July 3rd, 2018

~~CO~~ CO454

Recall: (Algorithm A) $(*)$

For $t = 0, 1, 2, \ldots, \Delta_0$

Let A_t = matrix of remaining processing times

Let Δ_t = max {row or column sum of A_t}

Let $I_t = \{i \in I : \text{sum of row } i \text{ in } A_t = \Delta_t\}$

Let $J_t = \{j \in J : \text{sum of col } j \text{ in } A_t = \Delta_t\}$

Let G_t = bipartite graph on $I \cup J$ s.t. $ij \in E(G_t)$ iff $[A_t]_{ij} > 0$

Compute M_t : a $(I_t \cup J_t)$-perfect matching in G_t

Schedule jobs according to M_t from time t to $t+1$ $(**)$

Recall:

We were finishing $O|pmtn|C_{max}$.

$\Rightarrow$ We can reduce $R|pmtn|C_{max} \leq_p O|pmtn|C_{max}$

Claim (from last class)

Algorithm A computes a schedule of makespan Δ_0

[Proof]

We showed that since $\Delta_{t+1} = \Delta_t - 1 \Rightarrow \Delta_t = \Delta_0 - t$

$\Rightarrow$ @ time Δ_0, all jobs are completed

$$C^*_{max} \geq \sum_{i \in I} P_{ij} \quad \forall j \in J$$

$$C^*_{max} \geq \sum_{j \in J} P_{ij} \quad \forall i \in I$$

$$C(LB) = \max\left\{ \max_{j \in J} \sum_{i \in I} P_{ij},\ \max_{i \in I} \sum_{j \in J} P_{ij} \right\}$$

Issues to resolve:

(1) @ time t, why does a $J_t \cup I_t$-perfect matching exist?

(2) When can we reuse M_t as M_{t+1}?

Goal: Bound # of matchings we use $\leq mn + m + n$.

Idea from last class:

What prevents M_t from being a $(I_{t+1} \cup J_{t+1})$-perfect matching in G_{t+1}?

$\leq mn$ 1) Complete a job if $p_{ij} = 1$.

$\leq m$ 2) $I_{t+1} \neq I_t$, there's a new tight machine

$\leq n$ 3) $J_{t+1} \neq J_t$, there's a new tight job

How long to use M_t?

"Run" the schedule given by M_t for a time interval of length δ ~~...~~ UNTIL

- one of the p_{ij}'s goes to 0,
- $\Delta_t - \delta$ becomes equal to row $\hat{i}$'s sum for $\hat{i}$ not matched by M_t
- $\Delta_t - \delta$ becomes equal to column $\hat{j}$'s sum for $\hat{j}$ not matched by M_t

Now, we can replace (in Alg ~~...~~ A)

→ (*) w/ : Let $t=0$, while $t < \Delta_0$

→ (**) w/ :

Compute:

$$\delta = \min \begin{cases} \min\limits_{ij \in M_t} p_{ij} \\ \min\limits_{\hat{i} \text{ not matched by } M_t} \left(\Delta_t - \text{row sum}_{A_t}(\hat{i}) \right) \\ \min\limits_{\hat{j} \text{ not matched by } M_t} \left(\Delta_t - \text{col sum}_{A_t}(\hat{j}) \right) \end{cases}$$

Schedule according to M_t from time t to $t+\delta$. Advance $t \leftarrow t+\delta$

Claim:
We can always find a $(I_t \cup J_t)$-perfect matching in G_t at time t

[Proof]
If G is bipartite, recall given fractional S-perfect matching, can find integral S-perfect matching

Consider:

$$x_{ij} = \frac{p_{ij}}{\Delta_t} \geq 0$$

For $i \in I$

$$\sum_{j \in J} x_{ij} = \frac{1}{\Delta_t} \underbrace{\sum_{j \in J} p_{ij}}_{\leq \Delta_t} \leq 1 \quad \text{and } =1 \text{ iff } i \in I_t.$$

For $j \in J$:

$$\sum_{i \in I} x_{ij} = \frac{1}{\Delta_t} \underbrace{\sum_{i \in I} p_{ij}}_{\leq \Delta_t} \leq 1 \quad \text{and } =1 \text{ iff } j \in J_t.$$

