Exercises §7

7.1 Prove the properties of $\mathcal{E}(G)$

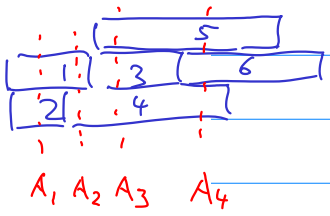
7.2 Characterization of interval orders

Show that the following conditions are equivalent:

(1) H is an interval order

(2) H does not contain $\begin{matrix} \circ \to \circ \\ \circ \to \circ \end{matrix}$ $(2+2)$

as induced sub order (may be transitive arcs)



(3) There is an ordering $j_1, j_2, \dots, j_n$ of V with

$$\text{Pred}(j_1) \subseteq \text{Pred}(j_2) \subseteq \dots \subseteq \text{Pred}(j_n)$$

(4) There is a numbering $A_1, A_2, \dots, A_m$ of the maximal antichains of H such that, for every $j \in V$, all A_k containing j occur consecutively in the numbering

[consecutiveness property of maximal antichains]

$x_j = 1$ in machines, precedence constraints

7.3 The m-machine problem can be solved in polynomial time on interval orders

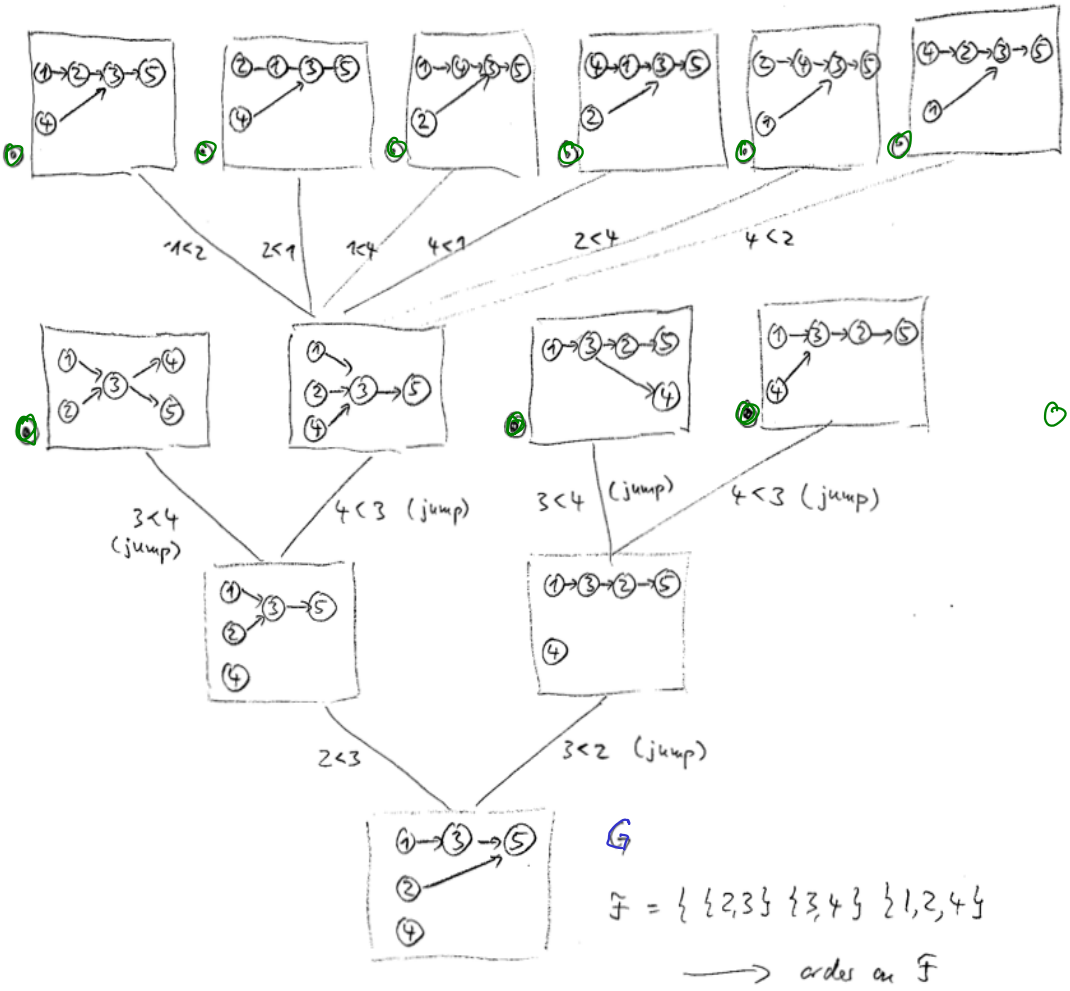
7.4 For every minimal feasible order H, there are K and Q such that H is the only optimal (minimal) feasible order

