CS528
Scheduling of Dependent Tasks

A Sahu

Dept of CSE, IIT Guwahati

Outline

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

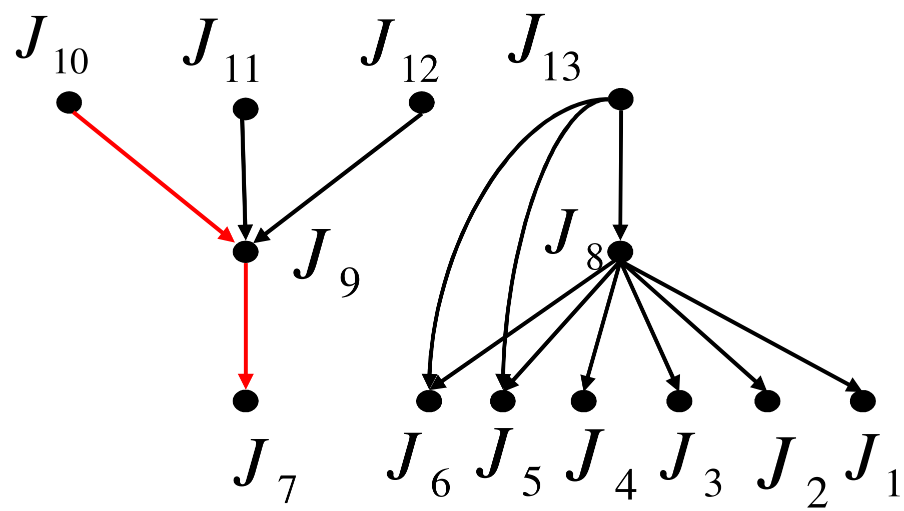
Scheduling of Dependent Tasks

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



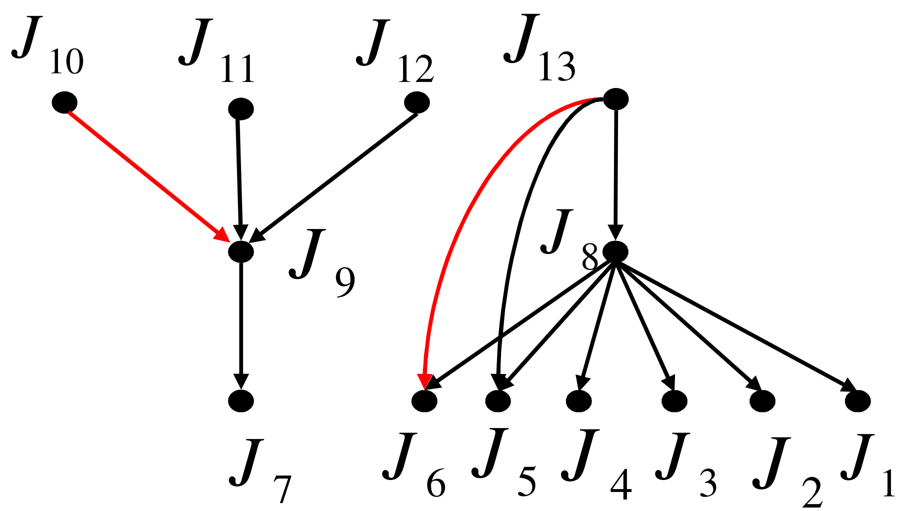
$$p(J_i) = 1$$

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



