# Due Date Related Objectives

# Mean Lateness

Mean lateness is minimized by SPT rule

$$\bar{L} = \frac{\sum_{j=1}^{n} L_j}{n} = \frac{\sum_{j=1}^{n} (C_j - d_j)}{n} = \frac{1}{n}\sum_{j=1}^{n} C_j - \frac{1}{n}\sum_{j=1}^{n} d_j$$

$$\bar{L} = \frac{1}{n}\sum_{j=1}^{n} C_j - K$$

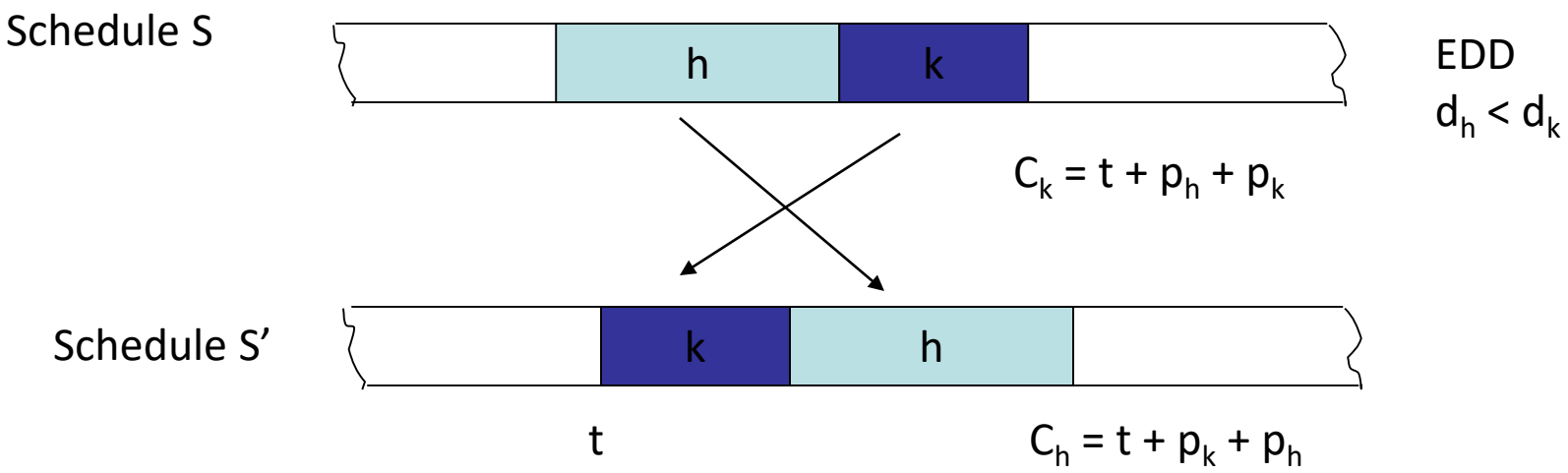
# Maximum Lateness

Maximum lateness ($L_{max}$) is minimized by Earliest Due Date (EDD) rule

EDD

Arrange the jobs in increasing order of their due dates

i.e. $d_{(1)} \leq d_{(2)} \leq \ldots \leq d_{(n)}$




$L_h(S) = t+p_h - d_h$

$L_k(S) = t+p_h+p_k- d_k$

$L_h(S') = t+p_h+p_k- d_h$

$L_k(S') = t+p_k- d_k$

$L_{max} = \max (L_h(S'), L_k(S))$

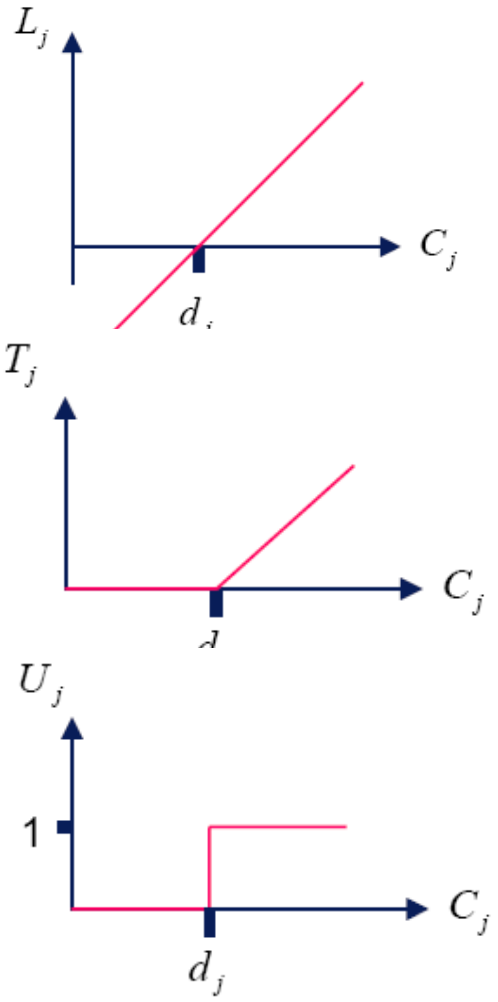
Schedule S is better

# Maximum Tardiness

- Maximum Tardiness ($T_{max}$) is minimized by EDD rule
  - $T_j = \max(0, C_j - d_j)$

