PAGE NO.
DATE / /

AOA week 14 Notes.

As we cannot find exact answers for NP-complete problems, we will try to find approximate answers for such problems.

* Load Balancing problems
  # Greedy Approach.

Input: 'm' resources with equal processing power

'n' jobs where job j takes t_j to process

Objective: Assignment of jobs to resources such that the maximum load on any machine is minimized.

let T_i denote load on machine i

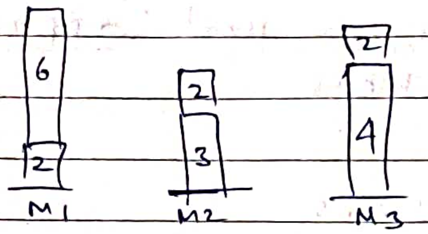
T^* : value of the optimal solution.

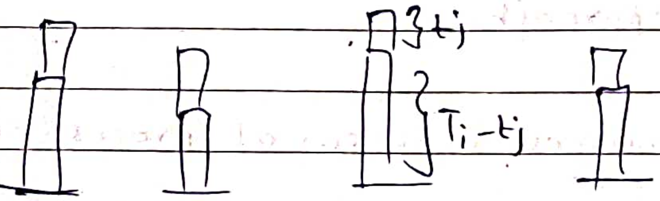
we have to minimize this

Greedy Balancing

Place the next job j with t_j on machine that has least load currently.

eg : Schedule : 2, 3, 4, 6, 2, 2 on 3 machines





we know that

eq1 $T^* \geq t_j$ → load of any task j

T^* → load in optimal answer

Also if number of jobs n > number of machines 'm'

eq2 $T^* \geq T_i - t_j$

Pg 603 explanation : load of machine i before assignment was $T_i - t_j$

And the cost can only increase if we add a new task on same machine i
it can never decrease.

$\therefore$

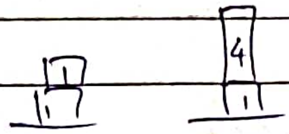
$$T^* \geq t_j$$
$$T^* \geq T_i - t_j$$

$$2T^* \geq T_i$$

$\boxed{T_i \leq 2T^*}$ 2-approximation.

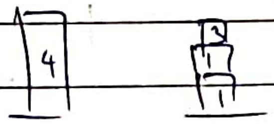
Consider a case

(1, 1, 1, 4) aligned on 2 machines



∴ T = 5

But we can have a better solution like



Here T = 4.

our previous greedy approach balances all resources among machines, and then a large resource comes and spoils the balancing

what can we do?

we can sort the resources in decreasing order and then apply greedy balancing

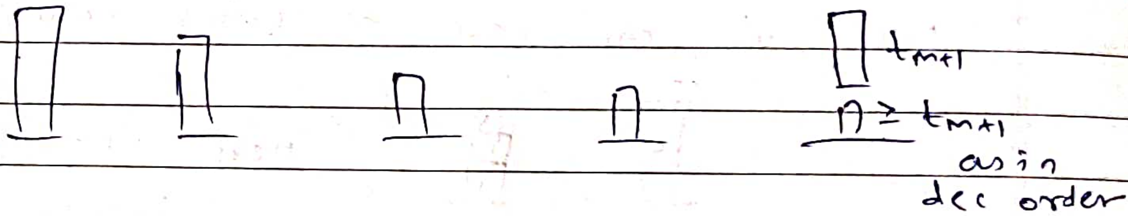
As first if we balance all large jobs, small jobs cannot cause much imbalance.

Improved approximation to greedy Balancing

Initially sort jobs in dec order of length then use the same greedy balancing.

if resources < machines
our solution is optimal.

if resources > machines



and $t_j \leq t_{m+1}$

$T^* \geq 2\, t_{m+1} \implies T^* \geq 2 t_j$ eq 1

$T^* \geq T_i - t_j$ eq 2

$T^* \geq 2\, T_j$

$2 \times \left(T^* \geq T_i - t_j \right)$

$3T^* \geq 2T_i$

$\therefore \boxed{T_i \leq 1.5\, T^*}$

1.5 approximation.

Vertex Cover Problem (Approximate Example)

problem statement : Find smallest vertex cover in G

start with S = Null

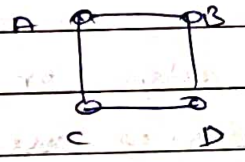
while S is not a vertex cover :

select an edge e not covered by S

Add both ends of e to S.

end while

example



Pick AB

S = {A, B}

covers edges AB, AC, BD

Pick CD

S = {A, B, C, D}

Covers all edges

2 - approximate

As vertex cover has one end of each edge.

Our solution can have atmost two ends of each edge.

