Scheduling
Shop Scheduling

Tim Nieberg

Shop models: General Introduction

Remark: Consider non preemptive problems with regular objectives

Notation Shop Problems:

- m machines, n jobs $1, \ldots, n$
- operations $O = \{(i,j) | j = 1, \ldots, n; \; i \in M^j \subset M := \{1, \ldots, m\}\}$ with processing times p_{ij}
- M^j is the set of machines where job j has to be processed on
- $PREC$ specifies the precedence constraints on the operations

Notation Shop Problems:

- m machines, n jobs $1, \dots, n$
- operations
  $O = \{(i,j) | j = 1, \dots, n; \ i \in M^j \subset M := \{1, \dots, m\}\}$ with processing times p_{ij}
- M^j is the set of machines where job j has to be processed on
- $PREC$ specifies the precedence constraints on the operations
- Flow shop: $M^j = M$ and
  $PREC = \{(i,j) \to (i+1,j) | i = 1, \dots, m-1; j = 1, \dots, n\}$

Shop models: General Introduction

Notation Shop Problems:

- m machines, n jobs $1, \dots, n$
- operations $O = \{(i,j) | j = 1, \dots, n;\ i \in M^j \subset M := \{1, \dots, m\}\}$ with processing times p_{ij}
- M^j is the set of machines where job j has to be processed on
- $PREC$ specifies the precedence constraints on the operations
- Flow shop: $M^j = M$ and $PREC = \{(i,j) \to (i+1,j) | i = 1, \dots, m-1; j = 1, \dots, n\}$
- Open shop: $M^j = M$ and $PREC = \emptyset$
- Job shop: $PREC$ contain a chain $(i_1, j) \to \dots, \to (i_{|M^j|}, j)$ for each j

Disjunctive Formulation of the constraints

- C_{ij} denotes completion time of operation (i, j)
- *PREC* have to be respected:

Disjunctive Formulation of the constraints

- C_{ij} denotes completion time of operation (i, j)
- $PREC$ have to be respected:

$$C_{ij} - p_{ij} \geq C_{kl} \quad \text{for all } (k, l) \rightarrow (i, j) \in PREC$$

- no two operations of the same job are processed at the same time:

Disjunctive Formulation of the constraints

- C_{ij} denotes completion time of operation (i,j)
- $PREC$ have to be respected:

$$C_{ij} - p_{ij} \geq C_{kl} \quad \text{for all } (k,l) \rightarrow (i,j) \in PREC$$

- no two operations of the same job are processed at the same time:

$$C_{ij} - p_{ij} \geq C_{kj} \text{ or } C_{kj} - p_{kj} \geq C_{ij} \quad \text{for all } i,k \in M^j; i \neq k$$

- no two operations are processed jointly on the same machine:

Shop models: General Introduction

Disjunctive Formulation of the constraints

- $PREC$ have to be respected:

$$C_{ij} - p_{ij} \geq C_{kl} \quad \text{for all } (k,l) \to (i,j) \in PREC$$

- no two operations of the same job are processed at the same time:

$$C_{ij} - p_{ij} \geq C_{kj} \text{ or } C_{kj} - p_{kj} \geq C_{ij} \quad \text{for all } i,k \in M^j; i \neq k$$

- no two operations are processed jointly on the same machine:

$$C_{ij} - p_{ij} \geq C_{il} \text{ or } C_{il} - p_{il} \geq C_{ij} \quad \text{for all } (i,j),(i,l) \in O; j \neq l$$

- $C_{ij} - p_{ij} \geq 0$
- the 'or' constraints are called disjunctive constraints
- some of the disjunctive constraints are 'overruled' by the $PREC$ constraints

Disjunctive Formulation - makes pan objective

$$
\begin{array}{ll}
\min C_{max} & \\
s.t. & \\
C_{max} \geq C_{ij} & (i,j) \in O \\
C_{ij} - p_{ij} \geq C_{kl} & (k,l) \rightarrow (i,j) \in PREC \\
C_{ij} - p_{ij} \geq C_{kj} \text{ or } C_{kj} - p_{kj} \geq C_{ij} & i,k \in M^j; i \neq k \\
C_{ij} - p_{ij} \geq C_{il} \text{ or } C_{il} - p_{il} \geq C_{ij} & (i,j),(i,l) \in O; j \neq l \\
C_{ij} - p_{ij} \geq 0 & (i,j) \in O
\end{array}
$$

Shop models: General Introduction

<u>Disjunctive Formulation - sum objective</u>

$$
\begin{array}{ll}
\min \sum w_j L_j & \\
s.t. & \\
L_j \geq C_{ij} - d_j & (i,j) \in O \\
C_{ij} - p_{ij} \geq C_{kl} & (k,l) \rightarrow (i,j) \in PREC \\
C_{ij} - p_{ij} \geq C_{kj} \text{ or } C_{kj} - p_{kj} \geq C_{ij} & i,k \in M^j; i \neq k \\
C_{ij} - p_{ij} \geq C_{il} \text{ or } C_{il} - p_{il} \geq C_{ij} & (i,j),(i,l) \in O; j \neq l \\
C_{ij} - p_{ij} \geq 0 & (i,j) \in O
\end{array}
$$

<u>Remark</u>:

