (1978)

$$k\text{-clique} \leq P_m \mid \text{prec}, p_j = 1 \mid C_{\max}$$

$P_m \mid \text{prec}, p_j = 1 \mid C_{\max}$  is NP-complete.

*Proof: out of Syllabus*

# $P_m \mid \text{prec}, p_j = 1 \mid C_{\max}$

Mayr (1985)

- **Theorem 2**

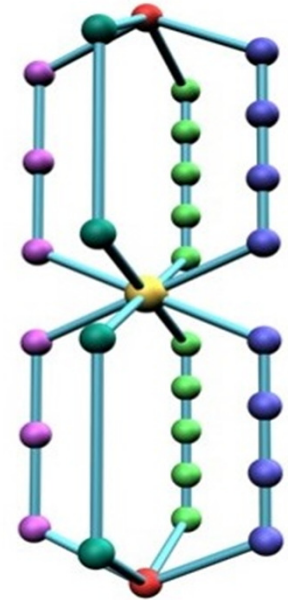
$P_m \mid p_j = 1, SP \mid C_{\max}$  is NP-complete.

SP: Series - parallel

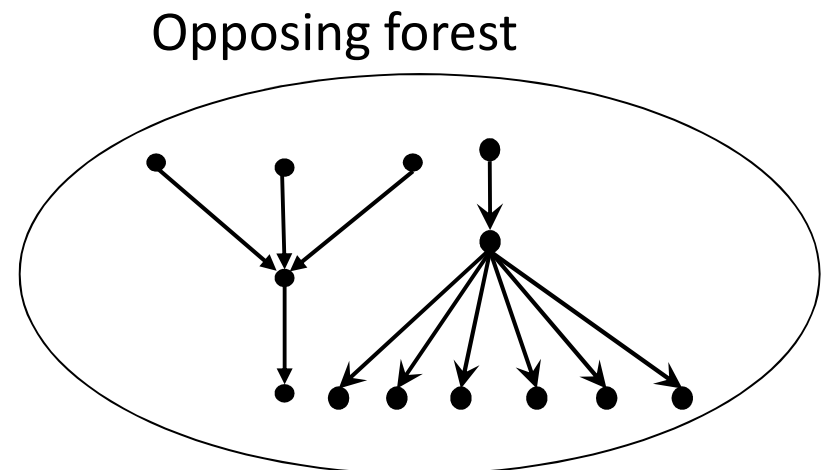
- **Theorem 3**

$P_m \mid p_j = 1, OF \mid C_{\max}$  is NP-complete.

OF: Opposing - forest



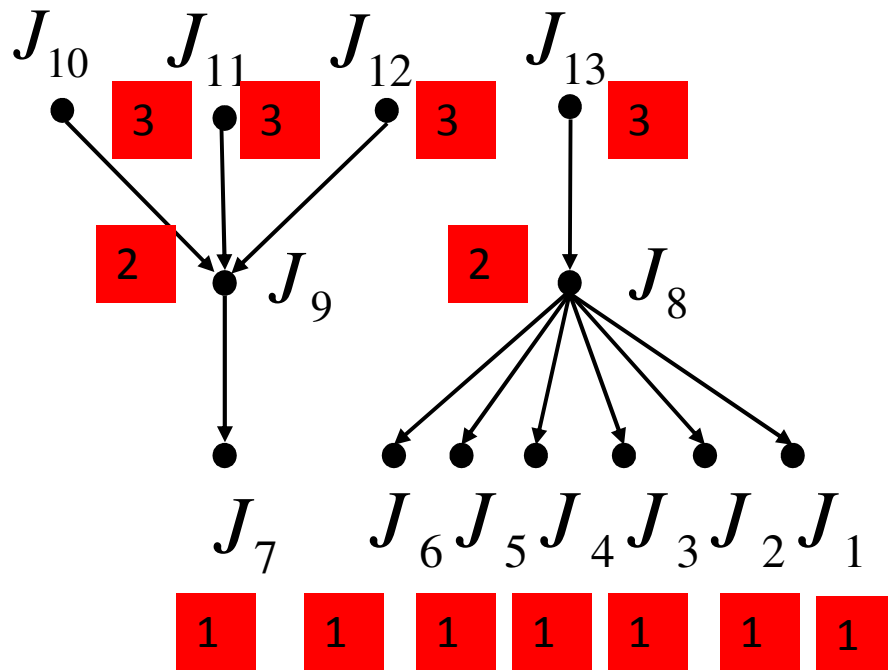
***Proof: out of Syllabus***



# Hu's HLF/CP Algorithm

- T. C. Hu (1961), **Critical Path/Highest Level First**
- Assign a level  $h$  to each job.
  - If job has no successors,  $h(j)$  equals 1.
  - Otherwise,  $h(j)$  equals one plus the maximum level of its immediate successors.
- Set up a priority list  $L$  by nonincreasing order of the jobs' levels.
- Execute the list scheduling policy on this level based priority list  $L$ .

