PARALLEL MACHINE MODELS (DETERMINISTIC)

Chapter 5, “Scheduling: Theory, Algorithms & Systems”, Pinedo

IE 661 Scheduling Theory
Fall 2003
Department of Industrial Engineering
University at Buffalo (SUNY)

September 2003

Outline

- Introduction
- Makespan without preemptions
- Makespan with preemptions
- Total completion time without preemptions
- Total completion time with preemptions
- Due-Date related objectives

Introduction

- Parallel machines: generalization of single machine, special case of flexible flow shop
- 2 step process
  1. allocation of jobs to machines
  2. sequence of jobs on a machine
- Assumption: $p_1 \geq p_2 \geq \ldots\ldots \geq p_n$
- Consider three objectives: minimize
  1. makespan
  2. total completion time
  3. maximum lateness

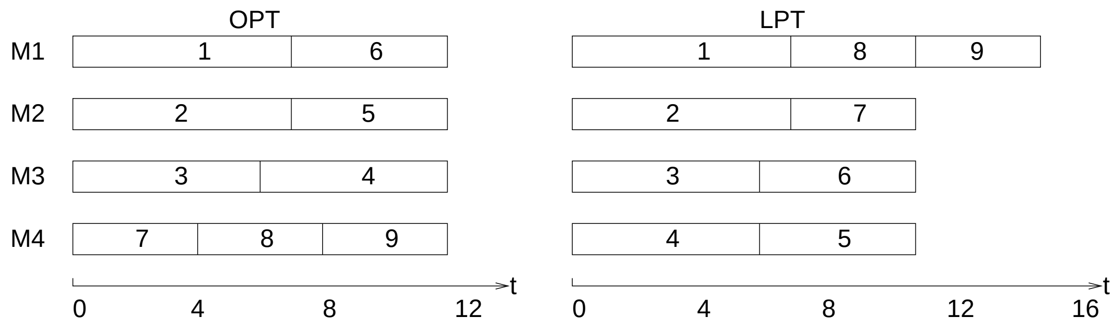
MAKESPAN WITHOUT PREEMPTIONS

Longest Processing Time Heuristic

- Consider $Pm||c_{max}$
- Special case: $P2||c_{max}$: NP-hard in the ordinary sense
- LPT:
  1. assign at $t = 0$, m largest jobs to m machines
  2. assign remaining job with longest processing time to next free machine
- Theorem 5.1.1: Upper bound for $\frac{c_{max}(LPT)}{c_{max}(OPT)}$: $\frac{c_{max}(LPT)}{c_{max}(OPT)} \leq \frac{4}{3} - \frac{1}{3m}$
- Proof: by contradiction

LPT: A Worst Case Example

Jobs	1	2	3	4	5	6	7	8	9
p_j	7	7	6	6	5	5	4	4	4



- 4 parallel machines
- $c_{max}(OPT) = 12$, $c_{max}(LPT) = 15$
- $\frac{c_{max}(LPT)}{c_{max}(OPT)} = \frac{15}{12}$
- $\frac{4}{3} - \frac{1}{3m} = \frac{15}{12}$

LPT: Proof

- Contradiction: Counter example with smallest n
  1. Property: Shortest job n is the
     1.1. last job to start processing (LPT)
     1.2. last job to finish processing
  2. If n is not the last job to finish processing, then:
     2.1. deletion of n does not change $c_{max}(LPT)$
     2.2. but it may reduce $c_{max}(OPT)$ (or remain same)
- A counter example with $n-1$ jobs
- All machines busy in time interval $[0, c_{max}(LPT) - p_n]$
- $c_{max}(LPT) - p_n \leq \frac{\sum_{j=1}^{n-1} p_j}{m}$

$$\Rightarrow c_{max}(LPT) \leq p_n + \frac{\sum_{j=1}^{n-1} p_j}{m} = p_n(1 - \frac{1}{m}) + \frac{\sum_{j=1}^{n} p_j}{m}$$

LPT: Proof .... Contd.