$$E_Q[K^H] < E_Q[K^{H'}] \quad \text{for all (minimal) feasible } H' \neq H$$

Does this also hold for $K = C_{max}$?

notice the analogy to extreme points in linear programming

§8 Construction of ES-policies



○ feasible

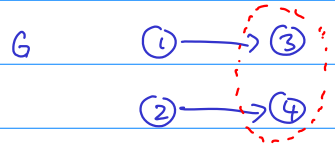
Remarks:

- Conflict settling tree may used for Brand & Bound algorithms for calculation an optimal schedule for given K, x
- It can be combined with the dynamic decision view, i.e. brand only at decision times
  In this way, only a subset of all forbidden sets will be considered

8.5 Remark: In the stochastic case, the optimal values obtained by priority / ES-policies are incomparable
[remember: in the the deterministic case, ES is better]

8.6 Example:

(1) $OPT^{PRIOR} < OPT^{ES}$



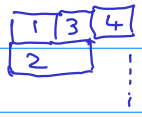
$\mathcal{F} = \{\{3,4\}\}$ $\quad \kappa = C_{max}$

$x^1 = (1,2,1,1)$ } each with prob $1/2$

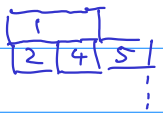
$x^2 = (2,1,1,1)$ }

Π priority policy $= \sigma$

$\Pi[x^1]$



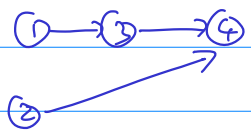
$\Pi[x^2]$:



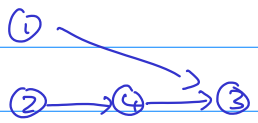
$E[\kappa^{\Pi}] = 3$

ES-policy: minimal feasible orders are

H_1



H_2



$E[\kappa^{H_1}] = E[\kappa^{H_2}] = 3.5$

(2) $OPT^{ES} < OPT^{PRIOR}$

already in the deterministic case

Exercises:

8.1 Construct an example in which the conflict settling tree contains leaves that are not minimal feasible

8.2 What is the complexity of adding a precedence constraint $i_F < j_F$ to an order H? What is a good data structure for the orders in the conflict settling tree to allow fast updating w.r.t. to adding precedence constraints?

§9 Preselective Policies

Generalize ES-planning rules by relaxing the "rule" for solving the conflict on a forbidden set

ES-policies:

for every $\overline{F} \in \overline{\mathcal{F}}$ choose $i_{\overline{F}}, j_{\overline{F}} \in \overline{F}$ and add $i_{\overline{F}} < j_{\overline{F}}$ to G

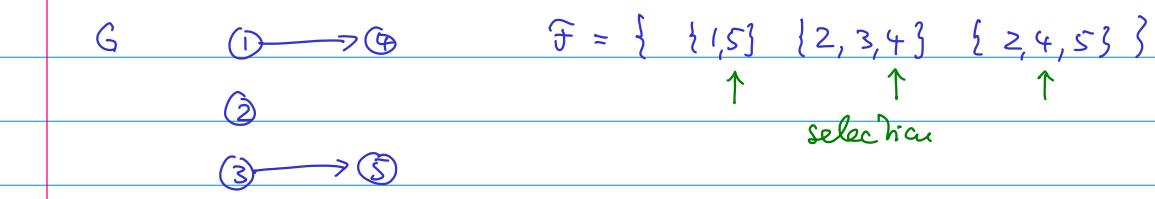
i.e. choose a waiting job $j_{\overline{F}}$ and a job $i_{\overline{F}}$ for which $j_{\overline{F}}$ must wait

