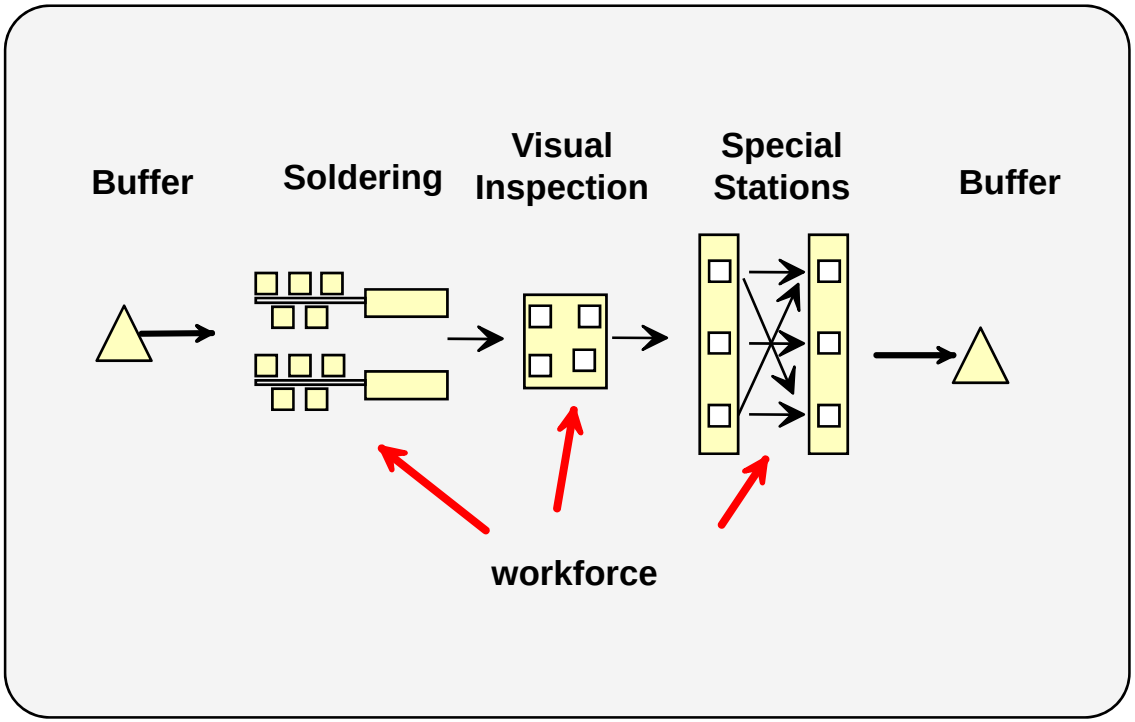
Chapter 8

Operations Scheduling

Operations Scheduling



Operations Scheduling

- **Scheduling is**
  - the process of organizing, choosing and timing resource usage to carry out all the activities necessary to produce the desired outputs at the desired times, while satisfying a large number of time and relationship constraints among the activities and the resources (Morton and Pentico, 1993).
- **Schedule specifies**
  - the time each job starts and completes on each machine, as well as any additional resources needed.
- **A Sequence is**
  - a simple ordering of the jobs.

Operations Scheduling

- Determining a **best** sequence
- 32 jobs on a single machine
- 32! Possible sequences approx. 2.6×10^{35}
  - suppose a computer could examine one billion sequences per second
  - it would take 8.4×10^{15} centuries
- real life problems are much more complicated
- Scheduling theory helps to
  - classify the problems
  - identify appropriate measures
  - develop solution procedures

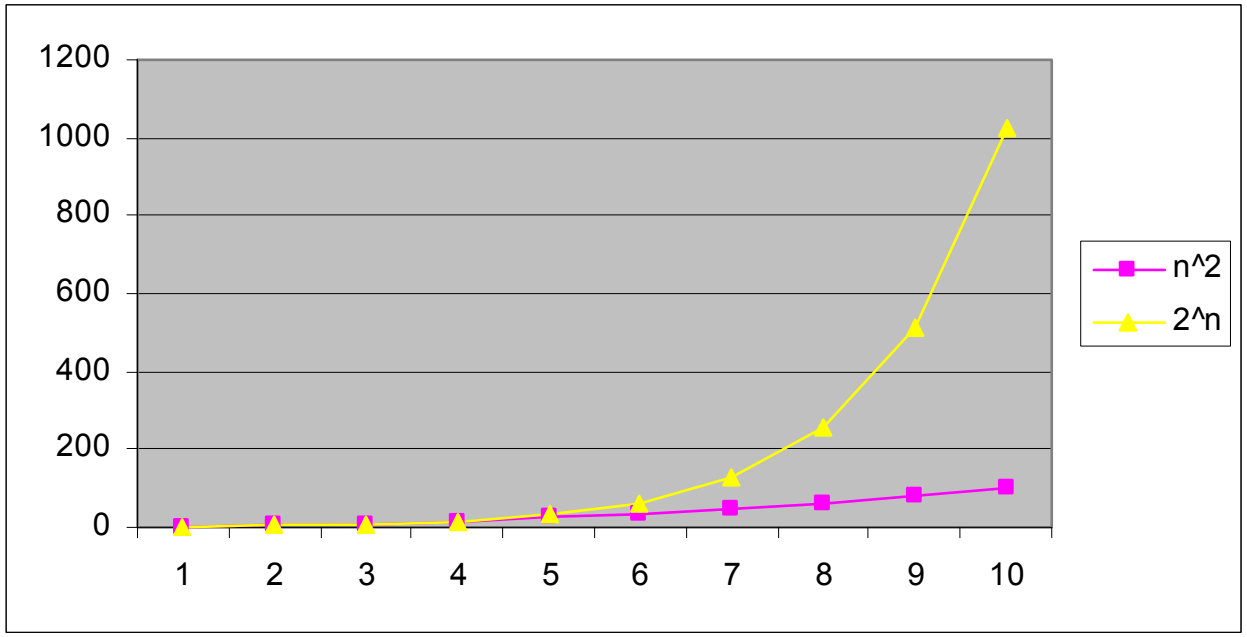
Operations Scheduling

- **Algorithmic complexity**
  - an efficient algorithm is one whose effort of any problem instance is bounded by a polynomial in the problem size, e.g. # of jobs
  - minimal spanning tree can be solved in at most n^2 iterations
  - n: number of edges
  - $O(n^2)$

  - if effort is exponential $O(2^n)$ the algorithm is not efficient
  - branch and bound algorithm for 0/1 variables

  - NP-hard problems: no exact algorithm in polynomial time is known. e.g. Traveling salesman problem
  - Heuristics are usually polynomial algorithms tailored to the specific problem structure

Operations Scheduling



Operations Scheduling

- **Scheduling Theory (Background)**
- **Jobs are**
  - activities to be done
  - processing time known
  - in general continously processed until finished (preemption not allowed)
  - due date
  - release date
  - precedence constraints
  - sequence dependent setup time
  - processed by at most one-machine at the same time

Operations Scheduling

- **Machines (resources)**
  - single machine, parallel machines
  - flow shop:
    - each job must be processed by each machine exactly once
    - all jobs have the same routing
    - a job cannot begin processing on the second machine until it has completed processing on the first
    - assembly line
  - job shop:
    - each job may have a unique routing
  - open shops:
    - job shops in which jobs have no specific routing
    - re-manufacturing and repair

Operations Scheduling

- **Measures**
  - profit, costs
  - it is difficult to relate a schedule to profit and cost
  - **regular measure** is a function of completion time
    - function only increases if at least one completion time in schedule increases

  - n= number of jobs to be processed
  - m= number of machines
  - p_{ik}= time to process job i on machine k
  - r_i = release date of job i
  - d_i = due date of job i
  - w_i = weight of job i relative to the other jobs

Operations Scheduling

- C_i = the completion time
- $F_i = C_i - r_i$, the flowtime
- $L_i = C_i - d_i$, lateness of job i
- $T_i = \max\{0, L_i\}$, tardiness of job i
- $E_i = \max\{0, -L_i\}$, earliness of job i

- $\delta_i = 1$, if job i is tardy ($T_i > 0$)
- $\delta_i = 0$, if job i is on time ($T_i = 0$)

$$C_{max} = \max_{i=1,n}\{C_i\}, \text{ makespan}$$

$$L_{max} = \max_{i=1,n}\{L_i\}, \text{ maximum lateness}$$

$$T_{max} = \max_{i=1,n}\{T_i\}, \text{ maximum tardiness}$$

Operations Scheduling

- **Common proxy objectives**
  - total flowtime
  - total tardiness
  - makespan
  - maximum tardiness
  - number of tardy jobs
  - if not all jobs are equally important weights should be introduced

  - minimizing total completion time is equivalent to minimizing total flowtime or minimizing total tardiness

Operations Scheduling

- **Algorithms:**
  - exact algorithms often based on (worst case scenario) enumeration (e.g. Branch and Bound, Dynamic Programming)
  - heuristic algorithm judged by quality (difference to the optimal solution) and efficacy (computational effort)
  - worst-case bounds are desirable to motivate use of a certain heuristic

Operations Scheduling

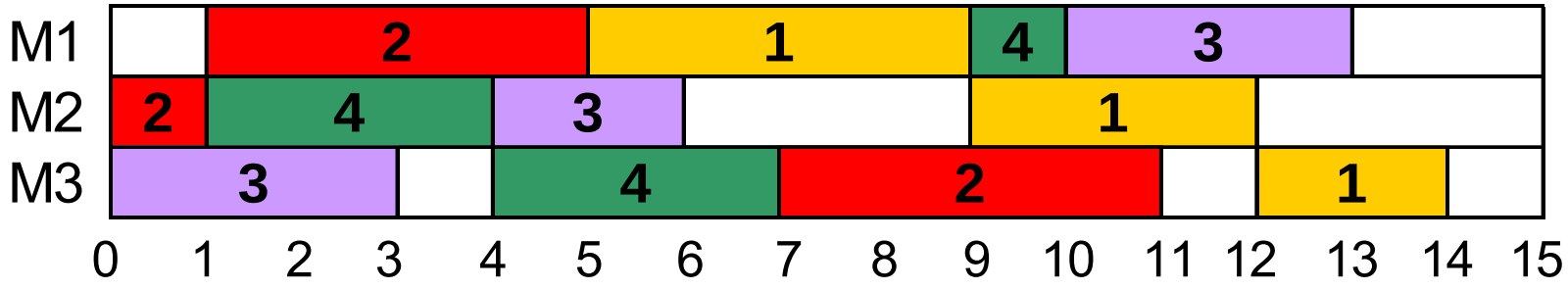
Consider the following four-job, three-machine job-shop scheduling problem:

	Processing time/machine number				
Job	**Op.1**	**Op.2**	**Op.3**	**Release Date**	**Due date**
1	**4/1**	**3/2**	**2/3**	**0**	**16**
2	**1/2**	**4/1**	**4/3**	**0**	**14**
3	**3/3**	**2/2**	**3/1**	**0**	**10**
4	**3/2**	**3/3**	**1/1**	**0**	**8**

- **Assume the following sequences:**
  - 2-1-4-3 on M1
  - 2-4-3-1 on M2
  - 3-4-2-1 on M3

Operations Scheduling

Gantt Chart (machine oriented)



Operations Scheduling

$$C_1 = 14, C_2 = 11, C_3 = 13, C_4 = 10$$

The makespan is

$$C_{\max} = \max\{C_1, C_2, C_3, C_4\} = \max\{14,11,13,10) = 14$$

The total flowtime is

$$\sum_i F_i = 14 + 11 + 13 + 10 = 48$$

Operations Scheduling

The lateness and the tardiness of a job:

$L_1 = 14 - 16 = -2$ $\quad$ $T_1 = \max\{0, -2\} = 0$

$L_2 = 11 - 14 = -3$ $\quad$ $T_2 = \max\{0, -3\} = 0$

$L_3 = 13 - 10 = 3$ $\quad$ $T_3 = \max\{0, 3\} = 3$

$L_4 = 10 - 8 = 2$ $\quad$ $T_4 = \max\{0, 2\} = 2$

The total lateness is

$$\sum_i L_i = (-2) + (-3) + 3 + 2 = 0$$

The total tardiness is

$$\sum_i T_i = 0 + 0 + 3 + 2 = 5$$

The maximum tardiness is

$$T_{\max} = \max\{0, 0, 3, 2\} = 3$$

Tardy jobs have δi =1, so

The number of tardy jobs is

$$T_1 = 0 \Rightarrow \delta_1 = 0$$
$$T_2 = 0 \Rightarrow \delta_2 = 0$$
$$T_3 > 0 \Rightarrow \delta_3 = 1$$
$$T_4 > 0 \Rightarrow \delta_4 = 1$$

$$N_T = 2$$

Operations Scheduling

- **Single Machine Scheduling**
- **Minimizing Flowtime**

  - Problem data

Job *i*	1	2	3	4	5
p *i*	4	2	3	2	4

  - Sequence: 1-2-3-4-5
  - Total Flowtime=?
  - $F=p_1 + (p_1+p_2) + (p_1+p_2+p_3)+...+(p_1+p_2+...+p_n)$
  - $F= np_1 + (n-1)p_2 +...+p_n$

Operations Scheduling

- Theorem. SPT sequencing minimizes total flowtime on a single machine with zero release times.
- Proof. We assume an optimal schedule is not an SPT sequence.
  - $p_i > p_j$
  -
  - $TF(S) = TF(B) + (t+p_i) + (t+p_i+p_j) + TF(A)$
  - $TF(S') = TF(B) + (t+ p_j) + (t+ p_j +p_i) + TF(A)$
  - $TF(S)-TF(S')= p_i - p_j > 0$

Operations Scheduling

SPT-rule ⇒ sequence: 2-4-3-1-5

$$C_1 = 11$$
$$C_2 = 2$$
$$C_3 = 7$$
$$C_4 = 4$$
$$C_5 = 15$$

Total flowtime = total completion time =39

- **SPT rule also minimizes**
  - total waiting time
  - mean # of jobs waiting (mean work in progress)
  - total lateness
- **Why?**

Operations Scheduling

- **Minimize weighted Flow-time:** $\sum_{i=1}^{n} w_i F_i$

- weighted SPT (WSPT): order ratios $\frac{p_i}{w_i}$ (nondecreasing**)**

- exact algorithm for weighted flow-time with zero release time (completion time)

Operations Scheduling

Weighted Flowtime

WSPT scheduling

$$w_1 = 1, w_2 = 4, w_3 = 3, w_4 = 1, w_5 = 3$$

the processing-time-to-weight ratio gives: 4; 0,5; 1; 2; 1,33

the WSPT sequence is the following: 2-3-5-4-1

$$C_1 = 15$$
$$C_2 = 2$$
$$C_3 = 5$$
$$C_4 = 11$$
$$C_5 = 9$$

the value of weighted flowtime is

$$\sum_{i=1}^{5} w_i F_i = 76$$

Operations Scheduling

- **Maximal Tardiness and Maximal Lateness**
  - due date oriented measure
  - earliest due date sequence (EDD)
  - EDD minimizes
    - Maximal Tardiness and
    - Maximal Lateness

Job i	1	2	3	4	5
Due date	16	10	7	7	5
Proc. Time	4	2	3	2	4

- EDD-sequence: 5-3-4-2-1
- Tardiness of the jobs is (0, 0, 2, 1, 0)

Operations Scheduling

- Number of Tardy Jobs
  - Hodgson's algorithm
  - **Step1.** Compute the tardiness for each job in the EDD sequence. Set N_T=0, and let k be the first position containing a tardy job. If no job is tardy go to step 4.
  - **Step 2.** Find the job with the largest processing time in positions 1 to k.

$$\text{Let } \mathrm{p}_{[\mathrm{j}]} = \max_{i=1,k} p_{[i]} \text{ then } \mathrm{j}^* = [j]$$

  - **Step 3.** Remove job j* from the sequence, set N_T=N_T+1, and repeat Step1.
  - **Step 4**. Place the removed N_T jobs in any order at the end of the sequence.
  - **This sequence minimizes the number of tardy jobs**

Operations Scheduling

- **Consider the previous example:**
- **EDD-sequence: 5-3-4-2-1**

- Step1: The tardiness is (0, 0, 2, 1, 0) ⇒ Job 4 in the third position is the first tardy job;

- Step2: The processing times for jobs 5, 3 and 4 are 4, 3, 2, respectively;
  ⇒ largest processing time for job 5

- Step 3: Remove job 5, goto step 1

- Step 1: EDD-sequence is 3-4-2-1; completion times (3, 5, 7, 11) and tardiness (0, 0, 0, 0) ⇒ Go to step 4

- Step 4: schedule that minimizes the number of tardy jobs is 3-4-2-1-5 and has only one tardy job: Job 5

Operations Scheduling

- **Minimize the weighted number of tardy jobs!**
- **NP-hard Problem**
- Heuristic approach: processing-time-to-weight ratio (not exact!)

- Consider the previous example with the following weights:

$$w_1 = 1, w_2 = 4, w_3 = 3, w_4 = 1, w_5 = 3$$

- EDD-sequence was 5-3-4-2-1
- **Step 1** first tardy job is job 4
- **Step 2** the processing-time-weight-ratio for jobs 5, 3 and 4 are 4/3, 3/3 and 2/1
- **Step 3** Remove job 4
- **Step 1** EDD-sequence is 5-3-2-1 with no tardiness
- **Step 4** new schedule 5-3-2-1-4 has one tardy job: job 4 with weight 1

Operations Scheduling

- **Minimize Flowtime with no tardy jobs**
  - for all jobs to be on time, the last job must be on time
  - schedulable set of jobs contain all jobs with due dates greater than or equal to the sum of all processing times
  - Start from the end and choose the job with the **largest** proc time among the schedulable jobs, schedule this job last, remove from the list and continue
  - **Optimal algorithm ! (**corresponding alg. For weighted flowtime is only heuristic)
  - **Problem data**

Job *i*	**1**	**2**	**3**	**4**	**5**
p *i*	**4**	**2**	**3**	**2**	**4**
due date	**16**	**11**	**10**	**9**	**12**

Operations Scheduling

- Step 1: Sum of the processing time is 15
  Job 1 has a due-date greater to 15 ⇒ schedule x-x-x-x-1
- Step 2: Sum of the remaining processing-times is 11
  Job 5 has a larger processing time ⇒ schedule x-x-x-5-1
- Step 3: remaining processing time is 7
  All remaining jobs have due dates at least that big
  ⇒ choose the one with the largest processing time ⇒ x-x-3-5-1
- Step 4: Continue ⇒2-4-3-5-1

Operations Scheduling

- **Minimizing total Tardiness**
- general single-machine tardiness problem is NP-hard

- **Heuristic approach for the weighted problem(Rachamadugu/Morton)**
- if all jobs are tardy, minimizing weighted tardiness is equivalent to minimizing weighted completion time, which is accomplished by the WSPT sequence.

- Weight-to-processing-time ratio is used

- Slack of job i, $S_i = d_i - (p_i + t)$ where t is the current time

Operations Scheduling

- A job should not get full WTPTR „credit“ if its slack is positive

$$S_i^+ = \max\{0, S_i\}$$

- Average processing time of the jobs:

$$p_{av} = 1/n \sum_{i=1}^{n} p_i$$

- Ratio of the slack to the average processing time of jobs:

$$S_i^+ / p_{av}$$

which is the number of average job lengths until job j is tardy

- Weight of a job is discounted by an exponential function: $\exp(-S_i^+ / \kappa p_{av})$

Operations Scheduling

- Define the priority of job i by

$$\gamma_i = \left(\frac{w_i}{p_i}\right) e^{-[S_i^+ / \kappa \cdot p_{av}]}$$

κ is a parmeter of the heuristic to be chosen by the user (e.g. $\kappa = 2$)

- Sequence jobs in descending order of priorities.

Operations Scheduling

- Rachamadugu and Morton (1982) R&M Heuristics:
  - **The owner of** Pensacola Boat Construction **has currently** 10 boats **to construct;**
  - If PBC delivers a boat after the delivery date, a penalty proportional to both the value of the boat and the tardiness must be paid.

Job	1	2	3	4	5	6	7	8	9	10
p(i)	8	12	6	10	3	11	9	11	13	7
w(i)	4	1	6	5	1	4	5	9	8	1
d(i)	26	28	32	35	38	48	50	51	53	64

  - How should PBC schedule the work to minimize the penalty paid?