- $\frac{\sum_{j=1}^{n} p_j}{m} \leq c_{max}(OPT)$

- $\frac{4}{3} - \frac{1}{3m} < \frac{c_{max}(LPT)}{c_{max}(OPT)} \leq \frac{p_n(1-\frac{1}{m})+\frac{\sum_{j=1}^{n} p_j}{m}}{c_{max}(OPT)} =$

$$\frac{p_n(1-\frac{1}{m})}{c_{max}(OPT)} + \frac{\frac{\sum_{j=1}^{n} p_j}{m}}{c_{max}(OPT)} \leq \frac{p_n(1-\frac{1}{m})}{c_{max}(OPT)} + 1$$

- $\frac{4}{3} - \frac{1}{3m} < \frac{p_n(1-\frac{1}{m})}{c_{max}(OPT)} + 1 \Rightarrow c_{max}(OPT) < 3p_n$
- On each machine at most 2 jobs
- LPT is optimal for this case □

Precedence Constraints

- Arbitrary ordering of jobs: $\frac{c_{max}(LIST)}{c_{max}(OPT)} \leq 2 - \frac{1}{m}$ for LPT
- Better algorithms (bounds) exist
- $P_m|prec|c_{max} \Rightarrow$ at least as hard as $P_m||c_{max}$ (strongly NP hard for $2 \leq m < \infty$)
- Special case $m \geq n \Rightarrow P\infty|prec|c_{max}$
  - $P_m|p_j = 1, prec|c_{max} \rightarrow$ NP hard
  - $P_m|p_j = 1, tree|c_{max} \rightarrow$ easily solvable with Critical Path Method (CPM)
    * intree
    * outtree

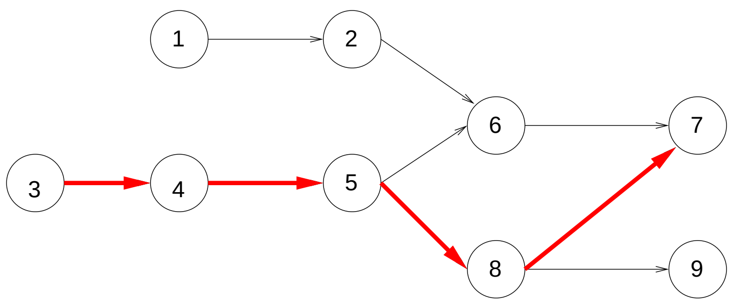
CPM: An Example

jobs	1	2	3	4	5	6	7	8	9
p_j	4	9	3	3	6	8	8	12	6

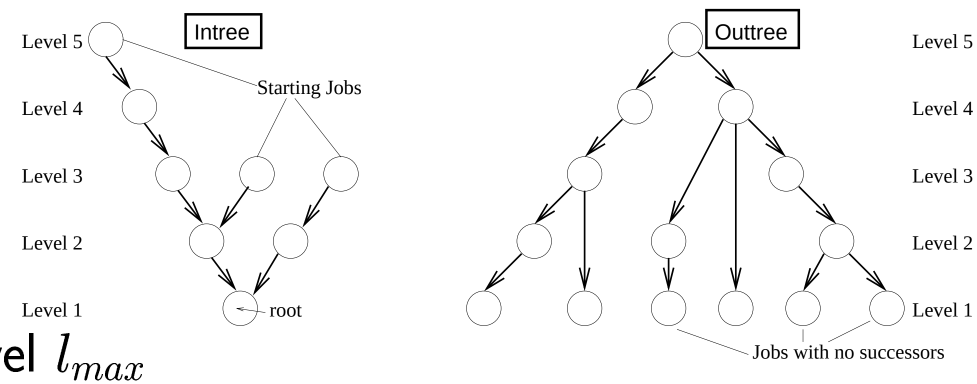
c'_j = earliest completion time of job j

c''_j = latest possible completion time of job j

jobs	1	2	3	4	5	6	7	8	9
c'_j	4	13	3	6	12	21	32	24	30
c''_j	7	16	3	6	12	24	32	24	32



