Contents college 8

Interval Scheduling, Reservation Systems, Timetabling
(Chapter 8, book)

- Reservation Systems without Slack (Interval Scheduling)
  $\rightarrow$ first part lecture
- Reservation Systems with Slack
  $\rightarrow$ read Section 8.3 book!
- Timetabling with Tooling Constraints
  $\rightarrow$ second part lecture
- Timetabling with Resource Constraints
  $\rightarrow$ read Section 8.5 book!

Definition Reservation System

Given:

- m parallel machines
- n jobs with
  - processing times $p_1, \ldots, p_n$
  - release dates $r_1, \ldots, r_n$
  - due dates $d_1, \ldots, d_n$
  - weights $w_1, \ldots, w_n$
- job has to be processed within given time interval
- it may not be possible to process all jobs

Goal: Select a subset of jobs which

- can be scheduled feasible and
- maximizes a given objective

Possible Objectives

- maximize number of jobs processed
- maximize total amount of processing
- maximize profit of jobs processed (here job weights are given)
- ...

Two principle models

1. Systems without slack

   job fills interval between release and due date completely, i.e.

$$p_j = d_j - r_j$$

   Also called fixed interval

2. Systems without slack

   interval between release and due date of a job may have some slack, i.e.

$$p_j \leq d_j - r_j$$

Applications Reservation Systems

- hotel room reservation
- car rental
- reserving machines in a factory
- timetabling (additionally constraints)
- ...

Reservation Systems without Slack (interval scheduling)

- m parallel machines
- n jobs; for job j:
  - release date r_j
  - due date d_j
  - processing time $p_j = d_j - r_j$
  - set M_j of machines on which j may be processed
  - weight w_{ij}: profit of processing j on machine i

Objective: maximize profit of the processed jobs:

- $w_{ij} = 1$: number of jobs processed
- $w_{ij} = w_j$: weighted number of jobs processed

Integer Programming Formulation

Notation and Variables:

- time periods $1, \ldots, H$
- J_l: set of jobs needing processing in period l
- $$x_{ij} = \begin{cases} 1 & \text{job } j \text{ on machine } i \\ 0 & \text{else} \end{cases}$$

Model:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} x_{ij}$$

$$\sum_{i=1}^{m} x_{ij} \leq 1 \quad j = 1, \ldots, n$$

$$\sum_{j \in J_l} x_{ij} \leq 1 \quad i = 1, \ldots, m$$

$$l = 1, \ldots, H$$

$$x_{ij} \in \{0, 1\}$$

Easy Special Cases 1

$p_j = 1$ for all jobs j

- each job is available exactly one time period
- the problem splits into independent problems, one for each time period
- resulting problem for period l:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} x_{ij}$$

$$\sum_{i=1}^{m} x_{ij} \leq 1 \;\; j = 1, \ldots, n$$

$$\sum_{j \in J_l} x_{ij} \leq 1 \;\; i = 1, \ldots, m$$

$$x_{ij} \in \{0, 1\}$$

This is an assignment problem and can be solved efficiently

Easy Special Cases 2

$w_{ij} = 1$ and $M_j = \{1, \ldots, m\}$ for all i, j

Thus, all machines are equal and the goal is to maximize the number of jobs processed

Assume: $r_1 \leq \ldots \leq r_n$

Notation: J: set of already selected jobs for processing; initial: $J = \emptyset$

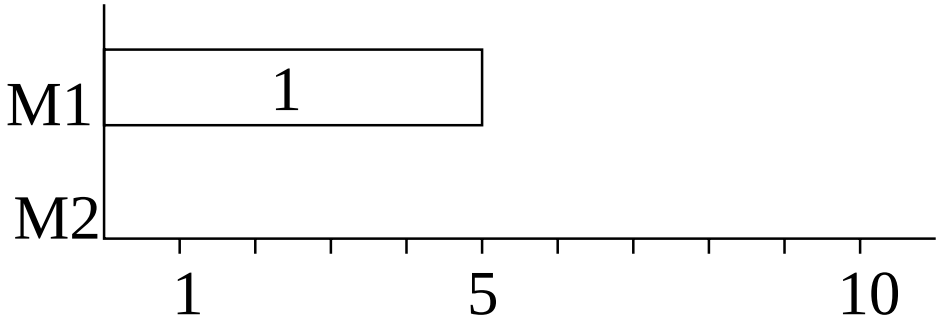
FOR $j = 1$ TO n DO
  IF a machine is available at r_j
  THEN
    assign j to that machine;
    $J := J \cup \{j\}$
  ELSE
    determine j^* such that
    $C_{j^*} = \max_{k \in J} C_k = \max_{k \in J} r_k + p_k$
    IF $C_j = r_j + p_j < C_{j^*}$ THEN
      assign j to machine of j^*;
      $J := J \cup \{j\} \setminus \{j^*\}$