- also other constraints, like e.g. release dates, can be incorporated
- the disjunctive constraints make the problem hard (lead to an ILP formulation)

Disjunctive Graph Formulation

- graph representation used to represent instances and solutions of shop problems
- can be applied for regular objectives only

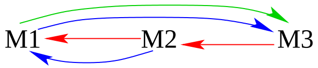
Disjunctive Graph $G = (V, C, D)$

- V set of vertices representing the operations O
- a vertex is labeled by the corresponding processing time;
- Additionally, a source node 0 and a sink node $*$ belong to V; their weights are 0
- C set of conjunctive arcs reflecting the precedence constraints: for each $(k, l) \rightarrow (i, j) \in PREC$ a directed arc belongs to C
- additionally $0 \rightarrow O$ and $O \rightarrow *$ are added to C
- D set of disjunctive arcs representing 'conflicting' operations: between each pair of operations belonging to the same job or to be processed on the same machine, for which no order follows from $PREC$, an undirected arc belongs to D

Disjunctive Graph - Example Job Shop

- Data: 3 jobs, 3 machines;

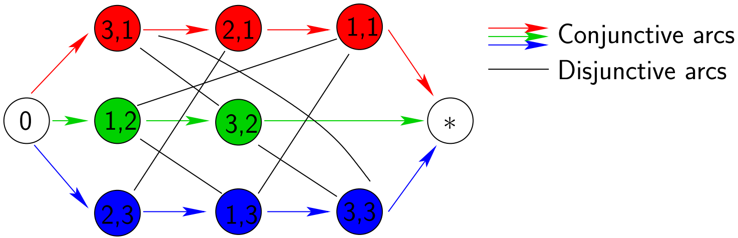
Jobs: 1	red	$(3,1) \rightarrow (2,1) \rightarrow (1,1)$	$p_{31} = 4, p_{21} = 2, p_{11} = 1$
2	green	$(1,2) \rightarrow (3,2)$	$p_{12} = 3, p_{32} = 3$
3	blue	$(2,3) \rightarrow (1,3) \rightarrow (3,3)$	$p_{23} = 2, p_{13} = 4, p_{33} = 1$



Disjunctive Graph - Example Job Shop

Jobs: 1	red	$(3,1) \to (2,1) \to (1,1)$	$p_{31} = 4, p_{21} = 2, p_{11} = 1$
2	green	$(1,2) \to (3,2)$	$p_{12} = 3, p_{32} = 3$
3	blue	$(2,3) \to (1,3) \to (3,3)$	$p_{23} = 2, p_{13} = 4, p_{33} = 1$

- Graph:



Disjunctive Graph - Example Open Shop

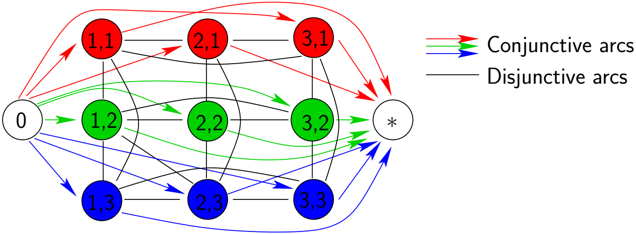
- Data: 3 jobs, 3 machines;

Jobs: 1	red	$(1,1),(2,1),(3,1)$	$p_{11}=4, p_{21}=2, p_{31}=1$
2	green	$(1,2),(2,2),(3,2)$	$p_{12}=3, p_{22}=1, p_{32}=3$
3	blue	$(1,3),(2,3),(3,3)$	$p_{13}=2, p_{23}=4, p_{33}=1$

Disjunctive Graph - Example Open Shop

Jobs: 1	red	$(1,1),(2,1),(3,1)$	$p_{11}=4, p_{21}=2, p_{31}=1$
2	green	$(1,2),(2,2),(3,2)$	$p_{12}=3, p_{22}=1, p_{32}=3$
3	blue	$(1,3),(2,3),(3,3)$	$p_{13}=2, p_{23}=4, p_{33}=1$

- Graph:



Disjunctive Graph - Selection

- basic scheduling decision for shop problems (see disj. formulation):
  define an ordering for operations connected by a disjunctive arc
- $\longrightarrow$ turn the undirected disjunctive arc into a directed arc
- selection S: a set of directed disjunctive arcs
  (i.e. $S \subset D$ together with a chosen direction for each $a \in S$)
- disjunctive arcs which have been directed are called 'fixed'
- a selection is a complete selection if
  - each disjunctive arc has been fixed
  - the graph $G(S) = (V, C \cup S)$ is acyclic

Selection - Remarks

- a feasible schedule induces a complete selection
- a complete selection leads to sequences in which operations have to be processed on machines
- a complete selection leads to sequences in which operations of a job have to be processed
- Does each complete selection leads to a feasible schedule?

