(5) Introduction to Computational Complexity

Def'n 5.1 (P)
We define $\underset{\sim}{P}$ to be:

$$P := \{\text{All problems } \pi \text{ s.t. there is a poly time algorithm for solving } \pi\}$$

Ex: $1||L_{max} \in P$ since we can use EDD

Def'n 5.2 (Decision Problem)
A decision problem is one that can be posed as a YES-NO question of the input.
The decision problem of an optimization problem asks:
"Is there a solution of objective value at most k?"

Ex:
Suppose our problem is $1||L_{max}$, the decision version of the problem asks:
"Is there a schedule S with $L^S_{max} \le k$?"
(i.e. $1||L_{max} \le k$)

Notation:
We sometimes denote a decision problem by its collection of YES instances.
Ex:

$$\pi_{(1||L_{max} \le k)} = \{\text{All instances of } (1||L_{max} \le k) \text{ s.t. } \exists \text{ schedule of } L_{max}\text{-value at most } k\}$$

Polynomial Reduction:

Def'n 5.3

Given 2 problems A, B, we say that B is polytime reducible to A, denoted by $B \le_p A$, if given an algorithm T for A, we can solve B by using a polynomial number of calls to T + a polynomial # of elementary operations.

Remark 5.1:

Suppose $B \le_p A$, then:

1) If $A \in P$, then $B \in P$
   (i.e. Polytime alg. for A admits a polytime alg. for B)
2) If B does not have a polytime alg (i.e $B \notin P$), then neither does A.

Ex: (Polytime Reductions)

1) $1 || \sum w_j C_j \le_p 1 | r_j | \sum w_j C_j$
   Since $1||\sum w_j C_j$ is the special case with $r_j = 0 \ \forall j$.

   Also: $1 | r_j | \sum C_j \le_p 1 | r_j | \sum w_j C_j$.

2) $(1 | r_j | L_{max} \le k) \le_p 1 | r_j | L_{max}$

3) $1 | r_j | L_{max} \le_p (1 | r_j | L_{max} \le k)$
   However, we need to be careful when searching for k (i.e. use a "smart" algorithm, like binary search)

Scanned with CamScanner

Def^n 5.4 (NP, Verifier)
A decision problem Π is in the class NP if there exists a polynomial verifier V, s.t.:
1) If $x \in \Pi$, there exists a certificate y s.t. $|y| \in poly(x)$ AND $V(x,y) = YES$
2) If $x \notin \Pi$, $V(x,y) = NO$ for all y

Ex:
1) Is $\Pi_{1|r_j|\Sigma C_j \leq k)} \in NP$?
Yes!
Certificate: A schedule S where $\Sigma C_j \leq k$
Verifier: Takes S and checks:
- The schedule is valid
- $\Sigma C_j \leq k$

2) Π_{CLIQUE}
Def^n 5.5 (Clique)
A clique in a graph $G = (V, E)$ is a subset $\mathcal{U} \subseteq V(G)$ s.t. for all $u, v \in \mathcal{U}$ there is an edge from u to v with $uv \in E(G)$.

And so,

$$\Pi_{CLIQUE} := \{(G,k) : G \text{ has a clique of size } \leq k\}$$

Π_{CLIQUE} is in NP.
Certificate: The subset $\mathcal{U}$.
Verifier: For every 2 vertices $u, v \in \mathcal{U}$ check that $uv \in E(G)$

$\blacksquare$

↪ continued.

Scanned with CamScanner

Ex (Continued)

3) Π composite

$$\Pi_{composite} := \{\text{Positive Integers } x \text{ s.t. } x \text{ is composite}\}$$

$\Pi_{composite} \in NP$:

Certificate : p, q integers where $p, q \neq 1$ or x

Verifier : Check $pq = x$.

4) Π prime

$$\Pi_{prime} := \{x \in \mathbb{Z}^+ \mid x \text{ is prime}\}$$

This problem seems harder, but, in fact, $\Pi_{prime} \in NP$, and the certificate is called the "Pratt Certificate".

Claim 5.2: $P \subseteq NP$

[Proof]

Let decision problem $\Pi \in P$. Then, we have an algorithm A s.t. on input x, A returns YES if $x \in \Pi$ and NO if $x \notin \Pi$.