# HLF/CP algorithm : Example



M2	$J_{10}$	$J_{13}$	$J_8$	$J_6$	$J_3$
M2	$J_{11}$	$J_9$	$J_7$	$J_5$	$J_2$
M1	$J_{12}$			$J_4$	$J_1$

$$L = (\underbrace{J_{10}, J_{11}, J_{12}, J_{13}}_{\text{Level 3}}, \underbrace{J_9, J_8}_{\text{Level 2}}, \underbrace{J_7, J_6, J_5, J_4, J_3, J_2, J_1}_{\text{Level 1}})$$

# HLF/CP algorithm

- **Time complexity**

$O(|V|+|E|)$  ( $|V|$  is the number of jobs and  $|E|$  is the number of edges in the precedence graph)

- **Theorem (Hu, 1961) : HLF/CP for Tree**

- The HLF algorithm is optimal for  $P_m \mid p_j = 1$ , in-tree (out-tree)  $\mid C_{\max}$ .
- The HLF algorithm is optimal for  $P_m \mid p_j = 1$ , in-forest (out-forest)  $\mid C_{\max}$ .



# HLF/CP algorithm

- N.F. Chen & C.L. Liu (1975)

The approximation ratio of HLF algorithm for the problem with general precedence constraints:

If  $m = 2$ ,  $\delta_{\text{HLF}} \leq 4/3$ .

If  $m \geq 3$ ,  $\delta_{\text{HLF}} \leq 2 - 1/(m-1)$ .



# PTAS Algorithms: $P_m \mid \text{prec}, p_j = 1 \mid C_{\max}$

- PTAS : Polynomial Time Approximation Scheme
- Approximation List scheduling policies
  - Graham's list algorithm/Greedy List
    - Discussed in Cilk Lectures :  $T \leq 2T^*$ , Also proved
    - CLR Book Chapter 27, Multi-threaded Algorithm
  - HLF algorithm
  - MSF algorithm

$$P_m \mid \text{prec}, p_j = 1 \mid C_{\max}$$

## Theorem 1

$P_m \mid \text{prec}, p_j = 1 \mid C_{\max}$  is NP-complete.

1. Ullman (1976)

$$3\text{SAT} \leq P_m \mid \text{prec}, p_j = 1 \mid C_{\max}$$

2. Lenstra and Rinnooy Kan (1978)

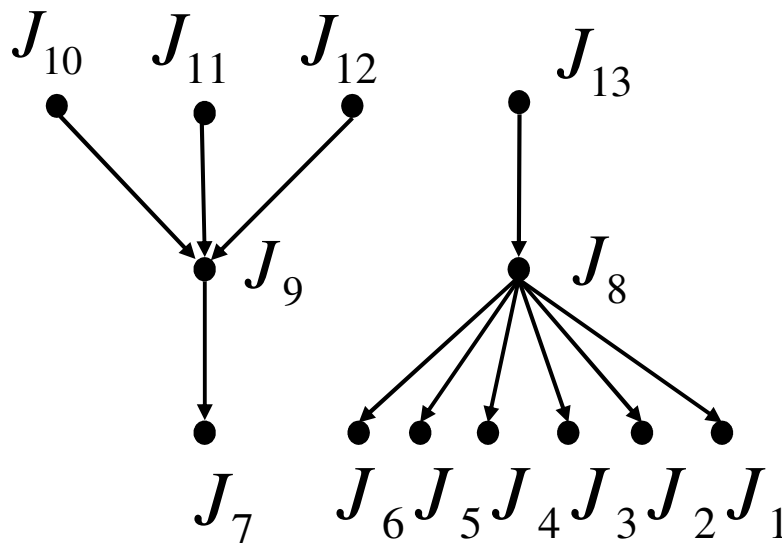
$$k\text{-clique} \leq P_m \mid \text{prec}, p_j = 1 \mid C_{\max}$$

$P_m \mid \text{prec}, p_j = 1 \mid C_{\max}$  is NP-complete.