Shop models: General Introduction

Calculate a Schedule for a Complete Selection S

- calculated longest paths from 0 to all other vertices in $G(S)$
- Technical description:
  - length of a path $i_1, i_2, \ldots, i_r$ = sum of the weights of the vertices $i_1, i_2, \ldots, i_r$
  - calculate length l_{ij} of the longest path from 0 to (i, j) (using e.g. Dijkstra)
  - start operation (i, j) at time $l_{ij} - p_{ij}$ (i.e. $C_{ij} = l_{ij}$)
  - the length of a longest path from 0 to $*$ (such paths are called **critical paths**) is equal to the makespan of the schedule
- resulting schedule is the semiactive schedule which respects all precedence given by C and S

Shop models: General Introduction

Reformulation Shop Problem

find a complete selection for which the corresponding schedule minimizes the given (regular) objective function

Makespan Minimization

- Lemma: For problem $F||C_{max}$ an optimal schedule exists with
  - the job sequence on the first two machines is the same
  - the job sequence on the last two machines is the same

  (Proof as Exercise)
- Consequence: For $F2||C_{max}$ and $F3||C_{max}$ an optimal solution exists which is a permutation solution
- For $Fm||C_{max}$, $m \geq 4$, instances exist where no optimal solution exists which is a permutation solution
  (Exercise)

Flow Shop models

Problem $F2||C_{max}$

- solution can be described by a sequence π
- problem was solved by Johnson in 1954

Johnson's Algorithm:

1. $L =$ set of jobs with $p_{1j} < p_{2j}$;
2. $R =$ set of remaining jobs;
3. sort L by SPT w.r.t. the processing times on first machine (p_{1j})
4. sort R by LPT w.r.t. the processing times on second machine (p_{2j})
5. sequence L before R (i.e. $\pi = (L, R)$ where L and R are sorted)

Example solution problem $F2||C_{max}$

- $n = 5; p = \begin{pmatrix} 4 & 3 & 3 & 1 & 8 \\ 8 & 3 & 4 & 4 & 7 \end{pmatrix}$

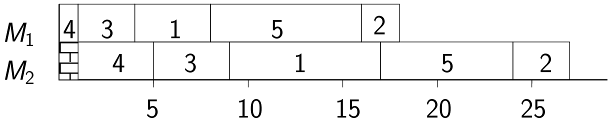
Flow Shop models

Example solution problem $F2||C_{max}$

- $n = 5; p = \begin{pmatrix} 4 & 3 & 3 & 1 & 8 \\ 8 & 3 & 4 & 4 & 7 \end{pmatrix}$
- $L = \{1, 3, 4\};\ R = \{2, 5\}$
- sorting leads to $L = \{4, 3, 1\};\ R = \{5, 2\}$

Example solution problem $F2||C_{max}$

- $n = 5; p = \begin{pmatrix} 4 & 3 & 3 & 1 & 8 \\ 8 & 3 & 4 & 4 & 7 \end{pmatrix}$
- $L = \{1, 3, 4\};\ \ R = \{2, 5\}$
- sorting leads to $L = \{4, 3, 1\};\ \ R = \{5, 2\}$
- $\pi = (4, 3, 1, 5, 2)$



Problem $F2||C_{max}$

- Lemma 1: If

$$\min\{p_{1i}, p_{2j}\} < \min\{p_{2i}, p_{1j}\}$$

then job i is sequenced before job j by Johnson's algorithm.

- Lemma 2: If job j is scheduled immediately after job i and

$$\min\{p_{1j}, p_{2i}\} < \min\{p_{2j}, p_{1i}\}$$

then swapping job i and j does not increase C_{max}.

- Theorem: Johnson's algorithm solves problem $F2||C_{max}$ optimal in $O(n\log(n))$ time.

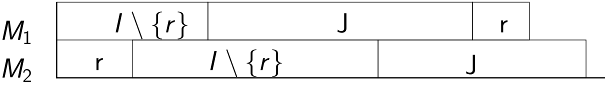
(Proofs on the board)

Problem $F3||C_{max}$

- $F3||C_{max}$ is NP-hard in the strong sense
- Reduction using 3-PARTITION
- Proof on the board

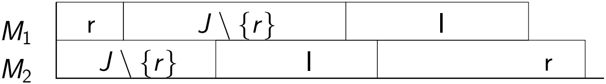
Algorithm Problem $O2||C_{max}$

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



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
  - $p_{1r} + p_{2r}$

Open Shop models

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}

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}
- 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 $O|pmtn|C_{max}$

Open Shop models

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 operataions is called a decrementing set if it contain 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

<u>Algorithm $O|pmtn|C_{max}$</u>

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.

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

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

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

Open Shop models

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

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

Open Shop models

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

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

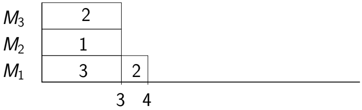
	p				ML
p	2	4	0	2	8
	0	1	2	3	6
	2	0	3	2	7
JL	4	5	5	7	$LB = 8$

Open Shop models

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

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

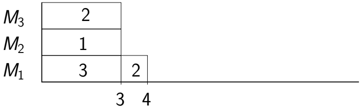
$\Delta = 1$	p				ML
p	2	4	0	2	8
	0	1	2	3	6
	2	0	3	2	7
JL	4	5	5	7	$LB = 8$



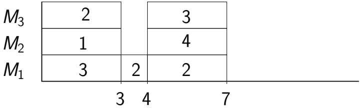
Open Shop models

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

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



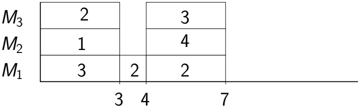
$\Delta = 3$	p				ML
p	2	3	0	2	7
	0	1	2	3	6
	2	0	3	2	7
JL	4	4	5	7	$LB = 7$



