Lec 19: Scheduling ~~[illegible]~~ Algorithms

Today we'll do several different scheduling problems, both to give you a quick overview of the kind of problems, and also the kind of techniques.

(1) A simple problem: single machine, release dates, min sum of completion times.

often written as $1 | r_j | \sum_j C_j$

Idea: Solve the "preemptive" problem first, and then use that to solve the general non-preemptive problem.

Preemptive problem solvable in poly time:-

- At each time work on the problem with shortest remaining processing time. (SRPT).
- An exchange argument says this is optimal.

- Non-preemptive problem: say preemptive schedule gives order
$$\bar{C}_1 \le \bar{C}_2 \le \dots \le \bar{C}_n$$

Schedule the jobs in this order. If job j not released yet, wait! ☺

Claim: this is a non-preemptive schedule with $C_j \le 2\bar{C}_j$.

Pf: take the preemptive schedule. Schedule a second copy of the job j once it is finished processing. Clearly, move C_j by $\sum_{i \le j} p_i \le \bar{C}_j$, so $C_j \le 2\bar{C}_j$. $\Rightarrow$ 2 approximation ☺

2 Lets Consider a more difficult problem $\boxed{1 \mid prec \mid \sum_i w_j C_j}$

jobs with procedences. (no release dates, though those will work too with a little more effort).
also weights

Each job j has :- w_j weight/importance
p_j processing requirement

and an order $\prec$ st $i \prec j$ means job i must finish before j starts.

Relaxation: vars are C_j. $\min \sum_i w_j C_j$

want $C_j \geq C_i + p_i$ $\forall$ jobs $i \prec j$

for other pairs of jobs would like

$C_j \geq C_k + p_k$ or $C_k \geq C_j + p_j$

But is not a convex constraint!
not linear, definitely

So here's a different constraint :-

$\forall$ sets of jobs $S \subseteq [n]$, define $p(S) = \sum_{j \in S} p_j$

$p^2(S) = \sum_{j \in S} p_j^2$.

Claim: this is a "valid" constraint.

Claim: $\sum_{j \in S} p_j C_j \geq \frac{1}{2}\left[p(S)^2 + p^2(S)\right]$. $\forall S \subseteq [n]$.

for ~~any schedule~~ $C_1 \ldots C_n$ arising from any ~~valid~~ schedule

Pf: Sps. $S = \{1 \ldots m\}$. and sps. $C_1 \leq C_2 \leq \ldots \leq C_m$

then $C_j \geq \sum_{i \leq j} p_i$ and $\sum_{j \in S} p_j C_j \geq \sum_{j=1}^{m} \sum_{k=1}^{j} p_j p_k \geq \frac{1}{2}\left[p^2(S) + p(S)^2\right]$

∎

So here's a linear program

$$\min \sum_i w_j C_j$$

$$\text{st. } C_j \geq C_i + p_i \quad \forall \text{ pair } i < j$$

$$\cdot \sum_{j \in S} p_j C_j \geq \frac{1}{2}\left[p^2(S) + p(S)^2\right] \quad \forall\, S \subseteq [n] \text{ jobs.}$$

Exponentially large LP! But that's OK, we'll see later how to solve it

Sps we get a solution to this, saying $\bar{C}_1 \leq \bar{C}_2 \leq \cdots \leq \bar{C}_n$

renumber jobs so that the solution of LP have increasing completion times.

Now schedule the jobs in this order.

<u>Claim</u>: for each job j, $C_j \leq 2\bar{C}_j$

<u>Pf</u>: Say $S = \{1, 2 .. j\}$

Then $C_j = p_1 + p_2 + \cdots p_j = p(S)$.

but $\sum_{i \in S} p_i \bar{C}_i \geq \frac{1}{2} p(S)^2$

$$\wedge\!\!|$$

$$\sum_{i \in S} p_i \cdot \bar{C}_j = p(S) \cdot \bar{C}_j$$

b/c we sorted in the order of increasing $\bar{C}_j$

$$\Rightarrow \bar{C}_j \geq \frac{1}{2} p(S) = \frac{1}{2} \cdot C_j .$$ ☺

Hence each job finishes at most twice later than in the fractional schedule.

<u>Summary</u>: find "valid" inequalities ~~for the~~ satisfied by the schedule you are looking for and add them in. (<u>strengthen</u> the LP!)

Will revisit this idea again very soon!

3. The Generalized Assignment Problem

- n jobs. m machines.

  each job j has some size $p_{ij} \geq 0$ when scheduled on m/c i.