Operations Scheduling

- **Penalty is weighted tardiness where weights measure the value of the boat.**
- **κ = 2**
- **Calculate:** $p_{av} = 9$

- **Job1:**

$$\gamma_1 = \left(\frac{w_1}{p_1}\right) e^{-[S_1^+/(\kappa\ p_{av})]} = \left(\frac{4}{8}\right) e^{-[(26-8)/(2x9)]} = 0{,}5e^{-1} = 0{,}18$$

Operations Scheduling

Job	1	2	3	4	5	6	7	8	9	10
p(i)	8	12	6	10	3	11	9	11	13	7
w(i)	4	1	6	5	1	4	5	9	8	1
d(i)	26	28	32	35	38	48	50	51	53	64
w(i)/p(i)	0,5	0,08	1	0,5	0,33	0,36	0,56	0,82	0,62	0,14
$S^+(i)$	18	16	26	25	35	37	41	40	40	57
$S^+(i)/kp_{av}$	1	0,89	1,44	1,39	1,94	2,06	2,28	2,22	2,22	3,17
priority	**0,18**	**0,03**	**0,24**	**0,12**	**0,05**	**0,05**	**0,06**	**0,09**	**0,07**	**0,01**

Jobs	**3**	**1**	**4**	**8**	**9**	**7**	**5**	**6**	**2**	**10**	**Sum**
gamma_i	0,24	0,18	0,125	0,09	0,07	0,06	0,05	0,047	0,03	0,01	
p_i	6	8	10	11	13	9	3	11	12	7	90
C_i	6	14	24	35	48	57	60	71	83	90	
d_i	32	26	35	51	53	50	38	48	28	64	
T_i	0	0	0	0	0	7	22	23	55	26	133
w_i	6	4	5	9	8	5	1	4	1	1	
w_i T_i	0	0	0	0	0	35	22	92	55	26	230

Operations Scheduling

- Minimizing Earliness and Tardiness with a Common Due-Date

$$Z = \sum_{i=1}^{n} (E_i + T_i)$$

- this is not a regular measure
- assume common due date: d_j=D

- Number jobs in LPT sequence: $p_1 \geq p_2 \geq \cdots \geq p_n$
- choose j* = n/2 or n/2+0.5

- if $p_1 + p_3 + \cdots + p_{j^*} \leq D$ then the following sequence is

optimal: 1 - 3 - 5 - 7 - . . . - n - . . .- 6 - 4 - 2

Operations Scheduling

- **Example: 10 Jobs with common due-date 80**

Jobs	A	B	C	D	E	F	G	H	I	J
proc Time	8	18	11	4	15	5	23	25	10	17

Operations Scheduling

- if $p_1 + p_3 + \cdots + p_{j^*} > D$ then apply a heuristic (by Sundararaghavan & Ahmed, 1984)

- **Step 0:** Set $B = D; A = \sum_{i=1}^{n} p_i - D; k = b = 1; \mathrm{a} = \mathrm{n}$; use the LPT sequence
- **Step 1:** If B>A:

  assign job k to position b
  b:=b+1
  B:=B-p_k

  else

  assign job k to position a
  a:=a-1
  A:=A-p_k
- **Step 2:** k:=k-1; if k<=n go to step 1.

Problems with non-zero release time

- Non-zero release times typically makes scheduling problems much harder, e.g. SPT does in general not minimize total flowtime

- **Heuristic Approach:**
- At each time t determine the set of **schedulable jobs:** jobs that have been released but not yet processed.

- Choose from the schedulable jobs according to some rule (e.g. SPT for minimizing flowtime)

Operations Scheduling

- **Preemption allowed:**

	j	**1**	**2**	**3**	**4**	**5**	**6**
	r	**12**	**2**	**0**	**11**	**4**	**10**
	p	**8**	**4**	**3**	**6**	**2**	**2**
t=0	rp			**3**			
t=2	rp		4	**1**			
t=3	rp		**4**	C			
t=4	rp		3			**2**	
t=6	rp		**3**			C	
t=9	rp		C				
t=10	rp						**2**
t=11	rp				6		**1**
t=12	rp	8			**6**		C
t=18	rp	**8**			C		
t=24	rp	C					

Operations Scheduling

- **Minimizing makespan with non-zero release time and tails**
- Given n jobs with release times r_i, procssing times p_i, and tails n_i

- **Schrage Heuristics:**
  - Start at t=0
  - 1. Determine schedulable jobs
  - 2. If there are schedulable jobs select the job j* among them with the largest tails, otherwise t=t+1 goto 1.
  - 3. Schedule j* at t
  - 4. If all jobs have been scheduled stop, otherwise set $t = t + p_{j*}$, goto 1.

Operations Scheduling

- **Schrage Heuristics Example: 6 jobs with release times and tails**

j	1	2	3	4	5	6
r_j	12	2	0	11	9	10
p_j	8	4	3	6	2	2
n_j	21	9	2	6	7	10

- **Minimize makespan!**

Operations Scheduling

- Denote by SJ the set of schedulable jobs and by S the scheduled sequence

- Step 1. t = 0, SJ = {3}, S = <3>, t = 3, C_{max} = 5
- Step 2. t=3, SJ = {2}, S = <3-2>, t = 7, C_{ma} = 16
- Step 3. t = 9, SJ = {5}, S = <3-2-5>, t = 11, C_{ma} = 18
- Step 4. t=11, SJ = {4, 6}, S = <3- 2- 5- 6>, t = 13, C_{ma} = 23
- Step 5. t=13, SJ = {1, 4}, S = <3- 2- 5- 6-1>, t = 21, C_{ma} = 42
- Step 6. T=21 SJ = {4}, S = <3- 2- 5- 6- 1- 4>, t = 27, C_{ma} = 42

- **Schrage heuristic is in general not optimal, e.g. B&B model can be used as an exact algorithm**

Operations Scheduling

- **Minimizing Set-Up Times**
  - sequence-dependent set-up times
  - the time to change from one product to another may be significant and may depend on the previous part produced
  - p_{ij} = time to process job j if it immediately follows job i

  - Examples:
    - electronics industry
    - paint shops
    - injection molding

  - minimizes makespan
  - problem is equivalent to the traveling salesman problem (TSP), which is NP-hard.

Operations Scheduling

SST(=shortest set-up time) heuristic

A metal products manufacturer has contracted to ship metal braces euch day fo four customers. Each brace requires a different set-up on the rolling mill:

Rolling mill set-up times

Job	A	B	C	D
A	∞	3	4	5
B	3	∞	4	6
C	1	6	∞	2
D	5	4	∞*	∞

*Job C cannot follow job D, because of quality problems

SST-heuristic:

Step 1 starting arbitrarily by choosing one Job: A

Step 2 B has the smallest set-up time following A; ⇒ A-B

Step 3 C has the smallest set-up time of all the remaining jobs following B; ⇒ A-B-C

Step 4 D is the last remaining job; ⇒ A-B-C-D-A with a makespan of 3 + 4 + 2 + 5 =14

Operations Scheduling

- **A regret based Algorithm**
  - makespan must be at least as big as the n smallest elements
  - reduced matrix
    - row reduction
    - column reduction
    - sum of reduced costs = lower bound for TSP
  - **find reduced matrix!**

Job	A	B	C	D
A	∞	3	4	5
B	3	∞	4	6
C	1	6	∞	2
D	5	4	∞*	∞

Operations Scheduling

- **The reduced matrix has a zero in every row and column**
- **what happens if we do not choose j to follow i**
- **regret: lower bound on not choosing j to follow i**

Job	A	B	C	D
A	∞	0	0	2
B	0	∞	0	3
C	0	5	∞	0
D	1	0	∞*	∞

Operations Scheduling

Regret heuristic

Find the cycle sequence that minimizes the set-up time.

Set-up times

Job	1	2	3	4	5
1	∞	18	3	3	6
2	19	∞	9	10	5
3	9	18	∞	13	20
4	6	6	1	∞	2
5	17	1	13	17	∞

Solution: TSP model – regret heuristic

Step 0 C(max) = 0 and L = 1
Step 1 Reduce the matrix:

Reduced matrix

Job	1	2	3	4	5	Min
1	∞	15	0	0	6	3
2	14	∞	4	5	0	5
3	0	9	∞	4	11	9
4	5	5	0	∞	1	1
5	16	0	12	16	∞	1
						19

Operations Scheduling

Step 2 Calculate the regret

Job	1	2	3	4	5	Min
1	∞	15	0(0)	0(4)	6	3
2	14	∞	4	5	0(5)	5
3	0(9)	9	∞	4	11	9
4	5	5	0(1)	∞	1	1
5	16	0(17)	12	16	∞	1
						19

New matrix

Job	1	3	4	5
1	∞	0	0	3
2	14	4	5	∞
3	0	∞	4	11
4	5	0	∞	1

Step 3 Choose the largest regret : 17

Step 4 Assign a job pair: Job 2 immediately follows job 5 (5-2)

L = 1+1;

We prohibit 2-5

Operations Scheduling

Step1 Reduce the matrix

$C_{max} = 19+4+1=24$

Reduced Matrix

Job	1	3	4	5
1	∞	0	0	2
2	10	0	1	∞
3	0	∞	4	10
4	5	0	∞	0

Step 2 Calculate the regret

Job	1	3	4	5
1	∞	0(0)	0(1)	2
2	10	0(1)	1	∞
3	0(9)	∞	4	10
4	5	0(0)	∞	0(2)

Step 3 Choose the largest regret: 9
Step 4 Assign a job pair: 3-1
Prohibit 1-3

Step 1 Reduce the matrix: not possible

Matrix

Job	3	4	5
1	∞	0	2
2	0	1	∞
4	0	∞	0

Operations Scheduling

Step 2 Calculate regret

Job	3	4	5
1	∞	0(3)	2
2	0(1)	1	∞
4	0(0)	∞	0(2)

Step 3 Choose the largest regret: 3
Step 4 Assign job pair : 1-4; partial sequence: 5-2, 3-1-4
Prohibit 4-1 and 4-3 (to keep 3-1-4-3 from being chosen)