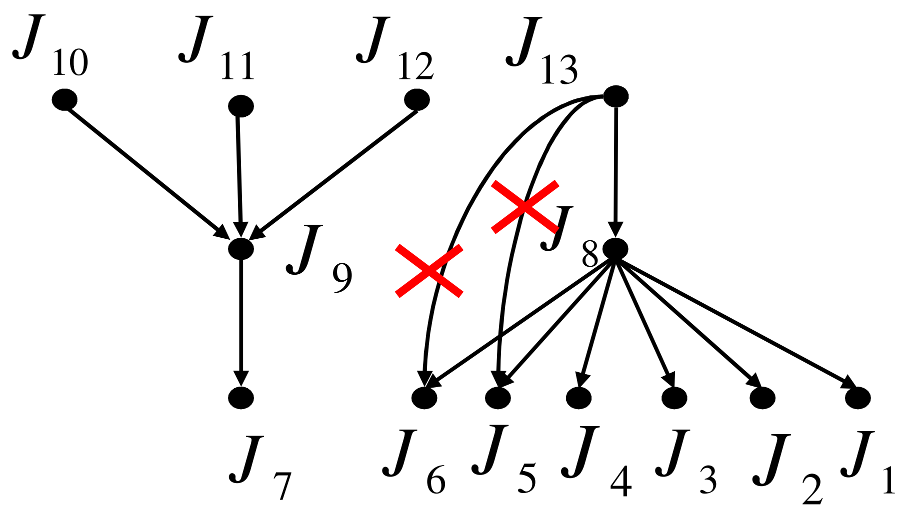
$$p(J_i) = 1$$

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

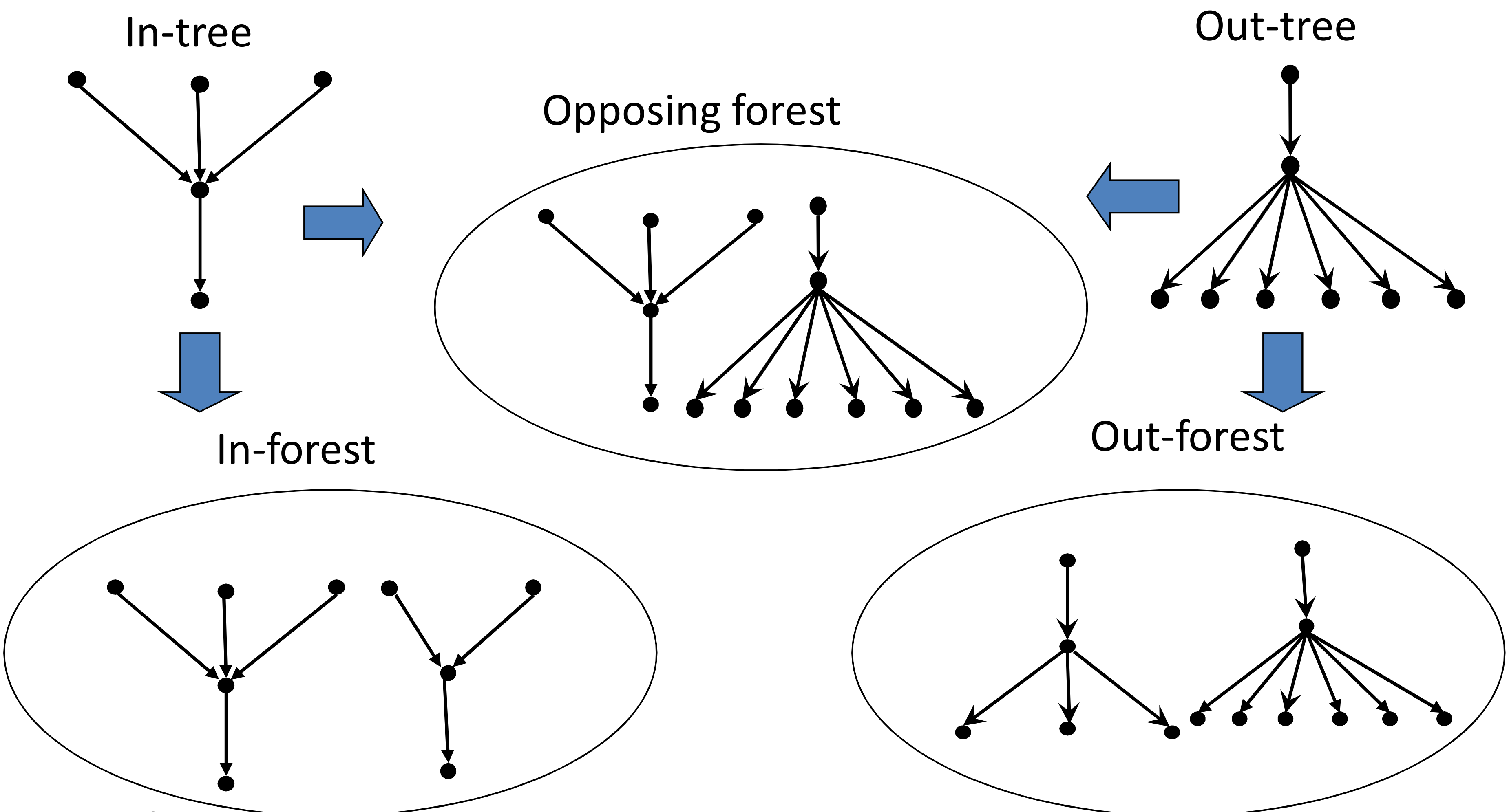


$$p(J_i) = 1$$

Special precedence constraints

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

Special precedence constraints



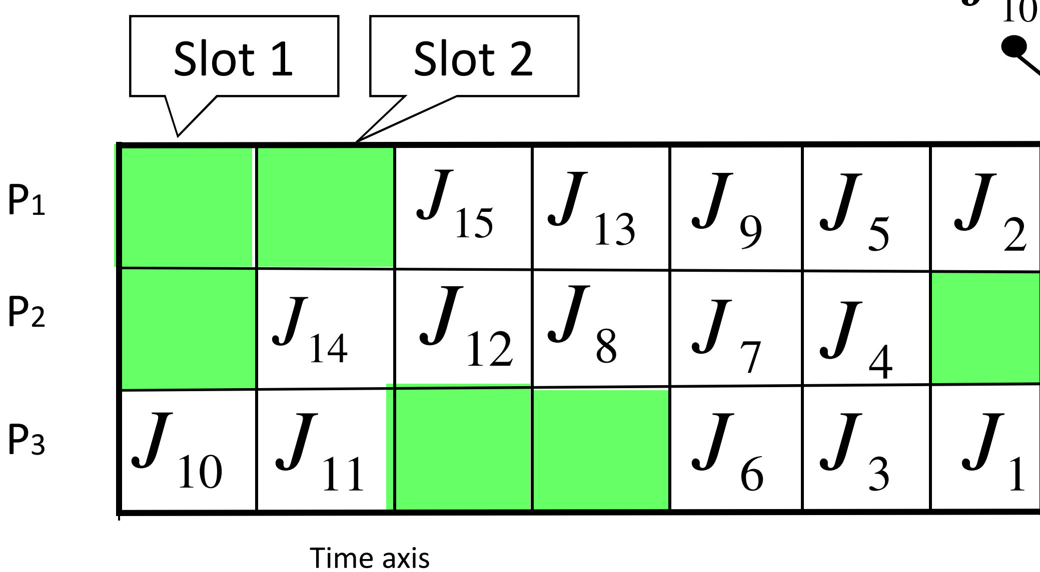
$P_m \mid prec, p_j = 1 \mid C_{max}$ (m≥1)