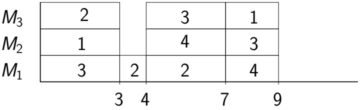
Open Shop models

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

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



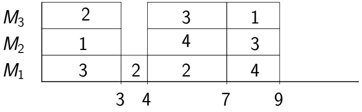
$\Delta = 2$	p				ML
p	2	0	0	2	4
	0	1	2	0	3
	2	0	0	2	4
JL	4	1	2	4	$LB = 4$



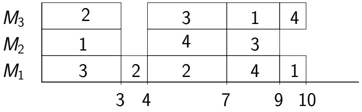
Open Shop models

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

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



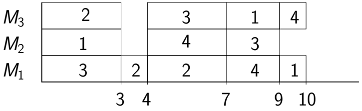
$\Delta = 1$	p				ML
p	2	0	0	0	2
	0	1	0	0	1
	0	0	0	2	2
JL	2	1	0	2	$LB = 2$



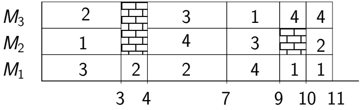
Open Shop models

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

$\Delta = 1$	p				ML
p	2	0	0	0	2
	0	1	0	0	1
	0	0	0	2	2
JL	2	1	0	2	$LB = 2$



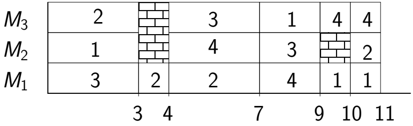
$\Delta = 1$	p				ML
p	1	0	0	0	1
	0	1	0	0	1
	0	0	0	1	1
JL	1	1	0	1	$LB = 1$



Open Shop models

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

	p				ML
p	2	4	3	2	11
	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 than on M_2
- I_{21}: set of jobs processed first on M_2 and than 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}

Job Shop models

Algorithm Problem $J2||C_{max}$

1. on M_1 first schedule the jobs from I_{12} in order π_{12}, than 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}, than the jobs from I_2, and last the jobs from I_{12} in order π_{12}

Job Shop models

Algorithm Problem $J2||C_{max}$

1. on M_1 first schedule the jobs from I_{12} in order π_{12}, than 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}, than 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 presented
  - a branch and bound approach
  - a heuristic approach called the Shifing 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

Generation of all active schedules

- 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

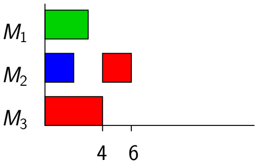
Generation of all active schedules (cont.)

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
   1. extend partial schedule by scheduling (i^*,j) next on machine i^*;
   2. delete (i^*,j) from Ω;
   3. add job-successor of (i^*,j) to Ω;
   4. Return to Step 2

Generation of all active schedules - example

Jobs: 1	(red)	$(3,1) \rightarrow (2,1) \rightarrow (1,1)$	$p_{31} = 4, p_{21} = 2, p_{11} = 1$
2	(green)	$(1,2) \rightarrow (3,2)$	$p_{12} = 3, p_{32} = 3$
3	(blue)	$(2,3) \rightarrow (1,3) \rightarrow (3,3)$	$p_{23} = 2, p_{13} = 4, p_{33} = 1$

Partial Schedule:



Job Shop models

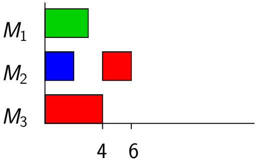
Generation of all active schedules - example

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:



$\Omega = \{(1,1),(3,2),(1,3)\}$;

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

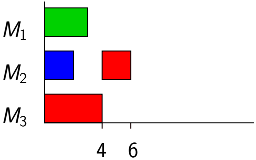
$t(\Omega) = \min\{6+1, 4+3, 3+4\} = 7$;

$i^* = M_1$;

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

Generation of all active schedules - example (cont.)

Partial Schedule:



Ω=$\{(1,1),(3,2),(1,3)\}$;

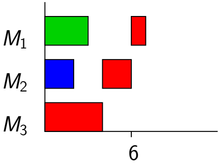
$r_{11} = 6, r_{32} = 4, r_{13} = 3$;

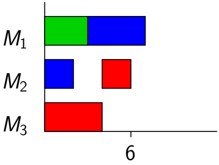
$t(\Omega) = \min\{6+1, 4+3, 3+4\} = 7$;

$i^* = M_1$;

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

Extended partial schedules:



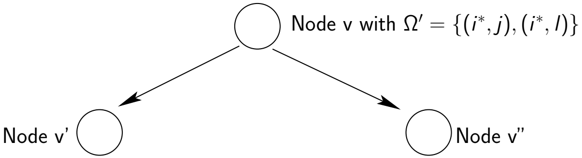


<u>Remarks on the generation</u>:

- 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^*
  $\rightarrow$ a branch fixes new disjunctions

Job Shop models

Disjunctions fixed by a branching



selection (i^*, j)

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

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

Job Shop models

Lower bounds for nodes of the branch and bound tree

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

Job Shop models

Lower bounds for nodes of the branch and bound tree

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

Job Shop models

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)