# Maximum Lateness

- 1| prec | $h_{max}$
  - $h_{max} = \max(h_1(C_1),\ldots,h_n(C_n))$
  - Lateness is a function of non-decreasing cost
- For lateness, $h_j = C_j - d_j$
- Backward dynamic programming
  - $C_{max} = \Sigma_j p_j$
  - J – scheduled jobs
  - $J^c$ – unscheduled jobs
  - J'- schedulable jobs
    - Subset of $J^c$



# Algorithm 1| prec | $h_{max}$

1. Set $J = \varnothing$, $J^c = \{1,\ldots,n\}$ and let $J'$ be all jobs that have no successors
2. Determine

$$j^* = \arg\min_{j \in J'} h_j\left(\sum_{k \in J^c} p_k\right)$$

   Add $j^*$ to $J$

   Delete $j^*$ from $J^c$

   Modify $J'$ to represent the new set of schedulable jobs (have no successors)
3. If $J^c = \varnothing$ STOP, otherwise go to 2

# Example

| jobs | 1 | 2 | 3 |
|---|---|---|---|
| $p_j$ | 2 | 3 | 5 |
| $h_j(C_j)$ | $1 + C_j$ | $1.2\ C_j$ | 10 |

$J=\Phi$, $J^c = \{1, 2, 3\}$, $J' = \{1, 2, 3\}$, $\Sigma_{j \text{ in } \{1,2,3\}} p_j = 10$

$h_1(10) = 1+10 = 11$ $h_2(10) = 1.2(10) = 12$ $h_3(10) = 10$

$j^* = 3$ is scheduled last

$J=\{3\}$, $J^c = \{1, 2\}$, $J' = \{1, 2\}$, $\Sigma_{j \text{ in } \{1,2\}} p_j = 10-5 = 5$

$h_1(5) = 1+5 = 6$ $h_2(5) = 1.2(5) = 6$

$j^* = 1$ or 2

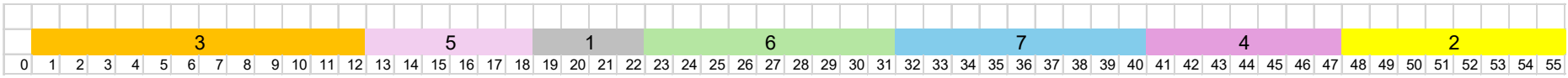
Optimal sequence is 1-2-3 or 2-1-3

# Exercise Problem 3.4

**3.4.** Find all optimal sequences for the instance of $1 \mid\mid h_{\max}$ with the following jobs.

| *jobs* | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $p_j$ | 4 | 8 | 12 | 7 | 6 | 9 | 9 |
| $h_j(C_j)$ | $3\,C_1$ | 77 | $C_3^2$ | $1.5\,C_4$ | $70+\sqrt{C_5}$ | $1.6\,C_6$ | $1.4\,C_7$ |

| j | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| p | 4 | 8 | 12 | 7 | 6 | 9 | 9 |




| *j* | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $p_j$ | 4 | 8 | 12 | 7 | 6 | 9 | 9 |
| $C_j$ | 22 | 55 | 12 | 47 | 18 | 31 | 40 |
| hj(Cj) | 66 | 77 | 144 | 70.5 | 74.24 | 49.6 | 56 |

# Exercise Problem 3.5

**3.4.** Find all optimal sequences for the instance of $1 \mid\mid h_{\max}$ with the following jobs.

| *jobs* | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $p_j$ | 4 | 8 | 12 | 7 | 6 | 9 | 9 |
| $h_j(C_j)$ | $3\,C_1$ | 77 | $C_3^2$ | $1.5\,C_4$ | $70+\sqrt{C_5}$ | $1.6\,C_6$ | $1.4\,C_7$ |

**3.5.** Consider $1 \mid prec \mid h_{\max}$ with the same set of jobs as in Exercise 3.4 and the following precedence constraints.

$$1 \rightarrow 7 \rightarrow 6$$

$$5 \rightarrow 7$$

$$5 \rightarrow 4$$

Find the optimal sequence.

**3.4.** Find all optimal sequences for the instance of $1 \mid\mid h_{\max}$ with the following jobs.

| *jobs* | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $p_j$ | 4 | 8 | 12 | 7 | 6 | 9 | 9 |
| $h_j(C_j)$ | $3\,C_1$ | 77 | $C_3^2$ | $1.5\,C_4$ | $70+\sqrt{C_5}$ | $1.6\,C_6$ | $1.4\,C_7$ |

**3.5.** Consider $1 \mid prec \mid h_{\max}$ with the same set of jobs as in Exercise 3.4 and the following precedence constraints.