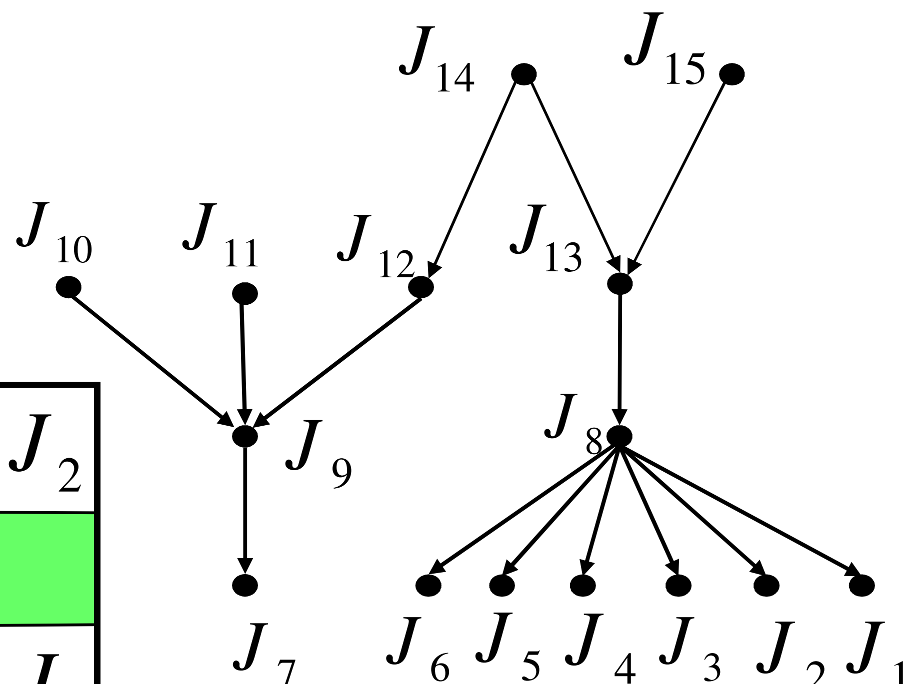
- Processor Environment
  - m identical processors are in the system.
- Job characteristics
  - Precedence constraints are given by a precedence graph;
  - Preemption is not allowed;
  - The release time of all the jobs is 0.
- Objective function
  - C_{max} : the time the last job finishes execution.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