Easy Special Cases 2 - Example-1

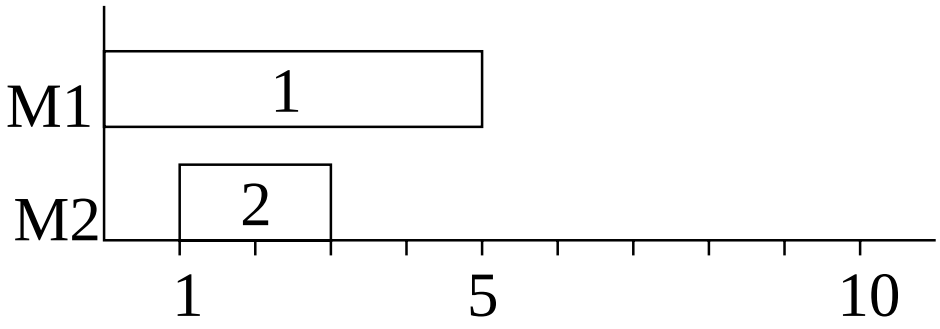
2 machines and 8 jobs

j	1	2	3	4	5	6	7	8
r_j	0	1	1	3	4	5	6	6
d_j	5	3	4	8	6	7	9	8

Iteration 1: $j = 1$



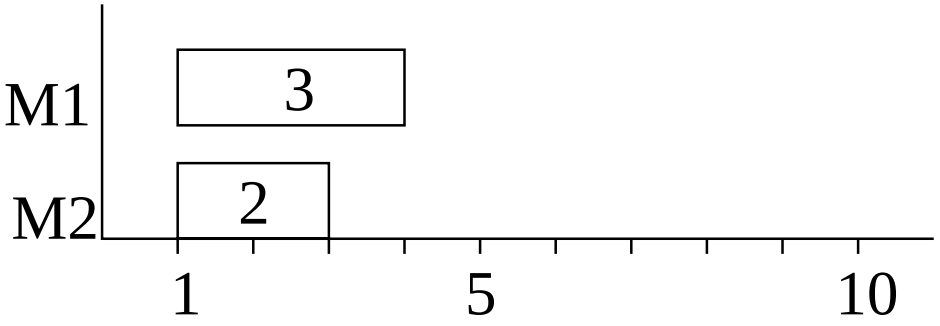
Iteration 2: $j = 2$



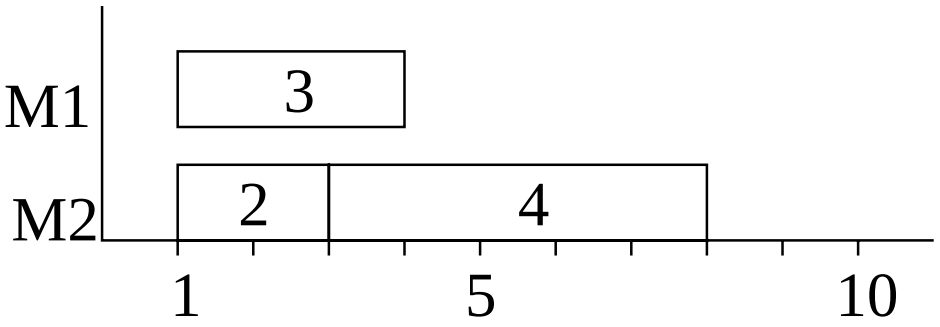
Easy Special Cases 2 - Example-2

j	1	2	3	4	5	6	7	8
r_j	0	1	1	3	4	5	6	6
d_j	5	3	4	8	6	7	9	8

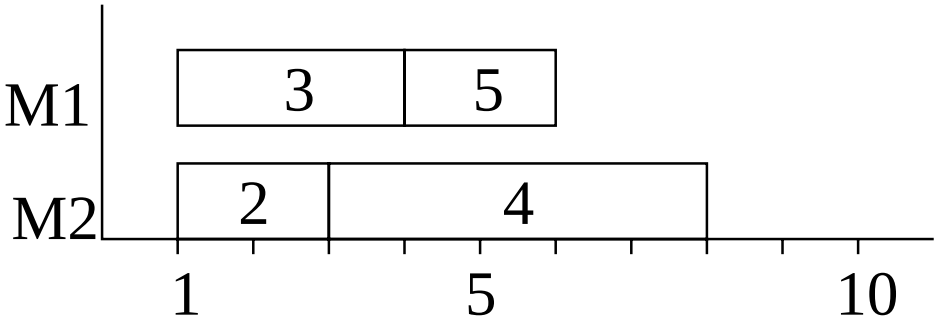
Iteration 3: $j = 3$, $j^* = 1$



Iteration 4: $j = 4$



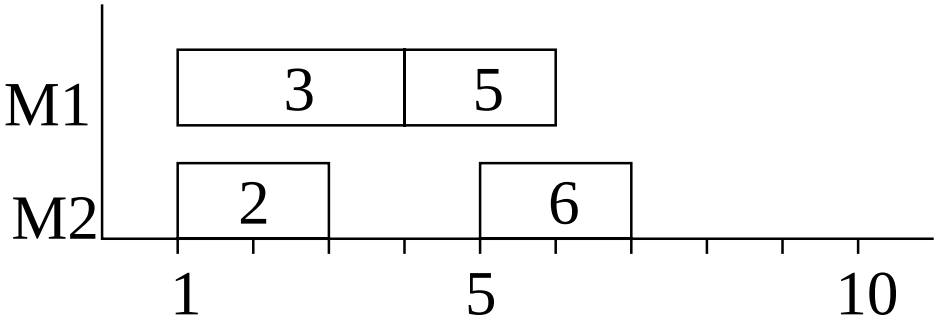
Iteration 5: $j = 5$, $j^* = 4$



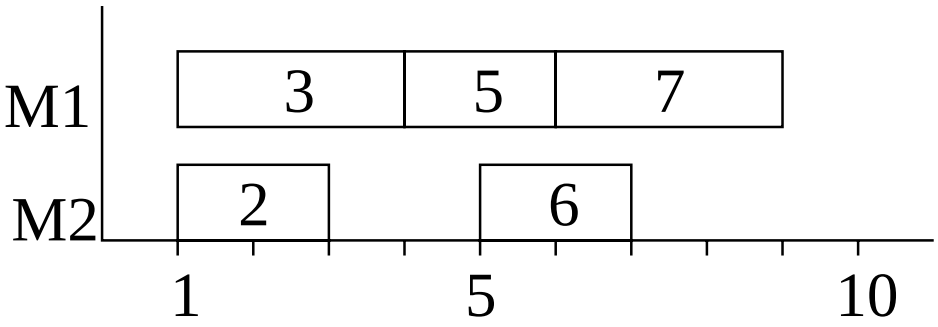
Easy Special Cases 2 - Example-3

j	1	2	3	4	5	6	7	8
r_j	0	1	1	3	4	5	6	6
d_j	5	3	4	8	6	7	9	8

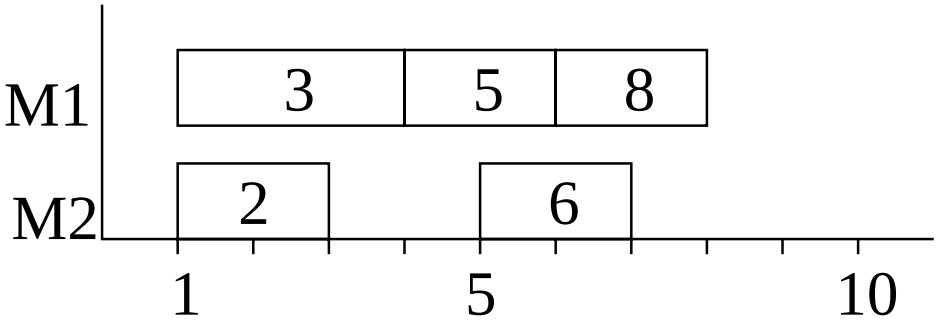
Iteration 6: $j = 6$, $j^* = 4$



Iteration 7: $j = 7$



Iteration 8: $j = 8$, $j^* = 7$



Another Version of the Reservation Problem

- $w_{ij} = 1$ for all i, j
- unlimited number of identical machines
- all jobs have to be processed
- Goal: use a minimum number of machines

