§4 Scheduling with scarce resources: Introduction and Complexity

The model

$V = \{1, \dots, n\}$ set of jobs

G = graph (V, E) of precedence constraints

$\mathcal{F} = \{F_1, \dots, F_k\}$ system of forbidden (resource constraints)

κ = cost function

$x = (x_1, \dots, x_n)$ / $X = (X_1, \dots, X_n)$ deterministic / stochastic processing times

<u>deterministic case</u>

Find a schedule S that respects $G, x, \mathcal{F}$ (<u>feasible</u> schedule)

such that $\kappa(S, x) \underset{\text{Def}}{=} \kappa(C_1, \dots, C_n)$ is minimum or "good"

$\uparrow$

$C_j = S_j + x_j$

<u>m-machine problems</u>

m identical machines / processors

every job needs one, every machine can process only one job at a time

$\Rightarrow \mathcal{F} = \{ F \subseteq V \mid |F| = m+1, F \text{ is antichain on } G \}$

4.1 Theorem: Let $m = 1$ and $E_G = \emptyset$

(1) idle time does not pay

$\uparrow$ processor is not busy, but there is a job that could be processed

(2) for $\sum w_j C_j = \kappa$, Smith's rule constructs an optimal schedule

- sort jobs s.t. $\frac{w_{i_1}}{x_{i_1}} \geq \frac{w_{i_2}}{x_{i_2}} \geq \dots$
- schedule the jobs in this order

(3) For $\kappa = L_{max}$, Jackson's rule constructs an optimal schedule

- sort jobs s.t. $d_{i_1} \leq d_{i_2} \leq \dots \leq d_{i_n}$ (d_j = due dates)
- schedule them in this order

(4) $K = L_{max}$, but jobs have also release dates r_j (i.e. $S_j \geq r_j$)

This problem is NP-hard

(5) $\sum_j w_j T_j$ is NP-hard

Proof: (1) obvious

(2) exchange argument for 2 adjacent jobs

(3) homework

(4) (5) without proof

Theorem shows: small changes turn a problem from polynomially solvable to NP-hard

typical for scheduling

1975 - 1985 many scheduling problems have been classified (about 8000)

about 80% of them are NP-hard

(Lawler, Lenstra, Rinnooy Kan)

webpage: http://www.informatik.uni-osnabrueck.de/knust/class

<u>Other example</u>

m-machine problem with $m = 2$, arbitrary precedence constraints G

$x_j = 1$ for all j $K = C_{max}$

models jobs in a parallel processor environment

$x_j = 1$ $\hat{=}$ time slots for processing

every long job is subdivided into a chain of pieces with unit length

Case $m = 2$ is polynomially solvable

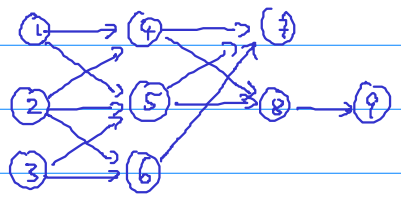
4.2 THEOREM: $(C^G_{max})^{OPT}(\mathbb{1})$ = maximum size of a matching in Incomp(G) + # unmatched jobs, and one can construct an optimal schedule from the matching in polynomial time [Fujii et al in '67, 71]

Incomp(G) has vertex set V (same as G)

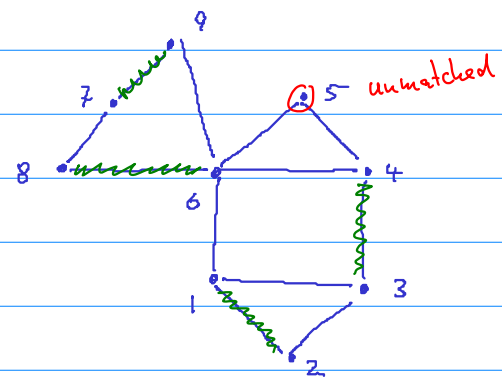
$\{i,j\}$ is an edge of Incomp(G) $:\Longleftrightarrow$ $i \parallel_G j$ (and $i \neq j$)

Example

G



Incomp(G)