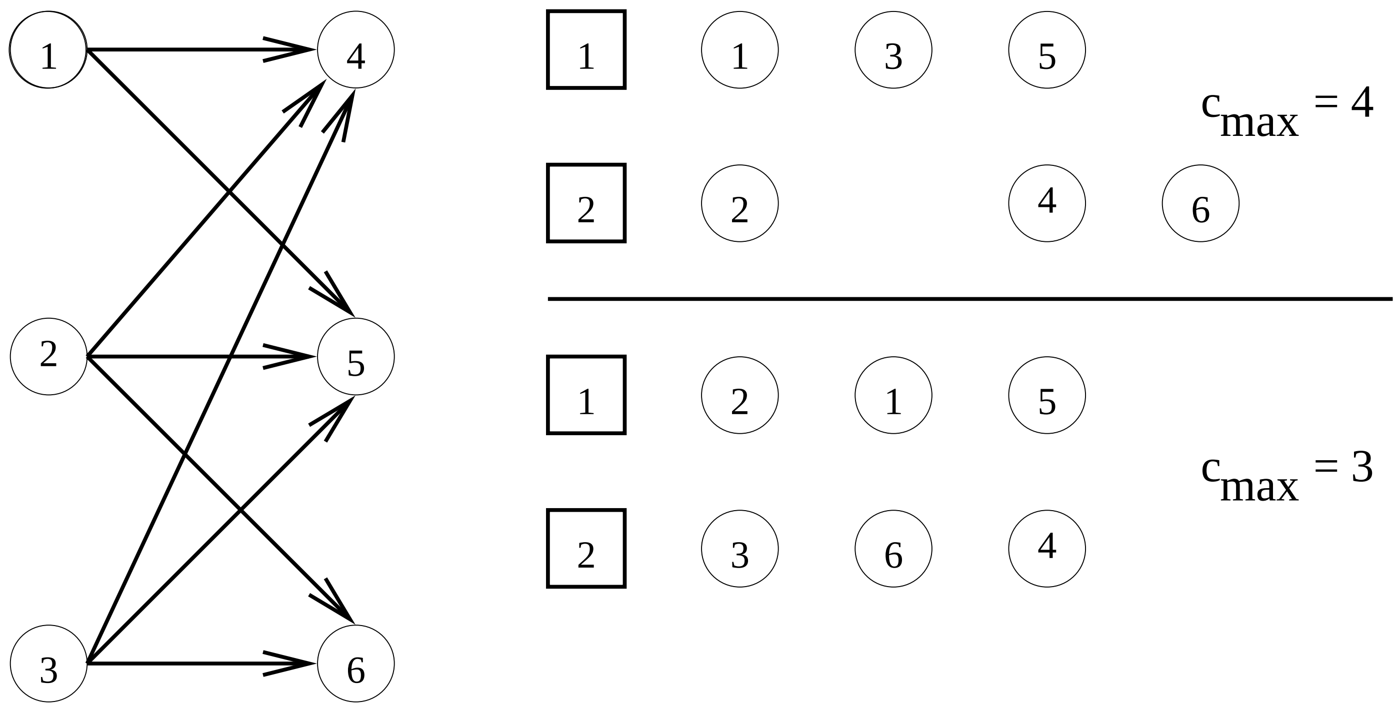
Tree Precedence



- Highest level l_{max}
- $N(l)$ = number of jobs at level l
- $H(l_{max}+1-r) = \sum_{k=1}^{r} N(l_{max}+1-k)$ = Total # of nodes at highest r levels
- Critical Path rule $\equiv$ Highest Level First rule for trees
- Theorem 5.1.5: CP rule optimal for $Pm|p_j=1, intree|c_{max}$ and $Pm|p_j=1, outtree|c_{max}$
- Arbitrary precedence constraints: $\frac{c_{max}(CP)}{c_{max}(OPT)} \leq \frac{4}{3}$ for 2 machines with Critical Path rule

