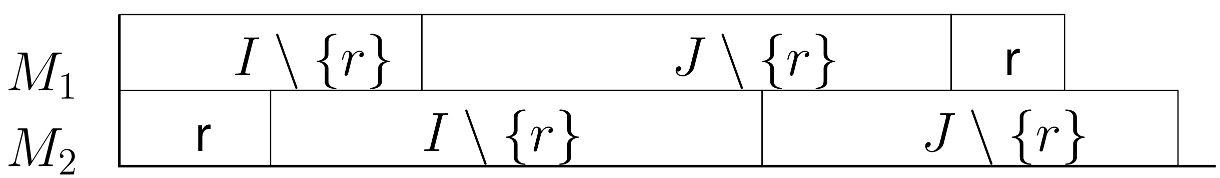
Open Shop models

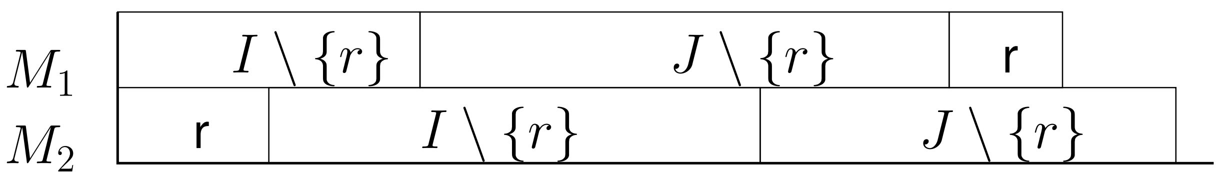
Algorithm Problem $O2||C_{max}$

1. $I =$ set of jobs with $p_{1j} \leq p_{2j}$; $J =$ set of remaining jobs;
2. LET $p_{kr} = \max\{\max_{j \in I} p_{1j}, \max_{j \in J} p_{2j}\}$;
    - order on M_1: $(I \setminus \{r\}, J \setminus \{r\}, r)$; order on M_2: $(r, I \setminus \{r\}, J \setminus \{r\})$
    - r first on M_2, then on M_1; all other jobs vice versa



Algorithm Problem $O2||C_{max}$

1. $I =$ set of jobs with $p_{1j} \leq p_{2j}$; $J =$ set of remaining jobs;
2. LET $p_{kr} = \max\{\max_{j \in I} p_{1j}, \max_{j \in J} p_{2j}\}$;
   - order on M_1: $(I \setminus \{r\}, J \setminus \{r\}, r)$; order on M_2: $(r, I \setminus \{r\}, J \setminus \{r\})$
   - r first on M_2, then on M_1; all other jobs vice versa



3. OBSERVE that $p_{3-k,r} \geq p_{kr} \geq \max\{\max_{j \in I} p_{1j}, \max_{j \in J} p_{2j}\}$;

Remarks Algorithm Problem $O2||C_{max}$

- complexity: $O(n)$
- algorithm solves problem $O2||C_{max}$ optimally
- Proof builds on fact that C_{max} is either
  - $\sum_{j=1}^{n} p_{1j}$ or
  - $\sum_{j=1}^{n} p_{2j}$ or
  - $p_{1r} + p_{2r}$

Remarks Algorithm Problem $O2||C_{max}$

- complexity: $O(n)$
- algorithm solves problem $O2||C_{max}$ optimally
- Proof builds on fact that C_{max} is either
  - $\sum_{j=1}^{n} p_{1j}$ or
  - $\sum_{j=1}^{n} p_{2j}$ or
  - $p_{1r} + p_{2r}$

Problem $O3||C_{max}$

- Problem $O3||C_{max}$ is NP-hard
  Proof as Exercise (Reduction using PARTITION)

Open Shop models

Problem $O|pmtn|C_{max}$

- define $ML_i := \sum_{j=1}^{n} p_{ij}$ (load of machine i)
- define $JL_j := \sum_{i=1}^{m} p_{ij}$ (load of job j)
- $LB := \max\{\max_{i=1}^{m} ML_i, \max_{j=1}^{n} JL_j\}$ is a lower bound on C_{max}

Problem $O|pmtn|C_{max}$

- define $ML_i := \sum_{j=1}^{n} p_{ij}$ (load of machine i)
- define $JL_j := \sum_{i=1}^{m} p_{ij}$ (load of job j)
- $LB := \max\{\max_{i=1}^{m} ML_i, \max_{j=1}^{n} JL_j\}$ is a lower bound on C_{max}
- Theorem: For problem $O|pmtn|C_{max}$ a schedule with $C_{max} = LB$ exists.
- Proof of the theorem is constructive and leads to a polynomial algorithm for problem $O|pmtn|C_{max}$

Notations for Algorithm $O|pmtn|C_{max}$

- job j (machine i) is called tight if $JL_j = LB$ ($ML_i = LB$)
- job j (machine i) has slack if $JL_j < LB$ ($ML_i < LB$)
- a set D of operations is called a *decrementing set* if it contains for each tight job and machine exactly one operation and for each job and machine with slack at most one operation
- Theorem: A decrementing set always exists and can be calculated in polynomial time
  (Proof based on maximal cardinality matchings; see e.g. P. Brucker: Scheduling Algorithms)

Open Shop models

Algorithm $O|pmtn|C_{max}$

REPEAT

1. Calculate a decrementing set D;
2. Calculate maximum value Δ with
   - $\Delta \leq \min_{(i,j) \in D} p_{ij}$
   - $\Delta \leq LB - ML_i$ if machine i has slack and no operation in D
   - $\Delta \leq LB - JL_j$ if job j has slack and no operation in D;
3. schedule the operations in D for Δ time units in parallel;
4. Update values $p,\ LB,\ JL$, and ML

UNTIL all operations have been completely scheduled.