*Proof: out of Syllabus*

# List scheduling policies

- Set up a priority list  $L$  of jobs.
- When a processor is idle, assign the first ready job to the processor and remove it from the list  $L$ .



$J_{11}$	$J_9$	$J_8$	$J_6$	$J_3$
$J_{10}$	$J_{13}$	$J_7$	$J_5$	$J_2$
$J_{12}$			$J_4$	$J_1$

First job of the list  
may not be ready

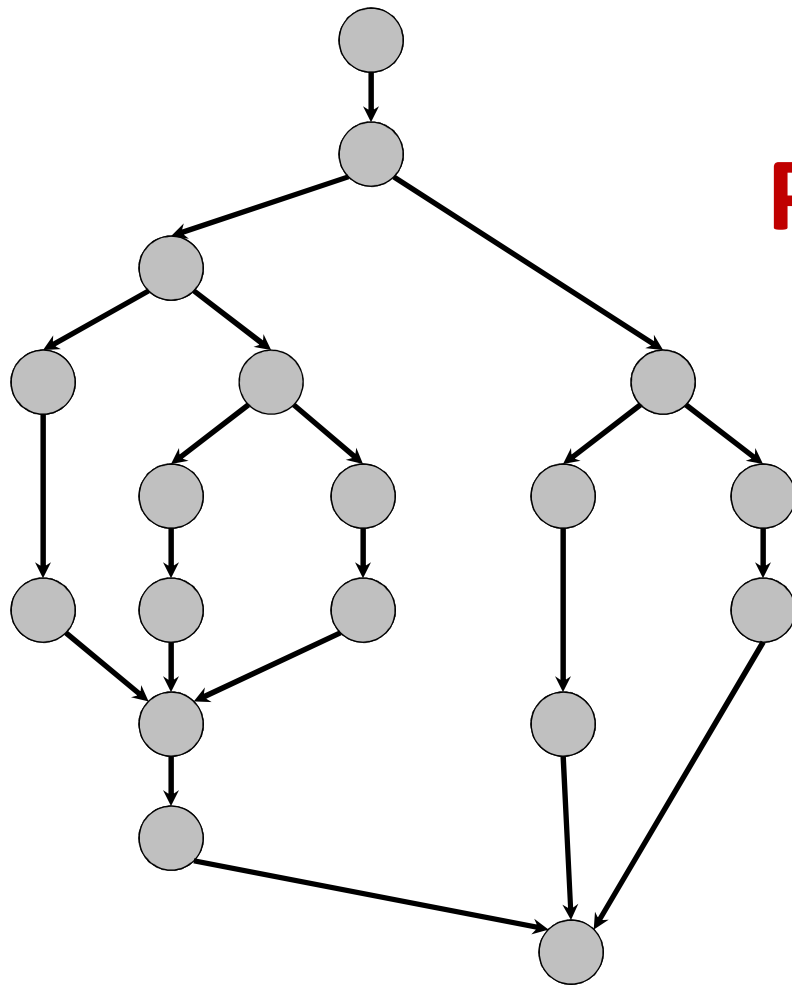
$$L = (J_9, J_8, J_7, J_6, J_5, J_{11}, J_{10}, J_{12}, J_{13}, J_4, J_3, J_2, J_1)$$

# Graham's list algorithm

- Graham first analyzed the performance of the simplest list scheduling algorithm.
- List scheduling algorithm with an arbitrary job list is called Graham's list algorithm.
- Approximation ratio for  $P_m \mid \text{prec}, p_j = 1 \mid C_{\max}$   
 $\delta = 2 - 1/m$ . (Tight bound!)
  - Approximation ratio is  $\delta$  if for each input instance, the makespan produced by the algorithm is at most  $\delta$  times of the optimal makespan.