Job Shop models

Better lower bound

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

Job Shop models

Better lower bound

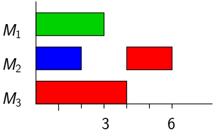
- solve 1-machine problem for all machines
- this results in values $f_1, \ldots, f_m$
- $LB^{new}(V) = \max_{i=1}^{m} f_i$

Remarks:

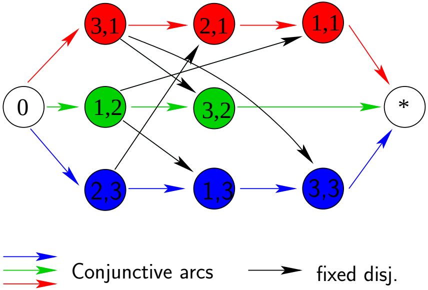
- 1-machine problem is NP-hard
- computational experiments have shown that it pays of to solve these m NP-hard problems per node of the search tree
- 20×20 job-shop instances are already hard to solve by branch and bound

Better lower bound - example

Partial Schedule:

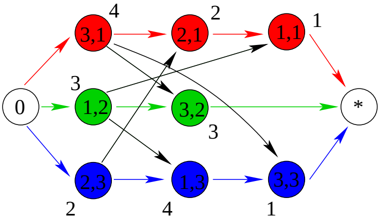


Corresponding graph $G(S')$:



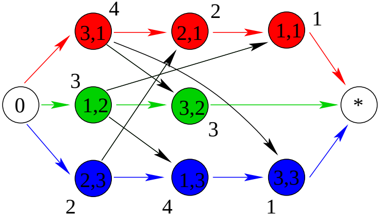
Job Shop models

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



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

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

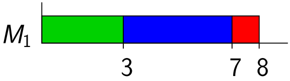


$LB(V) = l(0, (1,2), (1,3), (3,3), *) = 8$

Data for jobs on Machine 1:

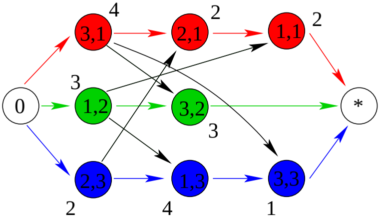
green	blue	red
$r_{12} = 0$	$r_{13} = 3$	$r_{11} = 6$
$q_{12} = 5$	$q_{13} = 1$	$q_{11} = 0$

Opt. solution: $Opt = 8$, $LB^{new}(V) = 8$



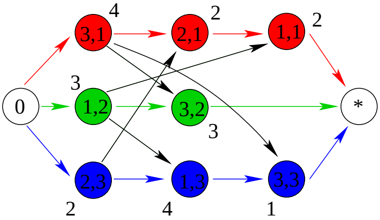
Job Shop models

Change p_{11} from 1 to 2!



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

Change p_{11} from 1 to 2!

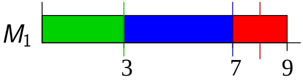


$LB(V) = l(0, (1, 2), (1, 3), (3, 3), *) = l(0, (3, 1), (2, 1), (1, 1), *) = 8$

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: $OPT = 9$, $LB^{new}(V) = 9$



Job Shop models

<u>The Shifting Bottleneck Heuristic</u>

- successful heuristic to solve makespan minimization for job shop
- iterative heuristic
- determines in each iteration the schedule for one additional machine
- uses reoptimization to change already scheduled machines
- can be adapted to more general job shop problems
  - other objective functions
  - workcenters instead of machines
  - set-up times on machines
  - ...

Shifting Bottleneck Heuristic for Job Shop

Basic Idea

- Notation: M set of all machines
- Given: fixed schedules for a subset $M^0 \subset M$ of machines (i.e. a selection of disjunctive arcs for cliques corresponding to these machines)
- Actions in one iteration:
  - select a machine k which has not been fixed (i.e. a machine from $M \setminus M^0$)
  - determine a schedule (selection) for machine k on the base of the fixed schedules for the machines in M^0
  - reschedule the machines from M^0 based on the other fixed schedules

Shifting Bottleneck Heuristic for Job Shop

Selection of a machine

- Idea: Chose unscheduled machine which causes the most problems (bottleneck machine)
- Realization:
  - Calculate for each operation on an unscheduled machine the earliest possible starting time and the minimal delay between the end of the operation and the end of the complete schedule based on the fixed schedules on the machines in M^0 and the job orders
  - calculate for each unscheduled machine a schedule respecting these earliest release times and delays
  - chose a machine with maximal completion time and fix the schedule on this machine

Technical realization

- Define graph $G' = (N, A')$:
  - N same node set as for the disjunctive graph
  - A' contains all conjunctive arcs and the disjunctive arcs corresponding to the selections on the machines in M^0
- $C_{max}(M^0)$ is the length of a critical path in G'

Technical realization

- Define graph $G' = (N, A')$:
  - N same node set as for the disjunctive graph
  - A' contains all conjunctive arcs and the disjunctive arcs corresponding to the selections on the machines in M^0
- $C_{max}(M^0)$ is the length of a critical path in G'

Comments:

- with respect to G' operations on machines from $M \setminus M^0$ may be processed in parallel
- $C_{max}(M^0)$ is the makespan of a corresponding schedule

Technical realization (cont.)