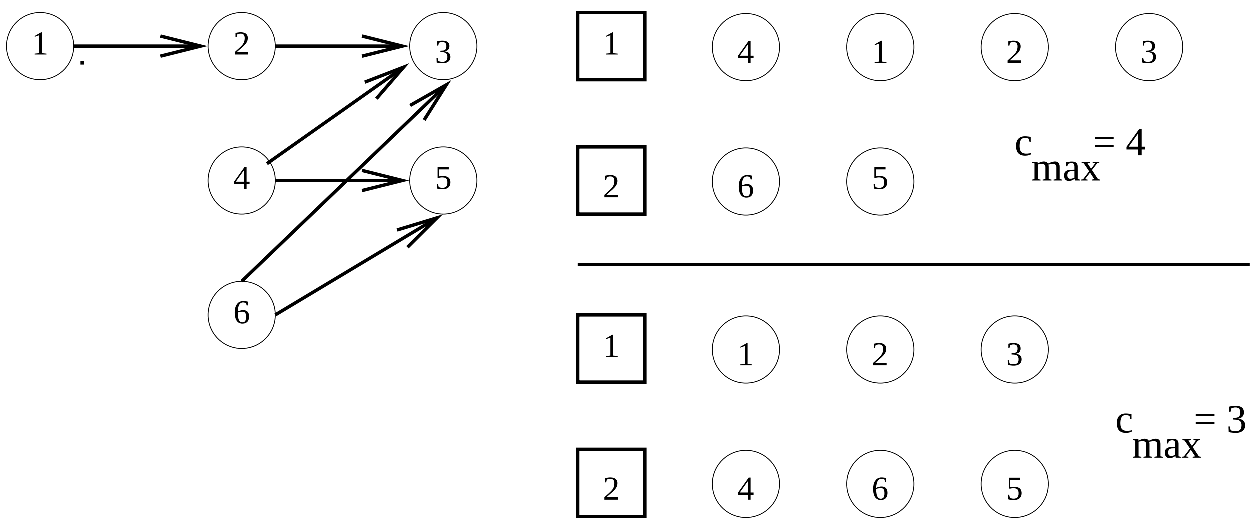
Worst Case Example of CP

6 jobs, 2 machines, unit processing times



Example: Application of LNS Rule

- LNS: Largest Number of Successors First
- Optimal for in and outtree
- 6 jobs, 2 machines, unit processing times
- Sub-optimal for arbitrary precedence constraints



$Pm|M_j|C_{max}$

- $Pm|p_j = 1, M_j|C_{max}$
- M_j are nested: 1 of 4 conditions is valid for jobs j and k
  1. $M_j = M_k$
  2. $M_j \subset M_k$
  3. $M_k \subset M_j$
  4. $M_j \cap M_k = \emptyset$
- Least Flexible Job First (LFJ) rule
- Machine is free $\rightarrow$ Pick job that can be scheduled on least number of machines
- Drawback: Pick which machine when several machines available at the same time?
- LFJ optimal for $Pm|p_j = 1, M_j|C_{max}$ if M_j are nested

Proof of Optimality of LFJ for Nested M_j's

- Proof by contradiction
  - j is the first job that violates LFJ rule
  - $j*$ could be placed at the position of j
  - by use of LFJ rules
    - $M_j \cap M_{j*} = \emptyset$ and $|M_{j*}| < |M_j|$ (Note $M_{j*} \subset M_j$)
  - Exchange of j and $j*$ still results in an optimal schedule
- LFJ optimal for $P2|p_j = 1, M_j|C_{max}$ (M_j's are always nested)

Example of LFJ

- $P4|p_j = 1, M_j|C_{max}$
- 8 jobs $\Rightarrow$ 8 M_j sets:
  1. $M_1 = \{1, 2\}$
  2. $M_2 = M_3 = \{1, 3, 4\}$
  3. $M_4 = \{2\}$
  4. $M_5 = M_6 = M_7 = M_8 = \{3, 4\}$

MAKESPAN WITH PREEMPTIONS

$Pm|prmp|C_{max}$

- Linear Programming formulation
- x_{ij} = total time job j spends on machine i

minimize C_{max}

subject to

$$\sum_{i=1}^{m} x_{ij} = p_j, \quad \forall j = 1, \ldots, n \text{ [processing time]}$$

$$\sum_{i=1}^{m} x_{ij} \leq C_{max}, \forall j = 1, \ldots, n \text{ [processing less than } C_{max}\text{]}$$

$$\sum_{j=1}^{n} x_{ij} \leq C_{max}, \forall i = 1, \ldots, m \text{ [makespan on each m/c]}$$

$$x_{ij} \geq 0 \quad \forall i = 1, \ldots, m, \quad \forall j = 1, \ldots, n \text{ [non-negativity]}$$

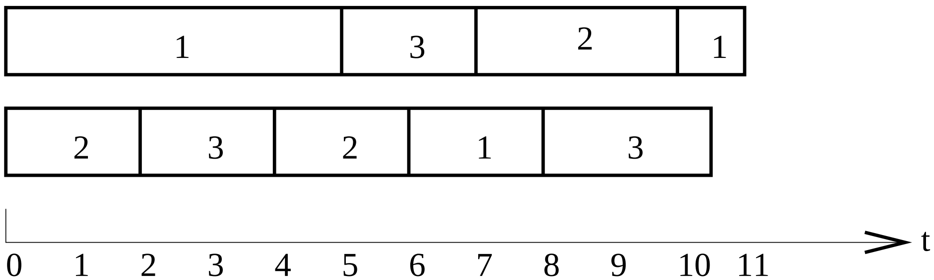
$Pm|prmp|C_{max}$ - LP Formulation

- C_{max} is a variable
- Solution of LP: optimal values of x_{ij} and $C_{max} \Rightarrow$ generation of a schedule
- Lower Bound