Preselective planning rules (will show that they are policies)

for every $\overline{F} \in \overline{\mathcal{F}}$, choose a waiting job $j_{\overline{F}} \in \overline{F}$ that must wait for any job $\in \overline{F}$, i.e. $j_{\overline{F}}$ can start after the first job in $\overline{F} \setminus \{j_{\overline{F}}\}$ completes

The sequence $s = (j_{\overline{F}_1}, j_{\overline{F}_2}, \dots, j_{\overline{F}_k})$ of waiting jobs $j_{\overline{F}_r} \in \overline{F}_r$ for $\overline{\mathcal{F}} = \{\overline{F}_1, \dots, \overline{F}_k\}$ is called a selection for $\overline{\mathcal{F}}$

9.1 Example

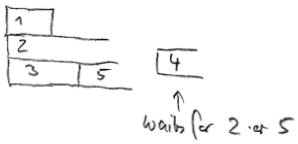


what are the possible schedules, when we do early start scheduling w.r.t. G and the waiting conditions coming from the selection s?

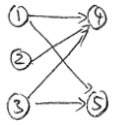
$t = 0$ 1 2 3 consider possible completions

↑ can be started at $t = 0$

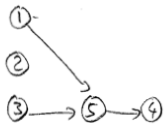
(2) $x_1 < x_3 < x_2$



H_{21}

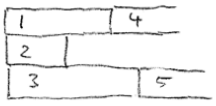


H_{22}

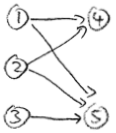


interval orders induced by the possible schedules

(3) $x_2 < x_1 < x_3$

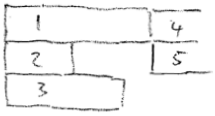


H_3

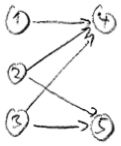


$H_3 = H_6$

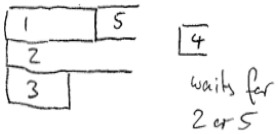
(4) $x_2 < x_3 < x_1$



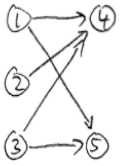
H_4



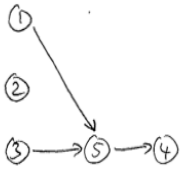
(5) $x_3 < x_1 < x_2$



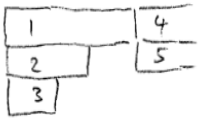
$H_{51} = H_{21}$



$H_{52} = H_{22}$



(6) $x_3 < x_2 < x_1$



H_6

$H_6 \succ H_1$

Cases with equality are subsumed by (1)-(6)
└ of processing times

Example shows:

- depending on x we obtain a different feasible order H with $\Pi[x] = ES_H[x]$ (holds for every elementary policy)

- $\Pi = \min \{ ES_H \mid H \text{ induced by } \Pi[x] \text{ for some } x \}$
(does not hold for every elementary policy)

Questions:

(1) When is a selection contradictory? Can this be easily checked?

(2) Does a "feasibe" selection define a policy?
I.e. is a preselective policy non-anticipative?

(3) What is the relationship with ES-policies?
Is $\Pi = \min\{ES_H \mid H \dots\}$ for some set of feasible ES-policies?
(as in the example)

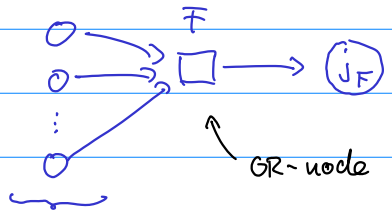
(4) How to calculate the start times of a preselective policy?

(5) Do preselective policies still have nice properties?