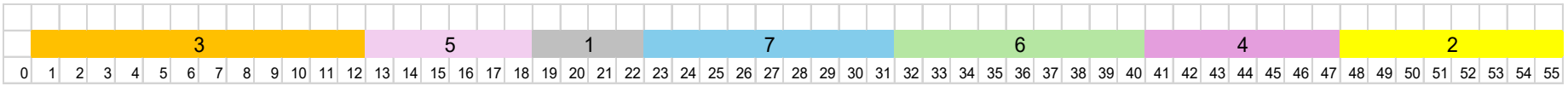
$$1 \rightarrow 7 \rightarrow 6$$

$$5 \rightarrow 7$$

$$5 \rightarrow 4$$

Find the optimal sequence.

| j | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| p | 4 | 8 | 12 | 7 | 6 | 9 | 9 |
| hj(55) | | 77 | 3025 | 82.5 | | 88 | |
| hj(47) | | | 2209 | 70.5 | | 75.2 | |
| hj(40) | | | 1600 | | | 64 | |
| hj(31) | | | 961 | | | | 43.4 |
| hj(22) | 66 | | 484 | | 74.69042 | | |
| hj(18) | | | 324 | | 74.24264 | | |
| hj(12) | | | 144 | | | | |



J={}, $J^c = \{1, 2, 3, 4, 5, 6, 7\}$ and J'={2, 3, 4, 6}. Then;

$$\begin{aligned} h_{J^*}(\sum_{j \in J^c} p_j) &= \min_{J \in J'}(h_j(\sum_{k \in J^c} p_k)) \\ h_{J^*}(55) &= \min\{h_2(55), h_3(55), h_4(55), h_6(55)\} \\ &= \min\{77, 3025, 82.5, 88\} \\ &= 77 \end{aligned}$$

So job 2 is scheduled last and it will finish at t=55.

Iteration 2:

J={2}, $J^c = \{1, 3, 4, 5, 6, 7\}$ and J'={3, 4, 6}. Then;

$$\begin{aligned} h_{J^*}(\sum_{j \in J^c} p_j) &= \min_{J \in J'}(h_j(\sum_{k \in J^c} p_k)) \\ h_{J^*}(47) &= \min\{h_3(47), h_4(47), h_6(47)\} \\ &= \min\{2209, 70.5, 75.2\} \\ &= 70.5 \end{aligned}$$

So job 4 is scheduled before job 2.

Iteration 3:

J=$\{4, 2\}$, $J^c = \{1, 3, 5, 6, 7\}$ and J'=$\{3, 6\}$. Then;

$$\begin{aligned} h_{J^*}(\sum_{j \in J^c} p_j) &= \min_{J \in J'}(h_j(\sum_{k \in J^c} p_k)) \\ h_{J^*}(40) &= \min(h_3(40), h_6(40)) \\ &= \min\{1600, 64\} \\ &= 64 \end{aligned}$$

So job 6 is scheduled before job 4.

Iteration 4:

J=$\{6, 4, 2\}$, $J^c = \{1, 3, 5, 7\}$ and J'=$\{3, 7\}$. Then;

$$\begin{aligned} h_{J^*}(\sum_{j \in J^c} p_j) &= \min_{J \in J'}(h_j(\sum_{k \in J^c} p_k)) \\ h_{J^*}(31) &= \min\{h_3(31), h_7(31)\} \\ &= \min\{961, 43.4\} \\ &= 43.4 \end{aligned}$$

So job 7 is scheduled before job 6.

Iteration 5:

J=$\{7,6,4,2\}$, $J^c = \{1,3,5\}$ and J'=$\{1,3,5\}$. Then;

$$\begin{aligned} h_{J^*}(\sum_{j\in J^c} p_j) &= \min_{J\in J'}(h_j(\sum_{k\in J^c} p_k)) \\ h_{J^*}(22) &= \min\{h_1(22), h_3(22), h_5(22),\} \\ &= \min\{66, 484, 74.69\} \\ &= 66 \end{aligned}$$

So job 1 is scheduled before job 7.

Iteration 6:

J=$\{1,7,6,4,2\}$, $J^c = \{3,5\}$ and J'=$\{3,5\}$. Then;

$$\begin{aligned} h_{J^*}(\sum_{j\in J^c} p_j) &= \min_{J\in J'}(h_j(\sum_{k\in J^c} p_k)) \\ h_{J^*}(18) &= \min\{h_3(18), h_5(18),\} \\ &= \min\{324, 74.24\} \\ &= 74.24 \end{aligned}$$

So job 5 is scheduled before job 1.

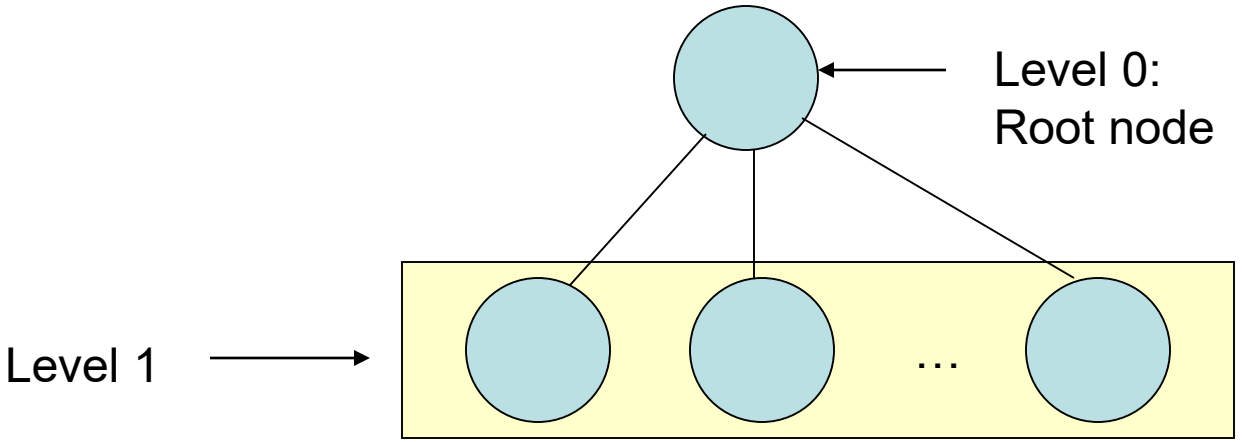
Optimal sequence is 3,5,1,7,6,4,2 and $\min(h_{max}) = 144$