Open Shop models

Correctness Algorithm $O|pmtn|C_{max}$

- after an iteration we have: $LB_{new} = LB_{old} - \Delta$
- in each iteration a time slice of Δ time units is scheduled
- the algorithm terminates after at most $nm + n + m$ iterations since in each iteration either
  - an operation gets completely scheduled or
  - one additional machine or job gets tight

Open Shop models

Example Algorithm $O|pmtn|C_{max}$

p_{ij}	j	ML
	2 4 3 2	11
i	3 1 2 3	9
	2 3 3 2	10
JL	7 8 8 7	$LB = 11$

Open Shop models

Example Algorithm $O|pmtn|C_{max}$

$\Delta = 3$	j	ML
	2 4 3 2	11
i	3 1 2 3	9
	2 3 3 2	10
JL	7 8 8 7	$LB = 11$

Open Shop models

Example Algorithm $O|pmtn|C_{max}$

$\Delta = 3$	j	ML
	2 4 3 2	11
i	3 1 2 3	9
	2 3 3 2	10
JL	7 8 8 7	$LB = 11$

Open Shop models

Example Algorithm $O|pmtn|C_{max}$

$\Delta = 3$	j				ML
	2	4	3	2	11
i	3	1	2	3	9
	2	3	3	2	10
JL	7	8	8	7	$LB = 11$

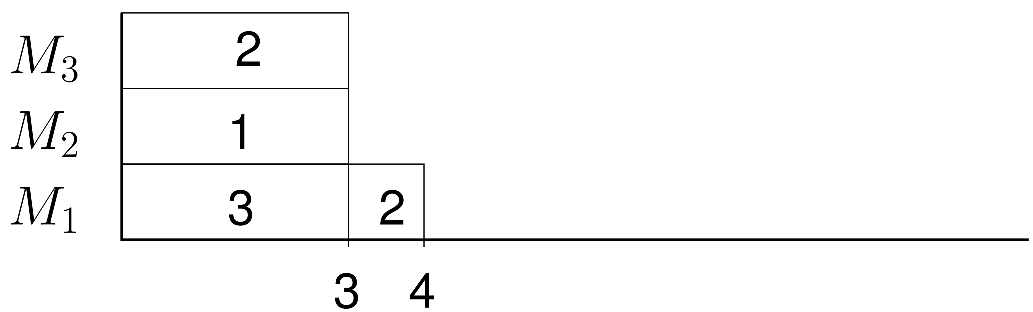
$\hat{p}_{ij}$	j				ML
	2	4	0	2	8
i	0	1	2	3	6
	2	0	3	2	7
JL	4	5	5	7	$LB = 8$

Open Shop models

Example Algorithm $O|pmtn|C_{max}$

$\Delta = 3$	j	ML
i	2 4 3 2	11
	3 1 2 3	9
	2 3 3 2	10
JL	7 8 8 7	$LB = 11$

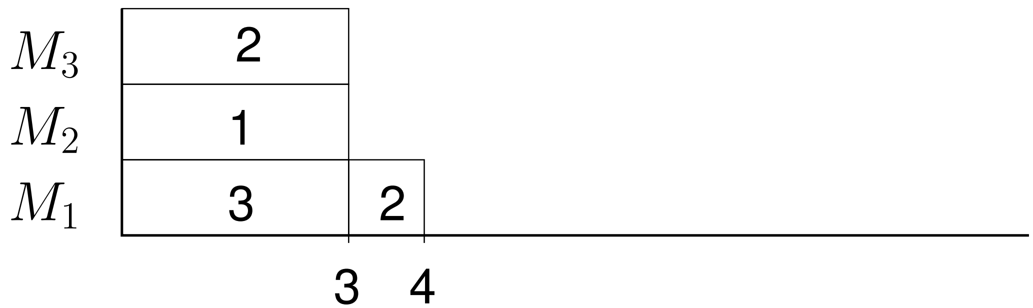
$\Delta = 1$	j	ML
i	2 4 0 2	8
	0 1 2 3	6
	2 0 3 2	7
JL	4 5 5 7	$LB = 8$



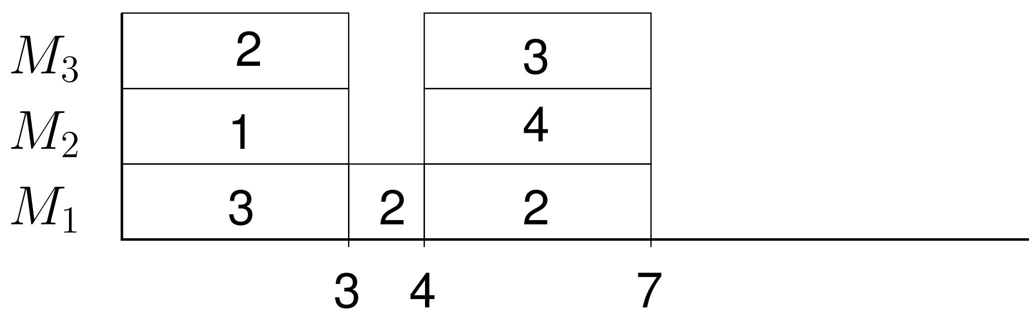
Open Shop models

Example Algorithm $O|pmtn|C_{max}$

$\Delta = 1$	j				ML
	2	4	0	2	8
i	0	1	2	3	6
	2	0	3	2	7
JL	4	5	5	7	$LB = 8$



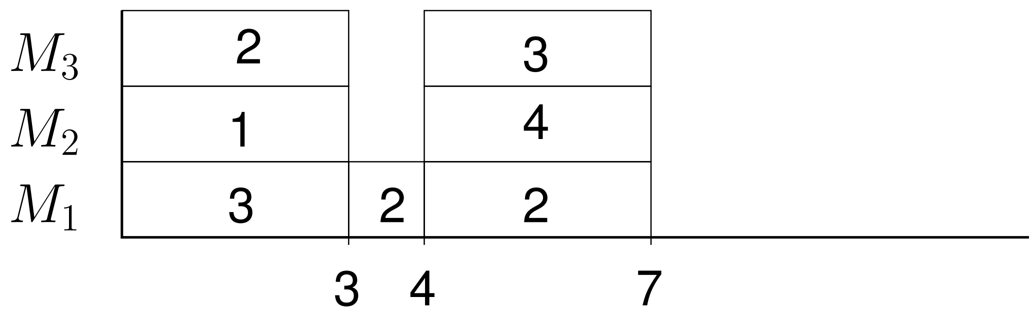
$\Delta = 3$	j				ML
	2	3	0	2	7
i	0	1	2	3	6
	2	0	3	2	7
JL	4	4	5	7	$LB = 7$