Consider the following verifier:

$$V(x,y) = \begin{cases} YES & \text{if } A \text{ returns YES on } x \\ NO & \text{if } A \text{ returns NO on } x \end{cases}$$

We claim that V shows that $\Pi \in NP$:

1) V runs in polytime, since A runs in polytime

2) Suppose $x \in \Pi$, we want to show that there exists a certificate y s.t. $V(x,y) = YES$.
$y = \emptyset$ is a possible certificate, since $V(x,y) = YES$ for all YES instances

3) Suppose $x \notin \Pi$, then $V(x,y) = NO$ for all y $\square$

Hilroy

Scanned with CamScanner

Def'n 5.6 (NP-hard)
Problem A is NP-hard if $X \leq_p A$ for all $X \in NP$

Def'n 5.7 (NP-Complete)
A (decision) problem Π is NP-Complete if $\Pi \in NP$ and Π is NP-hard

Remark 5.3:
Let Π be NP-Complete:
(1) If $\Pi \notin P$, then $P \neq NP$
(2) If $\Pi \in P$, then $P = NP$
{Proof}
(1) Trivial, if $\Pi \notin P$, then $\Pi \in NP \setminus P$, so $P \neq NP$.
(2) We want to show that $NP \subseteq P$. Let $X \in NP$, now, $X \leq_p \Pi$ and Π is solvable in polytime. Therefore, X is solvable in polytime. So $X \in P$. $\square$

Strategy: (for proving problem Π is NP-Complete)
1) Show $\Pi \in NP$
2) Choose a suitable NP-Complete problem Y
3) Show $Y \leq_p \Pi$

→ Starting menu of NP-Complete problems

Starting Menu of NP-Complete Problems:

1) 3-SAT:

Given a boolean formula:

$$F = C_1 \text{ AND } C_2 \text{ AND } \dots \text{ AND } C_m$$

where each clause:

$$C_i = (y_{i1} \text{ OR } y_{i2} \text{ OR } y_{i3}) \quad i \in \{1, \dots, m\}$$

each:

$$y_{ij} = \begin{cases} x_k \\ \text{NOT } x_k \end{cases} \quad \text{for some } k$$

Is there a setting (assignment) of TRUE/FALSE value to the x_k's under which F evaluates to TRUE?

2) CLIQUE:

Given a graph G and integer k, does G have a clique of size k?

3) SUBSET-SUM:

Given n integers $a_1, \dots, a_n \geq 0$ and a target integer B, is there a subset $S \subseteq \{1, \dots, n\}$, s.t. $\sum_{j \in S} a_j = B$?

4) PARTITION:

Given n integers $a_1, \dots, a_n \geq 0$ s.t. $\sum_{j=1}^{n} a_j = 2k$, where $k \in \mathbb{Z}$, is there $S \subseteq \{1, \dots, n\}$ s.t. $\sum_{j \in S} a_j = k$?

5) 3-PARTITION:

Given 3n integers $a_1, \dots, a_{3n} \geq 0$ with $\sum_{j=1}^{3n} a_j = nk$, $k \in \mathbb{Z}$, is there a partition $S_1, \dots, S_n$ of $\{1, \dots, 3n\}$ s.t.

$$|S_i| = 3 \quad \text{and} \quad \sum_{j \in S_i} a_j = k \quad \forall i \quad ?$$

Showing some scheduling problems are NP-Complete / NP-Hard

We will use ~~...~~ menu and the strategy introduced earlier.

i) $(1|r_j|L_{max} \le 0)$

Theorem 5.4 : $(1|r_j|L_{max} \le 0)$ is NP-Complete

[Proof]

To prove this, we must show:

(i) $(1|r_j|L_{max} \le 0) \in NP$

(ii) Choose $Y \in NP$-Complete, and show $Y \le_p (1|r_j|L_{max} \le 0)$

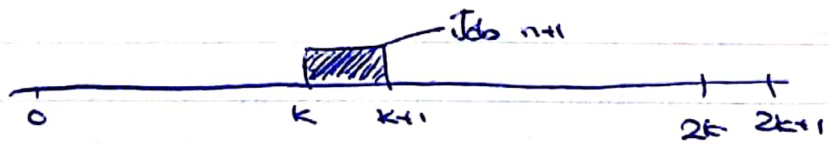
(i) is easy to show, since a polysize certificate for a YES instance is the schedule itself; and the verifier will check if said schedule is feasible and $L_{max} \le 0$

For (ii), we will show PARTITION $\le_p (1|r_j|L_{max} \le 0)$

Let $a_1, \dots, a_n \ge 0$ be integers with $\sum_{j=1}^{n} a_j = 2k$, k integer, be a partition instance.

Create n jobs with $p_j = a_j$ $\forall j = 1, \dots, n$, $r_j = 0$ and $d_j = 2k+1$.

Create job $n+1$ with $p_{n+1} = 1$, $r_{n+1} = k$, $d_{n+1} = k+1$



Cont'

[Proof] (con't)
We use the following lemma to complete the proof.

Lemma 5.4: There is a set $A \subseteq \{1, \dots, n\}$ with $\sum_{j \in A} = k$ iff there is a schedule S with $L_{max} \leq 0$.

