July 10th 2018

COMS4(

R||Cmax:
Consider the following LP-relaxation, which is a variant of the LP for R|pmtn|Cmax.

(LP)

min C_{max}

s.t.

(1) $\sum_{i=1}^{m} x_{ij} = 1 \quad \forall j = 1, \ldots, n$ (Ideally, $x_{ij} = 1$ if $j \to i$, and 0 otherwise)

(2) $\sum_{j=1}^{n} p_{ij} x_{ij} \leq C_{max} \quad \forall i = 1, \ldots, m.$

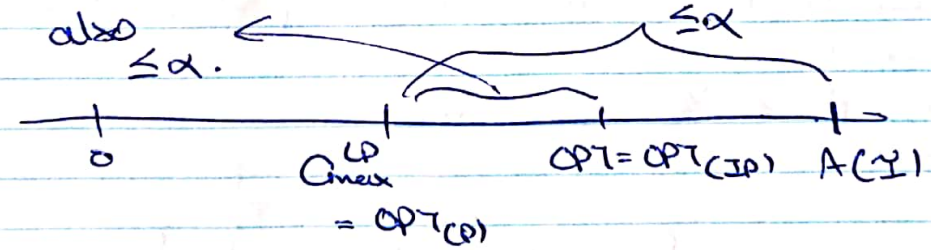
(3) $x, C_{max} \geq 0$

Observations:
(i) Any integral sol'ns to (P) corresponds to a schedule for R||Cmax. So:

OPT = {min C_{max} s.t. (1), (2), (3) and x_{ij} integral $\forall i, j$}

(IP)

(ii) (P) is the LP-relaxation of (IP)



Suppose we have an α-approx. alg A that for every instance I returns a solution of value $A(I) \leq \alpha \cdot C_{max}^{LP}(I)$
i.e. It proves an α-approx. using the LP-optimum $C_{max}^{LP}(I)$ as the lower bound on $OPT(I)$.

(cont'd)

Want to understand: What is an inherent lower bound on α?

$$OPT(I) \leq A(I) \leq \alpha \cdot C_{max}^{LP}(I)$$

$$\Rightarrow \frac{OPT(I)}{C_{max}^{LP}(I)} \leq \alpha.$$

$$\Rightarrow \alpha \geq \max_{\text{instances } I} \frac{OPT(I)}{C_{max}^{LP}(I)}$$ → let's call this β.

Integrality gap of (P)

So, any algorithm that uses C_{max}^{LP} as a lower bound to prove an approx. guarantee must have approximation factor at least β.

Claim: β is "large" for (P)

Proof:

Consider an instance of $R||C_{max}$ with one job with $p_{ij} = 1 = p_{i'j}$ $\forall i = 1, \ldots, m$

Clearly $OPT(I) = 1$. But, there is a feasible LP sol'n with $x_{ij} = 1/m$ $\forall i = 1, \ldots, m$ and $C_{max} = 1/m$.

So, $\beta \geq \frac{1}{1/m} = m$.

17

(Summary: This LP sucks, and doesn't provide anything meaningful.)

How can we fix (P)?

- Would like to add to (P) the constraint:

$$x_{ij} > 0 \Rightarrow C_{max} \geq p_{ij} \quad (*)$$

But this is not a linear constraint.
We can move to the decision version of $R||C_{max}$ i.e.

Given target makespan D, ~~decide~~ decide whether there is a schedule for $R||C_{max}$ of makespan $\leq D$?

Now we can incorporate (*) as follows:
Define $F = \{(i,j) : p_{ij} > D\}$.
Consider the feasibility LP:
(LP_D)

$$\sum_{i=1}^{m} x_{ij} = 1 \quad \forall j = 1, \ldots, n$$

$$\sum_{j=1}^{n} p_{ij} x_{ij} \leq D \quad \forall i = 1, \ldots, m.$$

$$x \geq 0$$

$$x_{ij} = 0 \quad \forall (i,j) \in F. \quad \leftarrow \text{This is new!}$$

Observation:
If $D \geq OPT(I)$, (LP_D) is feasible
(More generally, if the answer to the decision problem is YES, (LP_D) is feasible).

Theorem 1:
If (LP_D) is feasible, we can efficiently ~~produce~~ find a schedule of makespan $\leq 2D$.

Omit

Theorem 1 leads to 2-approx for R||Cmax:
(Assume all P_{ij}'s are integers, so OPT is also an integer)
- Use binary search to find smallest integer D (call this D^*) s.t. (LP_D) feasible
(Observation $\Rightarrow D \le OPT$)
- Use theorem 1 to obtain a schedule of makespan $\le 2 \cdot D^* \le 2 \cdot OPT$.

Running time:

(time taken for binary search) + (time to implement theorem 1)