- for an operation (i,j); $i \in M \setminus M^0$ let
  - r_{ij} be the length of the longest path from 0 to (i,j) (without p_{ij}) in G'
  - q_{ij} be the length of the longest path from (i,j) to $*$ (without p_{ij}) in G'

Comments:

- r_{ij} is the release time of (i,j) w.r.t. G'
- q_{ij} is the tail (minimal time till end) of (i,j) w.r.t. G'

Shifting Bottleneck Heuristic for Job Shop

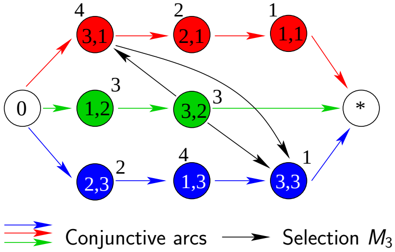
Technical realization (cont.)

- For each machine from $M \setminus M^0$ solve the nonpreemptive one-machine head-body-tail problem $1|r_j, d_j < 0|L_{max}$
- Result: values $f(i)$ for all $i \in M \setminus M^0$
- Action:
  - Chose machine k as the machine with the largest $f(i)$ value
  - schedule machine k according to the optimal schedule of the one-machine problem
  - add k to M^0 and the corresponding disjunctive arcs to G'
- $C_{max}(M^0 \cup k) \geq f(k)$

Shifting Bottleneck Heuristic for Job Shop

Technical realization - Example

- Given: $M^0 = \{M3\}$ and on M_3 sequence $(3,2) \longrightarrow (3,1) \longrightarrow (3,3)$
- Graph G':



- $C_{max}(M^0) = 13$

Technical realization - Example (cont.)

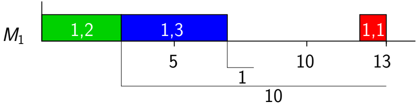
Machine M_1:

(i,j)	$(1,1)$	$(1,2)$	$(1,3)$
r_{ij}	12	0	2
q_{ij}	0	10	1
p_{ij}	1	3	4

Technical realization - Example (cont.)

Machine M_1:

(i,j)	$(1,1)$	$(1,2)$	$(1,3)$
r_{ij}	12	0	2
q_{ij}	0	10	1
p_{ij}	1	3	4

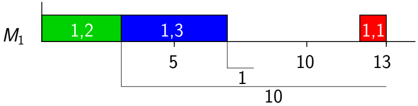


$f(M_1) = 13$

Technical realization - Example (cont.)

Machine M_1:

(i,j)	$(1,1)$	$(1,2)$	$(1,3)$
r_{ij}	12	0	2
q_{ij}	0	10	1
p_{ij}	1	3	4



$f(M_1) = 13$

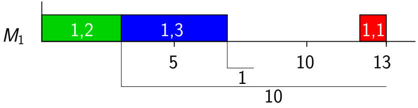
Machine M_2:

(i,j)	$(2,1)$	$(2,3)$
r_{ij}	10	0
q_{ij}	1	5
p_{ij}	2	2

Technical realization - Example (cont.)

Machine M_1:

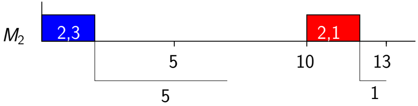
(i,j)	$(1,1)$	$(1,2)$	$(1,3)$
r_{ij}	12	0	2
q_{ij}	0	10	1
p_{ij}	1	3	4



$f(M_1) = 13$

Machine M_2:

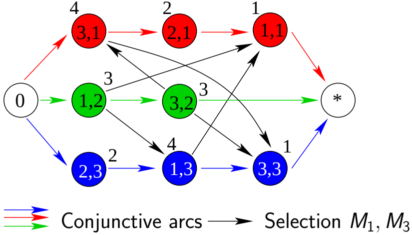
(i,j)	$(2,1)$	$(2,3)$
r_{ij}	10	0
q_{ij}	1	5
p_{ij}	2	2



$f(M_2) = 13$

Technical realization - Example (cont.)

- Choose machine M_1 as the machine to fix the schedule:
  - add $(1, 2) \rightarrow (1, 3) \rightarrow (1, 1)$ to G'
  - $M^0 = \{M_1, M_3\}$



- $C_{max}(M^0) = 13$

Shifting Bottleneck Heuristic for Job Shop

Reschedule Machines

- try to reduce the makespan of the schedule for the machines in M^0
- Realization:
  - consider the machines from M^0 one by one
  - remove the schedule of the chosen machine and calculate a new schedule based on the earliest starting times and delays resulting from the other machines of M^0 and the job orders

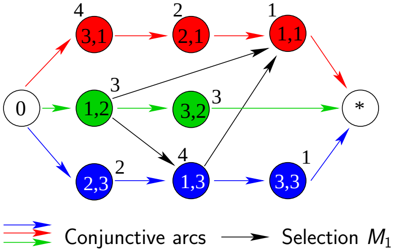
Technical realization rescheduling

For a chosen machine $l \in M^0 \setminus \{k\}$ do:

- remove the arcs corresponding to the selection on machine l from G'
- call new graph G''
- calculate values $r_{ij},\ q_{ij}$ in graph G''
- reschedule machine l according to the optimal schedule of the single machine head-body-tail problem

Technical realization rescheduling - Example

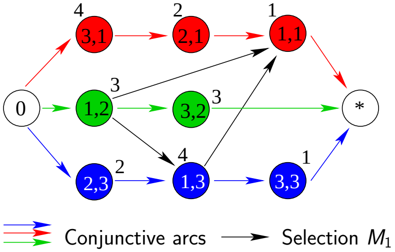
- $M^0 \setminus \{k\} = \{M_3\}$, thus $l = M_3$
- removing arcs corresponding to M_3 leads to graph G'':



$C_{max}(G'') = 8$;

Technical realization rescheduling - Example

- $M^0 \setminus \{k\} = \{M_3\}$, thus $l = M_3$
- removing arcs corresponding to M_3 leads to graph G'':

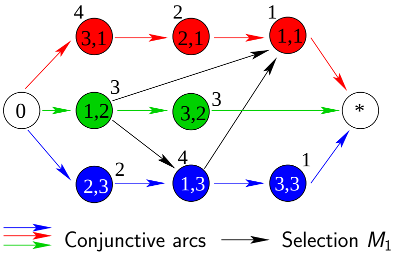


$C_{max}(G'') = 8$;

(i,j)	$(3,1)$	$(3,2)$	$(3,3)$
r_{ij}	0	3	7
q_{ij}	3	0	0
p_{ij}	4	3	1

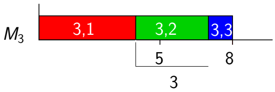
Technical realization rescheduling - Example

- $M^0 \setminus \{k\} = \{M_3\}$, thus $l = M_3$
- removing arcs corresponding to M_3 leads to graph G'':



$C_{max}(G'') = 8$;

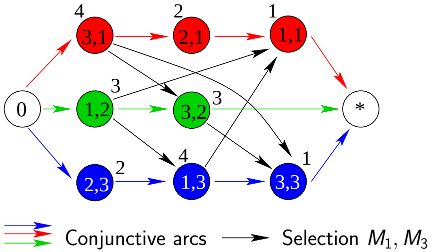
(i,j)	$(3,1)$	$(3,2)$	$(3,3)$
r_{ij}	0	3	7
q_{ij}	3	0	0
p_{ij}	4	3	1



$f(M_3) = 8$

Technical realization rescheduling - Example (cont.)

- add $(3,1) \rightarrow (3,2) \rightarrow (3,3)$ to G''
- New graph:



- $C_{max}^{new}(M^0) = 8$

Shifting Bottleneck Heuristic for Job Shop

Heuristic: summary

1. Initialization:
   1. $M^0 := \emptyset$;
   2. $G :=$ graph with all conjunctive arcs;
   3. $C_{max}(M^0) :=$ length longest path in G:
2. Analyze unscheduled machines:

   FOR ALL $i \in M \setminus M^0$ DO

   FOR ALL operation (i,j) DO

   1. $r_{ij} :=$ length longest path from 0 to (i,j) in G;
   2. $q_{ij} :=$ length longest path from (i,j) to $*$ in G;

   solve single machine head body tail problem $\rightarrow f(i)$

Heuristic: summary (cont.)

3. Bottleneck selection:
    1. determine k such that $f(k) = \max_{i \in M \setminus M^0} f(i)$;
    2. schedule machine k according to the optimal solution in Step 2;
    3. add corresponding disjunctive arcs to G;
    4. $M^0 := M^0 \cup \{k\}$;

Heuristic: summary (cont.)

4. Resequencing of machines:

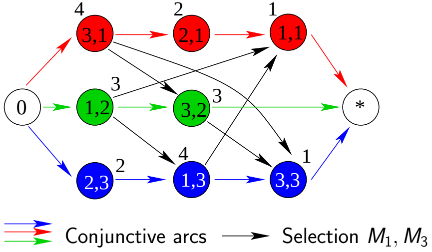
   FOR ALL $i \in M^0 \setminus \{k\}$ DO

   1. delete disjunctive arcs corresponding to machine k from G;
   2. FOR ALL operation (i, j) DO
      1. $r_{ij} :=$ length longest path from 0 to (i, j) in G;
      2. $q_{ij} :=$ length longest path from (i, j) to $*$ in G;
   3. solve single machine head body tail problem $\rightarrow f(i)$
   4. insert corresponding disjunctive arcs to G;
5. Stopping condition

   IF $M^0 = M$ THEN Stop ELSE go to Step 2;

SBH - Example (cont.)

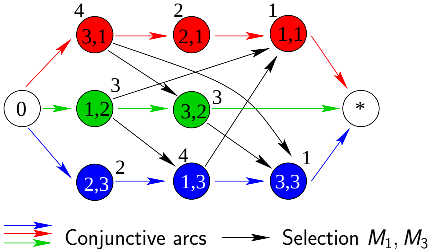
- $M^0 = \{M_1, M_3\}$; thus M_2 is bottleneck
- graph G:



- $C_{max}(G) = 8$;

SBH - Example (cont.)