# $1 \mid r_j, \text{prmp} \mid L_{max}$

- Preemptive EDD rule guarantees an optimum for $1 \mid r_j, \text{prmp} \mid L_{max}$

# $1 \mid r_j \mid L_{max}$

- Strongly NP-hard
- Branch and bound approach
  - Enumeration procedures



# $1 \mid r_j \mid L_{max}$

- Level 0; root node
- Branching
  - Level 1: n nodes; job *j* is scheduled first in node *j*
  - Level 2: n(n-1) nodes; a job is assigned to the second position
  - …
  - Level k: n(n-1)..(n-k-1) nodes; the first k positions are filled

# $1 \mid r_j \mid L_{max}$

- Suppose k-1 jobs are scheduled
- For the $k^{th}$ position, consider only jobs which satisfy

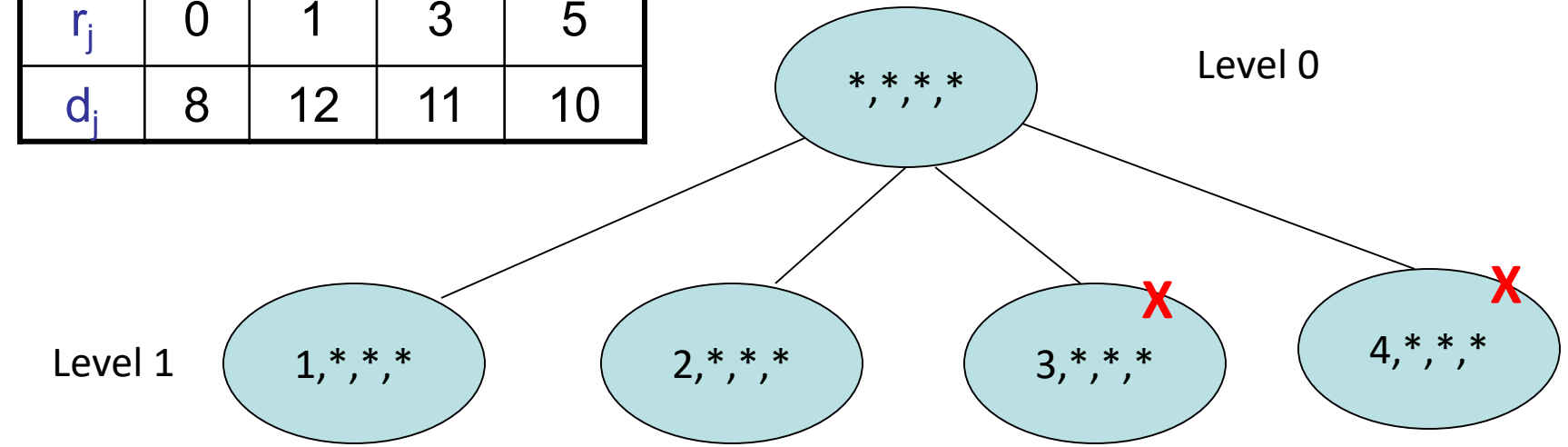
$$r_k < \min_{l \in J}(\max(t, r_l) + p_l)$$

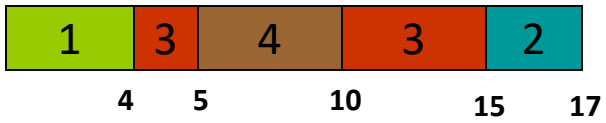
  - $t$ is the time when job $k$ is supposed to start
  - $J$ is the set of unscheduled jobs

- Bounding
  - $1 \mid r_j, \text{prmp} \mid L_{max}$ preemptive EDD rule
  - Non-delay schedule
  - It is optimal if it is non-preemptive

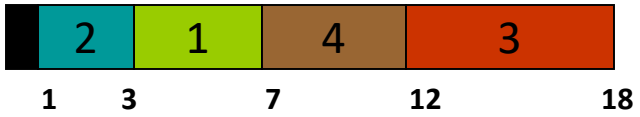
# $1 \mid r_j \mid L_{max}$ Example

| jobs | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $p_j$ | 4 | 2 | 6 | 5 |
| $r_j$ | 0 | 1 | 3 | 5 |
| $d_j$ | 8 | 12 | 11 | 10 |