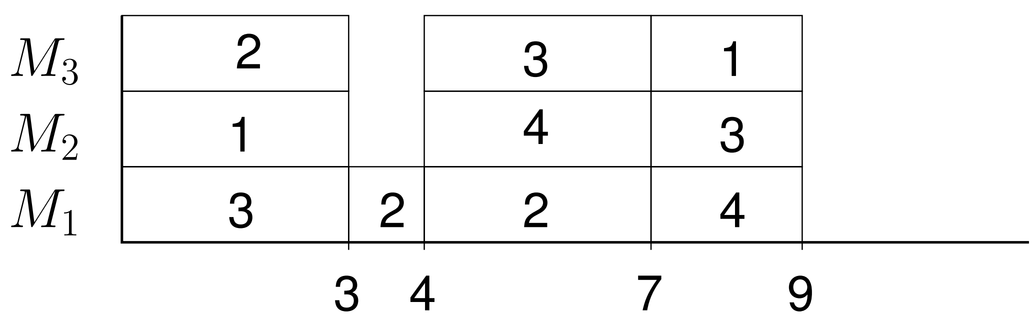
Open Shop models

Example Algorithm $O|pmtn|C_{max}$

$\Delta = 3$	j				ML
	2	3	0	2	7
i	0	1	2	3	6
	2	0	3	2	7
JL	4	4	5	7	$LB = 7$



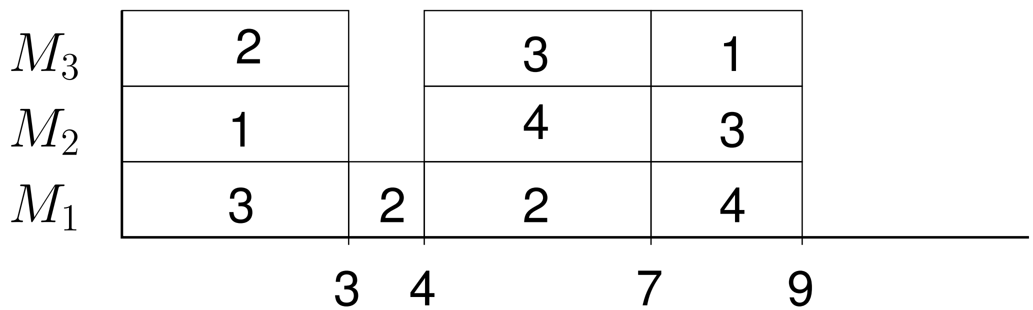
$\Delta = 2$	j				ML
	2	0	0	2	4
i	0	1	2	0	3
	2	0	0	2	4
JL	4	1	2	4	$LB = 4$



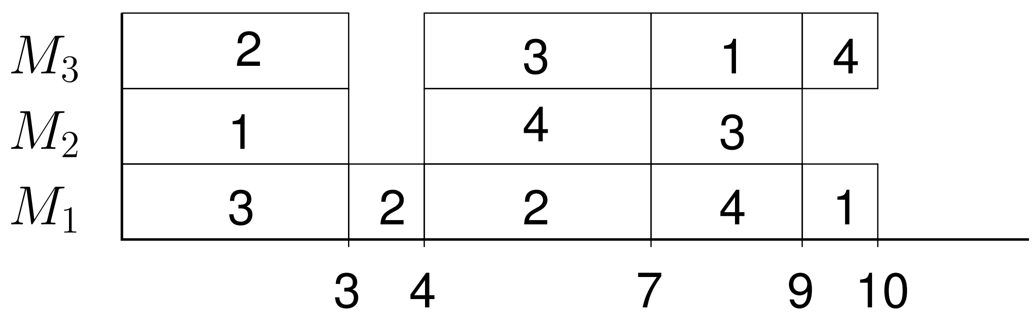
Open Shop models

Example Algorithm $O|pmtn|C_{max}$

$\Delta = 2$	j	ML
	2 0 0 2	4
i	0 1 2 0	3
	2 0 0 2	4
JL	4 1 2 4	$LB = 4$



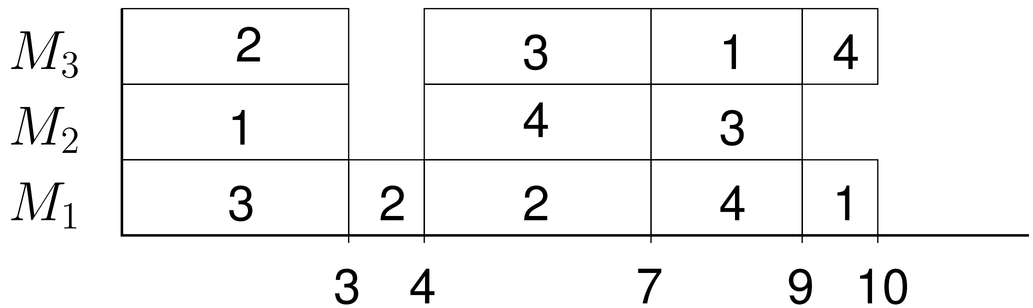
$\Delta = 1$	j	ML
	2 0 0 0	2
i	0 1 0 0	1
	0 0 0 2	2
JL	2 1 0 2	$LB = 2$