- $M^0 = \{M_1, M_3\}$; thus M_2 is bottleneck
- graph G:

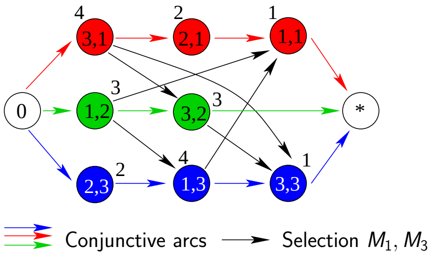


- $C_{max}(G) = 8$;

(i, j)	$(2, 1)$	$(2, 3)$
r_{ij}	4	0
q_{ij}	1	5
p_{ij}	2	2

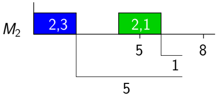
SBH - Example (cont.)

- $M^0 = \{M_1, M_3\}$; thus M_2 is bottleneck
- graph G:



- $C_{max}(G) = 8$;

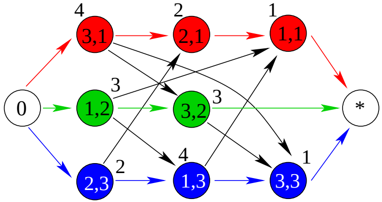
(i,j)	$(2,1)$	$(2,3)$
r_{ij}	4	0
q_{ij}	1	5
p_{ij}	2	2



$f(M_2) = 7$

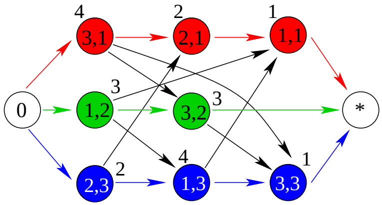
SBH - Example (cont.)

- Situation after Step 3: $M^0 = \{M_1, M_2, M_3\}$, $C_{max}(M^0) = 8$
- Graph G:

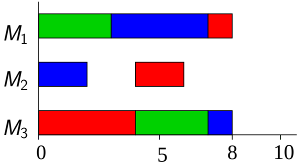


SBH - Example (cont.)

- Situation after Step 3: $M^0 = \{M_1, M_2, M_3\}$, $C_{max}(M^0) = 8$
- Graph G:



- Corresponding Schedule:



An Important Subproblem

- within the SBH the one-machine head-body-tail problem occurs frequently:
- this problem was also used within branch and bound to calculate lower bounds
- the problem is NP-hard (see Lecture 3)
- there are efficient branch and bound methods for smaller instances (see also Lecture 3)
- the actual one-machine problem is a bit more complicated than stated in Lecture 3 (see following example)

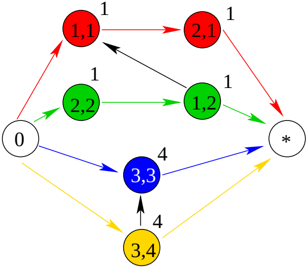
Shifting Bottleneck Heuristic for Job Shop

Example Delayed Precedences

Jobs:	red	$(1,1) \rightarrow (2,1)$
	green	$(2,2) \rightarrow (1,2)$
	blue	$(3,3)$
	yellow	$(3,4)$

Example Delayed Precedences (cont.)

- after 2 iterations SBH we get:
  $M^0 = \{M_3, M_1\}$; $(3,4) \rightarrow (3,3)$ and $(1,2) \rightarrow (1,1)$
- Resulting graph G: ($C_{max}(M^0) = 8$)

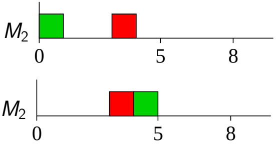


Example Delayed Precedences (cont.)

- 3. iteration: only M_2 unscheduled

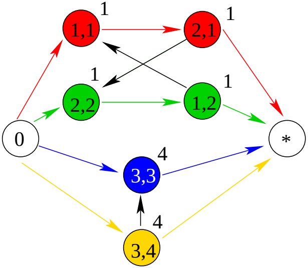
(i,j)	p_{ij}	r_{ij}	q_{ij}
$(2,1)$	1	3	0
$(2,2)$	1	0	3

- Possible schedules for M_2:



Example Delayed Precedences (cont.)

- Both schedules are feasible and might be added to current solution
- But: second schedule leads to



which contains a cycle

Delayed Precedences

- The example shows:
  - not all solutions of the one-machine problem fit to the given selections for machines from M^0
  - the given selections for machines from M^0 may induce precedences for machines from $M \setminus M^0$
- Example:
  - scheduling operation $(1,2)$ before $(1,1)$ on M_1 induces a delayed precedence constraint between $(2,2)$ and $(2,1)$ of length 3
  - $\rightarrow$ operation $(2,1)$ has to start at least 3 time units after $(2,2)$
  - this time is needed to process operations $(2,2),(1,2)$, and $(1,1)$

Shifting Bottleneck Heuristic for Job Shop

Rescheduling Machines

- after adding a new machine to M^0, it may be worth to put more effort in rescheduling the machines:
  - do the rescheduling in some specific order (e.g. based on their 'head-body-tail' values)
  - repeat the rescheduling process until no improvement is found
  - after rescheduling one machine, make a choice which machine to reschedule next (allowing that certain machines are rescheduled more often)
  - ...
- practical test have shown that these extra effort often pays off