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

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***~~





$$p(J_i) = 1$$

$P_m |$ prec, $p_j = 1 | C_{max}$

Theorem 1

$Pm |$ prec, $p_j = 1 | C_{max}$ is NP-complete.

1. Ullman (1976)

   $3SAT \leq Pm |$ prec, $p_j = 1 | C_{max}$

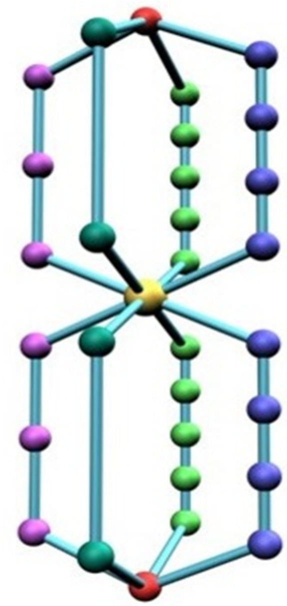
2. Lenstra and Rinooy Kan (1978)

   k-clique $\leq Pm |$ prec, $p_j = 1 | C_{max}$

$P_m |$ prec, pj = 1 $| C_{max}$ is NP-complete.

Proof: out of Syllabus

P_m | prec, $p_j = 1$ | C_{max}



Mayr (1985)

- **Theorem 2**

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

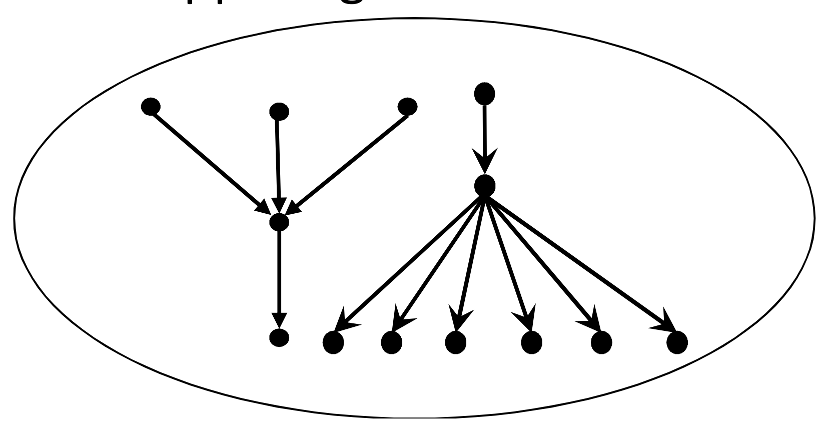
  SP: Series - parallel

- **Theorem 3**

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