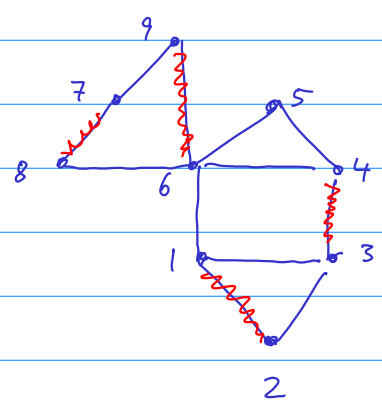


an optimal schedule

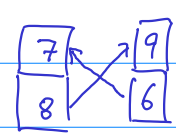
1	3	5	6	7
2	4		8	9

We see that every time slot with 2 jobs defines an edge of a matching in Incomp(G) (here: the green edges)

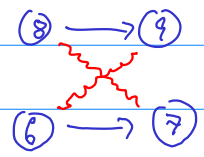
Now: from the matching to the schedule:



← this a maximum matching but it cannot be turned into a schedule s.t. matched jobs are scheduled in the same time slot



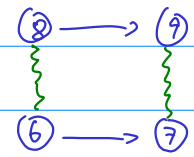
contradiction, there is no ordering of the time slots respecting the precedence constraints!



← call this a "crossing" (matching edge "cross" w.r.t. precedence constraints)

then $\{8,6\}$ and $\{9,7\}$ are also edges in $\mathrm{Incomp}(G)$

$\Downarrow$ "uncrossing" the matching

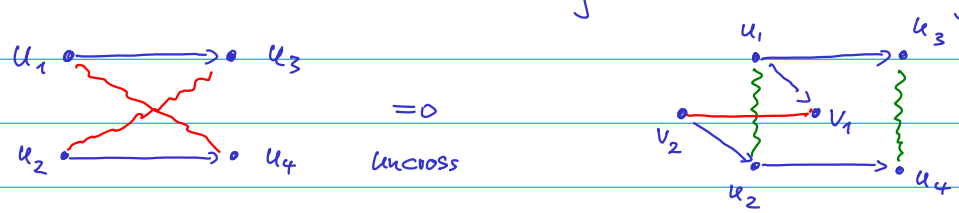


alway possible locally and locally schedulble

Claim: for every maximum matching M, there is a non-crossing matching of the same size that can be obtained by uncrossing crossings in any order

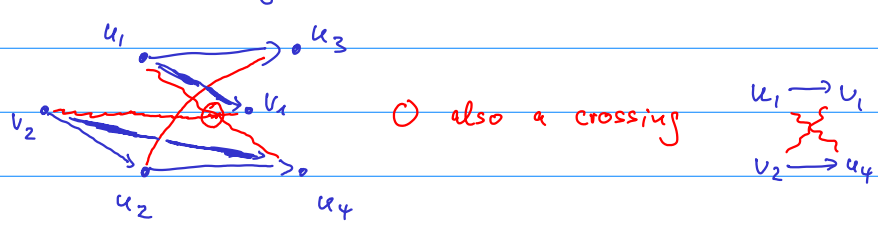
Proof of claim: $\boxed{\text{every uncrossing reduces the number of crossings}}$

suppose this is not the case $\Rightarrow$ uncrossing creates a new crossing



$\Rightarrow u_1 < v_1, \quad v_2 < u_2$

before uncrossing



not a crossing after uncrossing (since $\{u_1, u_4\}$ is no longer a matching edge)

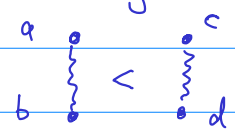
$\Rightarrow$ for every new crossing, another old crossing involving the same vertices and edges is also uncrossed

$\Rightarrow$ the number of crossings drops $\square$

Now assume that M is non-crossing

want to construct a schedule from M

order the matching edges



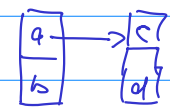
$:\Longleftrightarrow \quad a<c \text{ or } a<d \text{ or } b<c \text{ or } b<d$

in G

this is a well-defined ordering of the time slots represented by the matching edges

(otherwise there would be a crossing)

assume $a<c$ and slot ordering



if this is not a feasible slot ordering

then $c<b$ or $d<a$ or $d<b$

$\Downarrow$ $\qquad$ $\Downarrow$ $\qquad$ $\Downarrow$

$a<b$ $\qquad$ similar $\qquad$ crossing

(by transitivity)

contradiction to

$a \,\|_G\, b$

take a <u>linear extension</u> of that ordering of matching edges

$\hookrightarrow$ is a linear order preserving the ordering relations between matching edges