Preselective planning rules and AND/OR networks

↑
combinatorial representation of preselective planning rules

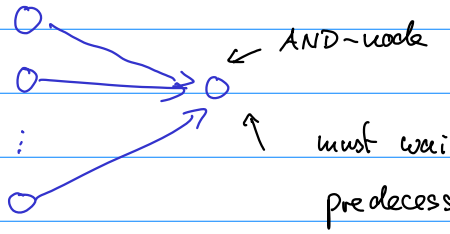
Consider a selected job $j_F \in F \in \mathcal{F}$
⇒ introduce an OR-precedence constraint



interpretation: j_F may start after one of the direct predecessors of □ has completed

all $j \in F \setminus \{j_F\}$

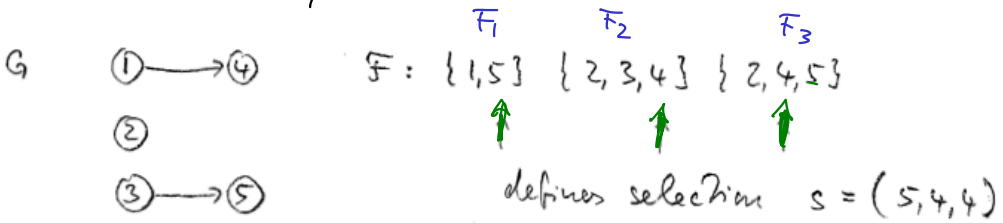
ordinary precedence constraints ≙ AND-precedence constraints



↑ must wait for the completion of all direct predecessors, AND-node

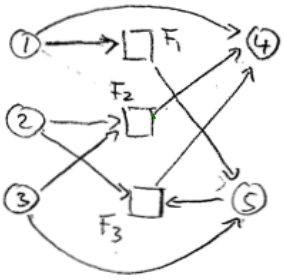
AND/OR-network = network with AND and OR nodes

Observation: Every selection defines an AND/OR-network



9.1 EXAMPLE (continued)

s = (5,4,4) defines



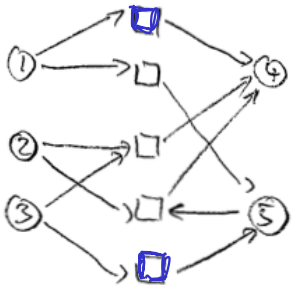
$F_1 = \{1,5\}$

$F_2 = \{2,3,4\}$

$F_3 = \{2,4,5\}$

Observation: every precedence constraint $i < j$ can be represented as OR-precedence constraint

9.1. EXAMPLE (continued) Represent all given prec. constraints by OR-prec. constraints

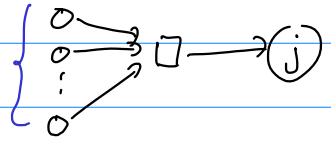


Then the AND/OR network is bipartite

Consequences:

(1) Resulting AND/OR network is bipartite

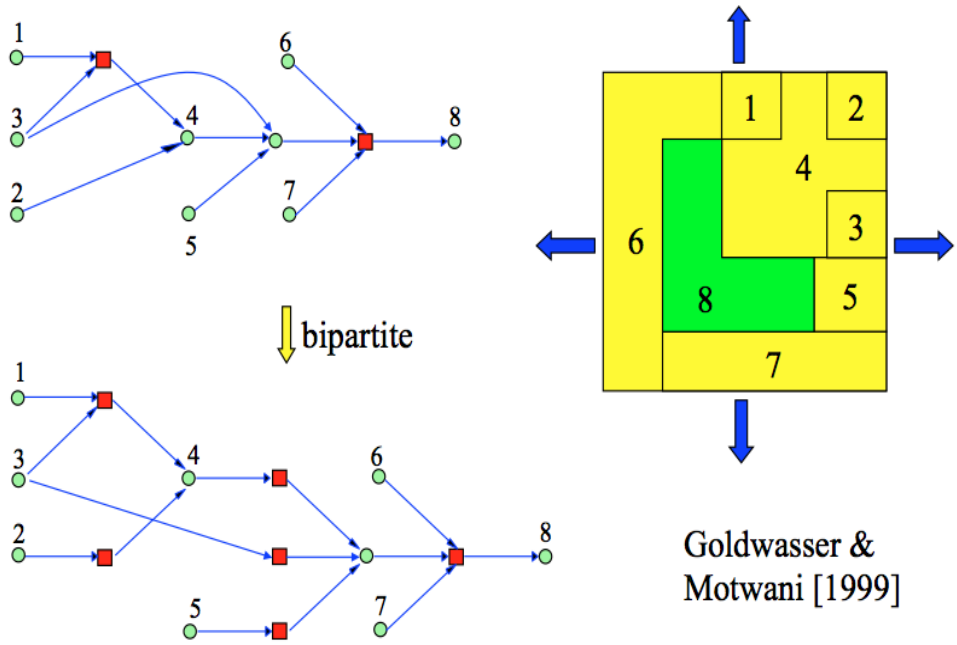
(2) May just speak of a system W of waiting conditions