Final Matrix

Job	3	5
2	0	∞
4	∞	0

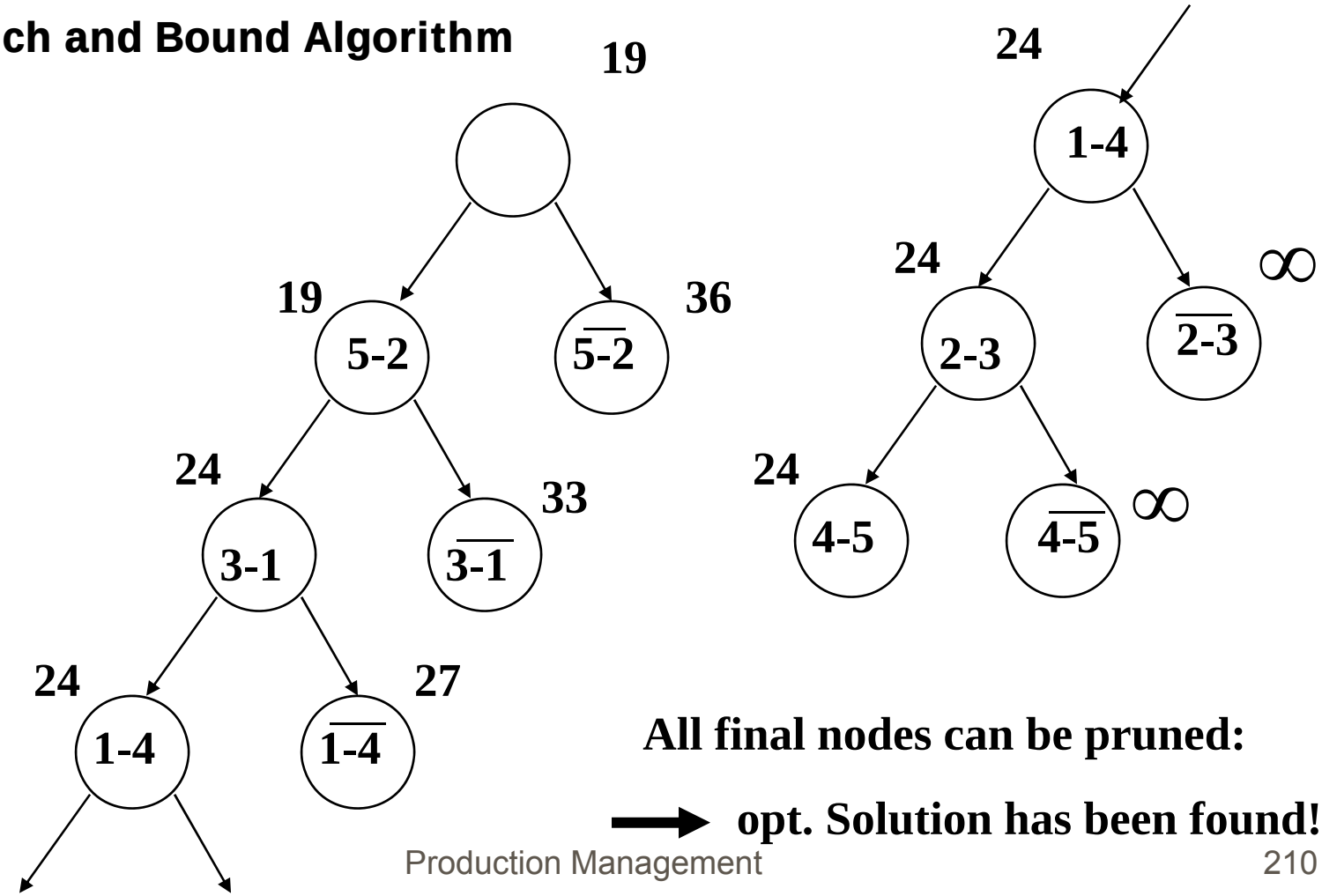
choose 2-3 and 4-5
-> sequence 3-1-4-5-2
the total set-up time is 24

Operations Scheduling

- **Branch and Bound Algorithm**
  - 1. Using the regret heuristic construct a (sub-)tree where each node represents the decision to let j follow i ($i - j$) or to prohibit that j follows i ($\overline{i - j}$)
  - 2. For each node a lower bound for the makespan is inferred from the regret heuristic
  - 3. Once a solution is obtained from the regret heuristic this is an upper bound for the optimal makespan. All nodes where the lower bound is above that level are pruned.
  - 4. If all but one final node are pruned (or no non-pruned node can be further branched) this final node gives the optimal solution.
  - 5. If 4. does not hold start again with 1. at one of nodes which are not pruned and can still be branched.

Operations Scheduling

- **Branch and Bound Algorithm**



All final nodes can be pruned:

→ **opt. Solution has been found!**

Operations Scheduling

- **Single-Machine Search Methods**
  - Neighborhood Search
  - Simulated Annealing
  - Ant System
  - Tabu Search
  - ...
- **Neighborhood Search**
  - seed
  - Neighborhood
  - any heuristic can be used to produce an initial sequence

Operations Scheduling

- adjacent pairwise interchange (API):
  - n-1 neighbors
  - 1-2-3-4-*5-6*-7-8-9
  - 1-2-3-4-*6-5*-7-8-9
- Pairwise interchange (PI):
  - n(n-1)/2 neighbors
  - 1-2-8-4-5-6-7-3-9
- Insertion (INS)
  - $(n-1)^2$ neighbors
  - 1-2-3-7-4-5-6-8-9
- Evaluation function
- Update function

Operations Scheduling

Neighborhood search

Consider the following single- machine tardiness problem;
Use the EDD sequence as the initial seed with an API neighborhood;

Data for neighborhood search

Job	1	2	3	4	5	6
Processing time	10	3	16	8	4	10
Due-date	15	16	24	30	35	37

- Step 1 Construct the EDD sequence and evaluate its total tardiness
  Set i = 1 and j = 2

- The EDD sequence S*: 1-2-3-4-5-6; tardiness-vector (0, 0 ,5 , 7 , 6, 14)

Operations Scheduling

- Step 2 Swap the jobs in the *i*-th and *j*-th position in S*; the sequence is S' with tardiness T'. If T' < T, go to step4
- Step 3 j = j +1: If j >n: go to step 5. Otherwise, i = j-1and go to step 2;
- Step 4 Replace S* with S'; i = 1, j = 2; go to step 2
- Step 5 Stop; S* is a local optimal sequence.

Operations Scheduling

Neighborhood search solution

Jobs		Schedule						Tardiness
i	j	1	2	3	4	5	6	32
1	2	2	1	3	4	5	6	32
2	3	1	3	2	4	5	6	42
3	4	1	2	4	3	5	6	33
4	5	1	2	3	5	4	6	30
1	2	2	1	3	5	4	6	30
2	3	1	3	2	5	4	6	40
3	5	1	2	5	3	4	6	34
5	4	1	2	3	4	5	6	32
4	6	1	2	3	5	6	4	32

Operations Scheduling

- **Single machine results**
  - Flowtime - SPT (E)
  - Lateness - SPT (E)
  - Weighted Flowtime -WSPT (E)
  - Maximal Tardiness (Lateness) - EDD (E)
  - Nb. Of tardy jobs - Hodgson (E)
  - weighted nb. Of tardy jobs - modified Hodgson (H)
  - No jobs tardy/flowtime - modified SPT (E)
  - Tardiness - R&M (H)
  - weighted Tardiness - R&M (H)
  - makespan with non-zero release time and tails - Schrage (H)
  - Sequence dependent - SST (H), regret (H), B&B (E)

Operations Scheduling

- **Parallel Machines**
  - Scheduling decisions:
    - which machine processes the job
    - in what order
  - **List Schedule**
    - to create a schedule, assign the job on the list to the machine with the smallest amount of work assigned.
    - **Step 0**. Let $H_i=0$, $i=1,2,\ldots,m$ be the assigned workload on machine i, $L=([1],[2],\ldots,[n])$ the ordered list sequence, $C_j=0$, $j=1,2,\ldots,n$, and $k=1$
    - **Step 1**. Let $j^*=L_k$ and $H_{i^*}=\min_{i=1,m}\{Hi\}$; Assign job j^* to be processed on machine i^*, $C_{j^*}=H_{i^*}+p_{j^*}$, $H_{i^*}=H_{i^*}+p_{j^*}$
    - **Step 2**. Set $k=k+1$, if $k>n$, stop. Otherwise go to step 1.

Operations Scheduling

Minimizing flowtime on parallel processors

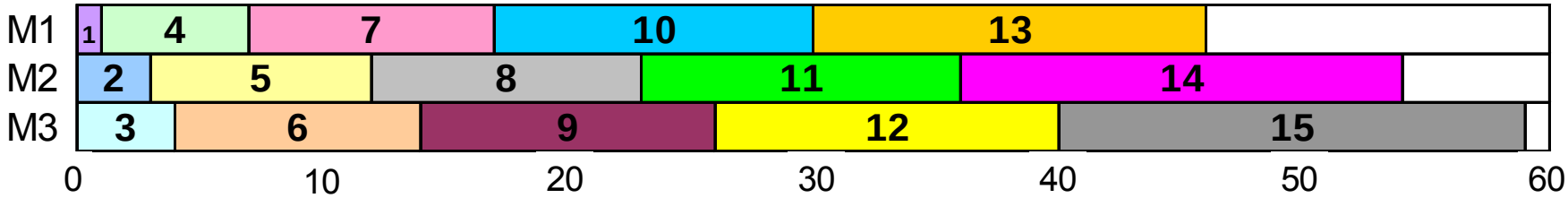
Consider a facility with 3 identical machines and 15 jobs that need to be done as soon as possible;
Processing times(after SPT):

Job	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Time	1	2	4	6	9	10	10	11	12	13	13	14	16	18	19

Optimal schedule:

Machine 1			Machine 2			Machine 3		
j	p(j)	C(j)	j	p(j)	C(j)	j	p(j)	C(j)
1	1	1	2	3	3	3	4	4
4	6	7	5	9	12	6	10	14
7	10	17	8	11	23	9	12	26
10	13	30	11	13	36	12	14	40
13	16	46	14	18	54	15	19	59

Total flowtime = 372



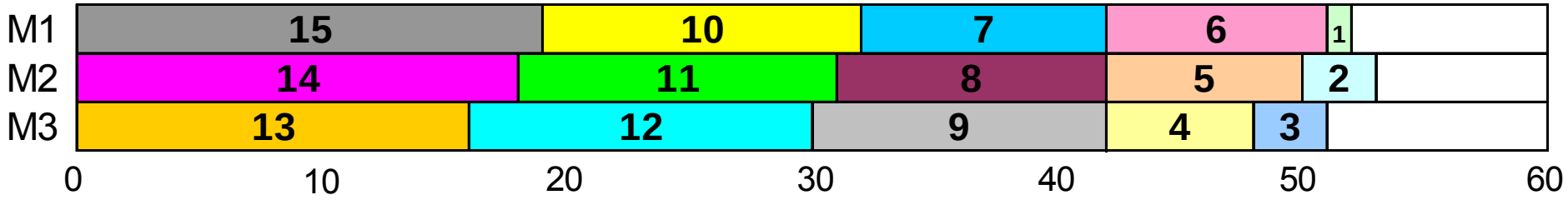
Operations Scheduling

Minimize the makespan

Use a longest processing time (LPT) first list;
Assign the next job on the list to the machine with the least total processing time assigned.

