§4 Scheduling subject to scarce resources: Introduction and complexity

The model:

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

G = graph of precedence constraints

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

κ = regular measure of performance

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

deterministic case:

find schedule S that respects $G, x, \mathcal{F}$ (feasible schedule)

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

$\uparrow$

$C_j = S_j + x_j$ completion times w.r.t. S

m-machine problems:

m identical machines / processors

every job needs one machine

every machine can process any job, but at most one at a time

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

4.1 THEOREM: Let $m = 1$ and $E_G = \emptyset$ (antichain). Then:

(1) idle time does not pay for any K

↑ can start a job, but wait

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

- sort jobs s.t. $\frac{x_{i_1}}{w_{i_1}} \leq \frac{x_{i_2}}{w_{i_2}} \leq \ldots \leq \frac{x_{i_n}}{w_{i_n}}$
- schedule jobs in this order

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

- sort jobs s.t. $d_{i_1} \leq \ldots \leq d_{i_n}$ (d_i = due date of job j)
- schedule jobs in this order

(4) Problem $K = L_{max}$ + release dates r_j [i.e. $S_j \geq r_j$] is NP-hard

(5) Problem $\sum w_j T_j$ 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 much research about complexity (Lawler, Lenstra, Rinnooy Kan)

about 8000 problems classified

about 80% NP-hard

webpage http://www

Other example:

m machine problem with $m \geq 2$, arbitrary precedence constraints G,

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

↳ models jobs in a parallel processor environment

$x_j = 1$ ≙ time slot for processing

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

Case $m = 2$ is polynomially solvable

4.2 THEOREM: $C_{max}^{opt\ G}(\mathbb{1})$ = maximum size of a matching in $Incomp(G)$ + # unmatched jobs [Fujii et al 67, 71]

↑

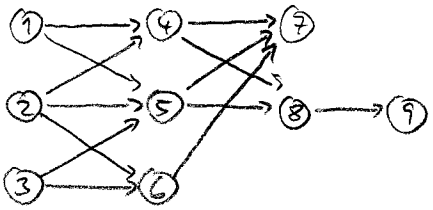
incomparability graph of G

vertex set V

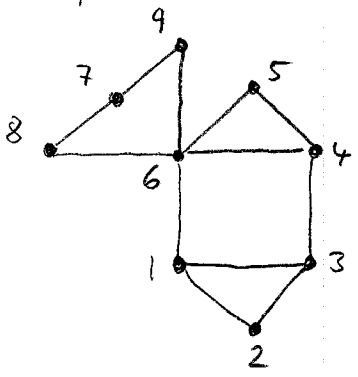
(i,j) edge in $Incomp(G)$ ⟺ $i \neq j$ and $i \parallel_G j$ (incomparable)

Example:

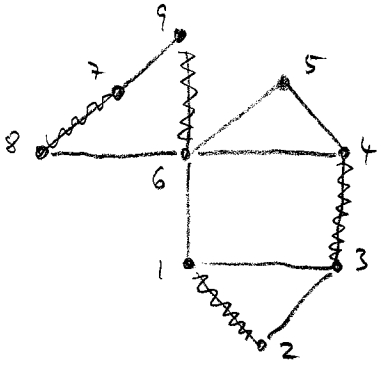
G:



Incomp(G)



a maximum matching M in Incomp(G)



size 4

unmatched jobs = 1

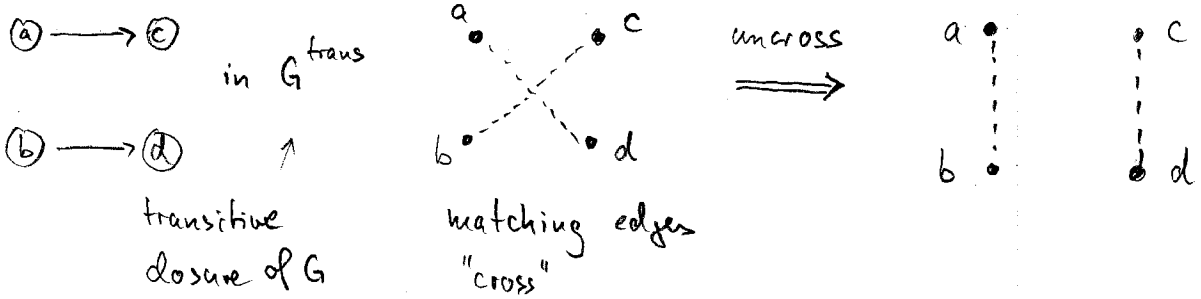
an optimal schedule

1	3	5	6	7
2	4		8	9

length 5 = |M| + 1

Proof: Idea: construct a schedule from the matching by processing matched jobs in parallel (makes use of $m = 2$)