$(X_{ij}) \triangleq$ 

So a system of waiting conditions ⟺ (bipartite) AND/OR network

AND/OR networks have been studied also in other contexts:
one of them is disassembly

An application: Disassembly

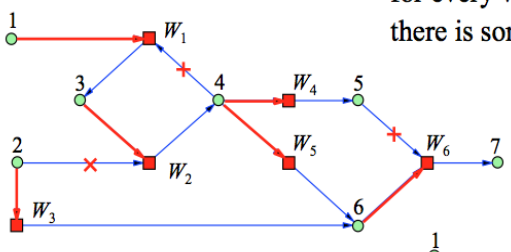


Goldwasser & Motwani [1999]

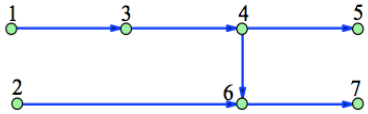
Question (1) When is a system of waiting conditions feasible?

Feasibility of waiting conditions $\mathcal{W}$

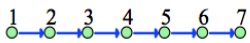
A realization of $\mathcal{W}$ is an acyclic graph $R = (V, A)$ on V s.t.
for every waiting condition (X, j),
there is some $i \in X$ preceding j



red arcs define a realization



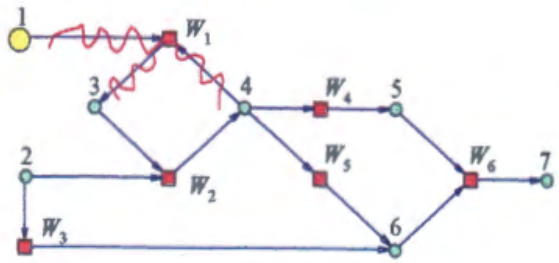
special case: linear realization



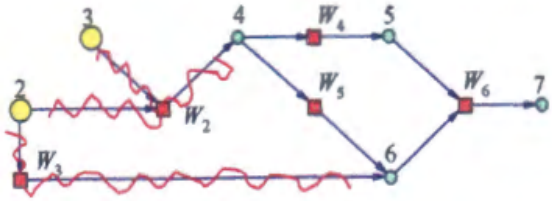
An algorithm for testing feasibility

- Tries to construct a linear realization for $\mathcal{W}$
- Imitates topological sort for digraphs

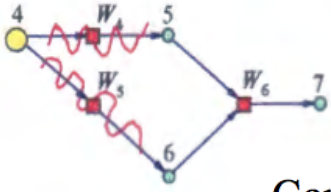
List L := [ ]
while (there is a job $i \in V$ that is not a waiting job of a condition in $\mathcal{W}$)
  begin
    insert i at the end of L and delete it from V
    if (some waiting condition (X, j) becomes satisfied)
      delete (X, j) from $\mathcal{W}$
  end
return L



L = [1,]



L = [1, 2, 3,]

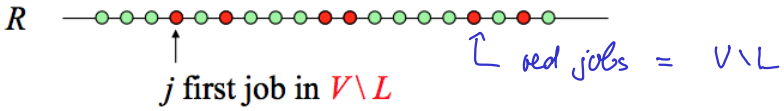


L = [1, 2, 3, 4, 5, 6, 7]

Correctness of the algorithm

$\mathcal{W}$ feasible $\Leftrightarrow$ L contains all jobs $\Leftarrow$ trivial

"$\Rightarrow$": Let R be a linear realization but $L \neq V$



j not in L $\Rightarrow$ there is a waiting condition (X, j) with $X \subseteq V \setminus L$
$\Rightarrow$ all jobs in X are red
but in R, there must be a job $i \in X$, i.e., a red job, before j
$\Rightarrow$ contradiction