Question

Since Indp set $\leq_p$ vertex cover

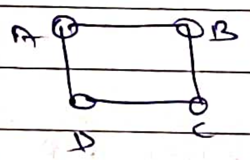
Can we use approximate soln of vertex cover to find a $\Im$-approximate to indp set?

No

example :

we know Indp set $= S$

then vertex cover $= V - S$



running our algo on this gives vertex cover $= \{A, B, C, D\}$

$\therefore$ indp set $= \phi$

Theorem

unless $P = NP$, there is no $1/n^{1-\varepsilon}$ approximation for the maximum independent set problem for any $\varepsilon > 0$ where n is the no. of nodes in the graph.

Question Since vertex cover $\leq_p$ set cover

Can I use 2-approx algo for set cover to find a 2-approx for vertex cover?

Yes

PAGE NO.
DATE / /

MAX - 3SAT

Given a set of clauses of length 3, find a truth assignment that satisfies the largest number of clauses.

A. 5-approximate to MAX-3SAT problem.

- set everything to true
  If less than 50% of clauses evaluate to true, then
- set everything to false

There are soln's that get 8/9 factor of optimal solution.

(*) Linear equations.

$$[A][x] = [B]$$

Linear programming

$$[A][x] \leq [B]$$

objective fnc: $[c^T][x]$

Goal: maximize the objective function subject to the above constraints

If $[A][x] \geq [B]$

The minimize the obj function.

1912 PPT

Linear programming standard Form.

- All constraints are of Form $(\leq)$
- All variables are non negative
- obj function is maximized.

eg: $x_1 - x_2 \geq 0$ $\quad x_1 \geq 0$ $\quad x_2 \geq 0$ $\quad x_1 + x_2 \leq 4$

maximize $x_1 + 2x_2$

(*) weighted vertex cover Problem

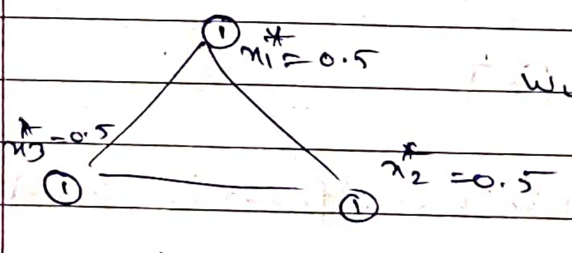
To Find an approximate soln using LP, drop the requirement that $x_i \in \{0, 1\}$ and solve the LP in poly time to find $\{x_i^*\}$ between 0 & 1

$$W_{LP} = \sum_i w_i x_i^*$$

Assume $S' \leftarrow$ optimal vertex cover set

$w(S') \leftarrow$ weight of optimal solution

$$W_{LP} \leq w(S')$$



$W_{LP} = 1.5$

$w(S') = 2$

$x_i^* = 0 \longrightarrow i \notin S$

$x_i^* = 1 \longrightarrow i \in S$

$x_i^* + x_j^* \geq 1$

Say $S_{\geq 1/2} = \{ i \in V ;\ x_i^* \geq 1/2 \}$

$W(S) = 3.$

PAGE NO.
DATE / /

$w(S) \le 2\, w_{LP}$

$w_{LP} \le w(S')$

$w(S) \le 2 \cdot w(S')$.

2 - approx.

* Maxflow problem

variables: Flow over edges

Maximize : $\sum_{e \text{ out of } s} f(e)$

Subject to :

$0 \le f(e) \le c_e$

$$\left(\sum_{e \text{ into } v} f(e) - \sum_{e \text{ out of } v} f(e) = 0 \right)$$

$$\hookrightarrow \sum_{e \text{ into } v} f(e) \ge \sum_{e \text{ out of } v} f(e)$$

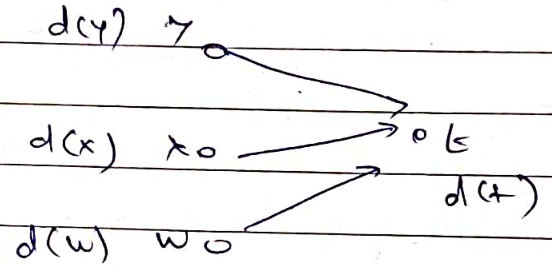
$$\sum_{e \text{ out of } v} f(e) \ge \sum_{e \text{ into } v} f(e)$$

A = B
Then in LP
A ≥ B
B ≥ A

look ppt

(*) shortest path in LP:

Find shortest path from s to t



$$d(t) \le d(y) + c_{yt}$$
$$d(t) \le d(x) + c_{xt}$$
$$d(t) \le d(w) + c_{wt}$$

$$\begin{cases} d(v) \le d(u) + w(u, v) & \text{For each edge } (u,v) \in E \\ d(s) = 0 \end{cases}$$

<u>objective</u>

~~maxi~~ ~~minimize~~ d(t)

Maximize