requires "uncrossing" of crossings

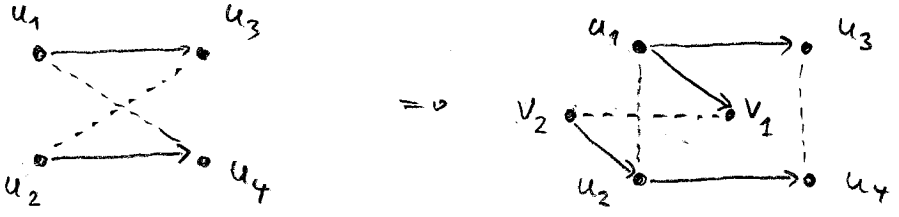


Crossings in example:

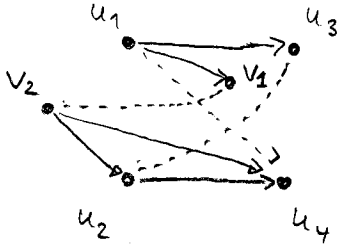
Claim: For every matching M, there is a non-crossing matching of the same size that can be obtained by uncrossing crossings in any order.

Proof of claim: Every uncrossing reduces the number of crossings

Suppose not ⇒ new crossing created, say by $u_1 \cdots u_2$



⇒ $\exists\ v_1 > u_1$, $v_2 < u_2$, $(v_1, v_2) \in M$

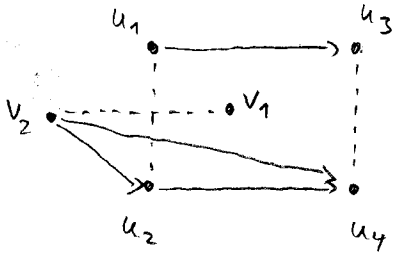


before uncrossing

⇒ (v_1, v_2) crossed with (u_1, u_4) before

[since $v_2 < u_4$]

⇒ that crossing no longer exists after uncrossing



⇒ For every new crossing, an old crossing involving the same edges is uncrossed

$\Rightarrow$ no double counting possible $\Rightarrow$ claim

Now assume that M is non-crossing

Construct an extension G^* of G

↳ i.e. $u <_G v \Rightarrow u <_{G^*} v$

that has exactly the matched edges as incomparabilities

(1) order matched edges by

$(a,b) \prec (c,d)$ if $a \leq_G b$ or $a <_G d$ or $b <_G c$ or $b \leq_G d$

[possible because of non-crossing]

(2) order jobs and matched pairs by

$a \prec (c,d)$ if $a <_G c$ or $a <_G d$ (similarly for $\succ$)

[well-defined since $c \| _G d$]

(3) Consider transitive closure of $\prec$ in (1), (2) and $<_G$ between unmatched jobs

(4) Take any <u>linear extension</u> L of that transitive closure

↓

extension in which any two elements are comparable, i.e., which is a single chain

(5) Replace matching edges by their 2 endpoints

This defines G^*

Consider $ES_{G^*}[x]$. Since $x_j = 1 \ \forall j$,

$C_{max}^{G^*}(x) =$ # matching edges + # unmatched jobs (*)

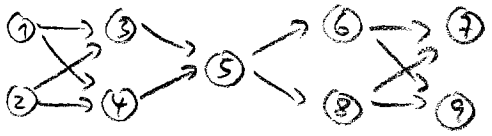
⇓

every matching gives a schedule S of length (*)
every schedule defines a matching fulfilling (*)
} ⇒ Theorem

Construction of G^* in example:

(1) $1, 2 < 3, 4 < 6, 8 < 7, 9$

(3) $1, 2 < 3, 4 < 5 < 6, 8 < 7, 9$ already linear

G^* 

ES_{G^*}

1	3	5	6	7
2	4		8	9

□

m-machine problem is open for $m = 3$

[the famous 3-machine problem]

If m is arbitrary, i.e. part of the input, then

4.3 THEOREM: The following problem is NP-complete

m-MACHINE-PROBLEM:

Given: G, m, time t, $x = 1$

Question: Is there a feasible schedule with makespan $\leq t$?