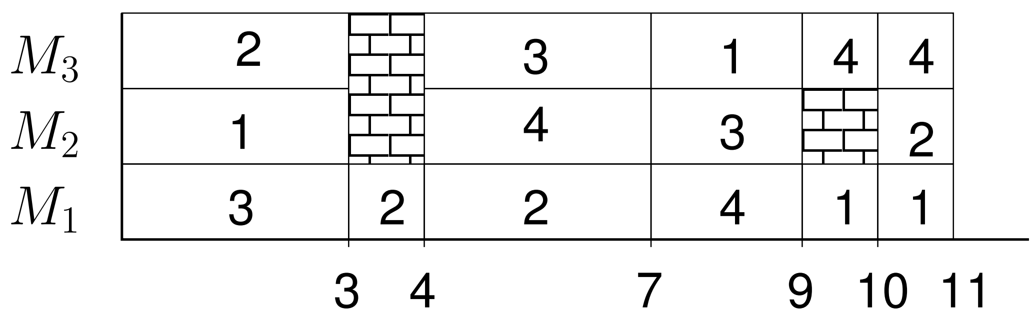
Open Shop models

Example Algorithm $O|pmtn|C_{max}$

$\Delta = 1$	j				ML
i	2	0	0	0	2
	0	1	0	0	1
	0	0	0	2	2
JL	2	1	0	2	$LB = 2$



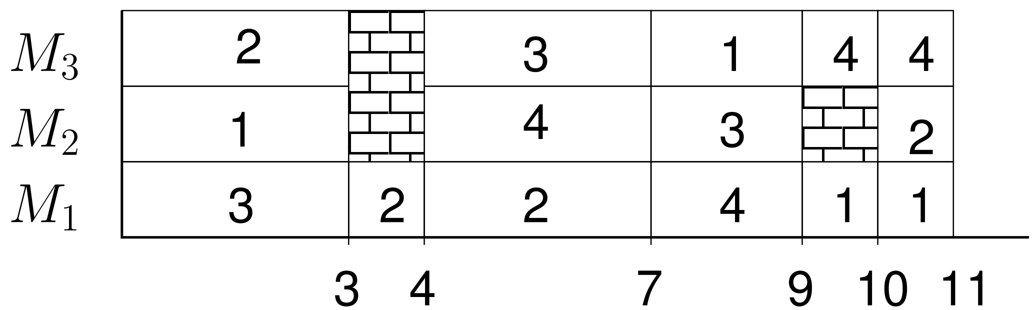
$\Delta = 1$	j				ML
i	1	0	0	0	1
	0	1	0	0	1
	0	0	0	1	1
JL	1	1	0	1	$LB = 1$



Open Shop models

Final Schedule Example Algorithm $O|pmtn|C_{max}$

	j				ML
	2	4	3	2	11
i	3	1	2	3	9
	2	3	3	2	10
JL	7	8	8	7	$LB = 11$



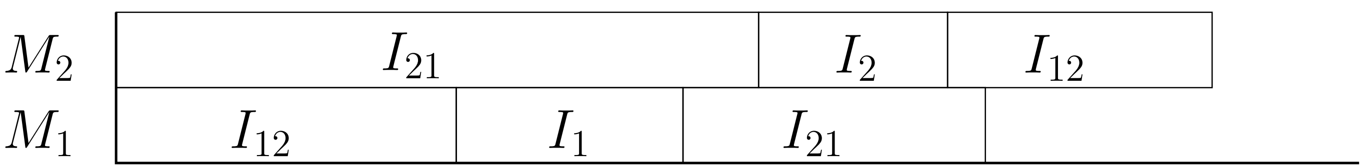
- 6 iterations
- $C_{max} = 11 = LB$
- sequence of time slices may be changed arbitrary

Problem $J2||C_{max}$

- I_1: set of jobs only processed on M_1
- I_2: set of jobs only processed on M_2
- I_{12}: set of jobs processed first on M_1 and then on M_2
- I_{21}: set of jobs processed first on M_2 and then on M_1
- π_{12}: optimal flow shop sequence for jobs from I_{12}
- π_{21}: optimal flow shop sequence for jobs from I_{21}

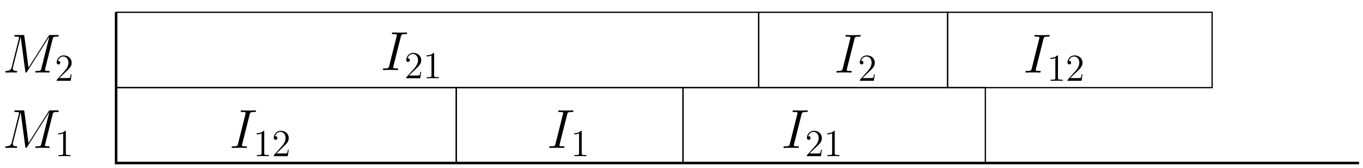
Algorithm Problem $J2||C_{max}$

1. on M_1 first schedule the jobs from I_{12} in order π_{12}, then the jobs from I_1, and last the jobs from I_{21} in order π_{21}
2. on M_2 first schedule the jobs from I_{21} in order π_{21}, then the jobs from I_2, and last the jobs from I_{12} in order π_{12}



Algorithm Problem $J2||C_{max}$

1. on M_1 first schedule the jobs from I_{12} in order π_{12}, then the jobs from I_1, and last the jobs from I_{21} in order π_{21}
2. on M_2 first schedule the jobs from I_{21} in order π_{21}, then the jobs from I_2, and last the jobs from I_{12} in order π_{12}



Theorem: The above algorithm solves problem $J2||C_{max}$ optimally in $O(n \log(n))$ time.
Proof: almost straightforward!

Problem $J||C_{max}$

- as a generalization of $F||C_{max}$, this problem is NP-hard
- it is one of the most treated scheduling problems in literature
- we present
  - a branch and bound approach
  - a heuristic approach called the Shifting Bottleneck Heuristic

for problem $J||C_{max}$ which both depend on the disjunctive graph formulation

Base of Branch and Bound

- The set of all active schedules contains an optimal schedule
- Solution method: Generate all active schedules and take the best
- Improvement: Use the generation scheme in a branch and bound setting
- Consequence: We need a generation scheme to produce all active schedules for a job shop
- $\rightarrow$ Approach: extend partial schedules

<u>Generation of all active schedules</u>

- Notations: (assuming that already a partial schedule S is given)
  - Ω: set of all operations which predecessors have already been scheduled in S
  - r_{ij}:earliest possible starting time of operation $(i,j) \in \Omega$ w.r.t. S
  - Ω': subset of Ω
- Remark: r_{ij} can be calculated via longest path calculations in the disjunctive graph belonging to S

Generation of all active schedules (cont.)