  OF: Opposing - forest

Opposing forest

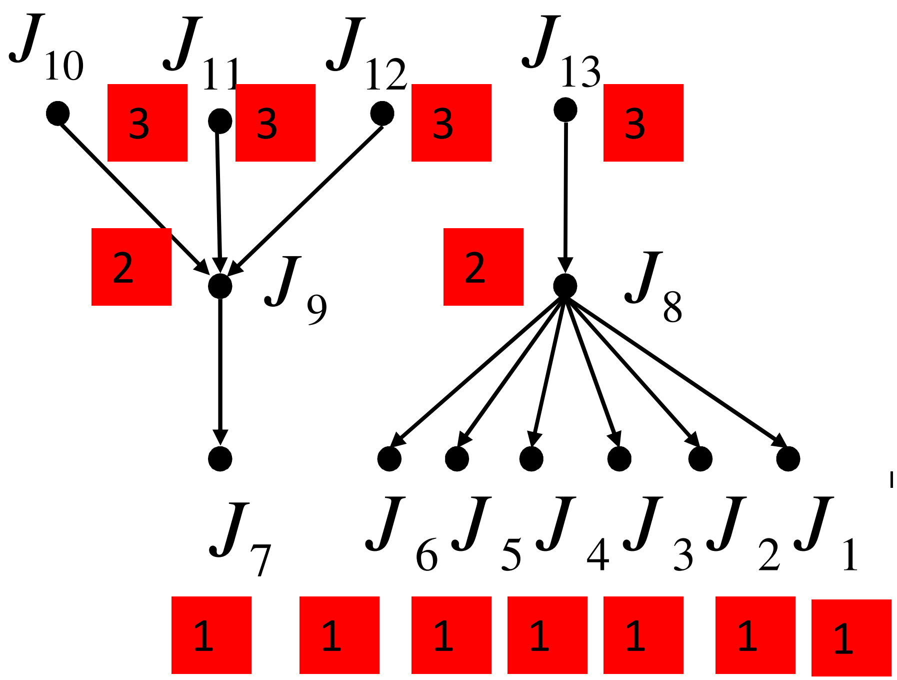


Proof: out of Syllabus

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

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

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

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

PTAS Algorithms: Pm | prec, p_j = 1 | 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

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

Theorem 1

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

1. Ullman (1976)

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

2. Lenstra and Rinooy Kan (1978)

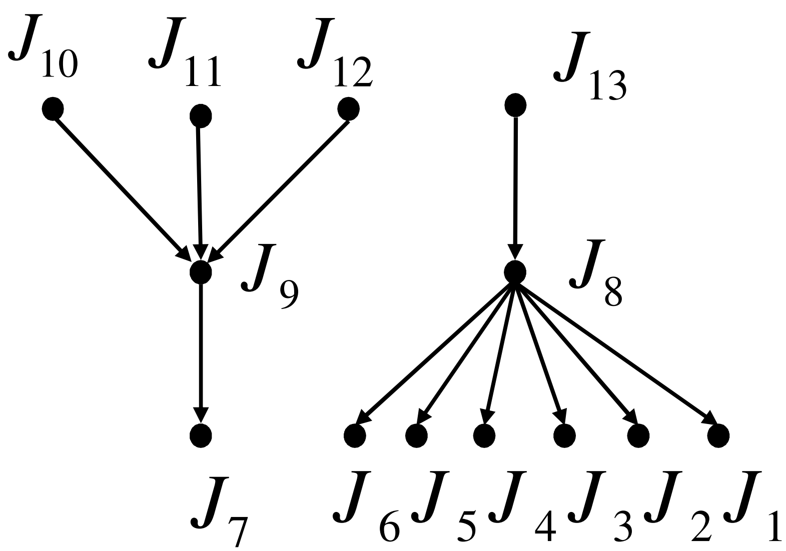
   $k\text{-clique} \leq Pm \mid prec, p_j = 1 \mid C_{max}$

