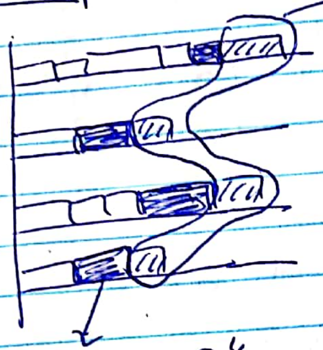
454-16

$P||\sum C_j$:

→ Call this "slot 1" on m m/c's



▨ = "slot 2", and etc.

A job j is assigned to slot k on m/c i means that there are $k-1$ jobs on m/c i after job j.

Why is this useful?: Every job j assigned to slot 1 on m/c i contributes p_j to our objective. And, in general, if j is assigned to slot k on a m/c, it contributes $k \cdot p_j$ to $\sum C_j$. (Since ~~there~~ it contributes p_j to its completion time, and $(k-1)p_j$ to the jobs that come after it on its m/c).

Goal: Assign jobs to slots so as to minimize

$$\sum_{j \to \text{slot } k} k p_j.$$

"Assign largest m jobs to m slot 1's, Next batch of m largest jobs to m slot 2's, and so on."

i.e. Sort jobs so that $p_1 \geq \dots \geq p_n$. Let $J = \{1, \dots, n\}$.
$k = 1$,

Repeat until $J = \emptyset$:
- Schedule min$(m, |J|)$ jobs from J in first slot k of the m m/c's (arbitrarily)
- $J \leftarrow J \setminus \{$ first min$(m, |J|)$ jobs of $J\}$
- $k \leftarrow k+1$.

This is clearly polynomial in time.

Theorem 1: The above algorithm produces an optimal schedule for $P||\sum C_j$.
[Proof] Exercise

Exercise: Show that running LIST-SCHEDULING with ~~[illegible]~~ jobs sorted in increasing p_j order mimics the above algorithm and produces an optimal schedule.

$R||\sum C_j$:
(Recall: job j takes time p_{ij} to be processed on m/c i)

If $j \mapsto$ Slot(i,k), then there are $(k-1)$ jobs following j on m/c i, and j contributes $k \cdot p_{ij}$ to $\sum C_j$.

Then, our task can be restated as:

min $\sum C_j \equiv$ ~~[illegible]~~ Job $\rightarrow$ slot assignment problem
Assign jobs to (i,k) slot to
$$\min \sum_{j \mapsto (i,k)} k p_{ij}.$$

Scanned with CamScanner

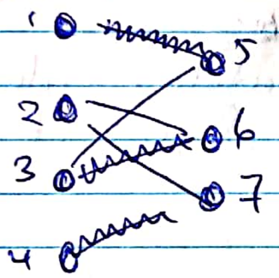
Excursion into matchings in bipartite graphs:

Let $G = (V := A \cup B, E)$ be a bipartite graph with bipartition $A \cup B$.

Defn: (Matching, S-perfect)

- A matching M is a set of edges s.t. no 2 edges of M share an endpoint
- Given $S \subseteq V$, a matching M is said to be S-perfect, if $\forall v \in S$, $\exists e \in M$ incident to v.

Ex:



$\sim\sim$ - matching M.

For $S = \{1, 5, 6, 7\}$, M is S-perfect.

For $S = \{2\}$, M is not S-perfect.

Defn (Fractional Matching, Fractional S-perfect)

A fractional matching x is an assignment $\{x_e \geq 0\}_{e \in E}$ of numbers to edges s.t.

$$\sum_{e \in \delta(v)} x_e \leq 1 \quad \forall v \in V. \quad (\delta(v) : \text{set of edges incident to } v).$$

Further, a fractional matching x is S-perfect (for $S \subseteq V$) if

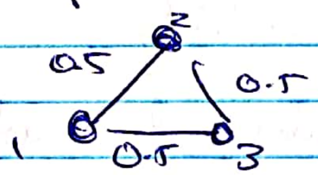
$$\sum_{e \in \delta(v)} x_e = 1 \quad \forall v \in S.$$

Note: If $x \in \{0,1\}^E$, then fractional matching corresponds to matchings and similarly for S-perfect matching.

Fact 1: Given edge costs $\{c_e\}_{e \in E}$, and any $S \subseteq V$, we can decide if $\exists$ S-perfect matchings and if so compute a min cost S-perfect matching in poly time.
Cost of matching $M = \sum_{e \in M} c_e$

Fact 2: In a bipartite graph, given edge costs $\{c_e\}_{e \in E}$ and a fractional S-perfect matching x, we can compute (in polytime) an S-perfect matching M of cost $\leq \sum_e c_e x_e$.