1. (Initial Conditions)
   $\Omega :=$ {first operations of each job}; $r_{ij} := 0$ for all $(i,j) \in \Omega$;
2. (Machine selection)
   Compute for current partial schedule $t(\Omega) := \min_{(i,j)\in\Omega}\{r_{ij} + p_{ij}\}$;
   $i^* :=$ machine on which minimum is achieved;
3. (Branching) $\Omega' := \{(i^*, j) | r_{i^*j} < t(\Omega)\}$
   FOR ALL $(i^*, j) \in \Omega'$ DO
   (a) extend partial schedule by scheduling (i^*, j) next on machine i^*;
   (b) delete (i^*, j) from Ω;
   (c) add job-successor of (i^*, j) to Ω;
   (d) Return to Step 2

Job Shop models

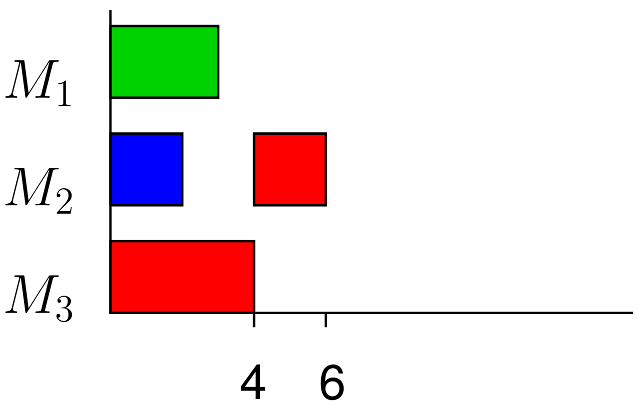
Generation of all active schedules - example

Jobs: 1 $(3,1) \rightarrow (2,1) \rightarrow (1,1)$ $p_{31} = 4, p_{21} = 2, p_{11} = 1$

2 $(1,2) \rightarrow (3,2)$ $p_{12} = 3, p_{32} = 3$

3 $(2,3) \rightarrow (1,3) \rightarrow (3,3)$ $p_{23} = 2, p_{13} = 4, p_{33} = 1$

Partial Schedule:



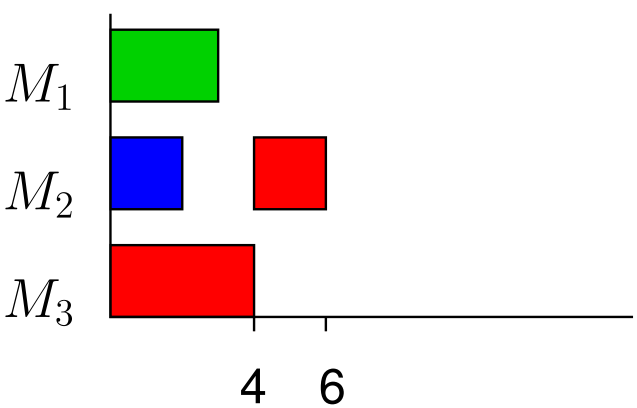
Generation of all active schedules - example

Jobs: 1 $(3,1) \rightarrow (2,1) \rightarrow (1,1)$ $p_{31} = 4, p_{21} = 2, p_{11} = 1$

2 $(1,2) \rightarrow (3,2)$ $p_{12} = 3, p_{32} = 3$

3 $(2,3) \rightarrow (1,3) \rightarrow (3,3)$ $p_{23} = 2, p_{13} = 4, p_{33} = 1$

Partial Schedule:



Ω=$\{(1,1),(3,2),(1,3)\}$;

$r_{11} = 6, r_{32} = 4, r_{13} = 3$;

$t(\Omega) = \min\{6+1, 4+3, 3+4\} = 7$;

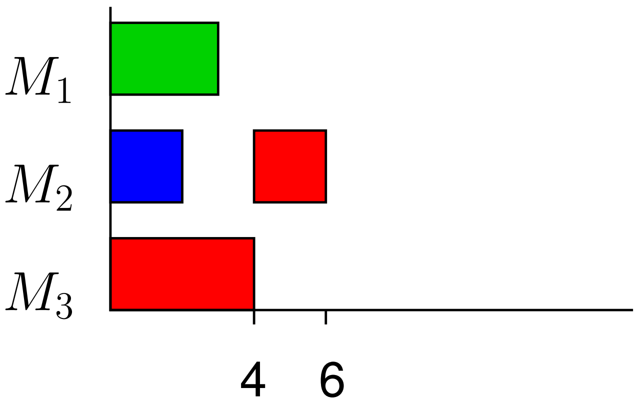
$i^* = M_1$;

$\Omega' = \{(1,1),(1,3)\}$

Job Shop models

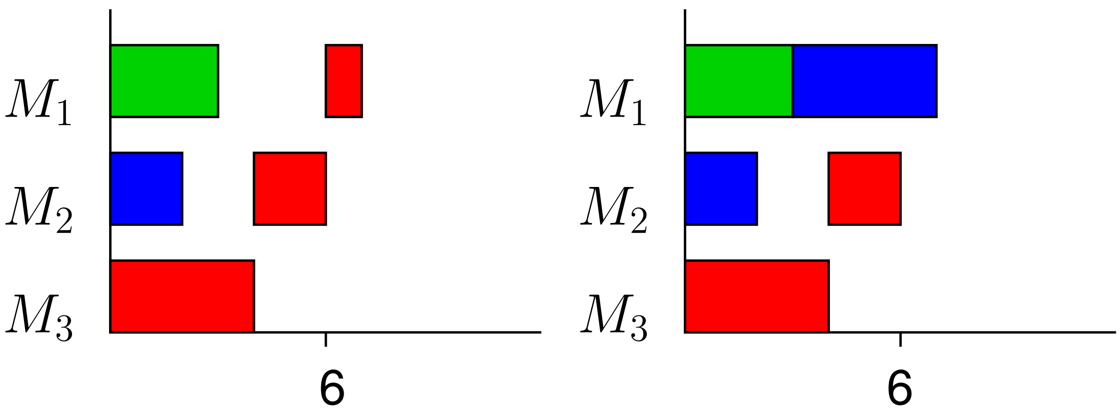
Generation of all active schedules - example (cont.)