# CP Algo: CLR Book Page 779-783

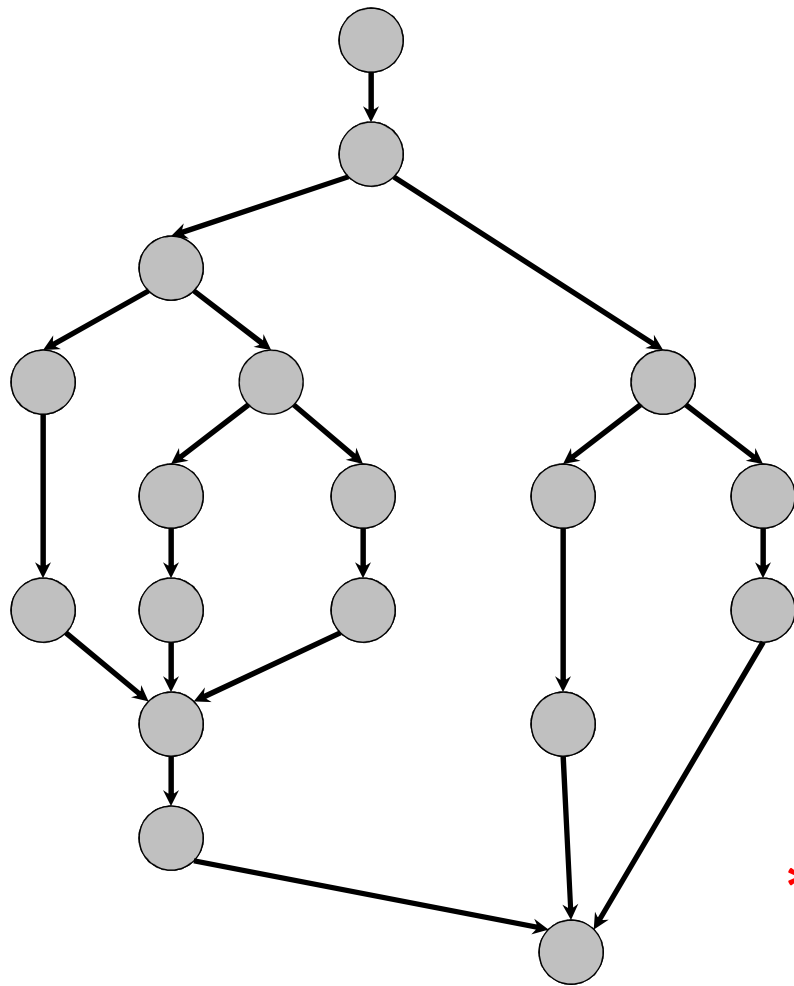
$T_P$  = execution time on  $P$  processors



$P_m \mid p_j = 1, \text{ prec} \mid C_{\max}$

# CP Algorithms

$T_P$  = execution time on  $P$  processors



$T_1$  = *work*

$T_\infty$  = *span*\*

## LOWER BOUNDS

- $T_P \geq T_1/P$
- $T_P \geq T_\infty$

\* Also called *critical-path length* or *computational depth*.

# CP: Greedy-Scheduling Theorem

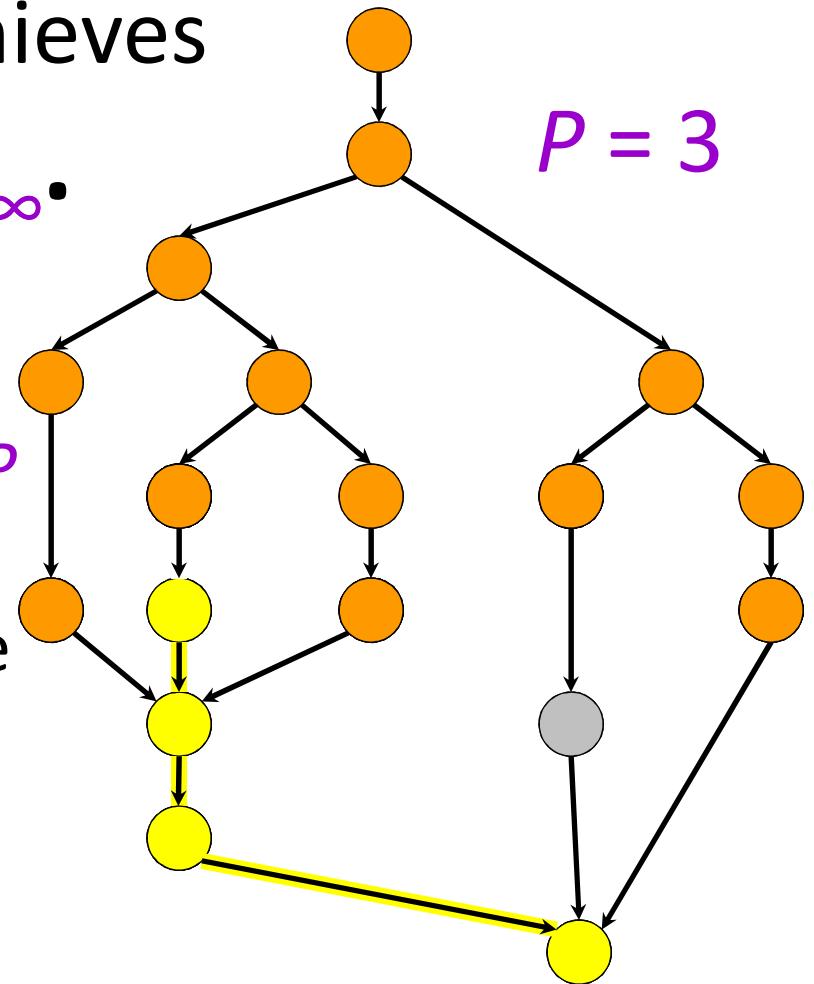
**Theorem** [Graham '68 & Brent '75].