Assume: $r_1 \leq \ldots \leq r_n$

Notation: M: set of machines used; initial: $M = \emptyset$

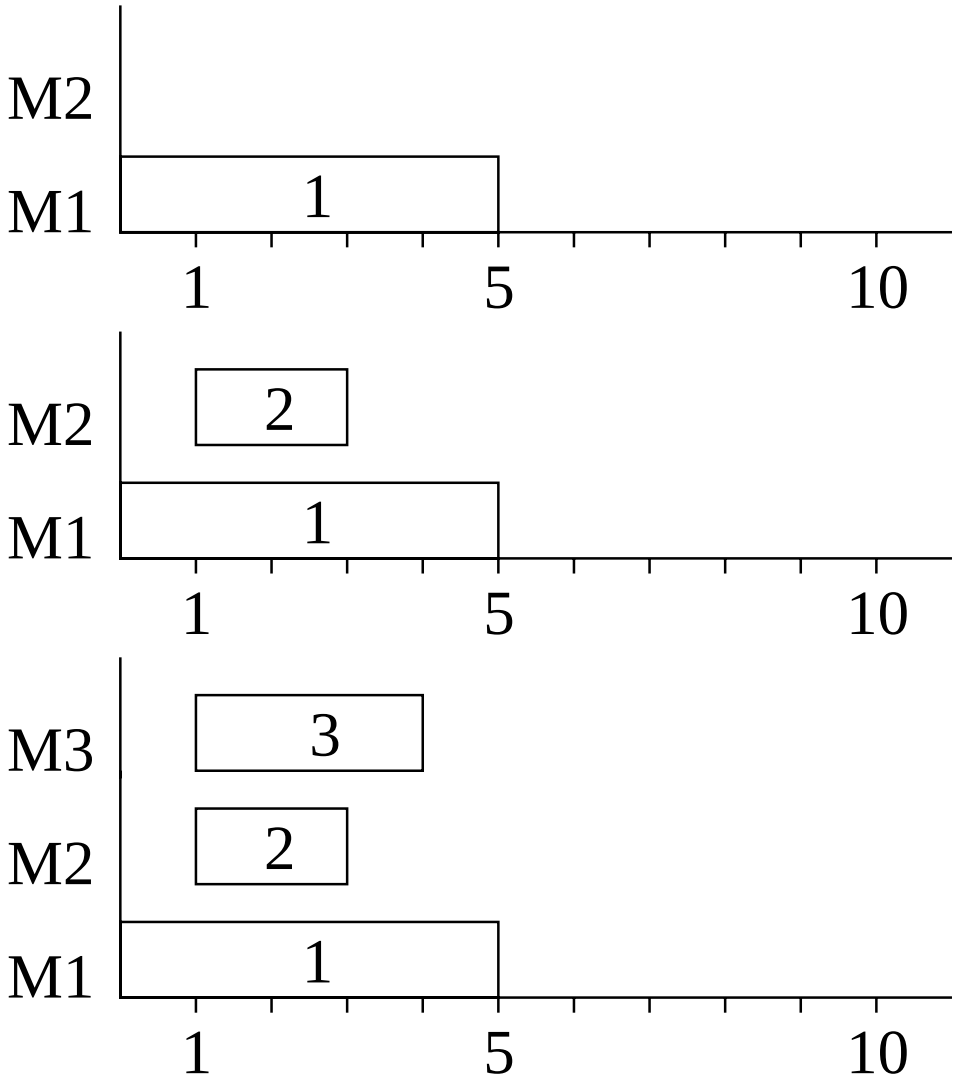
Algorithm for Another Version ...

$i = 0$;
FOR $j = 1$ TO n DO
    IF machine from M is free at r_j
    THEN
      assign j to a free machine
    ELSE
      i:=i+1;
      add machine i to M;
      assign job j to machine i.

Remark: The above algorithm gives the minimal number of machines to process all n jobs.

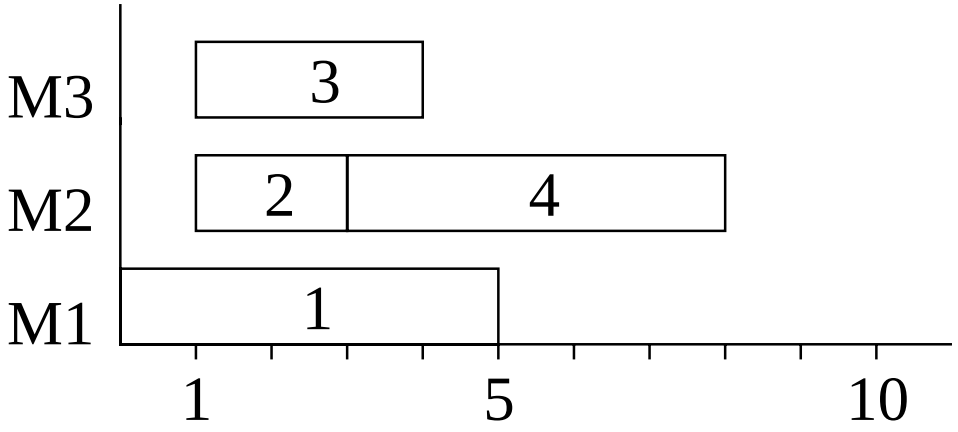
Another Version ... Example-1

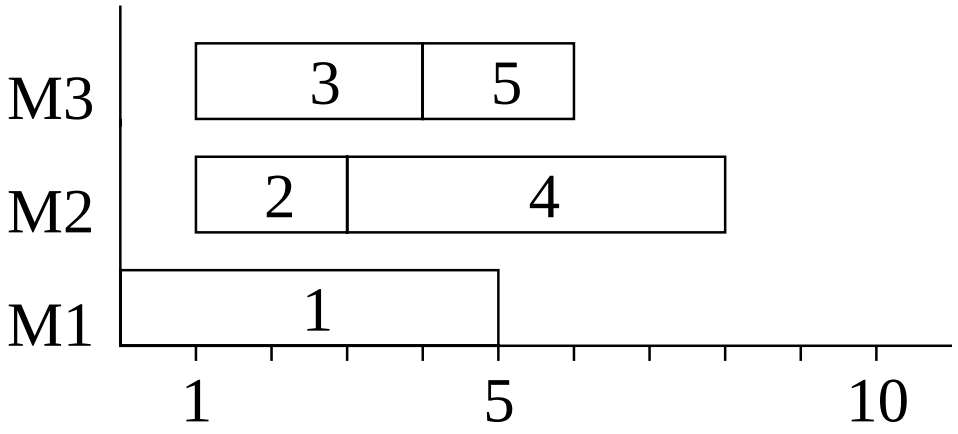
j	1	2	3	4	5	6	7	8
r_j	0	1	1	3	4	5	6	6
d_j	5	3	4	8	6	7	9	8