Partial Schedule:



Ω=$\{(1,1),(3,2),(1,3)\}$;
$r_{11} = 6, r_{32} = 4, r_{13} = 3$;
$t(\Omega) = \min\{6+1, 4+3, 3+4\} = 7$;
$i^* = M_1$;
$\Omega' = \{(1,1),(1,3)\}$

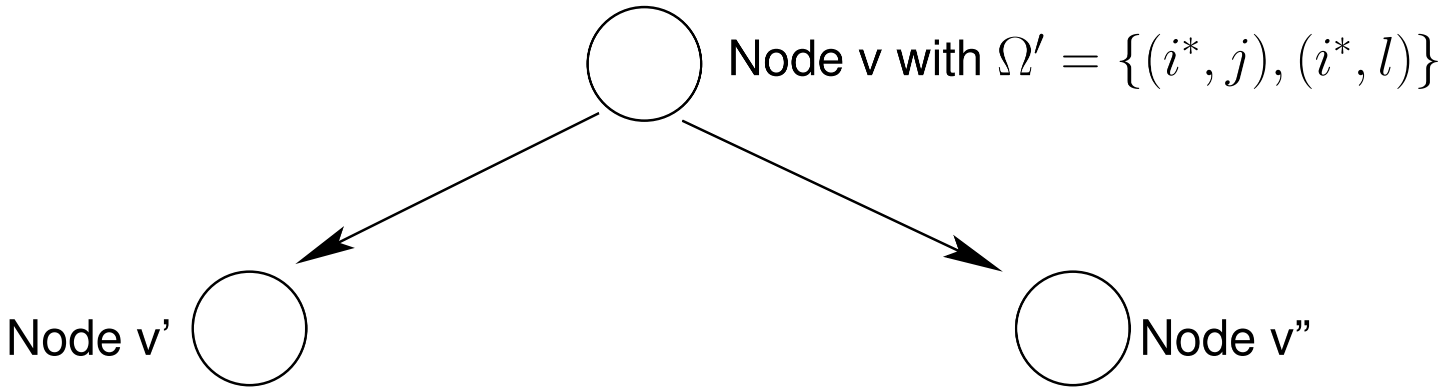
Extended partial schedules:



Remarks on the generation:

- the given algorithm is the base of the branching
- nodes of the branching tree correspond to partial schedules
- Step 3 branches from the node corresponding to the current partial schedule
- the number of branches is given by the cardinality of Ω'
- a branch corresponds to the choice of an operation (i^*, j) to be schedules next on machine i^*
  $\longrightarrow$ a branch fixes new disjunctions

Disjunctions fixed by a branching



selection (i^*, j)	selection (i^*, l)
Add disjunctions $(i^*, j) \rightarrow (i^*k)$ for all unscheduled operations (i^*, k)	Add disjunctions $(i^*, l) \rightarrow (i^*k)$ for all unscheduled operations (i^*, k)

Consequence: Each node in the branch and bound tree is characterized by a set S' of fixed disjunctions

Lower bounds for nodes of the branch and bound tree

- Consider node V with fixed disjunctions S':
- Simple lower bound:
  - calculate critical path in $G(S')$
  - $\longrightarrow$ Lower bound $LB(V)$

Lower bounds for nodes of the branch and bound tree

- Consider node V with fixed disjunctions S':
- Simple lower bound:
  - calculate critical path in $G(S')$
  - $\longrightarrow$ Lower bound $LB(V)$
- Better lower bound:
  - consider machine i
  - allow parallel processing on all machines $\neq i$
  - solve problem on machine i

1-machine problem resulting for better LB

1. calculate earliest starting times r_{ij} of all operations (i, j) on machine i (longest paths from source in $G(S')$)
2. calculate minimum amount q_{ij} of time between end of (i, j) and end of schedule (longest path to sink in $G(S')$)
3. solve single machine problem on machine i:
   - respect release dates
   - no preemption
   - minimize maximum value of $C_{ij} + q_{ij}$

Result: head-body-tail problem (see Lecture 3)

Better lower bound

- solve 1-machine problem for all machines
- this results in values $f_1, \ldots, f_m$
- $LB^{new}(V) = \max_{i=1}^{m} f_i$

<u>Better lower bound</u>

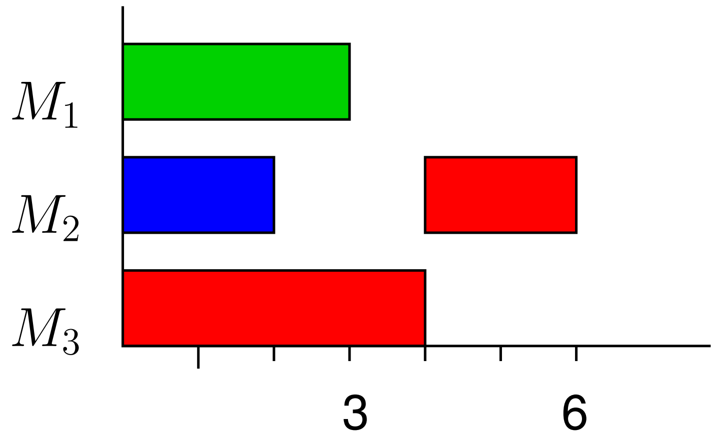
- solve 1-machine problem for all machines
- this results in values $f_1, \ldots, f_m$
- $LB^{new}(V) = \max_{i=1}^{m} f_i$