- Want to schedule jobs on machines to ~~minimize~~ have low "makespan"

$$\max_{i \in [m]} \underbrace{(\text{sum of } p_{ij} \text{ of jobs assigned to } i)}_{\text{load}(i)}$$

Called "makespan"

- Moreover also given a set of costs c_{ij}, the assignment cost is

$$\sum_{ij} c_{ij} \, \mathbb{1}(\text{job } j \text{ assigned to machine } i).$$

Let's first try to minimize makespan (and ignore assignment costs)

Attempt #1:

$$\min \; L$$
$$\text{st} \quad L \geq \sum_j p_{ij} x_{ij} \qquad \forall i$$
$$\sum_i x_{ij} = 1 \qquad \forall j$$
$$x_{ij} \geq 0.$$

Is this good? No. Say $\exists$ 1 job of size $p_{ij} = 1 \; \forall i$

then set $x_{ij} = 1/m \; \forall i$ and set $L = 1/m$.. but $OPT = 1$.

☹

Idea: Enumerate.

"Guess" OPT Load L^*. Now just set $x_{ij} = 0$ $\forall ij$ st $p_{ij} > L^*$.

And then find a feasible solution to
$$\left\{ \begin{array}{ll} \sum_j p_{ij} x_{ij} \le L^* & \forall i \\ \sum_i x_{ij} \ge 1 & \forall j \end{array} \quad x \ge 0. \right\}$$
$x_{ij} = 0$ if $p_{ij} > L^*$.

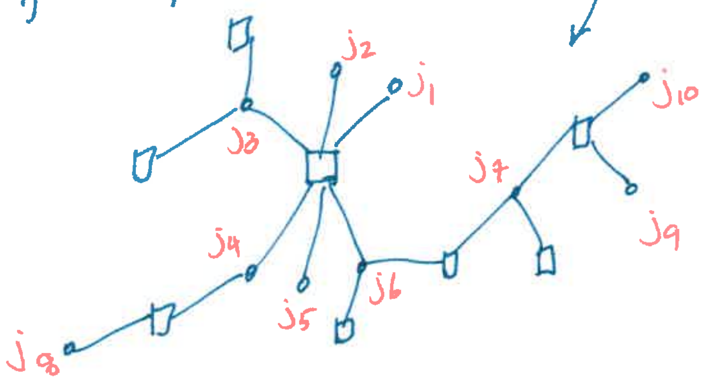
Let's try to find a solution of cost $\le 2L^*$.

Note: OPT is a solution to this LP

Approach 1:

(a) Show that OPTimal x has no cycles in its support. [HW ?]

(b) If its a forest, then looks like for each component.



~~Root the component at some j~~

How would you assign the jobs to machines.

Some jobs go only to a single m/c. Easy for them. (eg $j_1\, j_2\, j_5$ / $j_8\, j_9\, j_{10}$)

Remaining tree has all jobs as internal nodes, m/c as leaves.

Root component @ some ~~job~~ m/c, assign each job to parent.

So each m/c gets one extra job $\max_{x_{ij}>0} p_{ij} \le L^*$.

So total load $\le 2L^*$.

or really $\sum_j p_{ij} x_{ij} + L^*$. for m/c i.

Another Approach [Shmoys and Tardos].

Use integrality of the perfect matching polytope.

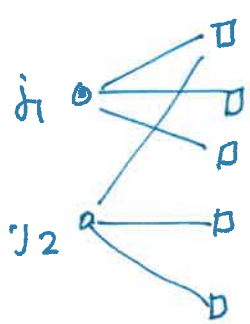
Know that the polytope $K = \{ y \geq 0 \mid \sum_i y_{ij} \geq 1 \ \forall j, \ \sum_j y_{ij} \leq 1 \ \forall i, \ y_{ij} = 0 \text{ for some } ij \in S \}$

can write any $y \in K$ as convex combo of integer matchings $\Leftarrow$ has all vertices / extreme points integral i.e. if y is an extreme pt of K then $y \in \{0,1\}^{m \times n}$.