Any greedy scheduler achieves

$$T_P \leq T_1/P + T_\infty.$$

*Proof.*

- # complete steps  $\leq T_1/P$ , since each complete step performs  $P$  work.
- # incomplete steps  $\leq T_\infty$ , since each incomplete step reduces the span of the unexecuted dag by 1. ■



## CP: Optimality of Greedy

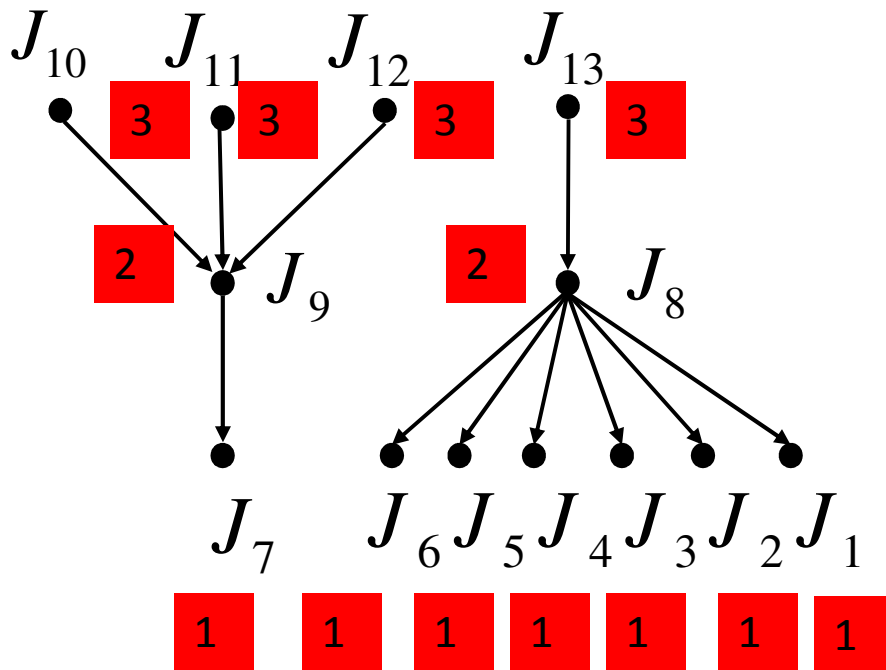
**Corollary.** Any greedy scheduler achieves within a factor of 2 of optimal.

**Proof.** Let  $T_p^*$  be the execution time produced by the optimal scheduler. Since  $T_p^* \geq \max\{T_1/P, T_\infty\}$  (lower bounds), we have

$$\begin{aligned} T_p &\leq T_1/P + T_\infty \\ &\leq 2 \cdot \max\{T_1/P, T_\infty\} \\ &\leq 2T_p^* . \quad \blacksquare \end{aligned}$$



# HLF/CP algorithm : Example



M2	$J_{10}$	$J_{13}$	$J_8$	$J_6$	$J_3$
M2	$J_{11}$	$J_9$	$J_7$	$J_5$	$J_2$
M1	$J_{12}$			$J_4$	$J_1$

$$L = (\underbrace{J_{10}, J_{11}, J_{12}, J_{13}}_{\text{Level 3}}, \underbrace{J_9, J_8}_{\text{Level 2}}, \underbrace{J_7, J_6, J_5, J_4, J_3, J_2, J_1}_{\text{Level 1}})$$

# HLF/CP algorithm

- **Time complexity**

$O(|V|+|E|)$  ( $|V|$  is the number of jobs and  $|E|$  is the number of edges in the precedence graph)

- **Theorem (Hu, 1961) : HLF/CP for Tree**

- The HLF algorithm is optimal for  $P_m \mid p_j = 1$ , in-tree (out-tree)  $\mid C_{\max}$ .
- The HLF algorithm is optimal for  $P_m \mid p_j = 1$ , in-forest (out-forest)  $\mid C_{\max}$ .