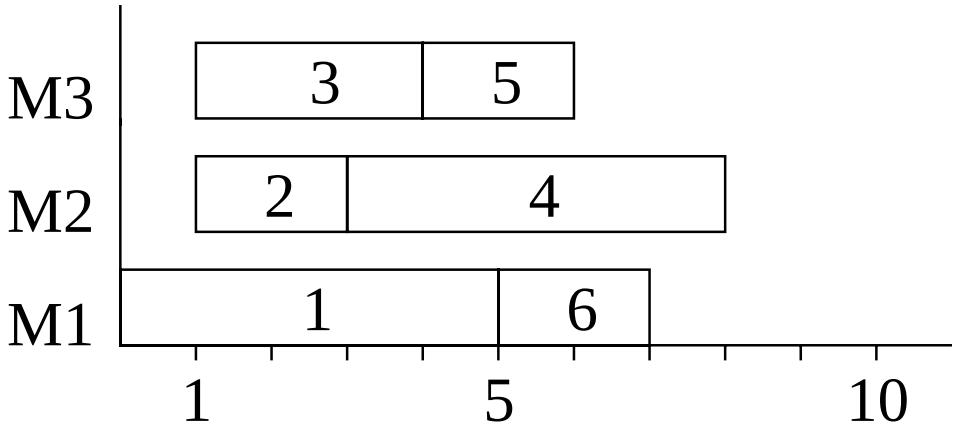


Another Version ... Example-2

j	1	2	3	4	5	6	7	8
r_j	0	1	1	3	4	5	6	6
d_j	5	3	4	8	6	7	9	8

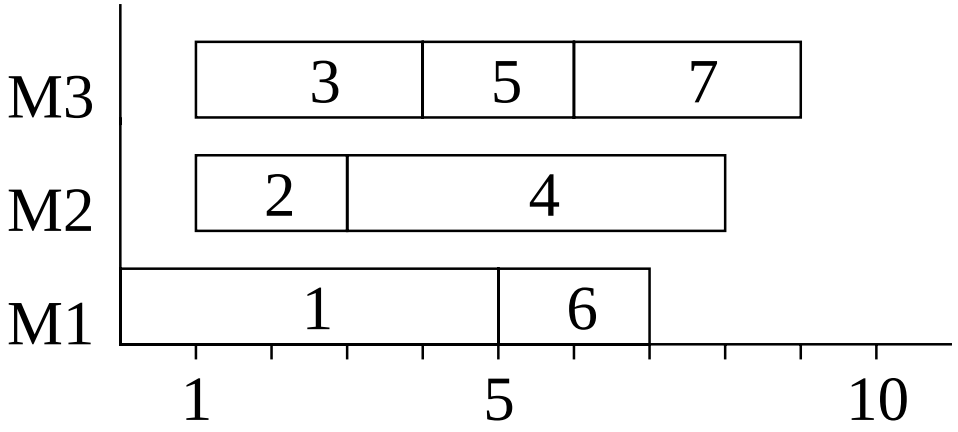


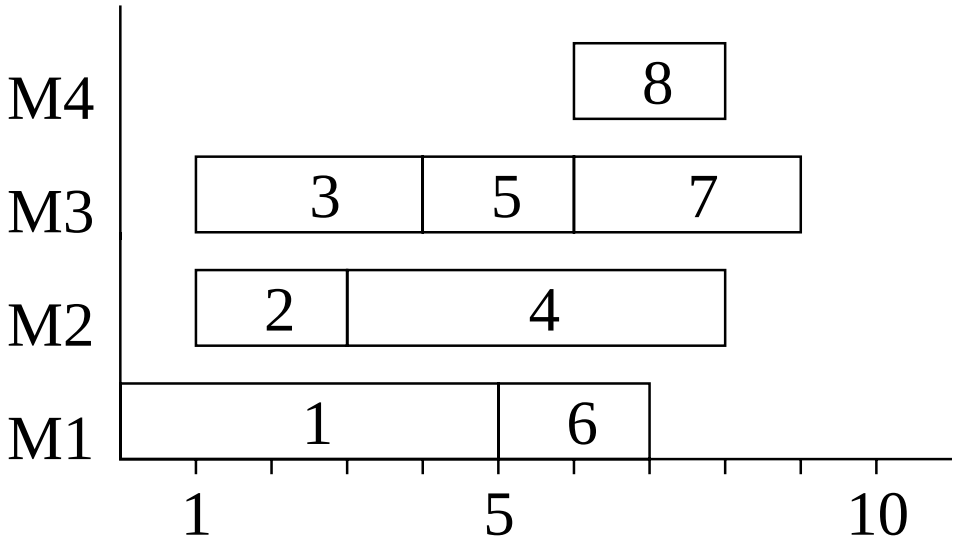




Another Version ... Example-3

j	1	2	3	4	5	6	7	8
r_j	0	1	1	3	4	5	6	6
d_j	5	3	4	8	6	7	9	8





Reformulation Another Version ...

The problem can be reformulated as a Graph Coloring problem

- n nodes (node $j \leftrightarrow$ job j)
- arc (j, k) if job j and k overlap
- assign a color to each node such that two nodes connected by an arc have different colors
- Goal: find a coloring with a minimal number of colors