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

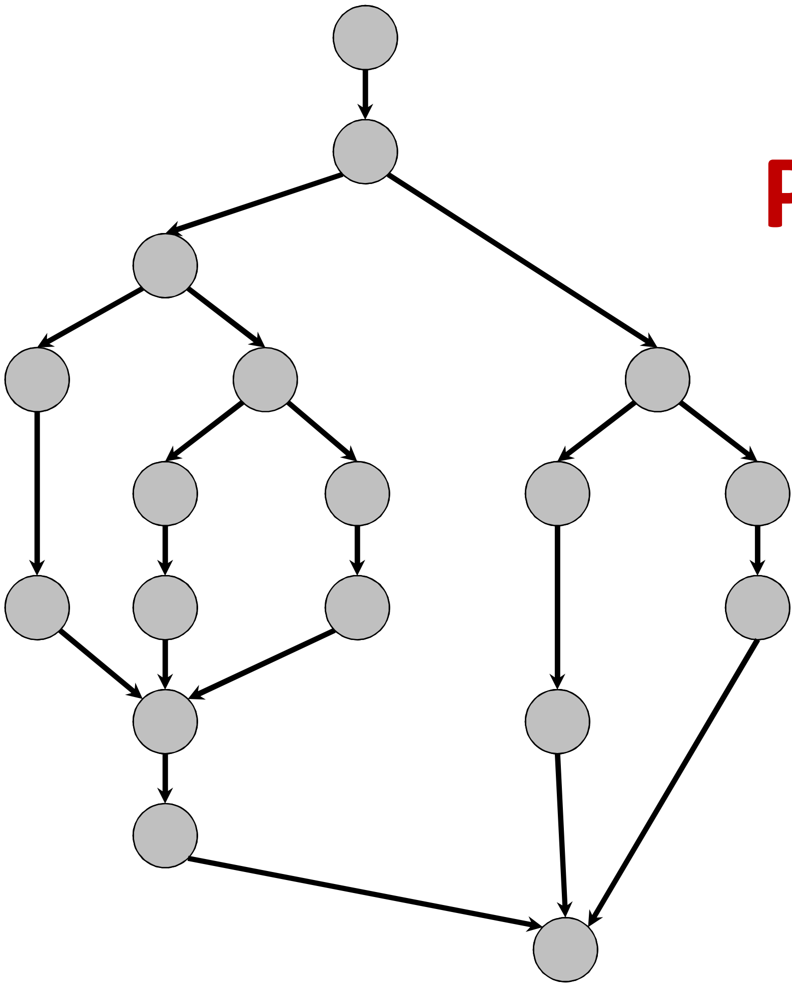
- Graham first analyzed the performance of the simplest list scheduling algorithm.
- List scheduling algorithm with an arbitrary job list is called Graham's list algorithm.
- Approximation ratio for $Pm \mid prec, p_j = 1 \mid C_{max}$

  $\delta = 2 - 1/m$. (Tight bound!)

  - Approximation ratio is δ if for each input instance, the makespan produced by the algorithm is at most δ times of the optimal makespan.

