Scheduling

Interval Scheduling, Reservations, and Timetabling

Tim Nieberg

Service Models

- activities, which are restricted by time windows, have to be assigned to resources
- often activities use several different resources in parallel
- the availability of resources may vary over time
- it may even be possible to influence the availability of resources for a certain cost
- a nice three field notation as for the manufacturing models does not exist, since the problems are more diverse

Service Models

Characteristics

- n activities/jobs with
  - processing times $p_1, \ldots, p_n$
  - release dates $r_1, \ldots, r_n$
  - due dates $d_1, \ldots, d_n$
  - weights $w_1, \ldots, w_n$
- m resources/machines with
  - time dependent availability
  - properties which allow only certain subsets of jobs to be processed on certain machines
  - possibility to extend resource availability for a certain price
  - ...

Service Models

Possible Objectives

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

Service Models

Areas of Application

- Reservation systems
- Timetabling
- Scheduling and timetabling in sport and entertainment
- Planning, scheduling and timetabling in transportation
- Workforce scheduling

Interval Scheduling, Reservation Systems

Definition Reservation System

- Given:
  - m parallel machines
  - n jobs
- 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

Interval Scheduling, Reservation Systems

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 with slack

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

   $$p_j \leq d_j - r_j$$

Interval Scheduling, Reservation Systems

Applications Reservation Systems

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

Reservation Systems with Slack

Relation with (Classical) Scheduling

- the reservation problem with slack is related to problem $Pm|r_j|L_{max}$ and problem $Pm|r_j|\sum w_j U_j$:
  - for problem $Pm|r_j|L_{max}$ a solution with $L_{max} \leq 0$ corresponds to a solution of the reservation problem with profit $= \sum_{j=1}^{n} w_j$
  - for problem $Pm|r_j|\sum w_j U_j$ a solution with $\sum w_j U_j = C$ corresponds to a solution of the reservation problem with profit $= \sum_{j=1}^{n} w_j - C$
- since $1|r_j|L_{max}$ is NP-hard in the strong sense, the reservation problem is also NP-hard in the strong sense
- due to this relation, we will not consider this type

Reservation Systems without Slack (interval scheduling)

Notations and Definition

- 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

Reservation Systems without Slack

Integer Programming Formulation - Notation and Variables

- time periods $1, \dots, H$
- J_l: set of jobs needing processing in period l
- variables x_{ij}:

$$x_{ij} = \begin{cases} 1 & \text{job } j \text{ on machine } i \\ 0 & \text{else} \end{cases}$$

- Remark: determining all sets J_l is not polynomial but already pseudo-polynomial since H may not be polynomially bounded

Integer Programming Formulation - Model

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

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

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

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

Reservation Systems without Slack

Easy Special Cases: $p_j = 1$ for all jobs j

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

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

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

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

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

Reservation Systems without Slack

Easy Special Cases: $p_j = 1$ for all jobs j (cont.)

- this problem is an assignment problem and can be solved polynomially
- the number of relevant time periods is at most n
- Consequence: the special case is polynomially solvable

Reservation Systems without Slack

<u>Easy Special Cases: $w_{ij} = 1$ and $M_j = \{1, \ldots, m\}$ for all i, j</u>

- all machines are equal and the goal is to maximize the number of jobs processed
- we assume $r_1 \leq \ldots \leq r_n$
- Notation: J is set of already selected jobs for processing
- initial: $J = \emptyset$

Reservation Systems without Slack

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

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^* s.t. $C_{j^*} = \max_{k \in J} C_k = \max_{k \in J} r_k + p_k$;

        IF $C_j = r_j + p_j < C_{j^*}$ THEN

            remove job j^* and assign job j to machine of j^*;

            $J := J \cup \{j\} \setminus \{j^*\}$

Theorem: The above algorithm solves the problem optimal.

(Proof almost straightforward)

Reservation Systems without Slack

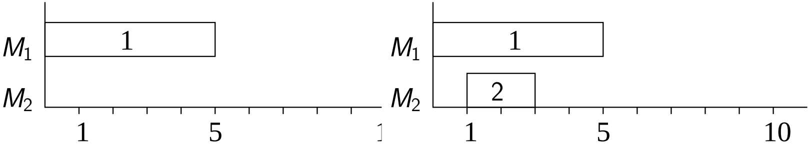
Example $w_{ij} = 1$ and $M_j = \{1, \ldots, m\}$ for all i, j

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: $j = 1$

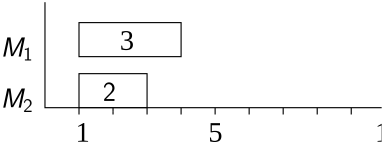
Iteration 2: $j = 2$



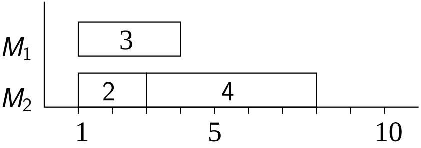
Example $w_{ij} = 1$ and $M_j = \{1, \ldots, m\}$ for all i, j (cont.)