How to use this? Here we have $\{ \sum_j p_{ij} x_{ij} \leq L^*, \ \sum_i x_{ij} \geq 1, \ x \geq 0 \}$. $x_{ij} = 0$ for $ij \in S$

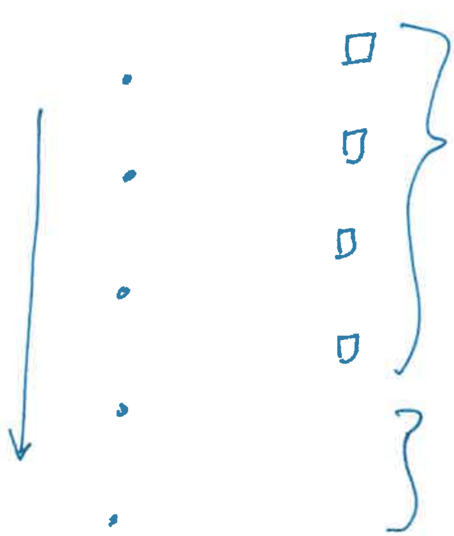
annoyance!

So do the following



each machine gets some "mass" $\sum_j x_{ij} = M_i$

Split machine i into $\lceil M_i \rceil$ mini-machines



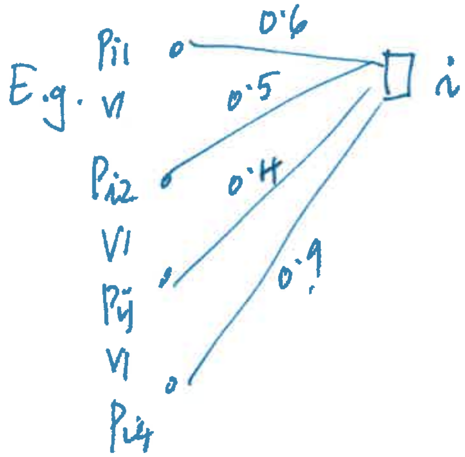
sort jobs in order of decreasing p_{ij} values (so diff for diff m/cs).

$M_i = 3.5$
$\Rightarrow 4$ m/cs.

Assign

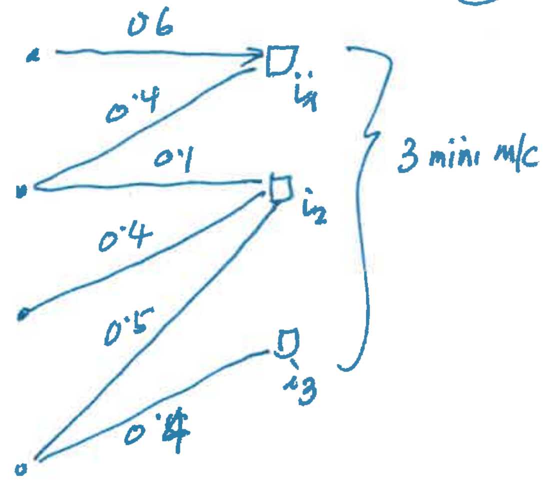
And assign 1 unit of x_{ij} to 1st m/c, 1 unit to 2nd ..., $M_i - \lfloor M_i \rfloor$ to last

in this order



(x solution)

$\Longrightarrow$ $M_i = 2.4$



(y solution)

Converted into instance where

- each job goes to ≥ 1 mini m/c fractionally
- each mini m/c gets ~~≤1~~ 1 unit of "mass".

$y_{ij} = 0$ for many of the edges.

<u>Now</u>: can ~~get an~~ write y as combo of integer matchings in K.
(i.e. whose support is same as support of y).

~~What~~ this means each mini m/c gets ≤ 1 job, each job to ≥ 1 mini m/c.
What does the load of i become?

for all but 1st mini m/c : load of job on $i_k \leq$ max size of job assigned fractionally to it
$\leq \sum p_{ij} y_{ij}$ of jobs assigned to previous mini m/c.

$\Rightarrow$ ~~load~~ size of jobs on all mini m/cs corresponding to i (except first)
$\leq$ fractional load on $i \leq L^*$.

size of job on 1st mini m/c $\leq \max\limits_{x_{ij}>0} p_{ij} \leq L^*$

$\Rightarrow$ at most $2L^*$ ☺

General Idea:

- Structure of LP solutions helped!
- Useful to try add inequalities to strengthen the LP!
  - the exponential many $\sum_i p_i C_i \geq \frac{1}{2}(p^2(S) + p(S)^2)$. type
  - or even the $x_{ij} = 0$ if $p_{ij} > L^*$.
- Guessing a parameter of OPT's sol^n (say L^*) can be useful.

Next time (x2)

- How to use the structure of the optimal LP solution (once again)
  - in a different way.
- How to add valid constraints to strengthen the LP automatically
  - "hierarchies", "Lift and project", "knapsack cover".