$L_{max} = \max(0,5,4,0) = 5$

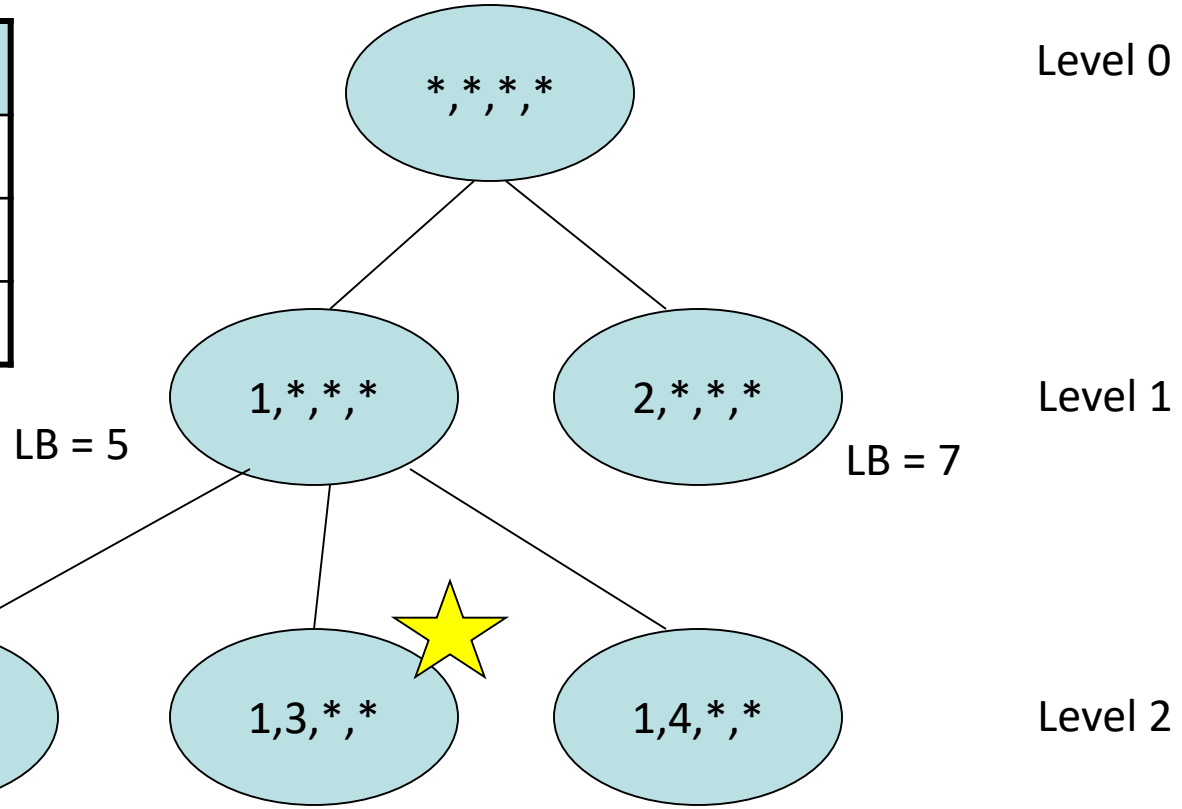


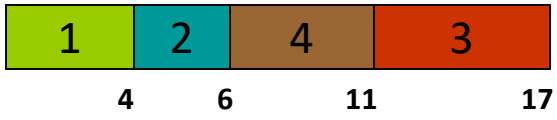

$L_{max} = \max(0,0,7,2) = 7$

Job 2 can be completed before jobs 3 and 4 are released

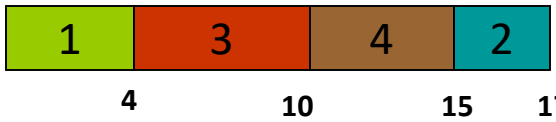
# 1 | $r_j$ | $L_{max}$ Example

| jobs | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $p_j$ | 4 | 2 | 6 | 5 |
| $r_j$ | 0 | 1 | 3 | 5 |
| $d_j$ | 8 | 12 | 11 | 10 |

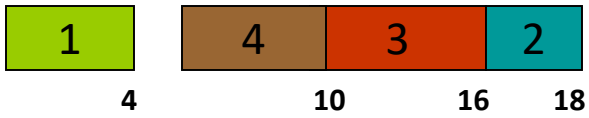



$L_{max} = \max(0,0,6,1) = 6$




$L_{max} = \max(0,5,0,5) = 5$




$L_{max} = \max(0,6,5,0) = 6$

# $1 \mid r_j, \text{prec} \mid L_{max}$

- Branch-and-bound
  - Similar to $1 \mid r_j \mid L_{max}$
  - Easier to solve

# Partition

- Given positive integers $a_1, a_2, \ldots, a_n$ and $b = (1/2)\Sigma_j a_j$, do there exist two disjoint subsets $S_1$ and $S_2$ such that

$$\sum_{j \in S_1} a_j = b$$

$$\sum_{j \in S_2} a_j = b$$

- For example, consider $a_j = \{1, 2, 2, 3\}$

  $b = 4$

  $S_1 = \{a_1, a_4\}$ and $S_2 = \{a_2, a_3\}$

# 3-Partition

Given positive integers $a_1, a_2, ..., a_{3t}, b$ with

$$\frac{b}{4} < a_j < \frac{b}{2} \qquad j = 1,2,...,3t$$

and

$$\sum_{j=1}^{3t} a_j = tb$$

do there exist $t$ pairwise three element subsets $S_i \subset \{1,2,...,3t\}$ such that $\sum_{j \in S_i} a_j = b$ for $i = 1,...,t$?