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



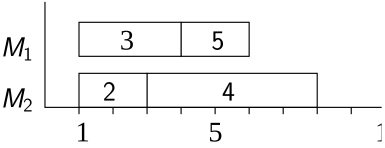
Iteration 4: $j = 4$



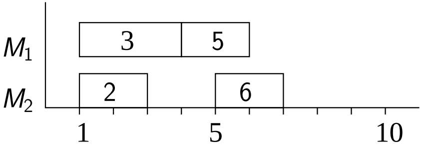
Example $w_{ij} = 1$ and $M_j = \{1, \ldots, m\}$ for all i, j (cont.)

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 5: $j = 5$



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

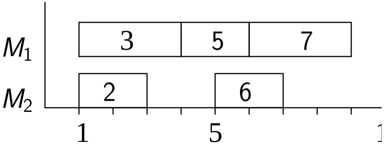


Reservation Systems without Slack

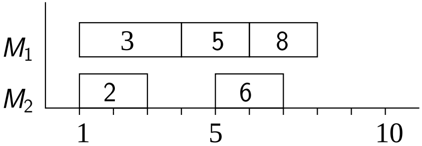
Example $w_{ij} = 1$ and $M_j = \{1, \ldots, m\}$ for all i, j (cont.)

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 7: $j = 7$



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



Reservation Systems without Slack

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: $M = \emptyset$

Reservation Systems without Slack

Algorithm for Another Version of the Reservation Problem

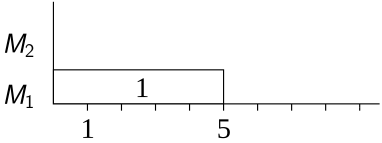
$i = 0$;
FOR $j = 1$ TO n DO
    IF machine from M is free at r_j THEN
        assign j to a free machine
    ELSE
        i:=i+1;
        add machine i to M;
        assign job j to machine i.

Theorem: The above algorithm gives the minimal number of machines to process all n jobs.
(Proof is straightforward)

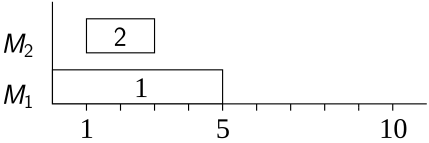
Algorithm for Another Version - Example

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

Iteration 1: $j = 1$



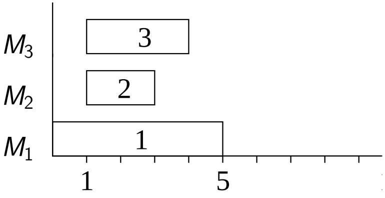
Iteration 2: $j = 2$



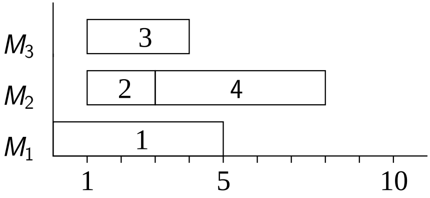
Algorithm for Another Version - Example (cont.)

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: $j = 3$



Iteration 4: $j = 4$

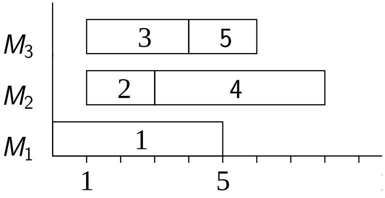


Reservation Systems without Slack

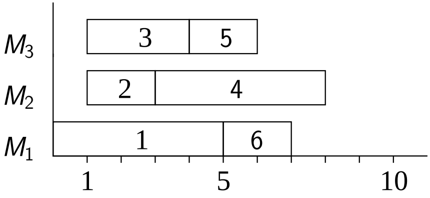
Algorithm for Another Version - Example (cont.)