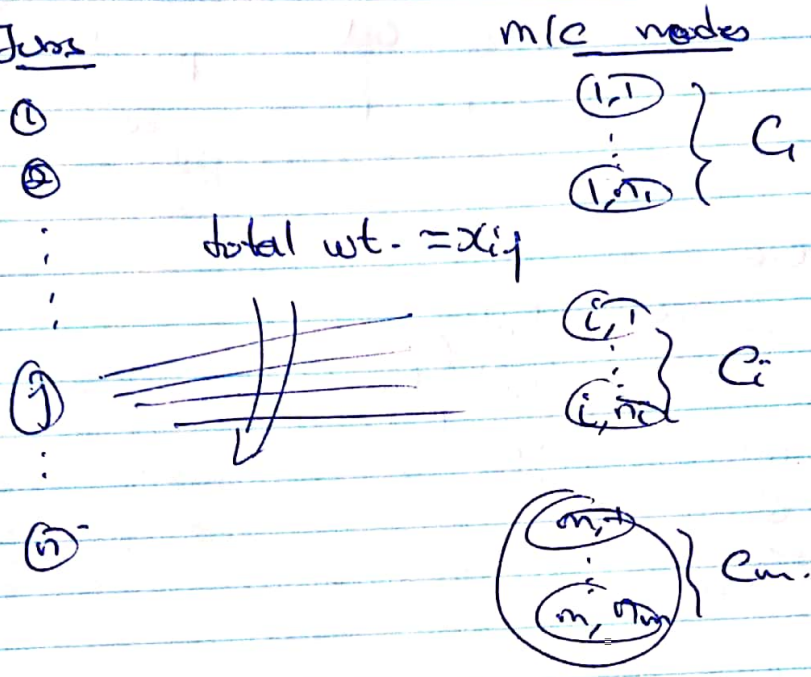
= ((poly # iterations of binary search) * (time to solve (LP_D)))
+ poly time
= poly time.

[Proof] (of Theorem 1)
Let $x = (x_{ij})_{\substack{i=1,\dots,m \\ j=1,\dots,n}}$: feasible solution to (LP_D)
We will build a bipartite graph with job nodes J and some nodes for m/c's s.t.
1) x translates to a fractional J-perfect matching, and.
2) Any J-perfect matching gives a schedule of makespan ~~at most~~ $\le 2D$.

Define $n_i = \lceil \sum_{j=1}^{n} x_{ij} \rceil$ = # jobs x assigns to m/c i.
Since # $\in [n_{i-1}, n_i]$

Create for each m/c i, n_i nodes $(i,1) \ldots (i,n_i)$ = copies

$\underbrace{\qquad\qquad}_{C_i}$



Will have edges b/w j and some copies of m/c i and assign some wts. $w_{j,(i,c)}$ on these edges

Actual bipartite graph ≡ edges $(j, (i,c))$ s.t $w_{j,(i,c)} > 0$

wts. assigned to $(j, (i,c))$ edges $\forall c = 1, \ldots, n_i$ will have the property that:

$$\sum_{c=1}^{n_i} w_{j,(i,c)} = x_{ij} \quad \forall j, i$$

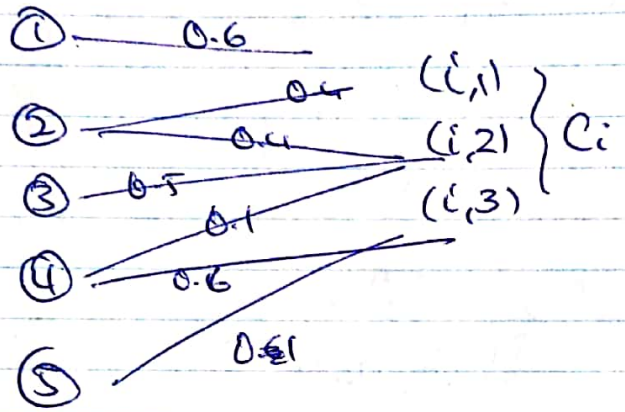
Defining $w_{j,(i,c)}$ wts for a fixed m/c i, but for all $j = 1, \ldots, n$, $\forall c = 1, \ldots, n_i$.

- Order the jobs $J_i = \{j : x_{ij} > 0\}$ in ↓ order of p_{ij}
- "Pack the x_{ij}'s on the n_i copies of m/c i" by using McNaughton's wrapround rule and considering jobs in J_i in above sorted order

Ex:

	1	2	3	4	5
x_{ij}	0.6	0.8	0.5	0.7	0.1
p_{ij}	8	4	3	1	1

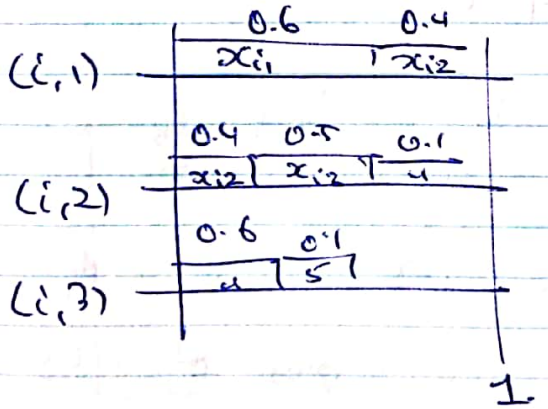
$\sum_{j=1}^{5} x_{ij} = 2.7$, so $n_i = 3$



- Think of each copy $C(i,q)$ as an mk
- Each $j \in J_i$ creates a "job" with length x_{ij}.

Pack x_{ij}'s on n_i copies → each copy can take ≤ 1 job.

Here, sorted order: 1, 2, 3, 4, 5



$W_{j,(i,c)}$ = amount of j assigned to any c under packing given by McNaughton's wraparound rule

Observations:

1) $\sum_{c \geq 1}^{n_i} w_{j,(i,c)} > 0 \Rightarrow x_{ij} > 0$

2) $\sum_{c=1}^{n_i} w_{j,(i,c)} = x_{ij}$

3) $w_{j,(i,c)} > 0$ for at most 2 (consecutive) copies c

4) For every copy $c > 1$, every j with $w_{j,(i,c)} > 0$ has $p_{ij} \leq p_{ik}$ for all jobs k with $w_{k,(i,c-1)} > 0$

Also $w_{j,(i,c)} > 0$ we have $\sum_k w_{k,(i,c-1)} = 1$.

so, $p_{ij} \leq \sum_k p_{ik} w_{k,(i,c-1)}$.