$$C_{max} \geq max\{p_1, \sum_{i=1}^{n} \frac{p_j}{m}\} = C_{max}^*$$

$Pm|prmp|C_{max}$ - LRPT

- Longest Remaining Processing Time first (LRPT)
- LRPT yields optimal schedule for $Pm|prmp|C_{max}$
- 2 machines, 3 jobs, $p_1 = 8$, $p_2 = 7$, $p_3 = 6$



- Notations:

  1. $p_j(t)$ = remaining processing time of job j at time t
  2. $\bar{p}(t) = (p_1(t), p_2(t), \ldots, p_n(t))$ = vector of remaining processing times at time t

LRPT - Majorization of Vectors

- $\bar{p}(t)$ majorizes $\bar{q}(t)$ if $\sum_{j=1}^{k} p_{(j)}(t) \geq \sum_{j=1}^{k} q_{(j)}(t) \qquad \forall k = 1, \ldots, n$
- $p_{(j)}(t) = j^{th}$ largest element of $\bar{p}(t)$
- Example:
  1. $\bar{p}(t) = (4, 8, 2, 4)$ and $\bar{q}(t) = (3, 0, 6, 6)$
  2. Arrange elements of each vector in descending order
  3. Verify $\bar{p}(t)$ majorizes $\bar{q}(t)$
- Result: If $\bar{p}(t)$ majorizes $\bar{q}(t)$, then LRPT applied to $\bar{p}(t)$ results in a larger or equal makespan than obtained by applying LRPT to $\bar{q}(t)$

TOTAL COMPLETION TIME WITHOUT PREEMPTIONS

$P_m || \sum C_j$ and SPT Rule

- Recall $p_1 \geq p_2 \geq \ldots\ldots \geq p_n$
- $p_{(j)}$ = processing time of job in position j on a single machine
- $\sum C_j = np_{(1)} + (n-1)p_{(2)} + \ldots\ldots + 2p_{(n-1)} + p_{(n)}$
- $p_{(1)} \leq p_{(2)} \cdots \leq p_{(n-1)} \leq p_{(n)}$ for optimal schedule
- SPT rule optimal for $P_m || \sum C_j$
- Proof:
  - $\frac{n}{m}$ is integer (otherwise add job with processing time 0) and mn coefficients:

n coefficients: m in number
$n-1$ coefficients: m in number
....
2 coefficients: m in number
1 coefficients: m in number

WSPT Rule - An Example

- WSPT minimizes $\Sigma w_j C_j$ for single machine
- Result does not extend for parallel machines
- $Pm || \Sigma w_j C_j \Rightarrow$ NP hard

jobs	1	2	3
p_j	1	1	3
w_j	1	1	3

- 2 machines
- Any schedule WSPT
  - Job 1 and 2 on M1 and M2 at t=0, Job 3 on M1 at t=1: $\Sigma w_j C_j = 14$
  - Job 3 on M1 at t=0, Job 1 and 2 on M2 at t =0 and t=1: $\Sigma w_j C_j = 12$
- $w_1 = w_2 = 1 - \epsilon \Rightarrow$ WSPT not necessarily optimal
- $\frac{\Sigma w_j C_j(WSPT)}{\Sigma w_j C_j(OPT)} < \frac{1}{2}(1 + \sqrt{2})$ (tight bound)

Precedence Constraints

- $Pm|prec|\Sigma C_j$: strongly NP-hard
- Result 1: Critical Path rule optimal for $Pm|p_j = 1, outtree|\Sigma C_j$
- Result 2: LFJ optimal for $Pm|p_j = 1, M_j|\Sigma C_j$ when M_j sets are nested
- $Pm|p_j = 1, M_j|\Sigma C_j$ special case of $Rm||\Sigma C_j$
- $Rm||\Sigma C_j$ can be formulated as an Integer Program

