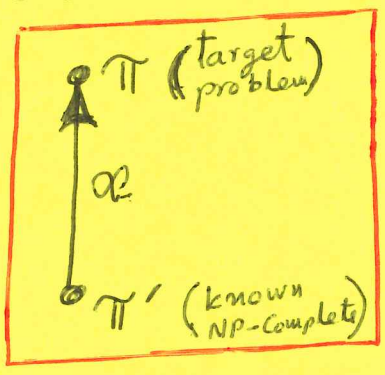
NP-Completeness Proofs by Component Design

Purdue Univ.
Avi Kak

- Of the three different approaches to proving NP-Completeness as mentioned at the beginning of Lecture 31, proofs by component design are the most complex.
- Of the various NP-Completeness proofs we have seen so far, the one for 3DM was arguably the most complicated. That one is a proof by component design. The other proofs by component design that we have gone through already are those for VERTEX COVER and PARTITION.
- The basic idea in a proof by component design is to identify the basic constituent units of the known NP-Complete problem Π', replace these units with structures appropriate to the target problem Π (this replacement may not be uniform as in the proofs by replacement), and then place whatever constraints are necessary on the structures so that we end with an instance of the target problem. The mapping from the Π' instance to a Π instance must obviously obey the fundamental conditions on polynomial transformability.



MINIMUM TARDINESS SEQUENCING (MINTARD):

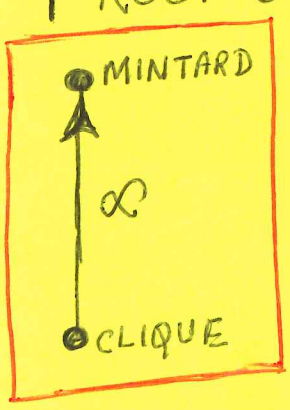
INSTANCE: A set T of tasks, each task $\in T$ having duration of 1 unit and a deadline of $\boxed{d(\text{task}) \in \mathbb{Z}^+}$, a partial order $\lessdot$ on T, and a non-negative integer $K \leq |T|$. [Partial order simply means that we want some tasks to be completed before ~~of~~ some other designated tasks. But such precedence contraints may not be known for all tasks in T.]

QUESTION: Is there a schedule $\boxed{\sigma : T \rightarrow \{0, 1, 2, \ldots, |T|-1\}}$ such that $\boxed{\sigma(\text{task}) \neq \sigma(\text{task}')}$ whenever $\boxed{\text{task} \neq \text{task}'}$; such that $\boxed{\sigma(\text{task}) < \sigma(\text{task}')}$ whenever $\boxed{\text{task} \lessdot \text{task}'}$; and such that

$$\boxed{\left| \{\text{task} \in T : \sigma(\text{task}) + 1 > d(\text{task})\} \right| \leq K}$$

EXPLANATION: We have tardiness in the completion of any task for which $\boxed{\sigma(\text{task}) + 1 > d(\text{task})}$. Note that a task starts at $\sigma(\text{task})$ and every task has a duration of one unit. Also note that $d(\text{task})$ is the deadline for a task. The decision question seeks to know if the number of tasks that are completed past their dead-lines is not greater than K.

PROOF: The fact that MINTARD $\in$ NP is established trivially. For the rest of the proof, we will construct a polynomial transformation from an arbitrary instance of CLIQUE to an instance of MINTARD. Recall from Lectures 29 and 30, an instance of CLIQUE is given by a graph $G = (V, E)$ and a positive integer $J \leq |V|$; and the decision question is whether exists a CLIQUE of size J or greater in G.



- So how do we map a CLIQUE instance to a MINTARD instance? We will set T of the latter to: $\boxed{T = V \cup E}$ and require that the two vertex tasks for a given edge take place before that edge task. That is,

if for vertices u and v we have $\{u, v\}$ as an edge e in E, then we want $task_u \lessdot task_e$ and $task_v \lessdot task_e$. Such a MINTARD instance will be one of the simpler examples of the MINTARD problem. [A more complicated example of MINTARD may involve transitive precedence constraints, such as when $\boxed{task_1 \lessdot task_2}$ and $\boxed{task_2 \lessdot task_3}$. But in the instance of MINTARD generated from the CLIQUE instance we do <u>not</u> have to deal with this additional complexity since the set of vertices is disjoint from the set of edges.]

- With regard to the K parameter of the mapped MINTARD instance, we set it to

$$\boxed{K = |E| - \frac{J(J-1)}{2}}$$

Note that a clique of size J, meaning when a CLIQUE has J vertices, will have $\frac{J(J-1)}{2}$ edges. So we are setting K to the total number of tasks corresponding to the edges that are NOT in the clique portion of G.

