Scheduling Unit Time Tasks (16.5)

A unit time task is simply a job which requires one unit of time to complete.

Given a set \( S = \{ a_1, a_2, \ldots, a_n \} \) of \( n \) unit time tasks, a schedule for \( S \) is simply a permutation of \( S \) giving the order in which the tasks are to be performed. We assume that any schedule begins at time \( t = 0 \) and ends at time \( t = n \).

Ex. \( S = \{ a_1, a_2, a_3 \} \) Schedule: \( a_3 \ a_1 \ a_2 \)

\[
\begin{array}{cccc}
& & G_1 & \\
G_3 & G_1 & G_2 & \\
0 & 1 & 2 & 3 \\
\end{array}
\]

Suppose each task \( a_i \in S \) has a deadline \( d_i \) satisfying \( 1 \leq d_i \leq n \), and a penalty \( w_i \geq 0 \) to be paid if \( a_i \) finishes later than its deadline (\( 1 \leq i \leq n \)).

**Problem**

Determine a schedule for \( S \) which minimizes the total penalty for missed deadlines.
Consider any schedule for S. We say a task is **early** in this schedule if it finishes before its deadline, otherwise the task is said to be **late** in the schedule.

A schedule is said to be in **early-first** form if all early tasks are completed before any late tasks are started.

If some late task \( q_j \) is performed before some early task \( q_i \), then upon swapping \( q_i \) with \( q_j \), \( q_j \) is still late and \( q_i \) is still early. Thus any schedule can be placed in early-first form without changing its penalty.

A schedule is said to be in **canonical** form if early tasks precede late tasks, and the early tasks are scheduled in order of increasing deadlines.
One can go from early-first form
to canonical form without changing
the penalty of a schedule.

Proof:
Suppose \( a_i \) and \( a_j \) are early tasks which
finish at times \( t_i \leq d_i \) and \( t_j \leq d_j \)
respectively. And suppose also that \( t_j < t_i \)
and \( d_i < d_j \).

Upon swapping \( a_i \) with \( a_j \) in this schedule,
we see that \( a_j \) finishes at time

\[
t_j' = t_i \leq d_i < d_j
\]

and \( a_i \) finishes at time

\[
t_i' = t_j < t_i \leq d_i.
\]

Hence both tasks are early in the
new schedule, and the penalty for
the new schedule is the same as for
the old.

Clearly any early-first schedule can be
placed in canonical form by performing
swaps of this kind.
Ex. $n = 7$

Task: $a_1, a_2, a_3, a_4, a_5, a_6, a_7$

ei: 4 1 5 2 6 1 6

wi: 1 2 1 1 2 3 1

A schedule with penalty 6:

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\text{late} & \text{late} & & & & & & \\
\end{array}
\]

Early-First Form:

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\text{early} & \text{late} & & & & & & \\
\end{array}
\]

Canonical Form:

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\text{early} & \text{late} & & & & & & \\
\end{array}
\]

An optimal schedule (Penalty = 2):

\[
\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\text{early} & \text{late} & & & & & & \\
\end{array}
\]
Define

\[ \mathcal{I} = \{ A \subseteq S \mid \text{there exists a schedule for } S \text{ in which all tasks in } A \text{ are early} \} \]

Obviously the set of early tasks in some schedule constitutes an independent set.

**Theorem**

\[ M = (S, \mathcal{I}) \] is a matroid.

Observe that the problem of finding a schedule for \( S \) which minimizes the penalty for missed deadlines is equivalent to finding a maximum weight independent set in this matroid.

\[ \begin{array}{cccccccc}
A & & S - A & & \ \ & & t \\
0 & \text{early} & \ & \text{late} & \ & n
\end{array} \]

i.e. we can minimize the fines which must be paid by maximizing the fines which need not be paid.
Define for $t = 0, 1, \ldots, n$ and $A \subseteq S$:

$$N_t(A) = \# \text{of tasks } a_i \in A \text{ such that } d_i \leq t$$

Note that $N_0(A) = 0$ and $N_n(A) = |A|$ for any $A \subseteq S$.

Lemma

Let $A \subseteq S$. The following are equivalent.

1. $A \in \mathcal{I}$

2. $N_t(A) \leq t$ for $t = 0, 1, \ldots, n$

3. If the tasks of $A$ are performed in order of increasing deadlines (starting at $t = 0$ and with no idle time between tasks), then no task is late.