$Rm||\Sigma C_j$ Formulation

$$x_{ikj} = \begin{cases} 1 \text{ if job } j \text{ scheduled as } k^{th} \text{ to last job on m/c } i \\ 0 \text{ otherwise} \end{cases}$$

minimize $\sum_{i=1}^{m} \sum_{k=1}^{n} \sum_{j=1}^{n} k p_{ij} x_{ikj}$

subject to

$\sum_{i=1}^{m} \sum_{k=1}^{n} x_{ikj} = 1 \ \forall j = 1, \ldots, n$ [Each job scheduled exactly once]

$\sum_{j=1}^{n} x_{ikj} \leq 1 \ \forall i = 1, \ldots, m, \ \forall k = 1, \ldots, n$ [Each position is not taken more than once]

$x_{ikj} = \{0, 1\} \ \forall i = 1, \ldots, m, \ \forall j = 1, \ldots, n, \ \forall k = 1, \ldots, n$

- Weighted bipartite matching problem: n jobs $\Rightarrow$ mn positions
- Relax integrality constraints on x_{ikj}
- LP solvable in polynomial time

TOTAL COMPLETION TIME WITH PREEMPTIONS

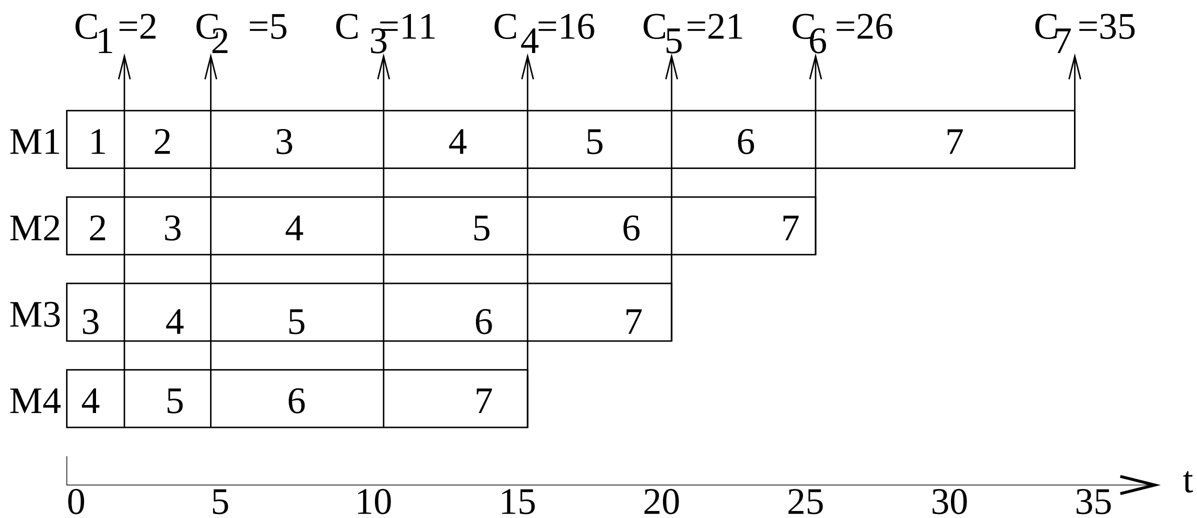
$Pm|prmp|\sum C_j$

- $Pm|prmp|\sum C_j$ special case of $Qm|prmp|\sum C_j$
- Result: There exists an optimal schedule with $C_j \leq C_k$, if $p_j \leq p_k$ $\forall j, k$
- SRPT-FM rule optimal for $Qm|prmp|\sum C_j$
- Shortest Remaining Processing Time on Fastest Machine
- $v_1 \geq v_2 \geq \ldots \geq v_n$
- $C_n \leq C_{n-1} \leq \ldots \leq C_1$
- There are n machines
  - more jobs than machines $\Rightarrow$ add machines with speed 0
  - more machines than jobs $\Rightarrow$ slowest machines are not used

Application with SRPT-FM - Example

M/C	1	2	3	4
v_j	4	2	2	1

Jobs	1	2	3	4	5	6	7
p_j	8	16	34	40	45	46	61



$$\Sigma C_j = 116$$

SRPT-FM is Optimal for $Qm|prmp|\Sigma C_j$ - Proof

$$\begin{aligned}
v_1 C_n &= p_n \\
v_2 C_n + v_1(C_{n-1} - C_n) &= p_{n-1} \\
v_3 C_n + v_2(C_{n-1} - C_n) + v_1(C_{n-2} - C_{n-1}) &= p_{n-2} \\
\ldots & \ldots \ldots \\
v_n C_n + v_{n-1}(C_{n-1} - C_n) + \ldots + v_1(C_1 - C_2) &= p_1
\end{aligned}$$