CP Algo: CLR Book Page 779-783

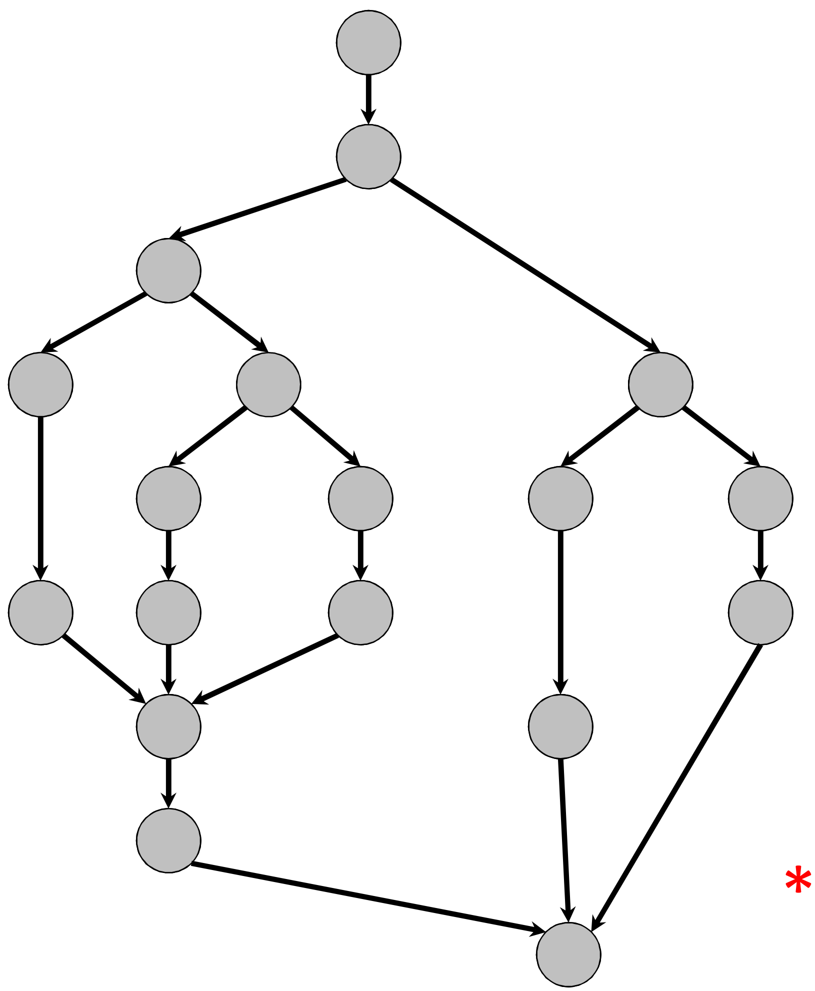
T_P = execution time on P processors



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

CP Algorithms

T_P = execution time on P processors



T_1 = ***work***

T_∞ = ***span****

Lower Bounds

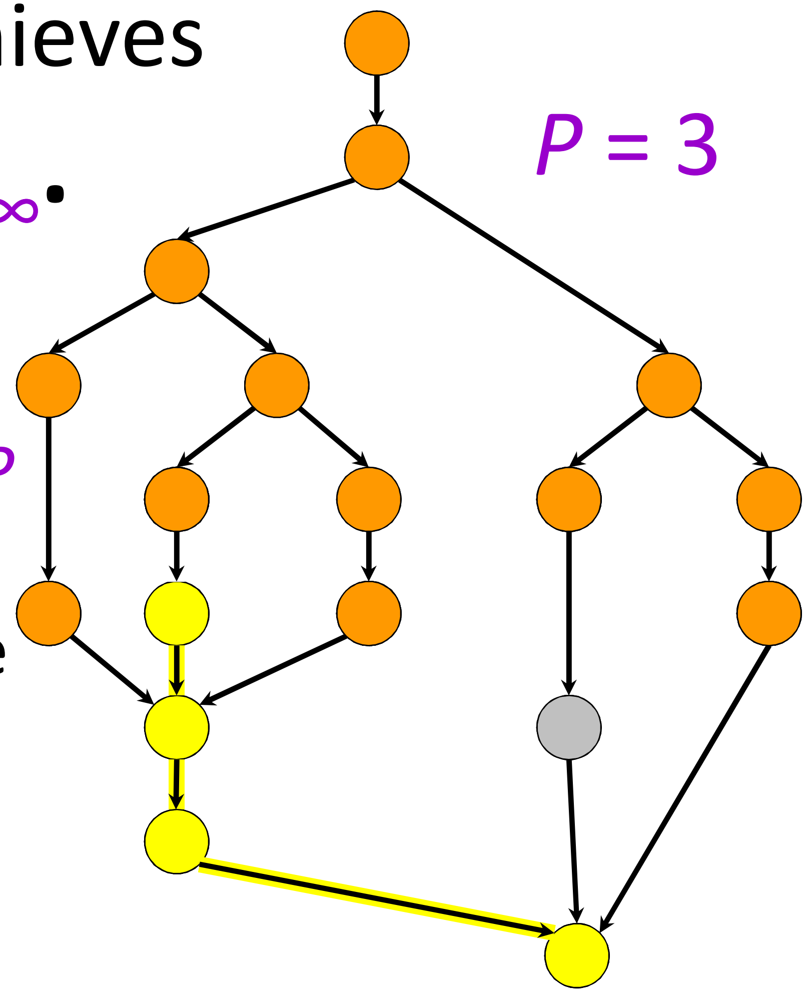
- $T_P \geq T_1/P$
- $T_P \geq T_\infty$

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

CP: Greedy-Scheduling Theorem

Theorem [Graham '68 & Brent '75].
Any greedy scheduler achieves

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



Proof.