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 5: $j = 5$



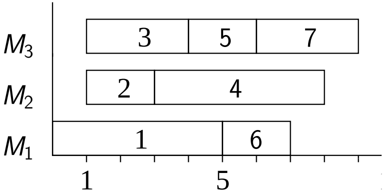
Iteration 6: $j = 6$



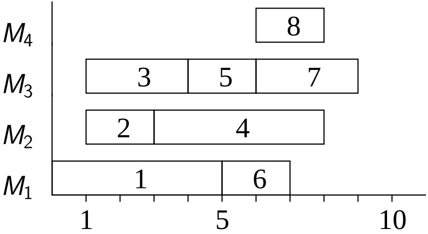
Algorithm for Another Version - Example (cont.)

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 7: $j = 7$



Iteration 8: $j = 8$



Reservation Systems without Slack

Reformulation Another Version

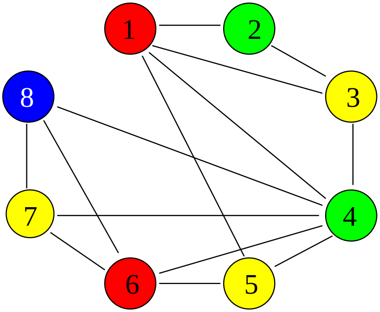
- The problem can be reformulated as a Graph Coloring problem
  - n nodes (node $j \leftrightarrow$ job j)
  - arc (j, k) if job j and k overlap
  - assign a color to each node such that two nodes connected by an arc have different colors
  - Goal: find a coloring with a minimal number of colors
- Remarks
  - 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 in general is NP-hard

Reservation Systems without Slack

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

Notations and Definition

- unlimited number of identical parallel machines
- n jobs with processing times $p_1, \ldots, p_n$
- 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 completing all jobs within a given time horizon H
  - Optimization Version:
    find a schedule for all jobs with a minimal makespan

Timetabling with Tooling Constraints

General Result:

- Theorem: Even for $p_j = 1$ for all j the problem is NP-hard. Proof (on the board) by reduction from Graph Coloring. It is based on the following
- Observation: The problem - for $p_j = 1$ - 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)

Timetabling with Tooling Constraints

Special Case: feasibility version with $p_j = 1$ for all j

- 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'

Special Case: feasibility version with $p_j = 1$ for all j (cont.)

- degree $d(v)$ for a node v: number of arcs adjacent to v
- given a partial coloring of the nodes:
  saturation level $sat(v)$ of a node v: number of different colored nodes already connected to v in the partial coloring

Timetabling with Tooling Constraints

Heuristic Special Case:

feasibility version with $p_j = 1$ for all j

Sort nodes in decreasing order of degrees;

Color a node v with maximal degree $d(v)$ with color 1;

WHILE nodes are uncolored DO

> calculate the maximal saturation level $max - sat$ of uncolored nodes v;
>
> from all nodes v with saturation level $sat(v) = max - sat$, choose any with maximal degree in the uncolored subgraph;
>
> Color the selected node with the color with lowest possible number;

Example Heuristic Special Case:

feasibility version with $p_j = 1$ for all j

Data:

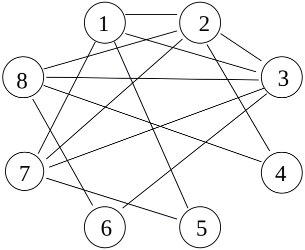
Jobs	1	2	3	4	5	6	7	8
p_j	1	1	1	1	1	1	1	1
Tool 1	1	1	1	0	0	0	1	0
Tool 2	0	1	0	1	0	0	0	1
Tool 3	1	0	0	0	1	0	1	0
Tool 4	0	0	1	0	0	1	0	1
Tool 5	0	0	0	1	0	0	0	0

Example Heuristic Special Case:
feasibility version with $p_j = 1$ for all j

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	0	1	0	0	1	0	1
Tool 5	0	0	0	1	0	0	0	0

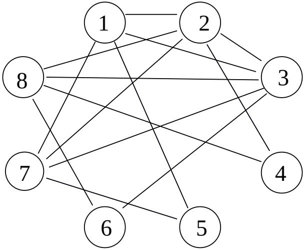
Corresponding Graph



Timetabling with Tooling Constraints

Example Heuristic Special Case:
feasibility version with $p_j = 1$ for all j

Corresponding Graph



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	0	1	0	0	1	0	1
Tool 5	0	0	0	1	0	0	0	0

Preprocessing:

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

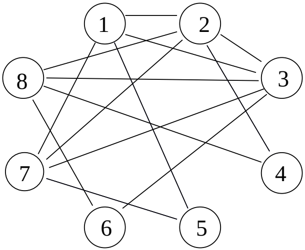
Timetabling with Tooling Constraints

Example Heuristic: feasibility version with $p_j = 1$ for all j (cont.)