# $1 \mid r_j \mid L_{max}$ NP-hard Proof

Reduction of 3-Partition to $1 \mid r_j \mid L_{max}$

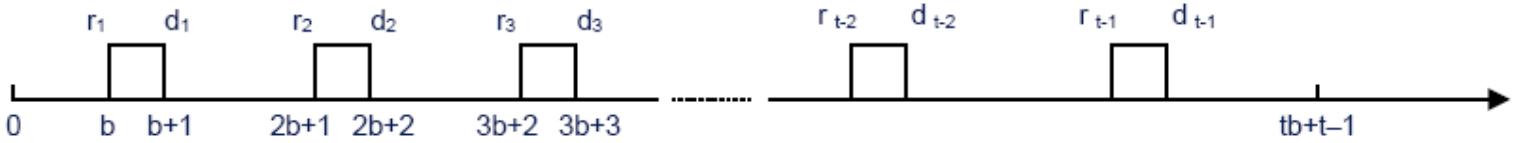
integers $a_1, \ldots, a_{3t}, b$ $\quad \frac{b}{4} < a_j < \frac{b}{2} \quad \sum_{j=1}^{3t} a_j = t \cdot b$

➔ n = 4t –1 jobs

$r_j = j \cdot b + (j - 1), \quad p_j = 1, \quad d_j = j \cdot b + j, \quad \forall\ j = 1, \ldots, t-1$

$r_j = 0, \quad p_j = a_{j-t+1}, \quad d_j = t \cdot b + (t-1), \quad \forall\ j = t, \ldots, 4t-1$

$L_{max} = 0$ if every job $j \in \{1, \ldots, t-1\}$ can be processed from $r_j$ to $r_j + p_j = d_j$ and all other jobs can be partitioned over t intervals of length b.



# Number of Tardy Jobs ($\Sigma_j U_j$)

- Easy problem
- Forward algorithm

- Jobs divided into two sets
  - Set 1: All jobs which can meet their due date
    - Sequenced in EDD rule
  - Set 2: All jobs which cannot meet their due date
    - Any order

# Algorithm

J set of scheduled jobs
$J^c$ set of unscheduled jobs
$J^d$ set of discarded jobs

**Step 1** Set $J = \varnothing$, $J^d = \varnothing$, and $J^c = \{1, \ldots, n\}$.

**Step 2** Let $j^*$ denote the job that satisfies $d_{j^*} = \min_{j \in J^c} (d_j)$.
Add $j^*$ to J.
Delete $j^*$ from $J^c$.
Go to Step 3.

**Step 3** If $\sum_{j \in J} p_j \leq d_{j^*}$ then go to Step 4,

otherwise
let $k^*$ denote the job which satisfies $p_{k^*} = \max_{j \in J} (p_j)$.
Delete $k^*$ from J.
Add $k^*$ to $J^d$.

**Step 4** If $J^c = \varnothing$ then STOP, otherwise go to Step 2.

# Example – Number of Tardy Jobs

| jobs | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $p_j$ | 7 | 8 | 4 | 6 | 6 |
| $d_j$ | 9 | 17 | 18 | 19 | 21 |

$J = \{\ \}$, $J^c = \{1,2,3,4,5\}$, $J^d = \{\ \}$

j* = 1 job 1 fits $J = \{1\}$, $J^c = \{2,3,4,5\}$, $J^d = \{\ \}$

j* = 2 job 2 fits $J = \{1,2\}$, $J^c = \{3,4,5\}$, $J^d = \{\ \}$

j* = 3 job 3 does not fit;
remove job 2 from J $J = \{1,3\}$, $J^c = \{4,5\}$, $J^d = \{2\}$

j* = 4 job 4 fits $J = \{1,3,4\}$, $J^c = \{5\}$, $J^d = \{2\}$

j* = 5 job 5 does not fit;
remove job 1 from J $J = \{3,4,5\}$, $J^c = \{\ \}$, $J^d = \{1,2\}$