Proof: CLIQUE reducible to m-MASCHINE PROBLEM

Consider instance of CLIQUE

└ graph $G = (V, E)$, number k

Question: does G have a clique of size k

Construct from this instance an instance of m-MASCHINE-PROBLEM.

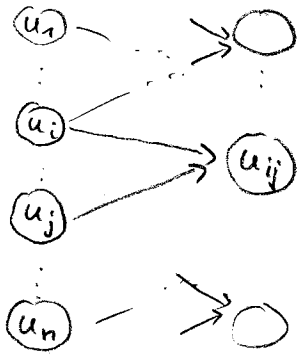
$V = \{1, \dots, n\}$ → jobs $u_1, \dots u_n$ ("vertex jobs")

$(i,j) \in E$ → job u_{ij} ("edge jobs")

Define precedence constraints among vertex and edge jobs by

$(i,j) \in E \Rightarrow u_i < u_{ij}, \; u_j < u_{ij}$

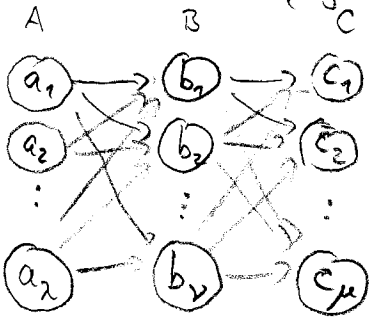
G_1



every edge job u_{ij} must wait for the two vertex jobs u_i, u_j

In addition, need dummy jobs and precedence constraints among them

G_2



every $a_i < b_j$, every $b_i < c_j$

with number of machines $m = \max\{k, \binom{k}{2} + n - k, |E| - \binom{k}{2}\} + 1$

$\lambda = m - k$

$\nu = m - \binom{k}{2} - n + k$ $\Rightarrow \min\{\lambda, \nu, \mu\} = 1$

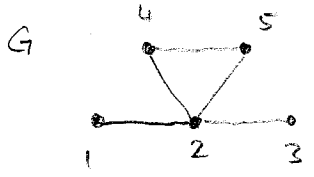
$\mu = m - |E| + \binom{k}{2}$

$G = G_1 + G_2$ (disjoint union)

Finally, $t = 3$

Claim: G has a clique of size k
$\Leftrightarrow$ there is an m-machine schedule of length $\leq t$

Expl:



$k = 3 \Rightarrow m = 6$

A ↓	B ↓	C ↓
///	//	//
///	u_1	//
//	u_3	//
u_2	u_{24}	//
u_4	u_{25}	u_{12}
u_5	u_{45}	u_{23}

dummy jobs create a frame

can schedule k vertex jobs in first time period, must use time slots for "clique" jobs

slots for remaining vertex jobs plus $\binom{k}{2}$ edge jobs from a k-clique

slots for remaining edge jobs

Proof claim:

"$\Rightarrow$": schedule k vertex jobs from the clique first
$\Rightarrow \binom{k}{2}$ edge jobs + remaining vertex jobs in time period 2
$\Rightarrow$ remaining edge jobs in time period 3

"$\Leftarrow$" Assume there is no clique of size k
$\Rightarrow$ at most $\binom{k}{2} - 1$ edge jobs in period 2
$\Rightarrow$ one idle time slot in period 2
$\Rightarrow$ length ≥ 4 $\square$

Exercises:

4.1 Show that Jackson's rule constructs an optimal 1-machine schedule for L_{max} (no precedence constraints).

4.2 Show that the 3-machine problem can be solved in polynomial time for precedence constraints of fixed width (i.e. the size of the largest antichain is bounded by a constant)

e
x