CSL 630 Lecture 13 Sept 11

Scheduling problem

Given a set of n Jobs $J_1, J_2 .. J_n$ with processing requirements 1 unit and (integral) deadlines, we want to schedule them in a way so that they get completed before deadlines. If not then job J_i incurs a penalty p_i. Goal: Minimise penalty.

(only one Job can be running at a time)

Eg.

	J_1	J_2	J_3
deadlines	1	2	1
penalty	5	2	8

T	0	1	2	3	4
	J_3	J_2	J_1		

penalty : 5

Strategy: Earliest deadline first
break ties on the basis of penalty

Suppose we have a schedule

$$\begin{array}{cccccccc} t_1 & t_2 & & & t_i & & t_k & \\ j_1 & j_2 & j_3 & - & j_i & - & j_k & \cdots \end{array}$$

$d_i > d_k$

If $d_i < t_k$ then we may incur penalty

Suppose the given schedule is "feasible" (no job incurs penalty) then

$d_k > t_k \qquad d_i > d_k \Rightarrow d_i > t_k$

<u>Objective</u> : Minimize the penalty of the jobs that missed the deadline

$\Leftrightarrow$ maximise the penalty of the "feasible" schedule

To apply "generic greedy" we must define the subset system framework

$S = \{J_1, J_2, \ldots J_n\}$

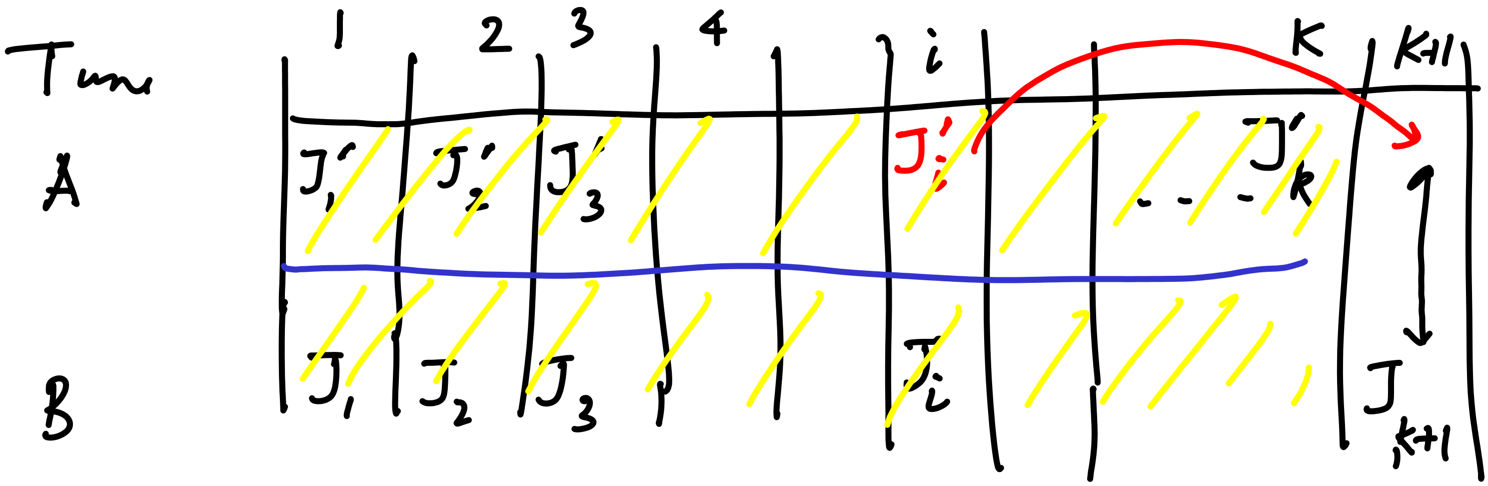
$\mathcal{I}$: subsets of S that are "feasible", i.e. they can be scheduled without missing any deadline.

Moreover, any subset of a feasible set of Jobs is also feasible.

We would like to see if we can satisfy properties (2) or (3) of the matroid theorem.

<u>exchange property</u>.

Given feasible sets A and B $|B| > |A|$ can we add a job $j \in B - A$ to A and still keep $A \cup \{j\}$ feasible?



<u>Case 1</u> : $J_{k+1} \notin A$ add J_{k+1} to A

<u>Case 2</u> $J_{k+1} \in A \Rightarrow J_{k+1} = J_i'$

Repeat the same argument with one job less in A and B.
either we terminate with case 1
or we are in a situation where
$A = \phi$ and B has 1 job

<u>h.w</u> : Find a feasible schedule (the above argument gives a feasible set)

"Generic greedy works"

How about minimizing in the matroid framework?

Since $\emptyset$ is independent by defn

$\emptyset$ is the min wt $= 0$

To deal with min. spanning trees, the underlying graph must be connected.

Then max spanning forest is a max spanning Tree.

Redefine the wt function

as $w(e) = w_{max} - w(e)$

w_{max} is the maximum wt of any edge.