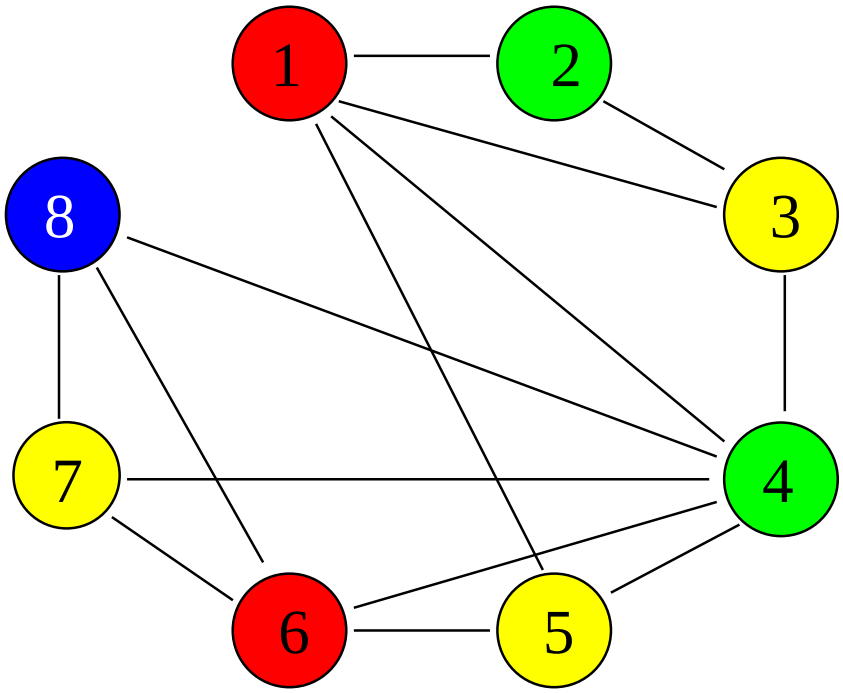
Remarks:

- jobs which overlap have to be on different machines,
  nodes connected by an arc have different colors,
  $\rightarrow$ each color corresponds to a machine
- graph coloring is NP-hard

Reformulation Example

j	1	2	3	4	5	6	7	8
r_j	0	1	1	3	4	5	6	6
d_j	5	3	4	8	6	7	9	8

corresponding graph coloring problem:



Timetabling with Tooling Constraints

- unlimited number of identical parallel machines
- n jobs with processing times $p_1, \ldots, p_n$, which can be processed at any time
- set T of tools
- job j needs a subset $T_j \subset T$ of tools for its processing
- jobs needing the same tool can not be processed in parallel

Objectives:

- Feasibility Version:
  find a schedule or timetable completing all jobs within a given time horizon H
- Optimization Version:
  find a schedule or timetable for all jobs with a minimal makespan

Timetabling with Tooling Constraints - cont

Remarks:

- the problem is a special case of the resource-constraint project scheduling problem (no precedences and all $R_i = 1$)
- even for $p_j = 1$ for all j no efficient optimal algorithms exist

Timetabling ... Special Case - 1

- feasibility version
- $p_j = 1$ for all j

Observation: The problem can be reformulated as a graph coloring problem in a similar way as for a special version of the interval scheduling problem!

- n nodes (node $j \leftrightarrow$ job j)
- arc (j,k) if job j and k require the same tool
- Question: Can the graph be colored with H different colors? (color $\leftrightarrow$ timeslot)

Remark: The associated optimization problem (using a minimal number of machines) is known as the chromatic number problem

Timetabling ... Special Case - 2

Remark: Even though the considered interval scheduling problem and the considered timetabling problem reduce to the same graph coloring problem, the timetabling problem with tooling constraints is harder!

Reason: For the interval scheduling problem the 'used resources' (time slots) are adjacent, whereas the tools may not be ordered in such a way

Remark: The graph resulting from the interval scheduling problem is a so called 'interval graph'

Heuristic Algorithm Special Case

Notation:

- degree of a node: number of adjacent arcs
- saturation level of a node: number of different colored nodes already connected to it in a partial coloring

Sort nodes in decreasing order of degrees;

Color a node of maximal degree with color 1;

WHILE nodes are uncolored DO

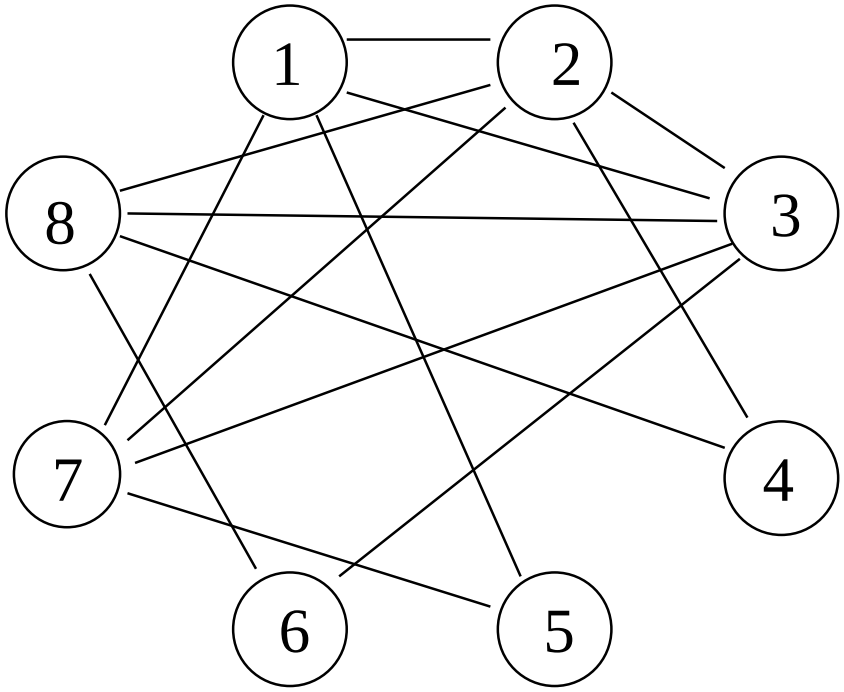
Chose an uncolored node with maximal saturation level;
(if tie, choose any with maximal degree in the uncolored subgraph);

Color selected node with color with lowest possible number;

Algorithm Example

Jobs	1	2	3	4	5	6	7	8
p_j	1	1	1	1	1	1	1	1
Tool 1	1	1	1	0	0	0	1	0
Tool 2	0	1	0	1	0	0	0	1
Tool 3	1	0	0	0	1	0	1	0
Tool 4	0	0	1	0	0	1	0	1
Tool 5	0	0	0	1	0	0	0	0

Corresponding Graph



Preprocessing:

Jobs(nodes)	1	2	3	4	5	6	7	8
degree	4	5	5	2	2	2	4	4

Algorithm Example cont. 1

Initial: choose node 2 and color it red (color 1)

Iteration 1: all neighbors of 2 (1,3,4, 7,8) have saturation level 1;

3 has highest degree

color node 3 green (color 2)

Iteration 2: 1,7,8 have max.sat. level;

all have degree 2;

choose node 1 and color it yellow (color 3)

Iteration 3: 7 has max.sat. level;

color node 7 blue (color 4)