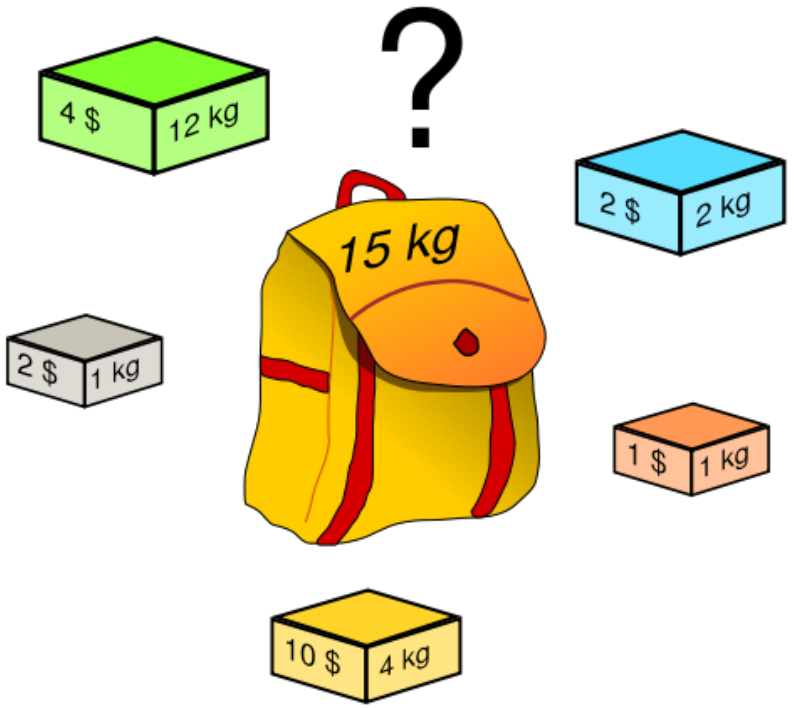
Optimal solution
3,4,5,1,2

$\Sigma_j U_j = 2$

# Weighted Number of Tardy Jobs ($\Sigma_j w_j U_j$)

- $1 \mid\mid \Sigma_j w_j U_j$ NP-hard
- $1 \mid d_j=d \mid \Sigma_j w_j U_j$
  - NP-hard in ordinary sense
  - Knapsack problem

# Knapsack Problem




Given n items and the value and weight of each item, maximize the value of the total items added to the knapsack

The knapsack can carry only limited items

$$Maximize \quad \sum_{j \in J} v_j x_j$$

$$subject\ to$$

$$\sum_{j \in J} s_j x_j \leq S$$

$$x_j \in \{0,1\}$$

# $1 \mid d_j = d \mid \Sigma_j w_j U_j$

- Knapsack problem
  - Due date $d$ ↔ max weight $S$ allowed
  - Weight of job $j$ ($w_j$) ↔ value of item $j$ ($v_j$)
  - Processing time of job $j$ ($p_j$) ↔ size of item $j$ ($s_j$)

$$\text{Maximize} \quad \sum_{j \in J} w_j x_j$$

$$\text{subject to}$$

$$\sum_{j \in J} p_j x_j \leq d$$

$$x_j \in \{0,1\}$$

- WSPT heuristic
  - Arrange the jobs in decreasing order of w/p

# Total Tardiness

- $1 \mid\mid \Sigma_j T_j$ – NP hard in ordinary sense
  - Pseudo polynomial time algorithm based on dynamic programming

- Properties
  - If $p_j \leq p_k$ and $d_j \leq d_k$ then there exists an optimal sequence in which job $j$ is scheduled before job $k$
    - Dominance result

  - $d_1 \leq d_2 \leq \ldots \leq d_n$ and $p_k = \max(p_1, p_2, \ldots, p_n)$
    - $k^{th}$ smallest due date has the largest $p$
    - jobs {1,..,$k$-1} appear in the same order before $k$
    - jobs {$k$+1, …,$n$} some before and some after $k$

# Total Tardiness

- There exists an integer $\delta$, $0 \le \delta \le n-k$, such that there is an optimal sequence S in which
  - Job $k$ is preceded by all jobs $j$ with $j \le k+\delta$, and
  - Followed by all jobs $j$ with $j > k+\delta$

- Optimal sequence for the set of jobs $1,\ldots,l$ is concatenation of
  - $1,2,\ldots,k-1,k+1,\ldots, k+\delta$
  - Followed by $k$
  - Followed by $k+\delta+1,\ldots,l$

# Total Tardiness

- J(j,l,k) = jobs in set {j, j+1, …, *l* } with the processing times of all the jobs in the set smaller than or equal to job k

- V(J(j,l,k),t) total tardiness of the set J when these jobs are started at time t

# Total Tardiness Algorithm

- Initial Conditions
  - $V(\Phi,t) = 0$, $V(\{ j \}, t) = \max(0, t + p_j - d_j)$
- Recursive Relations
  - $V(J(j,l,k),t) = \min(V(J(j,k'+\delta,k'),t) + \max(0, C_{k'}(\delta)-d_{k'}) + V(J(k'+\delta+1,l,k'), C_{k'}(\delta)) )$
  - $k'$ is such that $p_{k'} = \max\{p_j \mid j' \varepsilon J(j,l,k))$
  - $C_{k'}(\delta) = \Sigma_{j \le k+\delta} p_j$
- Optimal value function
  - $V(\{1,\ldots,n\},0)$

# Total Tardiness Example

| jobs | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $p_j$ | 121 | 79 | 147 | 83 | 130 |
| $d_j$ | 260 | 266 | 266 | 336 | 337 |

The jobs are already arranged in EDD

k' = job 3 $\quad 0 \le \delta \le (n-k) = 2$

V({1,2,…,5},0) = min {
V(J(1,3,3),0) + max(0,121+79+147-266) + V(J(4,5,3),347)
V(J(1,4,3),0) + max(0,121+79+147+83-266) + V(J(5,5,3),430)
V(J(1,5,3),0) + max(0,121+79+147+83+130-266) + V(Φ,560) }

V(J(1,3,3),0) = 0 optimal sequence 1→2 or 2 →1
V(J(4,5,3),347) = 94 + 223 = 317 optimal sequence 4→5