<u>Remarks</u>:

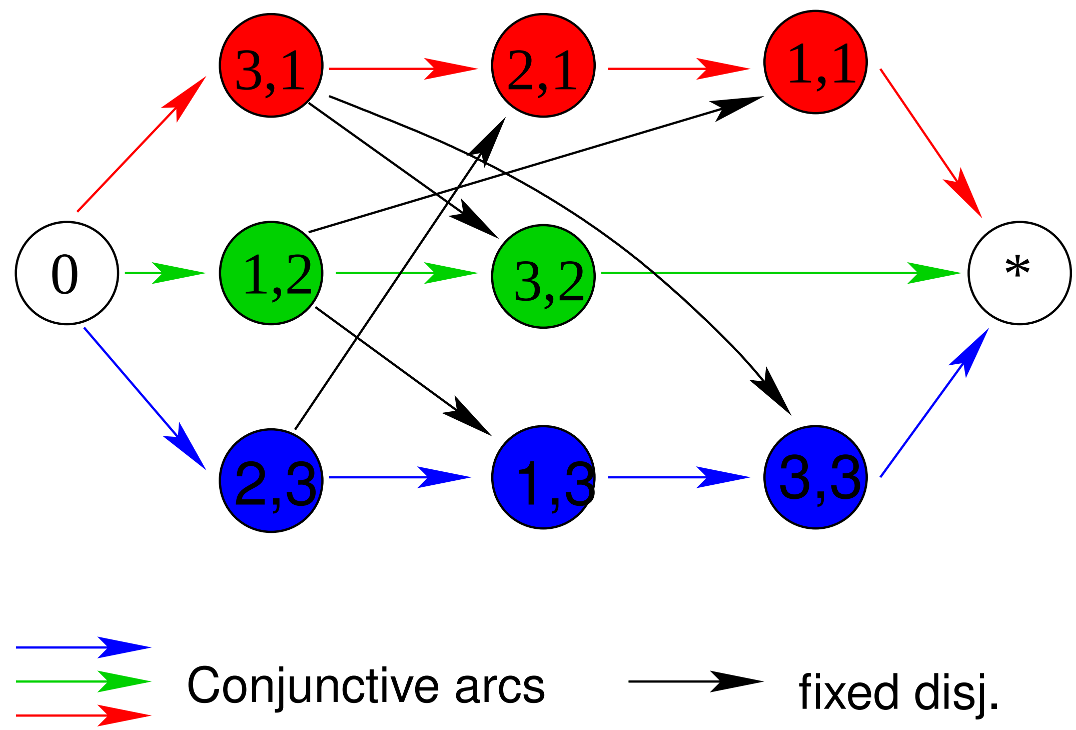
- 1-machine problem is NP-hard
- computational experiments have shown that it pays off to solve these m NP-hard problems per node of the search tree
- 20×20 job-shop instances are already hard to solve by branch and bound

Better lower bound - example

Partial Schedule:

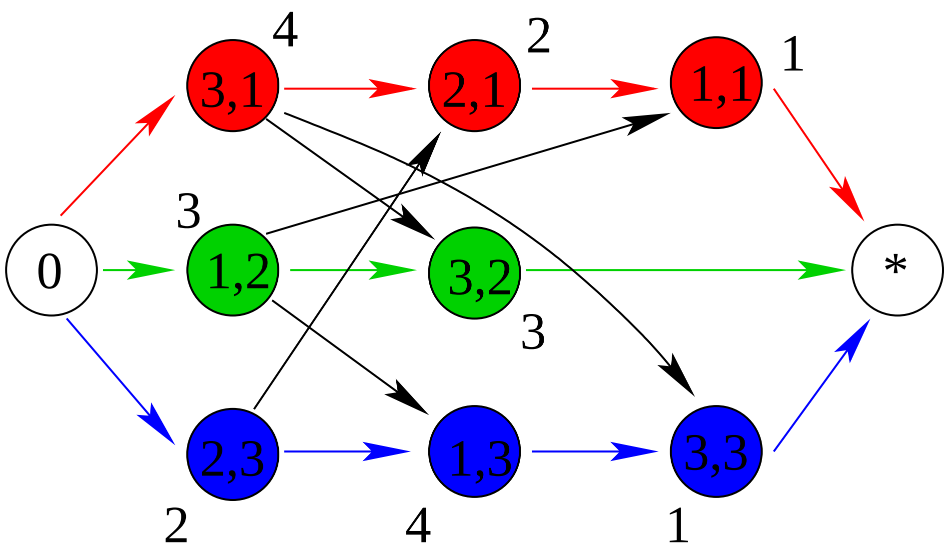


Corresponding graph $G(S')$:



Better lower bound - example (cont.)

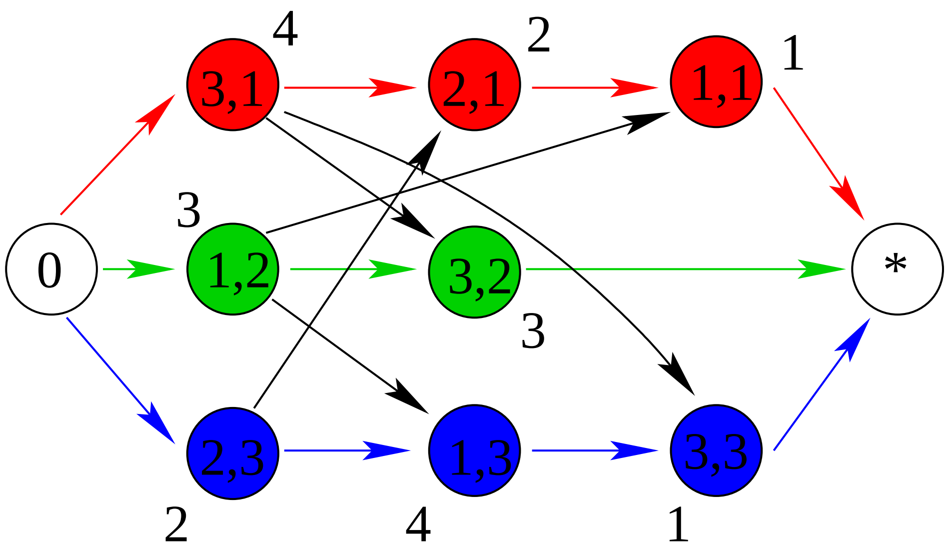
Example 1: Graph $G(S')$ with processing times:



$LB(V)$=$l(0, (1, 2), (1, 3), (3, 3), *)$=$8$

<u>Better lower bound - example (cont.)</u>

Example 1: Graph $G(S')$ with processing times:



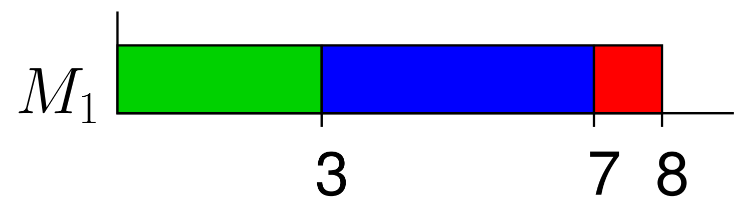
$LB(V)$=$l(0,(1,2),(1,3),(3,3),*)$=8

Data for jobs on Machine 1:

green	blue	red
$r_{12} = 0$	$r_{13} = 3$	$r_{11} = 6$
$q_{12} = 5$	$q_{13} = 1$	$q_{11} = 0$

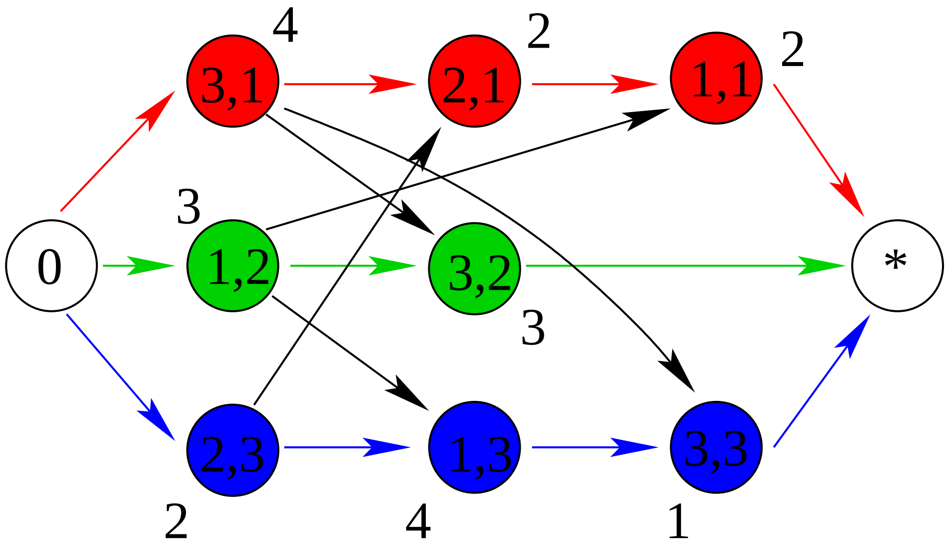
Opt. solution:

$Opt = 8$, $LB^{new}(V) = 8 = LB^{old}(V)$



Better lower bound - example (cont.)

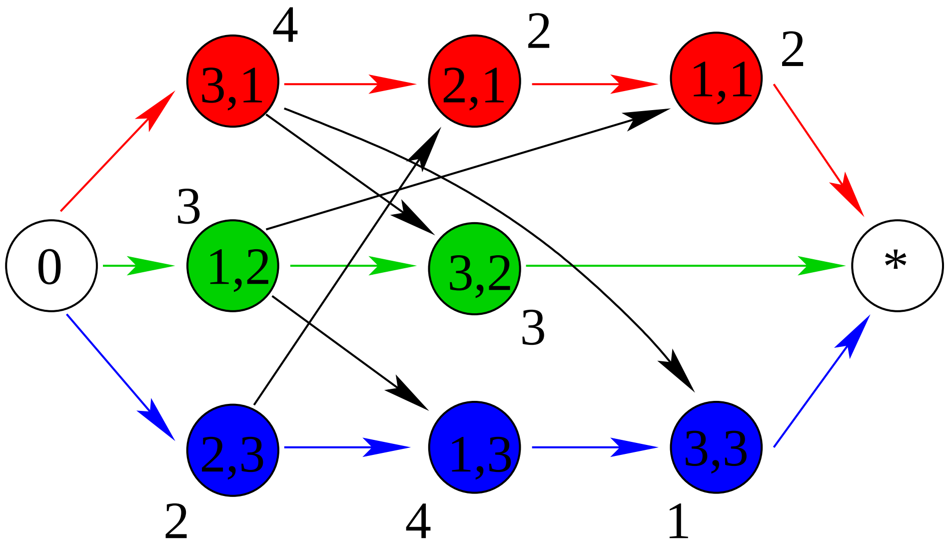
Example 2: Graph $G(S')$ with processing times: (p_{11} changed from 1 to 2!)



$$LB(V) = l(0,(1,2),(1,3),(3,3),*)$$
$$= l(0,(3,1),(2,1),(1,1),*) = 8$$

Better lower bound - example (cont.)

Example 2: Graph $G(S')$ with processing times: (p_{11} changed from 1 to 2!)



$LB(V) = l(0,(1,2),(1,3),(3,3),*)$
$= l(0,(3,1),(2,1),(1,1),*) = 8$

Data for jobs on Machine 1:

green	blue	red
$r_{12} = 0$	$r_{13} = 3$	$r_{11} = 6$
$q_{12} = 5$	$q_{13} = 1$	$q_{11} = 0$

Opt. solution:
$OPT = 9$, $LB^{new}(V) = 9 > LB^{old}(V)$