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

- Initial:
  - $d(2) = \max d(v)$;
  - color 2 red (color 1)
- Iteration 1:
  - $max - sat = 1$
  - $sat(v) = max - sat; v = 1, 3, 4, 7, 8$
  - $d(3) = \max\{d(v) | v = 1, 3, 4, 7, 8\}$
  - color 3 green (color 2)

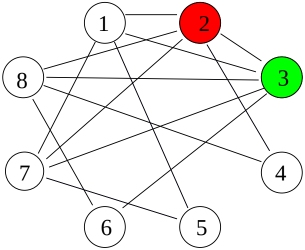
Initial Graph



Example Heuristic: feasibility version with $p_j = 1$ for all j (cont.)

- Initial:
  - $d(2) = \max d(v)$;
  - color 2 red (color 1)
- Iteration 1:
  - $max - sat = 1$
  - $sat(v) = max - sat; v = 1, 3, 4, 7, 8$
  - $d(3) = \max\{d(v) | v = 1, 3, 4, 7, 8\}$
  - color 3 green (color 2)

Graph after Iteration 1

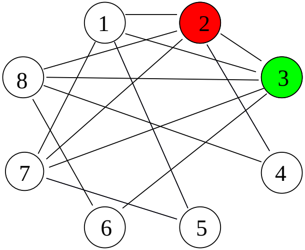


Timetabling with Tooling Constraints

Example Heuristic: feasibility version with $p_j = 1$ for all j (cont.)

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

Graph after Iteration 1



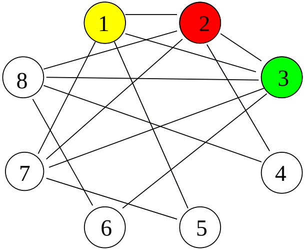
- Iteration 2:
  - $max - sat = 2$
  - $sat(v) = max{-}sat; v = 1, 7, 8$
  - $d(1) = \max\{d(v)|v = 1, 7, 8\}$
  - color 1 yellow (color 3)

Example Heuristic: feasibility version with $p_j = 1$ for all j (cont.)

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

- Iteration 2: $max - sat = 2$
  - $sat(v) = max - sat; v = 1, 7, 8$
  - $d(1) = \max\{d(v) | v = 1, 7, 8\}$
  - color 1 yellow (color 3)
- Iteration 3: $max - sat = 3$
  - $sat(7) = max - sat;$
  - color 1 blue (color 4)

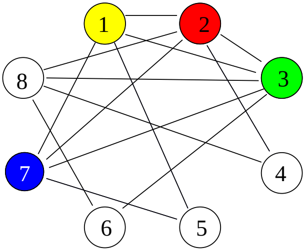
Graph after Iteration 2



Example Heuristic: feasibility version with $p_j = 1$ for all j (cont.)

- Iteration 2:
  - $max - sat = 2$
  - $sat(v) = max - sat$; $v = 1, 7, 8$
  - $d(1) = \max\{d(v) | v = 1, 7, 8\}$
  - color 1 yellow (color 3)
- Iteration 3:
  - $max - sat = 3$
  - $sat(7) = max - sat$;
  - color 1 blue (color 4)

Graph after Iteration 3

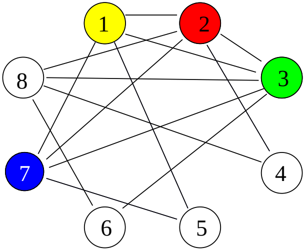


Timetabling with Tooling Constraints

Example Heuristic: feasibility version with $p_j = 1$ for all j (cont.)

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

Graph after Iteration 3



- Iteration 4:
  - $max - sat = 2$
  - $sat(v) = max - sat; v = 5, 8$
  - $d(8) = \max\{d(v)|v = 5, 8\}$
  - color 8 yellow (color 3)

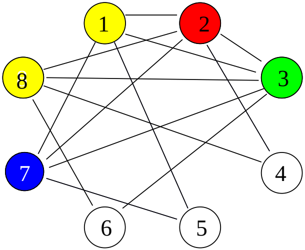
Timetabling with Tooling Constraints

Example Heuristic: feasibility version with $p_j = 1$ for all j (cont.)