(In particular, an S-perfect matching always exists if there is a fractional S-perfect matching)
(Ex. Counter-example:



x: Fractional $\{1,2,3\}$-perfect matching.

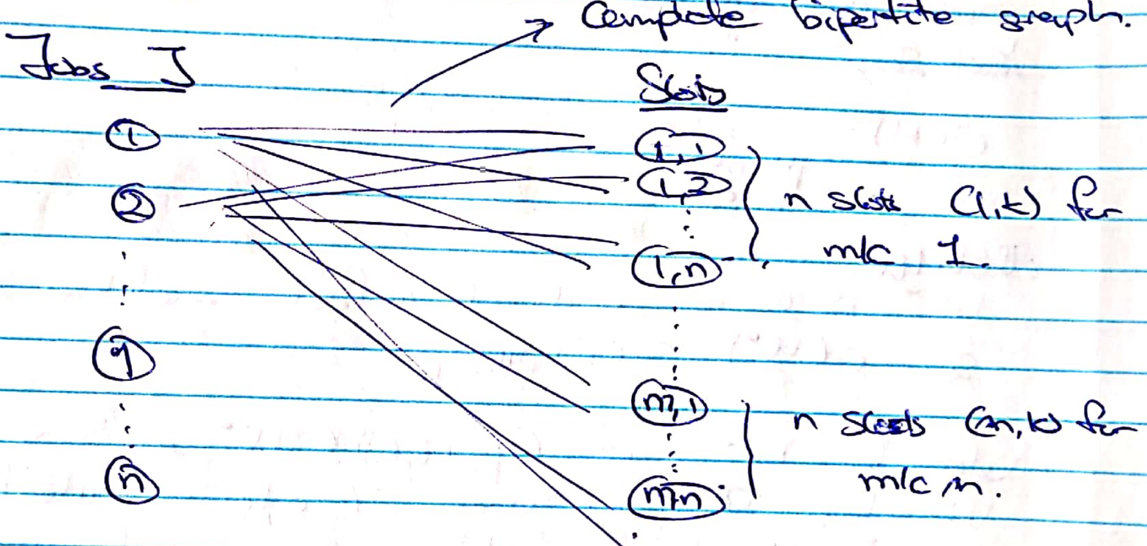
But, there is no way to obtain a $\{1,2,3\}$-perfect matching.)

i.e. The assumption of bipartite graphs is important.

(Back to $R||\sum C_j$)
(Recall we want to solve:
assign jobs to slots to minimize $\sum_{j \mapsto (i,k)} k p_{ij}$)

Strategy: We will view this as a min-cost S-matching problem in a bipartite graph.

Bipartite graph:



Create node-jobs j for all jobs, and node (i,k) for each slot $\forall i=1,\dots,m$, $k=1,\dots,n$.

The cost on edge $(j,(i,k))$ $\forall$ jobs j, slot (i,k) is $k \cdot p_{ij}$.

Theorem:
A min cost J-perfect matching M^* yields an optimal schedule for $R||\sum C_j$ where $j \mapsto (i,k)$ iff $(j,(i,k)) \in M^*$.

[Proof]

Suppose we have a schedule S. Then, consider the edge set $M = \{(j, (i,k)) : j$ is scheduled in slot (i,k) in $S\}$

Then M is a J-perfect matching since every job is assigned to exactly 1 slot and every slot is assigned at most one job.

So,

$$C(M) = \sum_{\substack{j \mapsto (i,k) \\ \text{in } S}} kp_{ij} = \sum C_j^S$$

M^* is a min-cost J-perfect matching, so $C(M^*) \leq OPT_{R||\sum C_j}$

(Note: In M^* if $(j, (i,k)) \in M^*$, then $\forall k' < k$, can assume $\exists$ edge incident to (i,k') in M^*, since otherwise, we can replace $(j, (i,k))$ in M^* by $(j, (i,k'))$ to get lower cost J-perfect matching)

So, schedule constructed by $j \mapsto (i,k)$ iff $(j, (i,k)) \in M^*$ is a valid schedule.

(i.e. $j \to (i,k)$ means $k-1$ jobs follow j on m/c i) and $\sum C_j$ of schedule $= C(M^*)$

$\leq OPT_{R||\sum C_j}$

$\square$