(This is a fractional S-perfect matching, so there is an integral S-perfect matching)

Ex:

Iteration 1 (t=0)

$$P = \begin{bmatrix} 2 & 1 & 0 & 2 \\ 0 & 4 & 1 & 0 \\ 2 & 0 & 0 & 2 \end{bmatrix} = A_0$$

1. Δ_0 = max rowsum or colsum

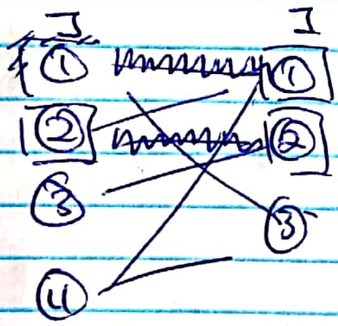
$$\begin{array}{cccc|c} \multicolumn{4}{c|}{\begin{bmatrix} 2 & 1 & 0 & 2 \\ 0 & 4 & 1 & 0 \\ 2 & 0 & 0 & 2 \end{bmatrix}} & \begin{matrix} =5 \\ =5 \\ =4 \end{matrix} \\ =4 & =5 & =1 & =4 & \end{array}$$

So, $\Delta_0 = 5$

2. $I_0 = \{1, 2\}$
3. $J_0 = \{2\}$
4. M_0

(J, I)

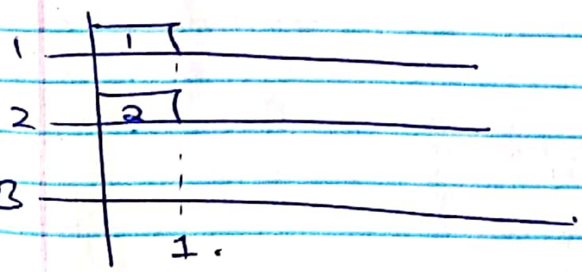
$M_0 = \{(1,1), (2,2)\}$



5. $\delta = \min \begin{cases} \min\{2, 4\} \\ \min\{5-4\} \\ \min\{5-1, 5-4\} \end{cases} = 1$

P(1,1)

P(2,2)



cont

Iteration 2 (it=1)

$$A_1 = \begin{array}{c} \begin{bmatrix} 1 & 1 & 0 & 2 \\ 0 & 3 & 1 & 0 \\ 2 & 0 & 0 & 2 \end{bmatrix} \begin{array}{l} =4 \\ =4 \\ =4 \end{array} \\ \begin{array}{cccc} \shortparallel & \shortparallel & \shortparallel & \shortparallel \\ 3 & 4 & 1 & 4 \end{array} \end{array}$$

$\Delta_1 = 4$

$I_1 = \{1, 2, 3\}$

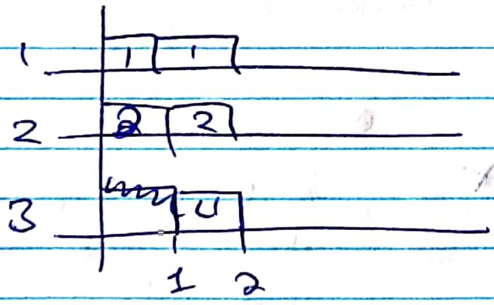
$J_1 = \{2, 4\}$

$M_1 =$ (See circles)

(J, I)

→ So, we have: $\{(1,1), (2,2), (3,4)\}$

$$\delta = \min \begin{cases} \min\{1,3,2\} \\ \cancel{\min\{} \ \infty \\ \min\{4-1\} \end{cases} = 1.$$



Iteration 3:

$$A_2 = \begin{array}{c} \begin{bmatrix} 0 & 1 & 0 & 2 \\ 0 & 2 & 1 & 0 \\ 2 & 0 & 0 & 1 \end{bmatrix} \begin{array}{c} 3 \\ 3 \\ 3 \end{array} \\ \begin{array}{cccc} 2 & 3 & 1 & 3 \end{array} \end{array}$$