# HLF/CP algorithm

- N.F. Chen & C.L. Liu (1975)

The approximation ratio of HLF algorithm for the problem with general precedence constraints:

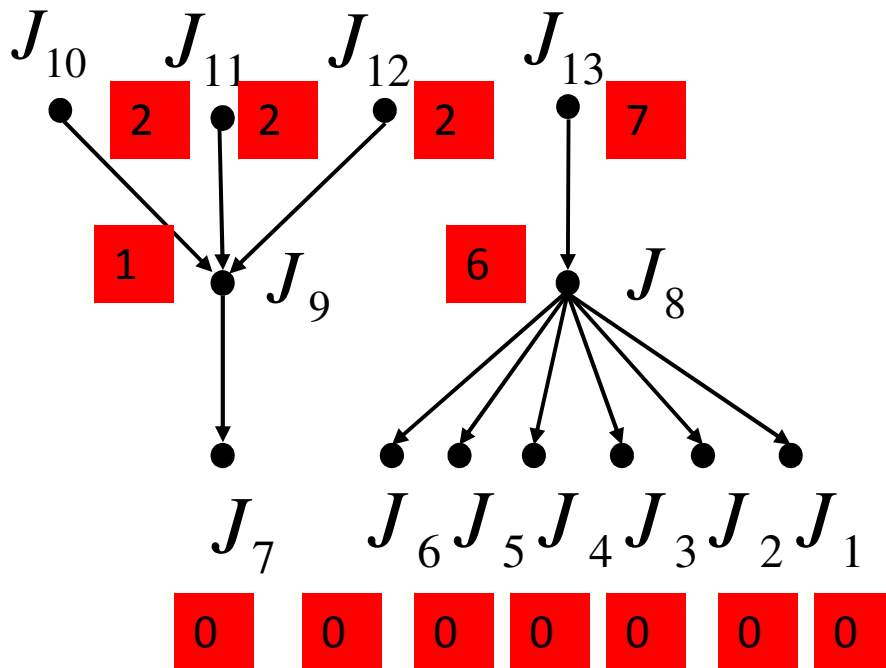
If  $m = 2$ ,  $\delta_{\text{HLF}} \leq 4/3$ .

If  $m \geq 3$ ,  $\delta_{\text{HLF}} \leq 2 - 1/(m-1)$ .

## Most Successors First (MSF)

- Algorithm:
  - Set up a priority list L by nonincreasing order of the jobs' successors numbers.
    - (i.e. the job having more successors should have a higher priority in L than the job having fewer successors)
  - Execute the list scheduling policy based on this priority list L.

# Most Successors First algorithm



M2	$J_{13}$	$J_{10}$	$J_9$	$J_7$	$J_2$
M2	$J_{12}$	$J_8$	$J_6$	$J_4$	$J_1$
M1	$J_{11}$		$J_5$	$J_3$	

$$L = (J_{13}, J_8, J_{12}, J_{11}, J_{10}, J_9, J_7, J_6, J_5, J_4, J_3, J_2, J_1)$$

7    6    2    2    2    1    0    0    0    0    0    0    0

# **Energy/Power/Temp Aware Scheduling of Tasks**

# Outline

- Power Aware
- Task with Hard Deadlines
- Energy Efficiency
- Energy Efficient Scheduling
- Real Time Tasks

# Power Aware Scheduling Vs Energy Aware Scheduling

- Power Budget should not exceed
  - Minimized
  - Monthly Expenses: CAP ==> Solution is EMI
  - Power CAP: If your system have 100 design, at any instance of time you should not run things above 100W
    - Suppose you have 3KW wiring in your home, you have 3 AC with each of 1.5KW rating, At a given time, you can run maximum of 2 AC.
- Total energy budget should not exceed
  - Battery capacity, mah (mobile), AH (UPS)
  - Minimized: EC
  - Power and Time