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

- Iteration 4: $max - sat = 2$
  - $sat(v) = max - sat; v = 5, 8$
  - $d(8) = \max\{d(v)|v = 5, 8\}$
  - color 8 yellow (color 3)
- Iteration 5: $max - sat = 2$
  - $sat(v) = max - sat; v = 4, 5, 6$
  - $d(4) = \max\{d(v)|v = 4, 5, 6\}$
  - color 4 green (color 2)

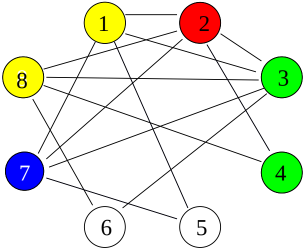
Graph after Iteration 4



Example Heuristic: feasibility version with $p_j = 1$ for all j (cont.)

- Iteration 4:
  - $max - sat = 2$
  - $sat(v) = max - sat$; $v = 5, 8$
  - $d(8) = \max\{d(v) | v = 5, 8\}$
  - color 8 yellow (color 3)
- Iteration 5:
  - $max - sat = 2$
  - $sat(v) = max - sat$; $v = 4, 5, 6$
  - $d(4) = \max\{d(v) | v = 4, 5, 6\}$
  - color 4 green (color 2)

Graph after Iteration 5

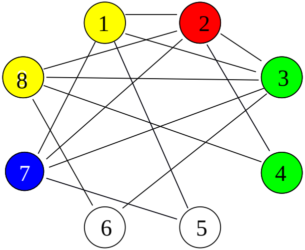


Timetabling with Tooling Constraints

Example Heuristic: feasibility version with $p_j = 1$ for all j (cont.)

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

Graph after Iteration 5



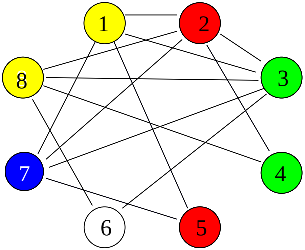
- Iteration 6:
  - $max - sat = 2$
  - $sat(v) = max - sat; v = 5, 6$
  - $d(5) = \max\{d(v) | v = 5, 6\}$
  - color 5 red (color 1)

Example Heuristic: feasibility version with $p_j = 1$ for all j (cont.)

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

- Iteration 6:
  - $max - sat = 2$
  - $sat(v) = max - sat; v = 5, 6$
  - $d(5) = \max\{d(v) | v = 5, 6\}$
  - color 5 red (color 1)
- Iteration 7:
  - only 6 is left
  - color 6 red (color 1)

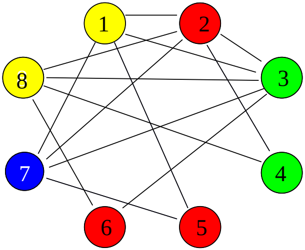
Graph after Iteration 6



Example Heuristic: feasibility version with $p_j = 1$ for all j (cont.)

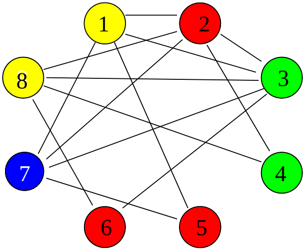
- Iteration 6:
  - $max - sat = 2$
  - $sat(v) = max - sat; v = 5, 6$
  - $d(5) = \max\{d(v) | v = 5, 6\}$
  - color 5 red (color 1)
- Iteration 7:
  - only 6 is left
  - color 6 red (color 1)

Final Coloring Graph



Example Heuristic: feasibility version with $p_j = 1$ for all j (cont.)

Final Coloring 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

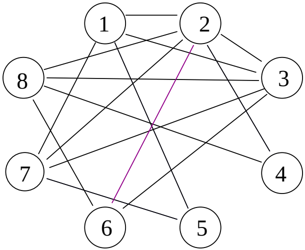
- Remark:
  For the given example the tools can not be ordered such that for all jobs the used tools are adjacent (i.e. the resulting graph is not an interval graph). Thus the instance can not be seen as an interval scheduling instance.
- Change of the data:
  assume job 2 needs besides tool 1 and 2 also tool 4

Relation to Interval Scheduling (cont.)

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

New Graph:



Relation to Interval Scheduling (cont.)

Tool 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

Transformation:

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

Relation to Interval Scheduling (cont.)

Transformation:

Tool 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

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

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