Iteration 4: 5,8 have max.sat. level;

8 has highest degree

color node 8 yellow (color 3)

Algorithm Example cont. 2

Iteration 5: 4,5,6 have max.sat. level;

all have degree 0;

choose node 4 and color it green (color 2)

Iteration 6: 5,6 have max.sat. level;

both have degree 0;

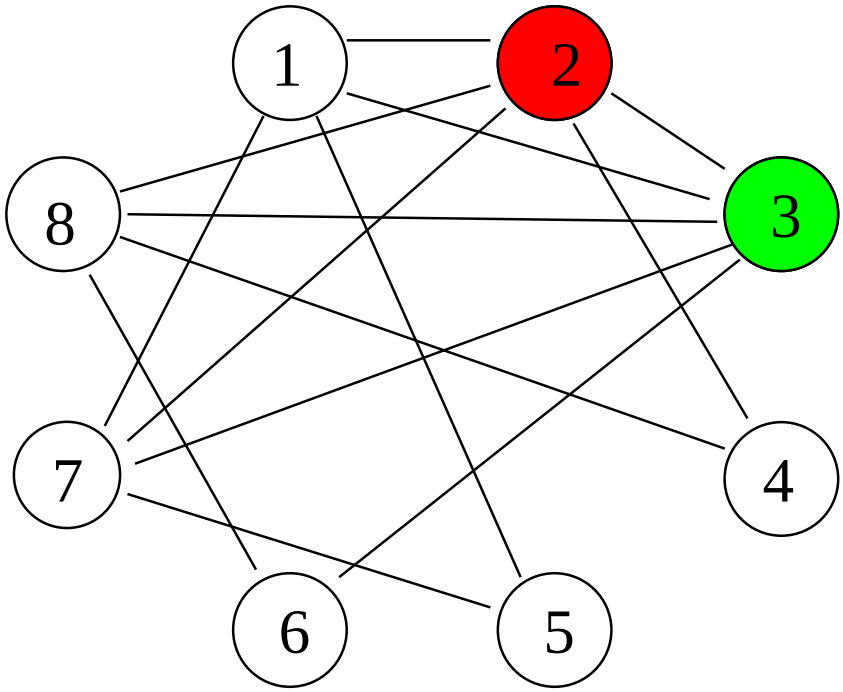
choose node 5 and color it red (color 1)

Iteration 7: only node 6 is left;

color node 6 red (color 1)

Algorithm Example cont. 3

Graph after Iteration 1

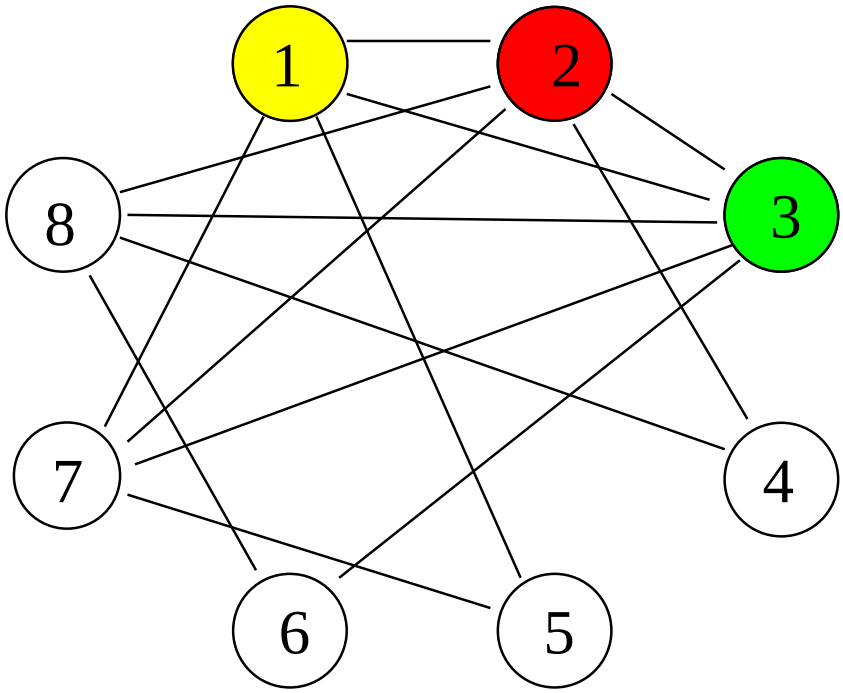


Data after Iteration 1:

Jobs(nodes)	1	2	3	4	5	6	7	8
saturation level	2	-	-	1	0	1	2	2
degree	2	-	-	1	2	1	2	2

Algorithm Example cont. 4

Graph after Iteration 2

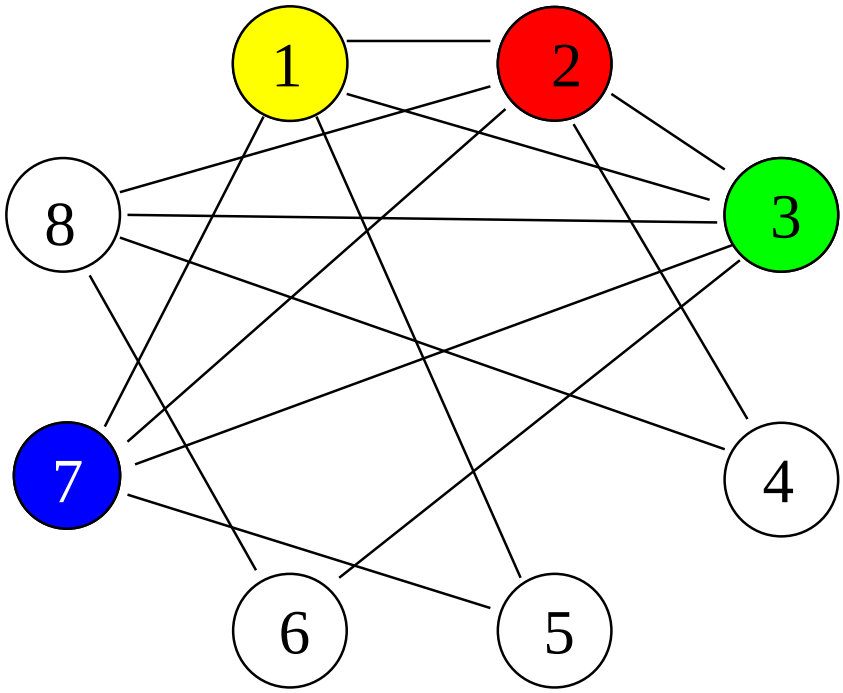


Data after Iteration 2:

Jobs(nodes)	1	2	3	4	5	6	7	8
saturation level	-	-	-	1	1	1	3	2
degree	-	-	-	1	1	1	1	2

Algorithm Example cont. 5

Graph after Iteration 3

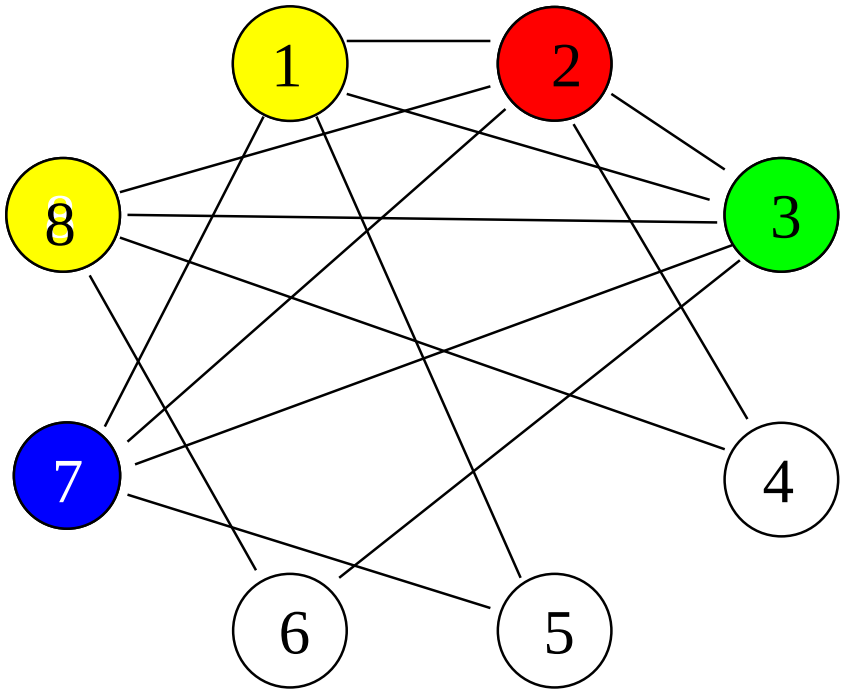


Data after Iteration 3:

Jobs(nodes)	1	2	3	4	5	6	7	8
saturation level	-	-	-	1	2	1	-	2
degree	-	-	-	1	0	1	-	2

Algorithm Example cont. 6

Graph after Iteration 4

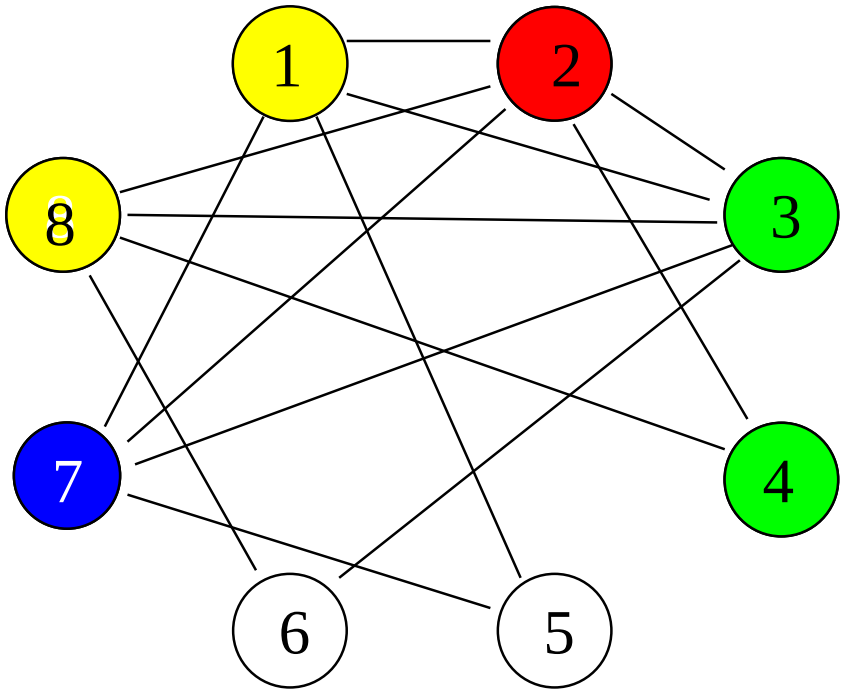


Data after Iteration 4:

Jobs(nodes)	1	2	3	4	5	6	7	8
saturation level	-	-	-	2	2	2	-	-
degree	-	-	-	0	0	0	-	-

Algorithm Example cont. 7

Graph after Iteration 5

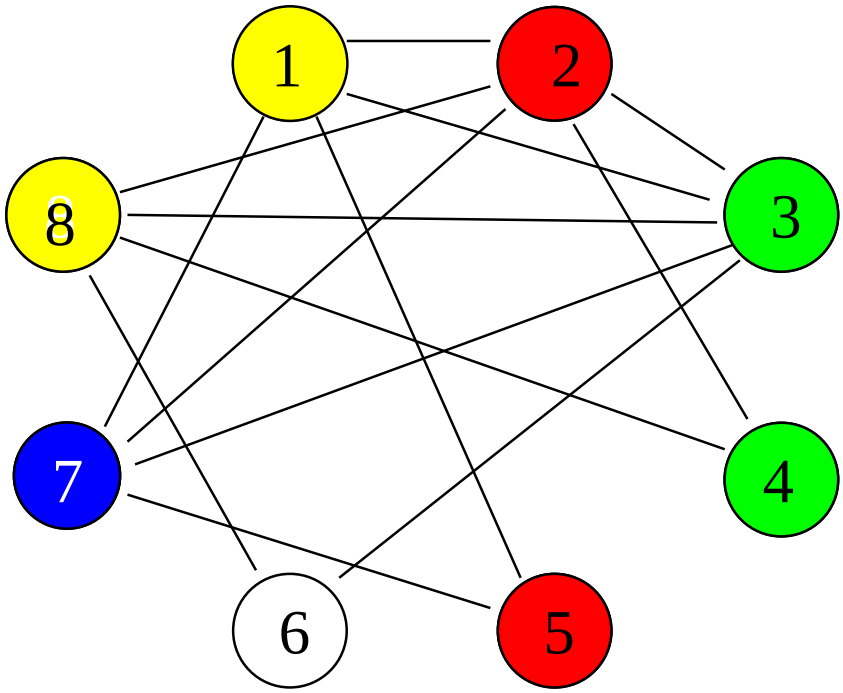


Data after Iteration 5:

Jobs(nodes)	1	2	3	4	5	6	7	8
saturation level	-	-	-	-	2	2	-	-
degree	-	-	-	-	0	0	-	-

Algorithm Example cont. 8

Graph after Iteration 6

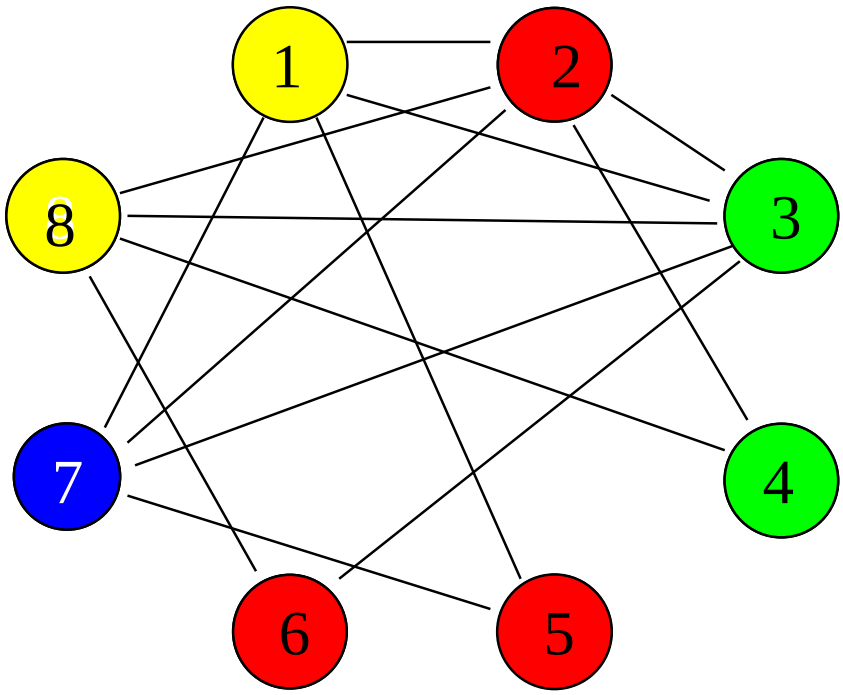


Data after Iteration 6:

Jobs(nodes)	1	2	3	4	5	6	7	8
saturation level	-	-	-	-	-	2	-	-
degree	-	-	-	-	-	0	-	-

Algorithm Example cont. 9

Final Graph



Solution:

jobs 2, 5, and 6 at time 1
jobs 3 and 4 at time 2
jobs 1 and 8 at time 3
job 7 at time 4

Relation to Interval Scheduling

Remark: For the given example the tools can not be ordered such that for all jobs the used tools are adjacent (i.e. the resulting graph is not an interval graph). Thus the instance can not be seen as an interval scheduling instance.

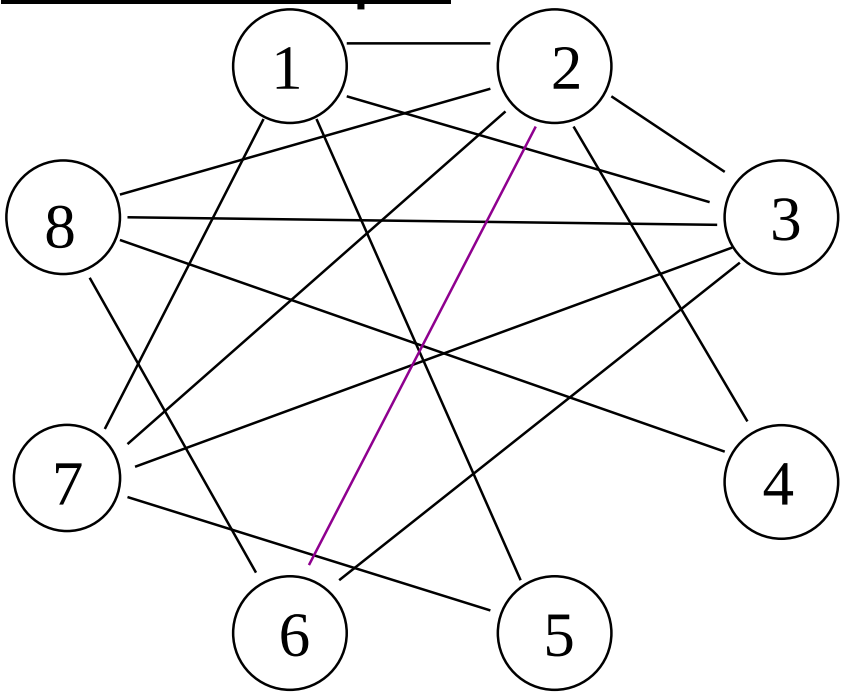
Change of the data:
assume job 2 needs besides tool 1 and 2 also tool 4

New data:

Jobs	1	2	3	4	5	6	7	8
p_j	1	1	1	1	1	1	1	1
Tool 1	1	1	1	0	0	0	1	0
Tool 2	0	1	0	1	0	0	0	1
Tool 3	1	0	0	0	1	0	1	0
Tool 4	0	1	1	0	0	1	0	1
Tool 5	0	0	0	1	0	0	0	0

Relation to Interval Sched. cont. 1

New Graph:



Toll renumbering:

Jobs	1	2	3	4	5	6	7	8
p_j	1	1	1	1	1	1	1	1
Tool 3	1	0	0	0	1	0	1	0
Tool 1	1	1	1	0	0	0	1	0
Tool 4	0	1	1	0	0	1	0	1
Tool 2	0	1	0	1	0	0	0	1
Tool 5	0	0	0	1	0	0	0	0

Relation to Interval Sched. cont. 1

Transform

time 1	tool 3
time 2	tool 1
time 3	tool 4
time 4	tool 2
time 5	tool 5

Resulting Interval Scheduling Prob.:

Job	1	2	3	4	5	6	7	8
r_j	0	1	1	3	0	2	0	2
d_j	2	4	3	5	1	3	2	4
p_j	2	3	2	2	1	1	2	2

Opgaven voor werkcollege

Date: Time:	Thursday, 07 June, 2001 13.45-15.30 (5+6)
Room:	CC 4
Exercises: College 8	 8.1 (a)+(c), 8.9 (a)