Optimal schedule:

Machine 1			Machine 2			Machine 3		
j	p(j)	C(j)	j	p(j)	C(j)	j	p(j)	C(j)
15	19	19	14	18	18	13	16	16
10	13	32	11	13	31	12	14	30
7	10	42	8	11	42	9	12	42
6	10	52	5	9	51	4	6	48
1	1	53	2	3	54	3	4	52



Operations Scheduling

- **Flow shops**
  - all jobs are processed in the same order
  - two machine makespan model: Johnson's Algorithm
  - Bound on makespan:

$$C^*_{\max} \geq \max\left\{\left(\min_{i=1,n} p_{i2} + \sum_{i=1}^{n} p_{i1}\right), \left(\min_{i=1,n} p_{i1} + \sum_{i=1}^{n} p_{i2}\right)\right\}$$

  - Formulate Johnson's Algorithm
  - For 2-machine Flow shops the optimal schedule is a **Permutation Schedule,** i.e. the job sequence is the same on every machine

Operations Scheduling

- **Makespan with more than two machines**
  - Johnson's algorithm will work in special cases, e.g. three machine problem where the second machine is dominated:

$$p_{i2} \leq \max(\min p_{i1}, \min p_{i3})$$

  - Formulate an artificial two machine problem with

$$p'_{i1} = p_{i1} + \mathrm{p}_{\mathrm{i2}} \quad \text{and} \quad p'_{i2} = p_{i2} + p_{i3}$$

and solve it using the Johnson algorithm gives the optimal solution for the three machine problem

Operations Scheduling

- **Heuristics for the m-machine problem**
  - Cambell, Dudek and Smith (1970)
  - convert a m-machine problem into a two machine problem
  - how?

$$p'_{i1} = \sum_{j=1}^{k} p_{ij} \quad \text{and} \quad p'_{i2} = \sum_{j=l}^{m} p_{ij}$$

  - Start with:k=1 and l=m; then k=2 and l=m-1; until: k=m-1 and l=2
  - m-1 schedules are generated
  - Use the best of these m-1 schedules

Operations Scheduling

Flow-shop heuristics

Processing data:

Job	1	2	3	4	5
M1	1	10	17	12	11
M2	13	12	9	17	3
M3	6	18	13	2	5
M4	2	18	4	6	16

Use the CDS heuristic to solve this problem.

(1) i.) Use the Johnson's algorithm only for M1 and M4:

Job	1	2	3	4	5
M1	1	10	17	12	11
M4	2	18	4	6	16
[j]	1	2	5	4	3

1-2-5-4-3, C_{max} = 88

Next combine M1 and M2 to pseudomachine 1 and M3 and M4 to pseudomachine 2.

Job	1	2	3	4	5
PM1	14	22	26	29	14
PM2	8	36	17	8	21
[j]	4	2	3	5	1

5-2-3-1-4, C_{max} = 85

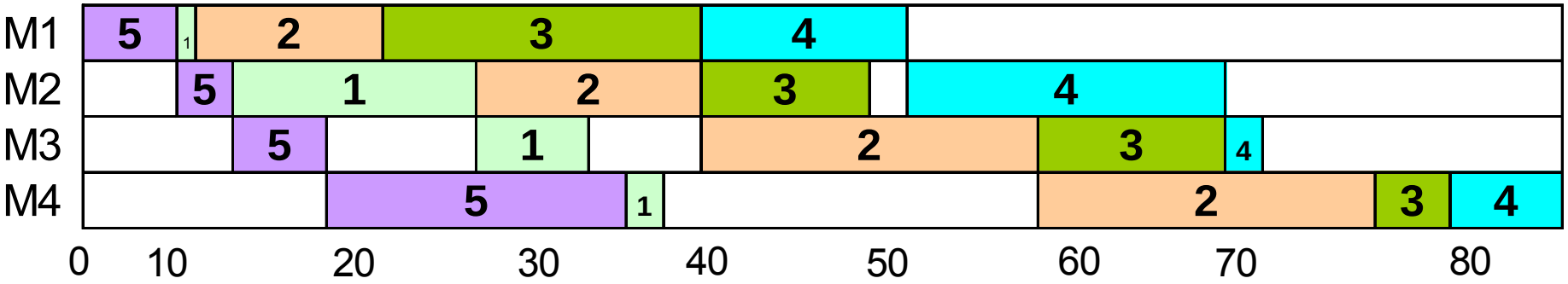
Finally combine M1, M2 and M3 to pseudomachine 1 and M2, M3 and M4 to pseudomachine 2.

Job	1	2	3	4	5
PM1	20	40	39	31	19
PM2	21	48	26	25	24
[j]	2	3	4	5	1

5-1-2-3-4, C_{max} = 85

Operations Scheduling

- **Gantt Chart for the CDS schedule**



Operations Scheduling

- **Gupta – Heuristic**
  - Gupta (1972)
  - exact for 2-machine problem and 3-machine problem, where the 2nd machine is dominated

$$e_i = \begin{cases} 1 & \text{if } p_{i1} < p_{im} \\ -1 & \text{if } p_{i1} \geq p_{im} \end{cases} \qquad s_i = \frac{e_i}{\min\limits_{k=1,\ldots,m-1} \{p_{i,k} + p_{i,k+1}\}}$$

  - Sorting jobs with nonincreasing s_i $(s_{[1]} \geq s_{[2]} \geq \ldots \geq s_{[n]})$

Job	p1+p2	p2+p3	p3+p4	min	ei	si	[i]
1	14	19	8	8	1	0.12	1
2	22	30	36	22	1	0.05	3
3	26	22	17	17	-1	-0.06	4
4	29	19	8	8	-1	-0.12	5
5	14	8	21	8	1	0.12	2

Operations Scheduling

- **Branch and Bound Approaches**
  - machine based bounds
  - job based bounds
  - three machines
    - H_j=current completion time of the last job scheduled on machine j
    - U=set of unscheduled jobs
    - makespan on machine 1 must be at least:

$$C^*_{\max} \geq H_1 + \sum_{i \in U} p_{i1} + \min_{i \in U}\{p_{i2} + p_{i3}\}$$

    - machine2:

$$C^*_{\max} \geq \max\left\{\left(H_1 + \min_{i \in U}\{p_{i1}\}\right), H_2\right\} + \sum_{i \in U} p_{i2} + \min_{i \in U}\{p_{i3}\}$$

Operations Scheduling

- Machine 3:

$$C^*_{\max} \geq \max\left\{ \left(H_1 + \min_{i \in U}\{p_{i1} + p_{i2}\}\right), \left[H_2 + \min_{i \in U}\{p_{i2}\}\right], H_3 \right\} + \sum_{i \in U} p_{i3}$$

- job oriented bounds:

$$C_{\max} \geq H_1 + \max_{i \in U}\left\{ \sum_{j=1}^{m} p_{ij} + \sum_{k \in U, k \neq i} \min\{p_{k1}, p_{k3}\} \right\}$$

Operations Scheduling

- **B&B algorithm for minimizing makespan in multi-machine Flow Shops**

- **1.** Create an initial incumbent solution, e.g. CDS heuristic
  → upper bound
- 2. Starting at t=0 with a root node; branch the tree by generating a node for each schedulable jobs.
- 3. In each node calculate the lower bounds and prune the node if at least one exceeds the upper bound.
- 4. If a non-pruned final node exists at the lowest level take the
- corresponding solution as new incumbent, update the upper bound
  and do the corresponding pruning.
- 5. If all final nodes are pruned current incumbent is the optimal solution, otherwise branch at the node with the lowest lower bound and goto 3.

Operations Scheduling

Makespan permutation schedule for a three-machine flow-shop

Processing data:

	Job i				
Machine j	1	2	3	4	5
1	1	10	17	12	11
2	13	12	9	17	3
3	6	18	13	2	5

Solution:
Start with CDS algorithm: sequence: 1-2-3-4-5, $C_{max} = 65$

Initial lower bound:

M1: $C_{max}^* >= H_1 + (p_{11} + p_{21} + p_{31} + p_{41} + p_{51})$
$+ \min\{ p_{12} + p_{13}, p_{22} + p_{23}, p_{32} + p_{33}, p_{42} + p_{43}, p_{52} + p_{53} \}$
$=0 + (1 + 10 + 17 + 12 + 11) + \min\{19, 30, 22, 19, 8\} = 51 + 8 =59$

M2: $C_{max}^* >= \max\{[H_1 + \min\{p_{11},p_{21},p_{31},p_{41},p_{51}\}],H_2\}$
$+ (p_{12} + p_{22} + p_{32} + p_{42} + p_{52}) + \min \{ p_{13} , p_{23}, p_{33}, p_{43}, p_{53}\}$
$=\max\{[0 + \min\{1, 10, 17, 12, 11\}], 0\}$
$+ (13 + 12 + 9 + 17 + 3) + \min\{6, 18, 13, 2, 5\}$
$=1 + 54 + 2 =57$

Operations Scheduling

M3: $C_{max}^* >= \max\{[H_1 + \min\{ p_{11} + p_{12}, p_{21} + p_{22}, p_{31} + p_{32}, p_{41} + p_{42}, p_{51} + p_{52} \}],$
$[H_2 + \min\{p_{12}, p_{22}, p_{32}, p_{42}, p_{52}\}], H_3\} + (p_{13} + p_{23} + p_{33} + p_{43} + p_{53})$
$=\max\{[0 + \min\{14, 22, 26, 29, 14\}],$
$[0 + \min\{13, 12, 9, 17, 3\}], 0\} + (6 + 18 + 13 + 2 + 5)$
$=\max\{14, 3, 0\} + 44 =58$