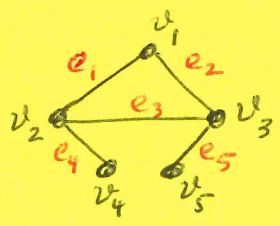
- With regard to the deadlines in the mapped MINTARD instance, we set

$$\boxed{d(task) = \begin{cases} \frac{J(J+1)}{2} & \text{if } task \in E \\ |V| + |E| & \text{if } task \in V \end{cases}}$$

Since $|V| + |E|$ is the total time required for the completion of <u>all</u> tasks, what the above implies is that we will <u>never</u> be tardy for any of the vertex tasks. Therefore, only the edge tasks are in the danger of being tardy.

- In the deadline specification shown above, note that $\frac{J(J+1)}{2}$ will give us time to finish all the tasks that ~~are in the~~ correspond to the vertices <u>and</u> the edges of the clique in G. If the clique has J vertices, then it has $\frac{J(J-1)}{2}$ edges, and $\boxed{J + \frac{J(J-1)}{2} = \frac{J(J+1)}{2}}$.

- Example:



$\boxed{J = 3}$

$T = \{v_1, v_2, v_3, v_4, v_5, e_1, e_2, e_3, e_4, e_5\}$

$v_1 \lessdot e_1$, $v_1 \lessdot e_2$, $v_2 \lessdot e_1$, $v_2 \lessdot e_3$, $v_2 \lessdot e_4$, $v_3 \lessdot e_2$, $v_3 \lessdot e_3$, $v_3 \lessdot e_5$, $v_4 \lessdot e_4$, $v_5 \lessdot e_5$

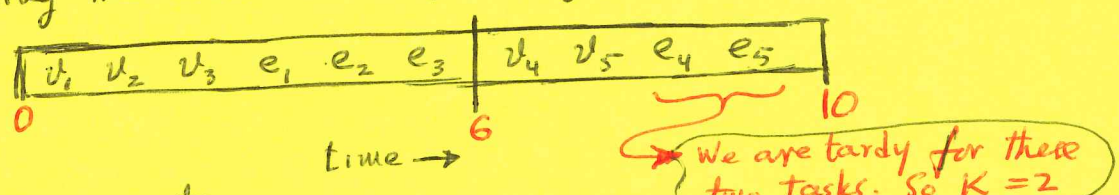
$K = |E| - \frac{J(J-1)}{2} = 2$

$d(e_1) = d(e_2) = d(e_3) = d(e_4) = d(e_5) = 6$

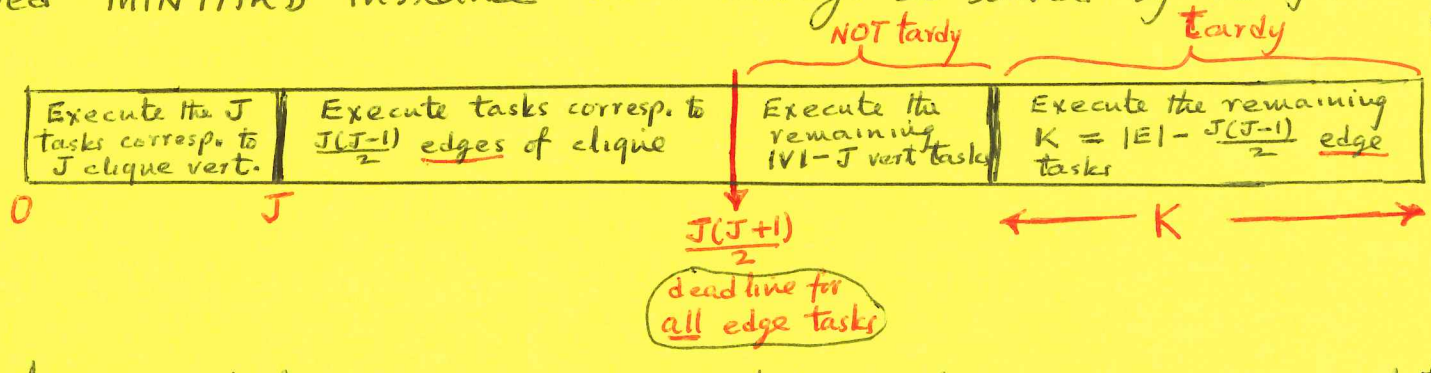
$d(v_1) = d(v_2) = d(v_3) = d(v_4) = d(v_5) = 10$

deadlines

There is a solution to this MINTARD instance: The solution consists of executing the tasks in the following order:



- Therefore, in general, if the CLIQUE instance has a solution, meaning that G contains a clique of size J or greater, then the mapped MINTARD instance can always be solved by the following schedule:



- Now we must argue that if the <u>mapped</u> instance of MINTARD has a solution, then the underlying CLIQUE instance must contain a clique of size J or greater. This argument rests on the fact that the only way to meet the edge deadline $\frac{J(J+1)}{2}$ for as many edges as possible and to also at the same time not violate the precedence constraints on the vertex tasks associated with the edges is to express $\frac{J(J+1)}{2}$ as $J + \frac{J(J-1)}{2}$ with J denoting the vertices and $\frac{J(J-1)}{2}$ the edges ~~that~~ connect them fully in ~~the~~ form of a clique.