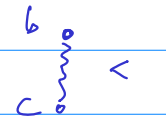
insert unmatched jobs into this linear order by

observing

$:\Longleftrightarrow \quad a<b \text{ or } a<c$

matching edge

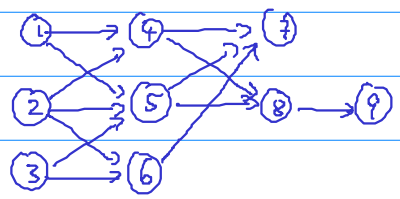
(symmetrically

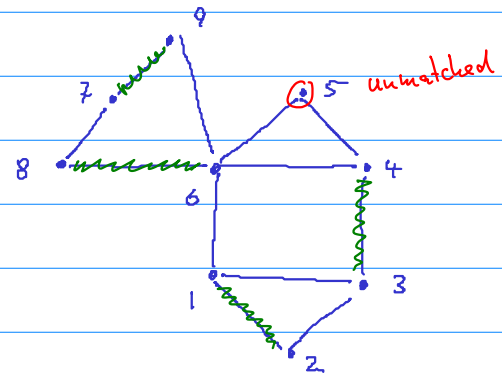


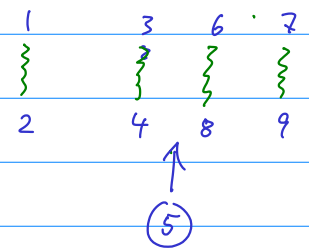
)

the linear order of matching edges and unmatched jobs defines a feasible schedule

G







1	3	5	6	7
2	4	=	8	9

$\square$

$\boxed{\text{m - machine problem is open for } m = 3}$ (complexity unknown)

famous 3-machine problem

4.3 THEOREM: The following problem is NP-complete

m - machine problem:

<u>Given</u>: G, m, time t, $x = \mathbb{1}$

<u>Question</u>: is there a feasible schedule on m machines with $C_{max} \leq t$

Proof: CLIQUE reducible to m-machine scheduling

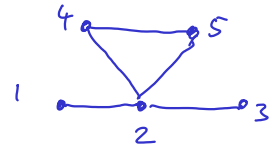
Construct from an instance of CLIQUE an m-machine problem in poly time

Graph G defines vertex jobs and edge jobs

every edge job must wait for the two vertex job that define the edge

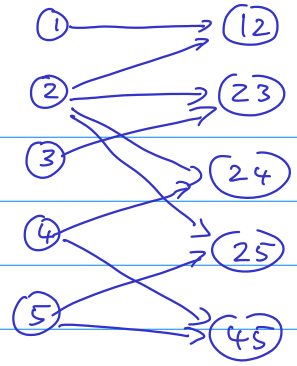
Example

G



$k = 3$

$\Rightarrow$

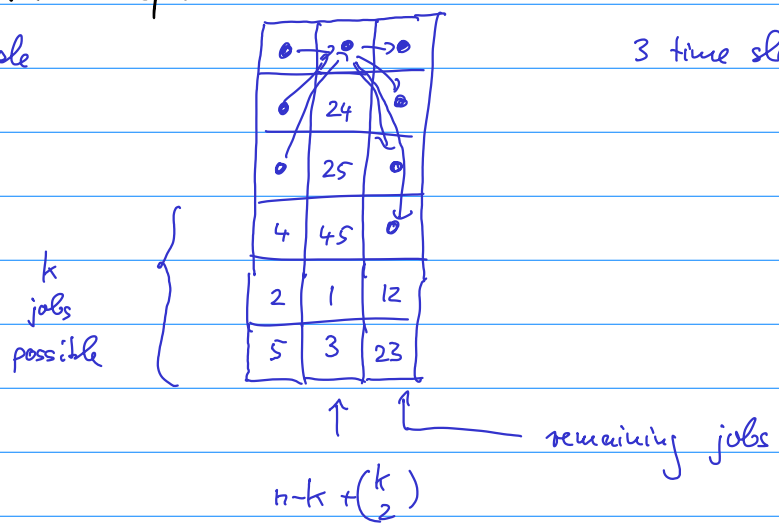


"real jobs"

in addition there are "dummy" jobs that define a "frame" in which the "real" jobs must be scheduled, together with the time bound $t = 3$ for the makespan

Example

3 time slots



The graph has a clique of size k $\Longleftrightarrow$ there is a schedule of length 3 for these precedence constraints