Job-based bounds are the following:

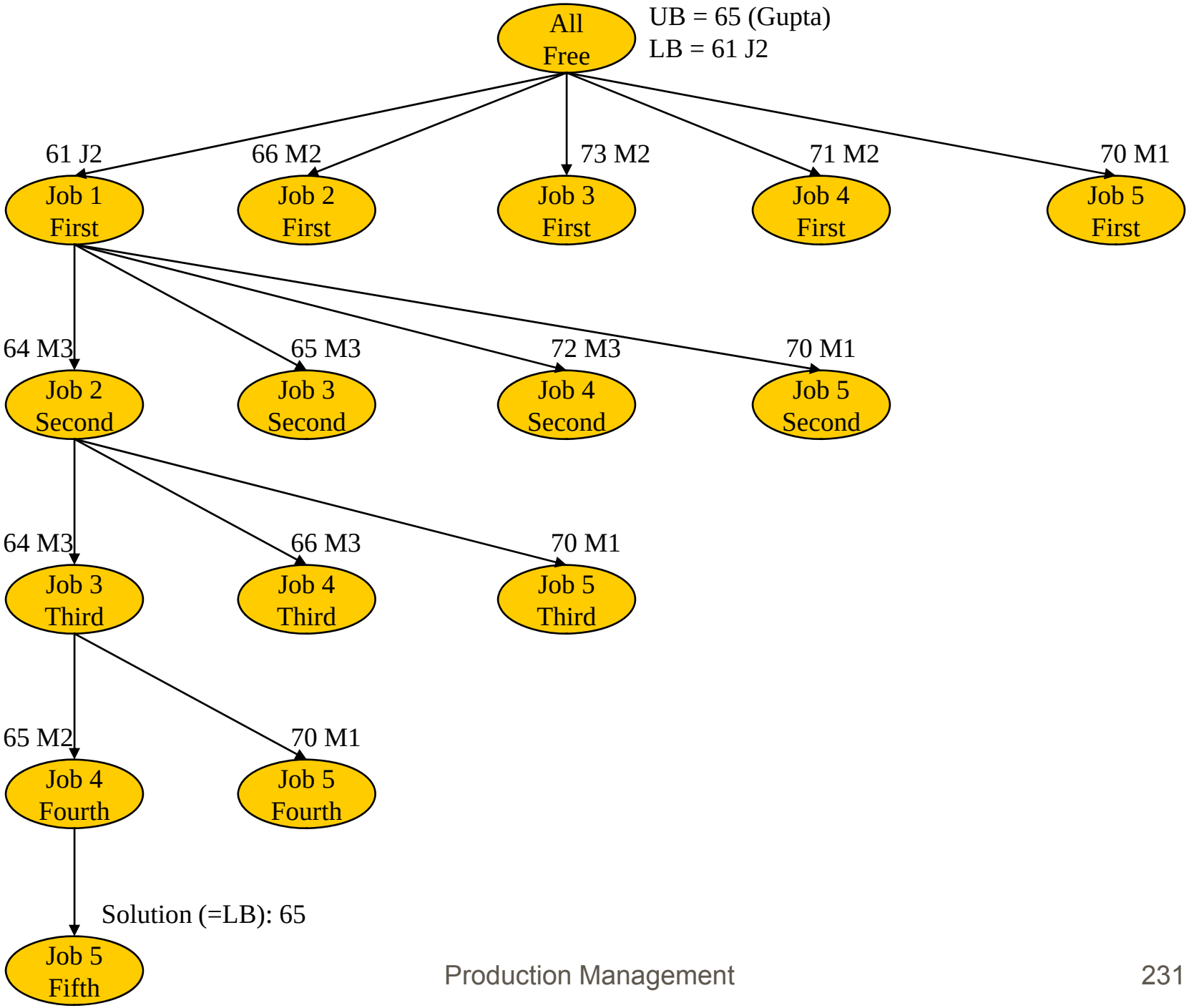
$$C_{\max} \geq H_1 + \sum_{j=1}^{3} p_{1j} + \sum_{k \in \{2,3,4,5\}} \min\{p_{k1}, p_{k3}\}$$

J1: $C_{max}^* >= H_1 + (p_{11} + p_{12} + p_{13})$
$+(\min\{ p_{21}, p_{23} \} + \min\{p_{31}, p_{33} \} + \min \{p_{41}, p_{43} \} + \min\{ p_{51}, p_{53} \})$
$=0 + (1 + 13 + 6) +(\min\{10, 18\} + \min\{17, 13\} + \min\{12,2\} + \min\{11,5\})$
$=0 + 20 + (10 + 13 + 2 + 5)=50$

Similarly, we have

J2: $C_{max}^* >= 61$, J3: $C_{max}^* >= 57$, J4: $C_{max}^* >= 60$, J5: $C_{max}^* >= 45$

LB: 61, UB: 65


All Free
UB = 65 (Gupta)
LB = 61 J2
61 J2
66 M2
73 M2
71 M2
70 M1
Job 1 First
Job 2 First
Job 3 First
Job 4 First
Job 5 First
64 M3
65 M3
72 M3
70 M1
Job 2 Second
Job 3 Second
Job 4 Second
Job 5 Second
64 M3
66 M3
70 M1
Job 3 Third
Job 4 Third
Job 5 Third
65 M2
70 M1
Job 4 Fourth
Job 5 Fourth
Solution (=LB): 65
Job 5 Fifth

Operations Scheduling

1st level:J2 at first place: $H_1 = 10$, $H_2 = 22$, $H_3 = 40$

U = {1, 3, 4, 5}

M1: $C_{max}^* >= 59$

M2: $C_{max}^* >= 66$, which is greater than the upper bound; thus we fathom the node;

J3, J4 and J5 at first place: we can fathom all of them;

2nd level: Consider Job 3: $H_1 = 18$, $H_2 = 27$, $H_3 = 40$, U = {2, 4, 5}

M1: $C_{max}^* >= 59$

M2: $C_{max}^* >= 62$

M3: $C_{max}^* >= 65$, so we fathom the job; only job 2 remains unfathomed;

3rd level: Job 3: $H_1 = 28$, $H_2 = 37$, $H_3 = 57$, U = { 4, 5}

M1: $C_{max}^* >= 59$

M2: $C_{max}^* >= 61$

M3: $C_{max}^* >= 64$

Machine-bounds did not fathom the node; so we have to calculate job-based bounds:

J4: $C_{max}^* >= 64$

J5: $C_{max}^* >= 49$

➔ best bound = 64; thus create nodes for J4 and J5

4th level: nodes J4 and J5 of level 3 will be fathomed; thus the algorithm is complete:

1-2-3-4-5 with a makespan of 65;

Operations Scheduling

- **Job Shops**
  - different routings for different jobs
  - precedence constraints
  - $(n!)^m$ possible schedules

Operations Scheduling

- **Two machine job shops**
  - Jackson (1956)
  - minimize makespan
    - Machine A: {AB}, {A}, {BA}
    - Machine B: {BA}, {B}, {A,B}

Jackson's algorithm

Job	1	2	3	4	5	6	7	8	9	10
Route	BA	AB	BA	B	A	AB	B	BA	BA	AB
p(i)1	3	1	11	0	3	9	0	8	13	2
p(i)2	8	10	13	1	0	8	6	10	6	6

Find a schedule that would finish all jobs as soon as possible!

Solution:
{A} = {5}, {B} = {4,7}, {AB} = {2, 6, 10} and {BA} = {1, 3, 8, 9}

Operations Scheduling

Johnson's algorithm for {AB}:

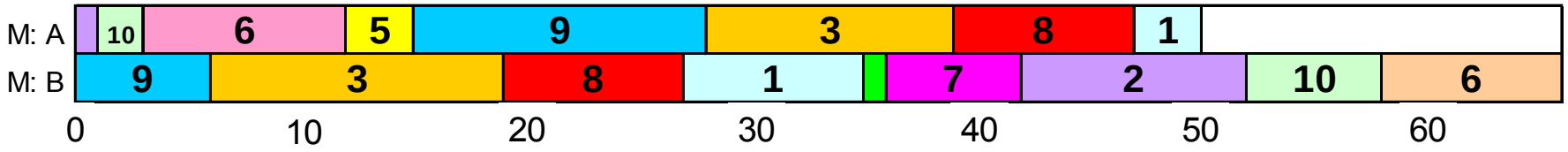
Job	2	10	6
p(i)1	1	2	9
p(i)2	10	6	8

Johnson's algorithm (reversed) for {BA}:

Job	9	3	8	1
p(i)1	13	11	8	3
p(i)2	6	13	10	8

sequence for A: 2-10-6-5-9-3-8-1
sequence for B: 9-3-8-1-4-7-2-10-6

makespan:67



Operations Scheduling

- **Dispatching**
  - job shop scheduling
  - dispatching rules
  - Basic idea:
    - schedule an operation of a job as soon as possible
    - if more than one job is waiting to be processed by the same machine schedule the one with best priority
  - Define:
    - A= set of idle machines
    - J_k= the index of the last job scheduled on machine k
    - U_k= the set of jobs that can be processed on machine k
    - H_k = the completion time of the job currently processed on machine k
    - u_{it} = the priority of job i at time t

Operations Scheduling

- Step 0. Initialize: t=0; H_k=0,k=1,2,…,m; A={1,2,…,m}; U_k={i|operation 1 of i is on machine k, i=1,2,…,n}; s_{ij}=c_{ij}=0. Go to step 4.
- Step 1. Increment t;

  Let

  $$t = \min_{k=1,m;k\notin A} H_k, \text{and } K = \{k \mid H_k = t\}$$

- Step 2. Find the job or jobs that complete at time t and the machine released. Set A = A∪K.
- Step 3. Determine the jobs ready to be scheduled on each machine; Let U_k={i|job i uses machine k and all operations of job i before machine k are completed}, k=1,2,…,m.
  If U_k=0 for k=1,2,…,m,Stop.
  If U_k=0 for k∈A, go to Step 1.