[Proof] (Lemma)
($\Rightarrow$) If $\exists A \subseteq \{1, \dots, n\}$ with $\sum_{j \in A} a_j = k$, then, we schedule jobs in A first in $[0, k]$, schedule job $n+1$ in $[k, k+1]$, and jobs $\{1, \dots, n\} \setminus A$ in $[k+1, 2k+1]$. No jobs are late and so $L_{max} \leq 0$.

($\Leftarrow$) S cannot have idle time, and must schedule job $n+1$ in $[k, k+1]$. So, we can take $A = \{j \mid j$ is scheduled in $[0, k]\}$ and then $\sum_{j \in A} a_j = k$.

□ (Lemma + Theorem)

2) $P2 || C_{max}$:
$P2$: 2 identical / parallel machines
C_{max}: Max completion time

Theorem 5.5: $P2 || C_{max}$ is NP-hard
[Proof]
We will show:

$$\text{PARTITION} \leq_p P2 || C_{max}$$

Let $a_1, \dots, a_n \geq 0$ be integers, $\sum_{j=1}^{n} a_j = 2k$ (k integer) be a PARTITION instance.
Create n jobs with $p_j = a_j$ $\forall j = 1, \dots, n$

→ con't

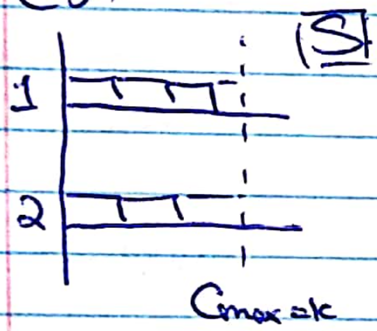
Hilroy

[Proof] (Cont)
We finish the proof with the following lemma.

Lemma 5.5: C_{max} of the above $P2||C_{max}$ instance equals k iff the PARTITION instance is a YES instance

[Proof] (Lemma)
($\Leftarrow$) If $\exists A \subseteq \{1,...,n\}$ with $\sum_{j\in A} a_j = k$, then schedule jobs in A on M/C 1 and jobs in [redacted] $\{1,...,n\} \setminus A$ on M/C 2. This gives a schedule with $C_{max} = k$

($\Rightarrow$)



Since $C_{max} = k$ and $\sum_{j=1}^{n} p_j = 2k$, there cannot be any idle time on either machine in $[0, C_{max} = k]$.

So, we can take:

$$A = \{j : j \text{ is scheduled on M/C } 1\}$$

Then, $\sum_{j\in A} a_j = k$, which is a YES instance of PARTITION.

□ (Lemma + Theorem)

Go [redacted]. continued

3) $1|prec|\sum U_j$:

We define:

$$U_j := \begin{cases} 1 & ; \text{ if } C_j > d_j \\ 0 & ; \text{ otherwise} \end{cases}$$

And so, the problem asks to minimize the number of late jobs

Theorem 5.6: $1|prec|\sum U_j$ is NP-hard

We will show:

$$\text{CLIQUE} \le_p 1|prec|\sum U_j$$

Let $G=(V,E)$, integer k be a CLIQUE instance.
Let $|V|=n$, $|E|=m$, and define $l = \frac{k(k-1)}{2}$

Further:

- For every $u \in V$, create node-job j_u
- For every $e \in E$, create edge-job j_e

And $N = n+m$ = total # of jobs.

- For every edge $e=(u,v)$ of G, create 2 precedence constraints: $j_u \to j_e$, $j_v \to j_e$

And, for the schedule:

- Set $p_j = 1 \ \forall j$
- Set $d_{j_e} = k+l \ \forall e \in E$
- Set $d_{j_u} = N+m \ \forall u \in V$

  (Since there are only $n+m$ jobs, node-jobs will never be late)

(Our goal is to force the schedule from $[0, k+l]$ to correspond to a clique)

(→ Continued

[Proof] (Cont.)

We complete the proof with the following lemma:

Lemma 5.6: G has a clique of size k iff there is a schedule with exactly $m-l$ late jobs

($\Rightarrow$) Let $A \subseteq V$ be a clique of size k. Schedule node-jobs j_u $\forall u \in A$ in $[0, k]$, and edge-jobs j_{uv} $\forall (u,v) \in A$ in $[k, k+l]$, and the remaining jobs arbitrarily in $[k+l, N]$. This gives l late jobs

($\Leftarrow$) Let S be a schedule with $m-l$ late jobs, so l edge jobs are scheduled in $[0, k+l]$. Define:

$$A = \{u \in V : j_u \text{ is scheduled in } [0, k+l]\}$$

$|A| \leq k$, since l edge jobs are scheduled in $[0, k+l]$.

By our precedence constraints for every edge job j_{uv} scheduled in $[0, k+l]$, we must have that j_u and j_v are also scheduled in $[0, k+l]$, so $u, v \in A$.

And we have that $l = \frac{k(k-1)}{2}$, which can only be satisfied if $|A| = k$, so A is a clique of size k.

$\square$