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

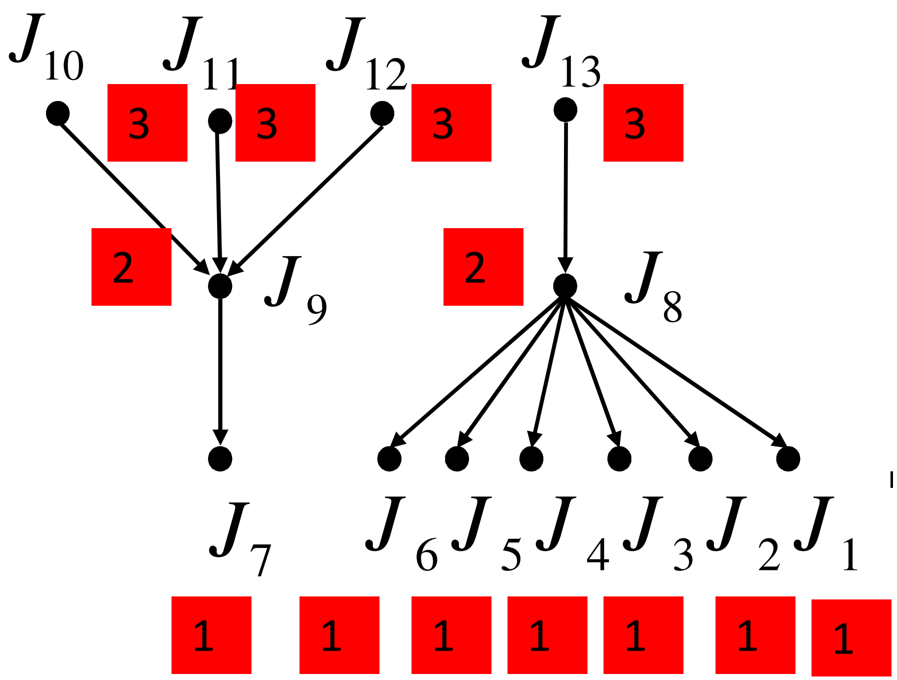
CP: Optimality of Greedy

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

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

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:

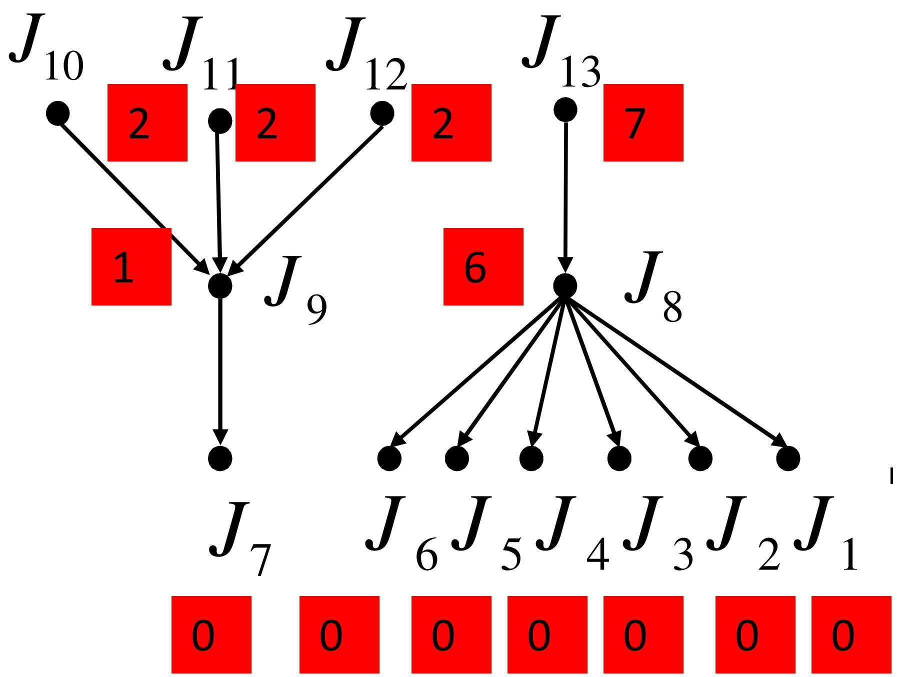
If $m = 2$, $\delta_{HLF} \leq 4/3$.

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

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 100W 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