Proof:

$(1) \Rightarrow (2)$

Equivalently we prove the contrapositive:

Not $(2) \Rightarrow$ Not $(1)$. Suppose $N_t(A) > t$ for some $t$. Then there are more than $t$ tasks in $A$ which must finish before time $t$. These tasks cannot be scheduled without at least one of them being late. $\therefore A \notin \mathcal{I}$.
(2) \implies (3)

Assume (2) holds and suppose the tasks in A are scheduled by increasing deadlines. By re-indexing the elements of \( S \) if necessary, we may assume

\[
A = \{a_1, a_2, ..., a_k\} \subseteq S
\]

with deadlines \( d_1 < d_2 < ... < d_k \). Our (partial) schedule is then

\[
\begin{array}{cccccccc}
  a_1 & a_2 & a_3 & \cdots & a_k \\
  \hline
  0 & 1 & 2 & 3 & \cdots & k-1 & k & t
\end{array}
\]

Since \( d_1 = 1 \), Task \( a_1 \) is not late.

But also

\[
\begin{align*}
N_1(A) &\leq 1 \implies d_2 \geq 2 \implies a_2 \text{ not late} \\
N_2(A) &\leq 2 \implies d_3 \geq 3 \implies a_3 \text{ not late} \\
N_3(A) &\leq 3 \implies d_4 \geq 4 \implies a_4 \text{ not late} \\
&\vdots \\
N_{k-1}(A) &\leq k-1 \implies d_k \geq k \implies a_k \text{ not late}
\end{align*}
\]

so no task in A is late.

(3) \implies (1) is obvious.
Exercise:
Write an algorithm which determines whether or not $A \subseteq S$ is independent. (Hint: use part (2) of the preceding lemma, and recall counting sort.)

Exercise:
Write an algorithm which determines a schedule of unit time tasks with minimum total penalty. (Hint: base your algorithm on the greedy algorithm for weighted matroids.)

It remains only to prove that $(Q, \Pi)$ is a matroid.

Proof:
Obviously $S$ is finite and non-empty, and $\Pi$ is a collection of subsets of $S$, so the first axiom is satisfied.

If $B \subseteq A \in \Pi$, then the same schedule in which the tasks of $A$ are early also has the tasks in $B$ early since $B \subseteq A$, thus the hereditary property is satisfied.
To prove the exchange property, let $A, B \subseteq X$ with $|B| > |A|$. We must show $B$ contains a task which extends $A$.

Define

\[ k = \max \{ t \mid 0 \leq t \leq n \text{ and } N_t(B) \leq N_t(A) \} \]

Recall $N_0(B) = N_0(A) = 0$ so the above set is non-empty, whence its maximum $k$ exists. Also note

\[ N_n(B) = |B| > |A| = N_n(A) \]

so that $k < n$. The definition of $k$ says that $N_k(B) \leq N_k(A)$ and

\[ N_t(B) > N_t(A) \text{ for } k < t \leq n \]

In particular

\[ N_{k+1}(B) > N_{k+1}(A) \]

thus

\[ N_{k+1}(B) - N_k(B) > N_{k+1}(A) - N_k(A). \]
This last inequality says that $R$ contains more tasks with deadline $k+1$ than does $A$.

Let $a_i \in R - A$ with $i = k+1$, and define

$$A' = A \cup \{a_i\}.$$ 

We use part (2) of the preceding lemma to show $A' \in \mathcal{I}$.  

By the definition of $A'$ we have $N_t(A') = N_t(A)$ for $0 \leq t \leq k$, and since $A \in \mathcal{I}$ we have $N_t(A) \leq t$.  Thus

$$N_t(A') \leq t \quad \text{for} \quad 0 \leq t \leq k.$$ 

Again by the definition of $A'$ we have $N_t(A') \leq N_t(A) + 1$ for any $t$.  But recall $k+1 \leq n$ implies $N_t(A) \leq N_t(R)$ whence $N_t(A) + 1 \leq N_t(R)$.  Also $N_t(R) \leq t$ since $R \in \mathcal{I}$.  Thus

$$N_t(A') \leq t \quad \text{for} \quad k+1 \leq t \leq n.$$  

$\therefore A' \in \mathcal{I}$ as required.