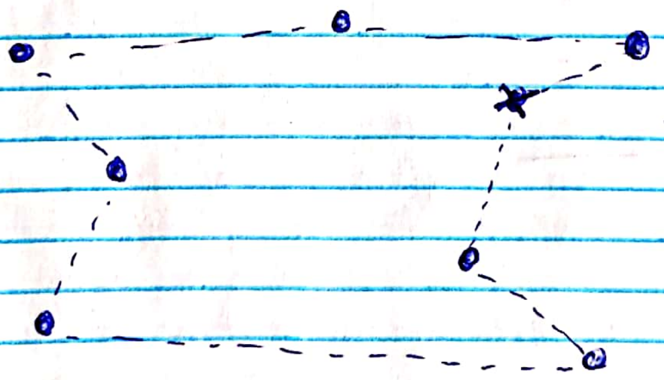
$\Delta_2 = \cancel{2}\, 3$, $I_2 = \{1,2,3\}$, $J_2 = \{2,4\}$

(continuing on $\delta = 2$)
(and, iteration 4 looks like:

$$A_3 = \begin{bmatrix} 0 & \boxed{1} & 0 & 0 \\ 0 & 0 & \boxed{1} & 0 \\ 0 & 0 & 0 & \boxed{1} \end{bmatrix}, \text{etc.})$$

Travelling Salesman Problem:

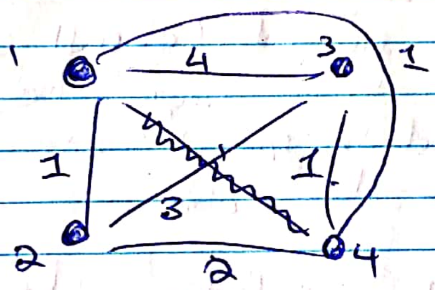
→ We can show that this is equivalent to "jobshop w/ setup times"



→ Want to travel through all nodes and returning to starting node ≈ w/ shortest distance/cost.

Def'n (Travelling Salesman Problem - TSP)
Given complete graph $G=(V,E)$ with edge costs $c_e \geq 0$ for all $e \in E$. We want to find a (simple) cycle of min. cost that visits all the nodes (called tours in this context)
No repeated vertices.

Ex!
$G = (\{1,2,3,4\}, \{12, 13, 14, 23, 24, 34\})$
$c_e = 1 \quad 4 \quad 1 \quad 3 \quad 2 \quad 1$



Q: Is this problem NP-hard?
A: YES.

Q: Can we find a k-approx algorithm?
A: No, not for arbitrary edge weights

(But, we can do this if ~~the edge wei~~ we assume the triangle inequality)

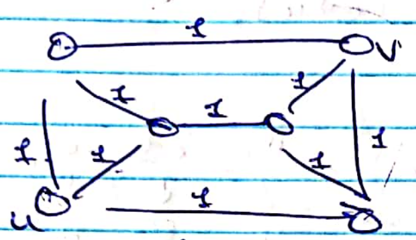
Def'n
Edge costs c_e for all $e \in E$ satisfy the triangle inequality if:
$$c_{uv} \leq c_{uw} + c_{wv} \quad \text{for } u,v \in V.$$

Def'n

We call TSP instance <u>metric</u> if the edge costs satisfy the triangle inequality

Ex: Metric TSP instance

G':



If $e \in G'$, then c_e = length of the shortest uv-path $(e = uv)$

↳ i.e. uv is not in this graph, the cost of uv would be 2. in order to preserve the metric.

Claim: If edge costs satisfy the triangle inequality, then $\forall u, v \in V$, $c_{uv} \le c(P)$ for every u,v-path P.

(If $P = u = w_0 w_1 \dots w_k = v$ then $c(P) = \sum_{i=0}^{k-1} c_{w_i w_{i+1}} = \sum_{e \in P} c_e$).

[Proof]

$$c(P) = \underbrace{\overbrace{c_{w_0 w_1} + c_{w_1 w_2}}} + \dots + c_{w_{k-1} w_k}$$

$$\ge c_{w_0 w_2}$$

$$\ge c_{w_0 w_3}$$

→ Do induction on the length of the path □