Operations Scheduling

- Step 4. For each idle machine try to schedule a job; for each k $\in$ A with Uk$\neq$0,

let i^* be the job with the best priority : $u_{i^*t} = \min_{i \in U_k} u_{it}$

Schedule job i * on machine k

Set $J_k = i, s_{i^*k} = t, c_{i^*k} = t + p_{i^* j(k)}, H_k = c_{i^*k}$

Remove the scheduled job from U_k

$U_k \leftarrow U_k - \{i\}$

and the machine from A

$A \leftarrow A - \{k\}$

Go to Step 1

Operations Scheduling

- **Many priority measures possible:**
  - SPT
  - FCFS
  - MWKR (most work remaining)
  - EDD
  - EDD/OP
  - SLACK, SLACK/OP
  - Critical ratio: slack/remaining time
  - ...

Operations Scheduling

Quick Closures: job-shop dispatch heuristic

Quick Closure has four machines in the shop: (1) brake, (2) emboss, (3) drill, (4) mill; The shop has currently orders for six different parts, which use all the four machines, but in a different order.
Processing time:

	Operation			
Job	1	2	3	4
1	6/1	8/2	13/3	5/4
2	4/1	1/2	4/3	3/4
3	3/4	8/2	6/1	4/3
4	5/2	10/1	15/3	4/4
5	3/1	4/2	6/4	4/3
6	4/3	2/1	4/2	5/4

Finish all six parts as soon as possible!

Solution: We use a dispatch procedure with MWKR as the priority.

Operations Scheduling

Step1 t = 0, $H_1 = H_2 = H_3 = H_4 = 0$, A = {1, 2, 3, 4}, U_1 = {1, 2, 5}, U_2 = {4}, U_3 = {6}, U_4 = {3}; $s_{ij} = c_{ij} = 0$, i = 1, 2, 3, 4, 5, 6; and j = 1, 2,3, 4; Go to step 4

Step 4 u_{10} = -(6+8+13+5) = -32, u_{20} = -12, u_{50} = -17; thus $s_{11} = 0$, $c_{11} = 0 + 6 = 6$, $H_1 = 6$.
Remove job 1 from U_1, U_1 = {2, 5} and machine 1 from A, A = {2, 3, 4}.
Set k = 2; there is only one job in U_2 so we schedule it on machine 2; i* = 4, $s_{41} = 0$, $c_{41} = 5$, $H_2 = 5$, U_2 = { }, and A = {3, 4}.
We schedule J6 and J3 on M3 and M4 (tab: t = 0 row). Go to step 1.

Step 1 t = $\min_{k=1,m:k\varepsilon A} H_k = \min\{6, 5, 4, 3\} = 3$, and K = $\{k \backslash H_k = 3\}$ = {4}; H_k *min* is bold in the table;

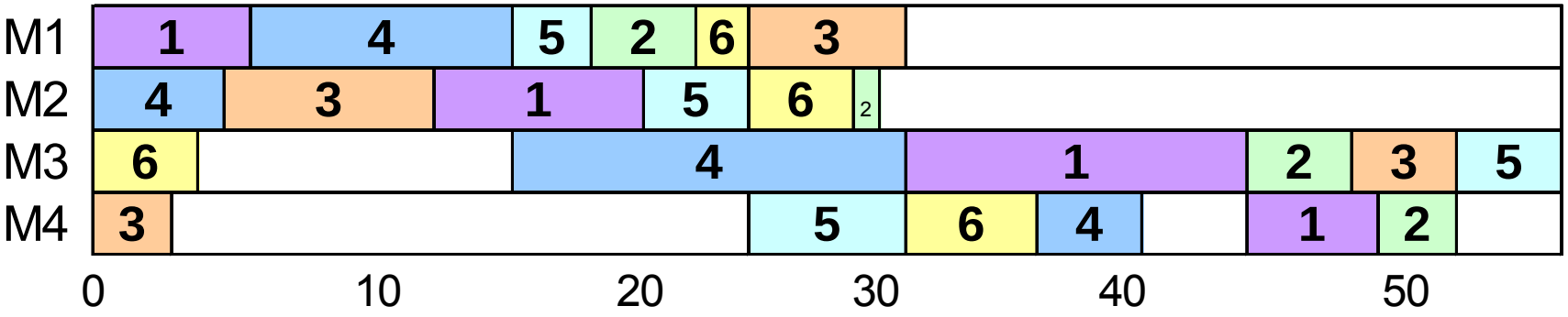
Step 2 J3 completes at time 3 on M4, so $i^3 = \{i \backslash J_k = i, k \varepsilon K\} = \{3\}$, K = {4},
and A = {} U {4} = {4}, (tab: t = 3 row)

Step 3 U1 = {2, 5}, U2 = {3}, U3 = U4 = { }; Since no jobs are waiting for M4, no jobs can be scheduled to start at time 3; go to step 1 etc.

Operations Scheduling

t	it	K	A	U1	U2	U3	U4	H1	H2	H3	H4
0	-	-	1,2,3,4	**1**,2,5	**4**	**6**	**3**	6	5	4	3
3	3	4	4	2,5	3			6	5	**4**	
4	6	3	3,4	2,5,6	3			6	**5**		
5	4	2	2,3,4	2,4,5,6	**3**			**6**	13		
6	1	1	1,3,4	2,**4**,5,6	1			16	**13**		
13	3	2	2,3,4	2,3,5,6	**1**			**16**	21		
16	4	1	1,3,4	2,3,**5**,6		**4**		**19**	21	31	
19	5	1	1,4	**2**,3,6	5			23	**21**	31	
21	1	2	2,4	3,6	**5**	1		**23**	25	31	
23	2	1	1,4	3,**6**	2	1		**25**	**25**	31	
25	6,5	1,2	1,2,4	**3**	2,**6**	1	**5**	31	**29**	31	31
29	6	2	2		**2**	1	6	31	**30**	31	31
30	2	2	2			1,2	6	**31**		**31**	**31**
31	3,4	1,3	1,2,3			**1**,2,3	4,**6**			44	36
36	6	4	1,2,4			2,3,5	**4**			44	**40**
40	4	4	1,2,4			2,3,5				**44**	
44	1	3	1,2,3,4			**2**,3,5	**1**			**48**	49
48	2	3	1,2,3			**3**,5	2			52	**49**
49	1	4	1,2,4			5	**2**			**52**	**52**
52	3,2	3,4	1,2,3,4			**5**				**56**	
56	5	3	1,2,3,4								

Operations Scheduling



Chapter 10

Section 5.5:
Bottleneck Scheduling

Operations Scheduling

- **Shifting Bottleneck Procedure**

- **heuristic to minimize makespan for multiple machine job shops**

- **Main idea:**

  1. for each job on each machine calculate the minimal amount of time needed before and after the processing of this job

     ⟶ generates minimal makespan problem with release times and tails

  2. for each machine solve this problem for each machine (e.g. Schrage heuristic) and determine the machine with the maximal makespan (bottleneck machine)

  3. Fix the found sequence on the bottleneck machine, update release times and tails on the remaining machines and repeat 2. for the remaining machines until schedules for all machines have been determined

Operations Scheduling

- **Shifting Bottleneck Procedure Example:**
- 3 machines (M1, M2, M3), 3 jobs (1,2,3)
- **Job routings:** 1: M1-M2-M3

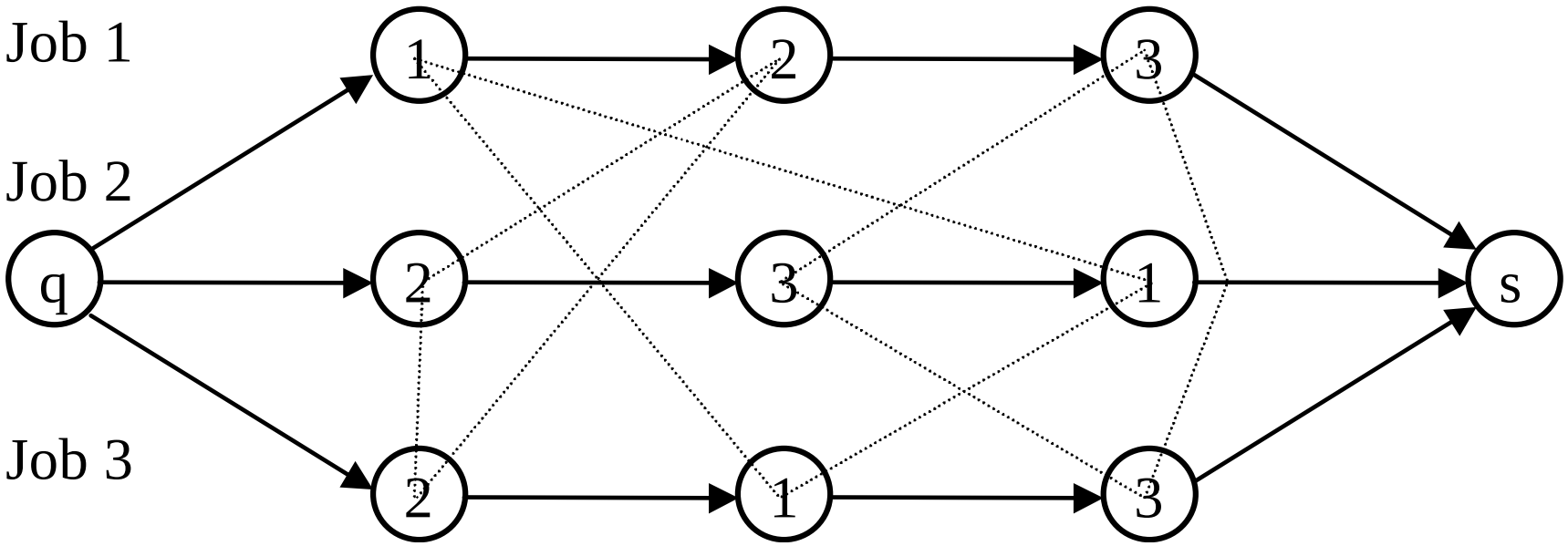
  2: M2-M3-M1

  3: M2-M1-M3
- **Processing times:**

p_{ik}	M1	M2	M3
1	3	3	2
2	3	2	3
3	3	4	1

Operations Scheduling

- **Machine-Flow-Graph:**



Operations Scheduling

- **Problems with release times and tails for each machine:**
- **M1:**

	1	2	3
r_j	0	5	4
p_j	3	3	3
n_j	5	0	1

M2:

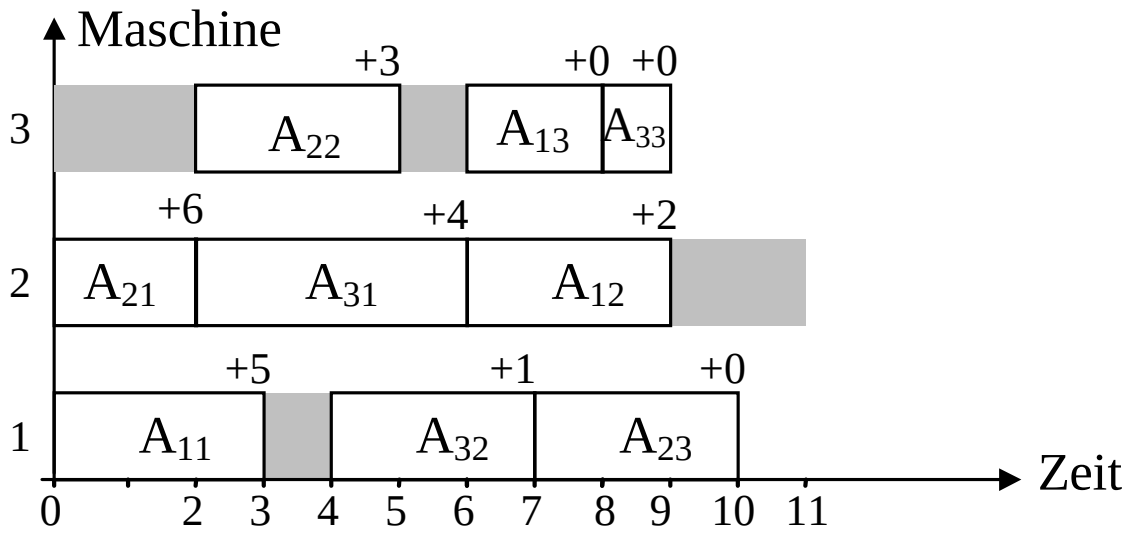
	1	2	3
r_j	3	0	0
p_j	3	2	4
n_j	2	6	4

M3:

	1	2	3
r_j	6	2	7
p_j	2	3	1
n_j	0	3	0

Operations Scheduling

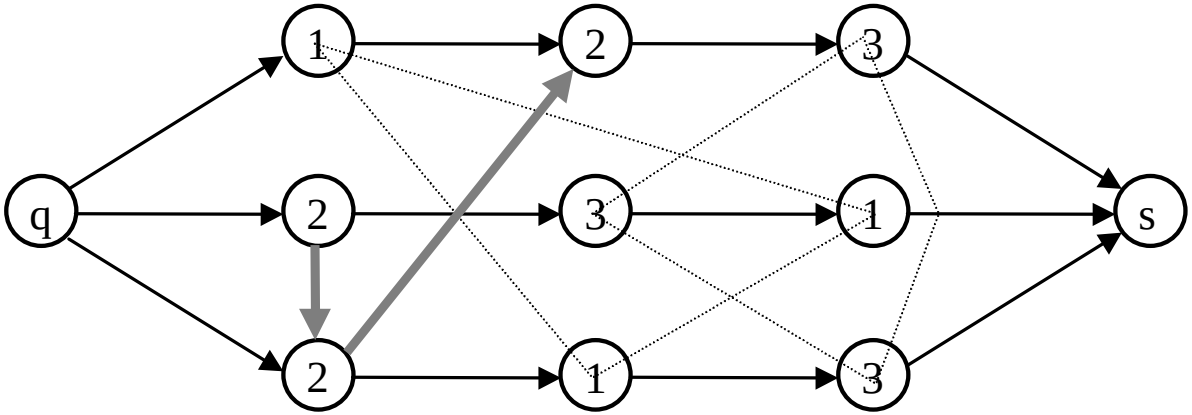
- **Schrage heuristic gives the following solutions for the three machines:**



- ⟹ machine 2 is bottleneck with $C_2 = 11$
- ⟹ fix sequence on machine 2

Operations Scheduling

- **Machine-flow-graph:**



- **Update release times and tails on M1 and M3:**
- **M1:**

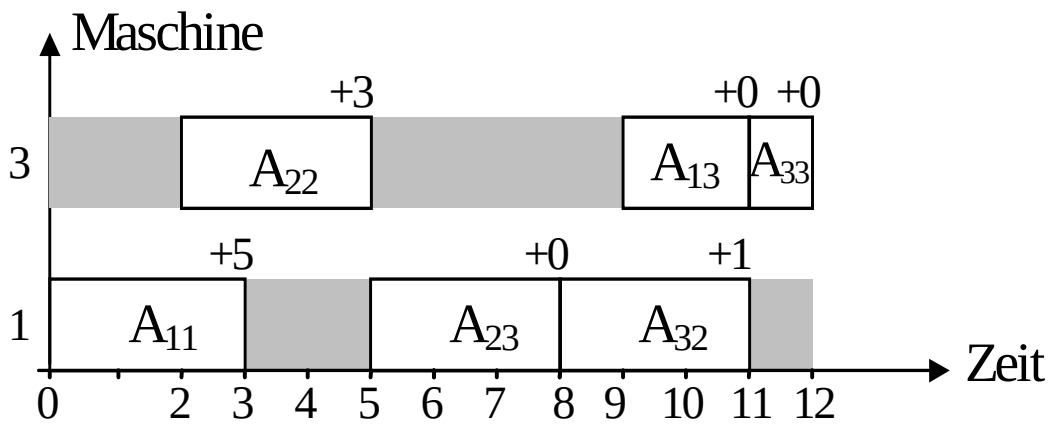
	1	2	3
r_j	0	5	**6**
p_j	3	3	3
n_j	5	0	1

M3:

	1	2	3
r_j	**9**	2	**9**
p_j	2	3	1
n_j	0	3	0

Operations Scheduling

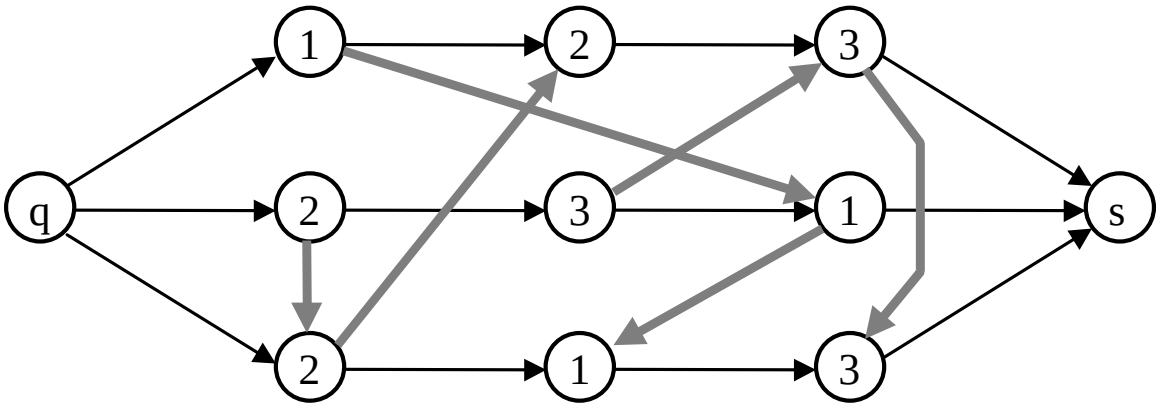
- **Schrage heuristic for M1, M3:**



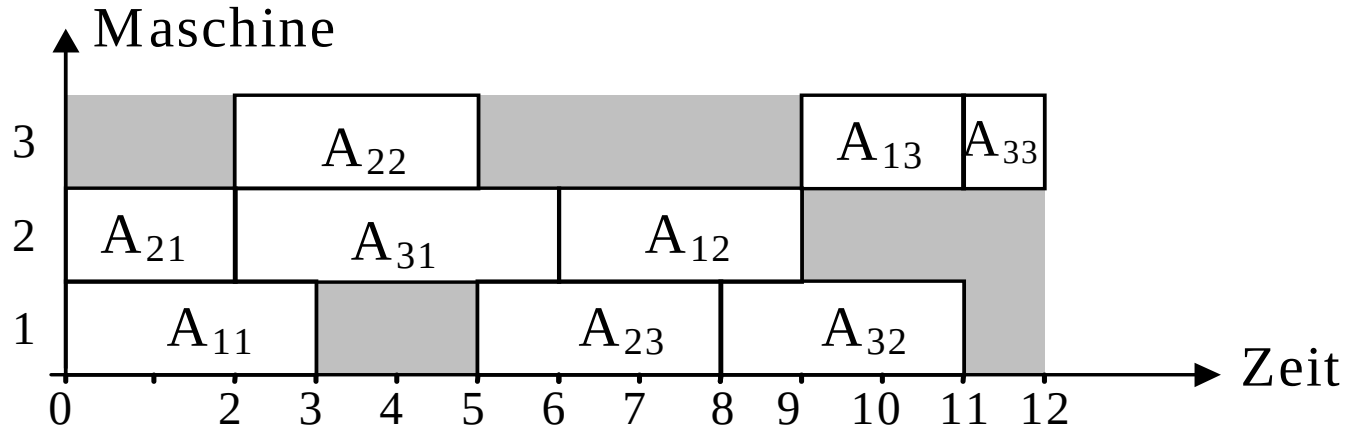
- ⟶ both machines could be considered the bottleneck with C=12, fix sequence on M1

Operations Scheduling

- **Updated machine-flow-graph:**



- **update relase time and tails and apply Schrage to M3. This gives**



- **with C_{max} = 12**

Operations Scheduling

- **Finite Capacity Scheduling**
  - MRP systems generally assume constant lead times, ignore setups
  - MRP plans might be unrealistic
  - Traditionally problem hiden by inventory and excess capacity
  - Reducing Inventory and capacity makes finite capacity scheduling crucial
  - Computer-assisted finitie capacity scheduling systems rather than manual scheduling by foreman

Operations Scheduling

- **Work to do: 8.3abcde, 8.4, 8.5, 8.6, 8.10, 8.14, 8.16, 8.18 (with the following due dates: 42, 50, 12, 63, 23, 34, 36, 42, 54, 32) 8.30ab, 8.32abc, 8.36ab, 8.43, 8.44, 8.49ab, 8.51ab, 8.56, 8.57 (apply shifting bottleneck procedure)**

- **Minicase: Ilana Designs**