Hence

$$\begin{aligned}
v_1 C_n &= p_n \\
v_2 C_n + v_1 C_{n-1} &= p_n + p_{n-1} \\
v_3 C_n + v_2 C_{n-1} + v_1 C_{n-2} &= p_n + p_{n-1} + p_{n-2} \\
\ldots & \ldots \ldots \\
v_n C_n + v_{n-1} C_{n-1} + \ldots + v_1 C_1 &= p_n + p_{n-1} + \ldots + p_1
\end{aligned}$$

SRPT-FM is Optimal for $Qm|prmp|\sum C_j$ - Proof - Contd...

- S' is optimal $\Rightarrow C'_n \leq C'_{n-1} \leq \ldots \leq C'_1$
- $c'_n \geq p_n/v_1 \Rightarrow v_1 C'_n \geq p_n$
- Processing done on jobs n and $n-1 \leq (v_1+v_2)C'_n + v_1(C'_{n-1} - C'_n)$
- $\Rightarrow v_2 C'_n + v_1 C'_{n-1} \geq p_n + p_{n-1}$
- Similarly $v_k C'_n + v_{k-1} C'_{n-1} + \ldots v_1 C'_{n-k+1} \leq p_n + p_{n-1} + \ldots + p_{n-k+1}$

$$
\begin{aligned}
v_1 C'_n &= v_1 C_n \\
v_2 C'_n + v_1 C'_{n-1} &= v_2 C_n + v_1 C_{n-1} \\
\ldots &\quad \ldots \quad \ldots \\
v_n C'_n + v_{n-1} C'_{n-1} + \ldots + v_1 C'_1 &= v_n C_n + v_{n-1} C_{n-1} + \ldots + v_1 C_1
\end{aligned}
$$

SRPT-FM is Optimal for $Qm|prmp|\Sigma C_j$ - Proof - Contd...

- Multiply inequality i by $\alpha_i \geq 0$ and obtain $\Sigma C'_j \geq \Sigma C_j$
- Proof is complete if these α_i exist
- α_i must satisfy

$$\begin{aligned} v_1\alpha_1 + v_2\alpha_2 + \ldots + v_n\alpha_n &= 1 \\ v_1\alpha_2 + v_2\alpha_3 + \ldots + v_{n-1}\alpha_n &= 1 \\ \ldots \quad \ldots & \quad \ldots \\ v_1\alpha_n &= 1 \end{aligned}$$

- These α_i exist as $v_1 \geq v_2 \geq \ldots v_n$

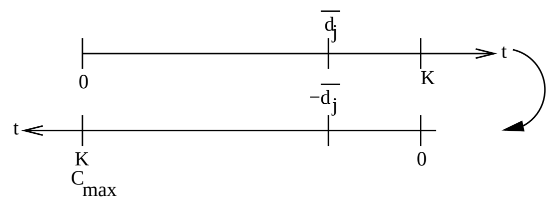
DUE-DATE RELATED OBJECTIVES

$Pm||L_{max}$

- $Pm||L_{max}$ with all due dates =0 $\equiv Pm||C_{max} \Rightarrow$ NP-hard
- $Qm|prmp|L_{max}$
- Assume $L_{max} = z$

$$C_j \leq d_j + z \Rightarrow \text{set } \overline{d_j} = d_j + z \text{ (hard deadline)}$$

- Finding a schedule for this problem equivalent to solving $Qm|r_j, prmp|C_{max}$
  - Reverse direction of time



  - Release each job j at $K - \overline{d_j}$ (for a sufficiently big K)
  - Solve problem with LRPT-FM for $L_{max} \leq z$ and perform search over z

Minimizing L_{max} with Preemptions

Jobs	1	2	3	4
d_j	4	5	8	9
p_j	3	3	3	8

- $P2|prmp|l_{max}$
- Is there a feasible schedule with $L_{max} = 0$? $(\overline{d_j} = d_j)$

Jobs	1	2	3	4
r_j	5	4	1	0
p_j	3	3	3	8

- Is there a feasible schedule with $C_{max} < 9$? YES, apply LRPT