V(J(1,4,3),0) = 0 optimal sequence 1→2→4 or 2 →1 →4
V(J(5,5,3),430) = 223

V(J(1,5,3),0) = 76 optimal sequence 1→2→4 →5 or 2 →1 →4→5

# Total Tardiness Example

- V({1,..,5},0) =

  min { 0 + 81 + 317, 0 + 164 + 223, 76 + 294 + 0 } = 370

  Optimal sequences

  1→2→4 →5 →3 or 2 →1 →4→5 →3

# Total Weighted Tardiness

- $1 \mid\mid \Sigma_j w_j T_j$ is strongly NP-hard

- If there are two jobs *j* and *k* with $d_j \leq d_k$, $p_j \leq p_k$ and $w_j \geq w_k$, then there exists an optimal sequence in which job *j* appears before job *k*

- Lower bounds
  - Transportation problem
  - $p_j$ is treated as $p_j$ jobs with unit processing time

# Total Weighted Tardiness

$$x_{jk} = \begin{cases} 1, \text{if one unit of job j is processed in the interval [k-1, k]} \\ 0, \text{otherwise} \end{cases}$$

$$\sum_{k=1}^{C_{\max}} x_{jk} = p_j \qquad \forall j = 1,\ldots,n$$

$$\sum_{j=1}^{n} x_{jk} = 1 \qquad \forall k = 1,\ldots,C_{\max}$$

$$c_{jk} = \begin{cases} 0, \text{for } k \le d_j \\ w_j, \text{for } k > d_j \end{cases}$$

The schedule need not be feasible

$$Minimize \sum_{k=1}^{C_{\max}} c_{jk} x_{jk}$$

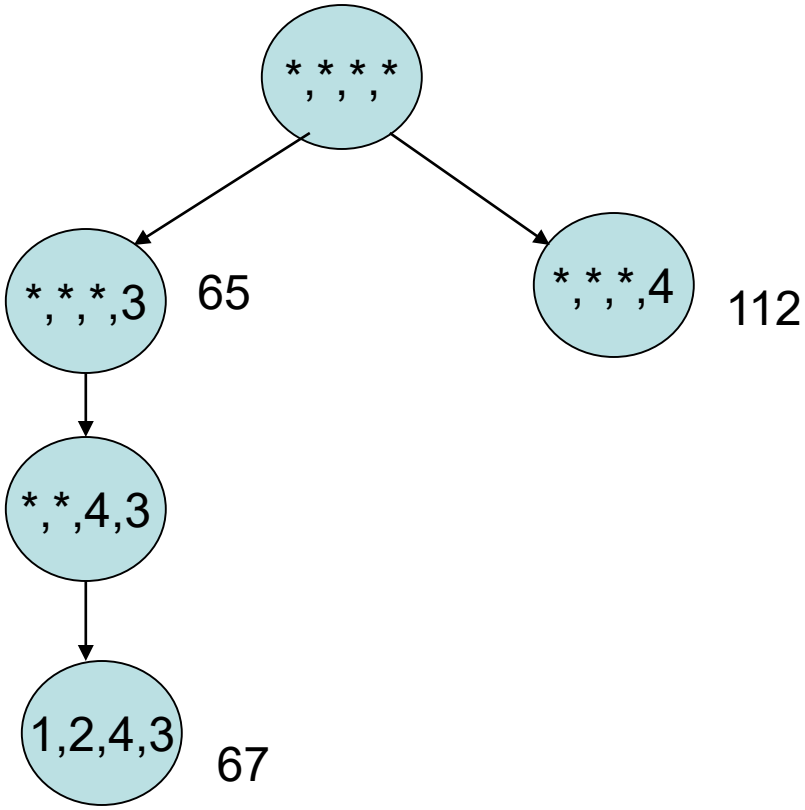
# Total Weighted Tardiness Example

| jobs | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $w_j$ | 4 | 5 | 3 | 5 |
| $p_j$ | 12 | 8 | 15 | 9 |
| $d_j$ | 16 | 26 | 25 | 27 |

$c_{1k} = 0$ for k = 1,…,16
$c_{1k} = 4$ for k = 17,…,44
$c_{2k} = 0$ for k = 1,…,26
$c_{2k} = 5$ for k = 27,…,44
$c_{3k} = 0$ for k = 1,…,24
$C_{3k} = 3$ for k = 26,…,44
$c_{4k} = 0$ for k = 1,…,27
$c_{4k} = 5$ for k = 28,…,44

$p_2 < p_4$, $d_2 < d_4$, $w_2 = w_4$ $2 \rightarrow 4$

$p_1 < p_3$, $d_1 < d_3$, $w_1 > w_3$ $1 \rightarrow 3$



# Due Date Related Objectives

- Mean Lateness – SPT
- Maximum Lateness – SPT
- Maximum Tardiness – EDD
- $1|prec|h_{max}$ backward DP
- $1|r_j, prmp| L_{max}$ - preemptive EDD
- $1|r_j| L_{max}$ – NP-hard, B&B
- $1|r_j, prec| L_{max}$ – B&B
- $1||\Sigma_j U_j$ – Heuristic
- $1||\Sigma_j w_j U_j$ – NP-hard, Knapsack
- $1|d_j=d|\Sigma_j w_j U_j$ – Knapsack, WSPT
- $1||\Sigma_j T_j$ – NP-hard, DP
- $1||\Sigma_j w_j T_j$ – B&B, Transportation problem