Lecture Notes: Scheduling to maximize profit

Problem:

1 machine. it can be working on at most one job at a time

n jobs $J_1, \dots, J_j, \dots, J_n$

unit length, non-preemptive

↳ once the job has started, it may not stop until it has completed

release time 0

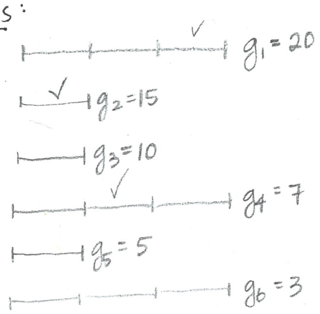
deadline $d_j \in \mathbb{Z}^+$

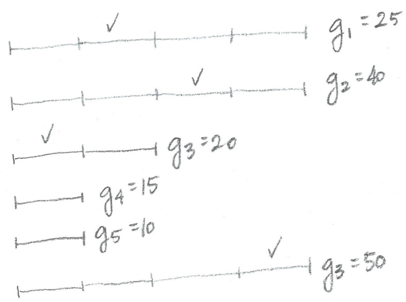
profit $g_j \in \mathbb{Z}^+$

goal: feasibly schedule a subset S of jobs where S has max profit.

↳ $\sum_{j \in S} g_j$

Examples:





The greedy algorithm for this problem bears heavy resemblance to Kruskal's alg. for minimum spanning trees.

High-level of Greedy Scheduler:

- sort jobs s.t. $g_1 \geq g_2 \geq \dots \geq g_n$
- $J \leftarrow \emptyset$
- for each j from 1 to n do
  - if $\boxed{feas(J \cup \{j\})}$ do
    - $J \leftarrow J \cup \{j\}$

→ oracle telling us whether $J \cup \{j\}$ is a feasible set of jobs, i.e. whether there exists a schedule of $J \cup \{j\}$.

(Recall Kruskal's alg:

- sort edges s.t. $w(e_1) \leq w(e_2) \leq \dots \leq w(e_m)$
- $T \leftarrow \emptyset$
- for each edge e from 1 to m do
  - if $\boxed{feas(T \cup \{e\})}$ do
    - $T \leftarrow T \cup \{e\}$

→ can think of this as an oracle telling us whether $T \cup \{e\}$ has cycles

)

Nevermind for now how the oracle determines whether a set of jobs is feasible. We will come back to this.

Note: $\exists$ optimal schedule having no idle time.

(GS)
Claim: Greedy Scheduler maximizes profit.

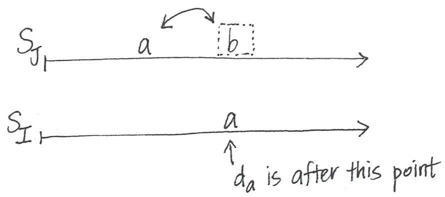
Proof: - let J denote jobs satisfied in GS' solution and S_J be some feasible schedule of jobs J

- let I denote jobs satisfied by the optimal solution (OPT) ; S_I a feasible schedule of I.
- one can rearrange S_J and S_I into S_J' and S_I' s.t. any $j \in I \cap J$ is done in same slot in S_J' and S_I':

  for every job $a \in I \cap J$

  (I) if a in same slot in S_J and S_I, nothing to rearrange.

  (II) if S_J schedules a earlier than S_I schedules a,

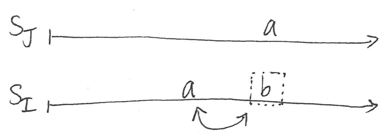


in S_J, swap a with b. (b may be nothing if S_J didn't schedule anything in the slot where S_I scheduled a. can still "swap")

both a and b still done before their deadlines.

  (III) if S_J sch's a later than S_I sch's a,

in S_I, swap a with b. again, still feasible.

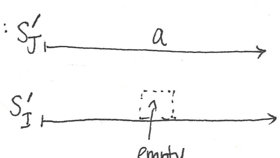


- once job a has been moved into agreement, it never needs to more again. can repeat this argument for all of $I \cap J$, each time decreasing number of common but unsynced jobs.

  so profit(S_J) = profit(S_J') and profit(S_I) = profit(S_I').

- S_I' and S_J' can still look different, but only because $I \neq J$. How can this happen?

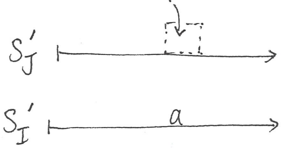
Case 1:



Some job a is sch'ed in S_J' opposite an empty slot in S_I' and $a \notin I$.

※ contradicts optimality of I since $I \cup \{a\}$ is feasible and more profitable.

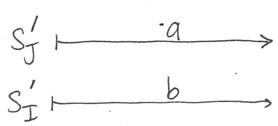
Case 2:



Some job a is sch'ed in S_I' opposite empty slot in S_J' and $a \notin J$.

$J \cup \{a\}$ is feasible and Greedy Scheduler wouldn't have skipped it. ※

Case 3:



$a \notin I$

$b \notin J$

Case 3.1 : $g_a > g_b$. $I \setminus \{b\} \cup \{a\}$ is more profitable than I.
※ optimality of I.

Case 3.2 : $g_a < g_b$. then Greedy Scheduler skipped over b to eventually pick a. ※ greediness

Case 3.3 : $g_a = g_b$. this is the only thing that can happen.

$\therefore \forall$ timeslots t, at t S_J' and S_I' schedule

- no jobs
- same job
- two jobs with same profit.

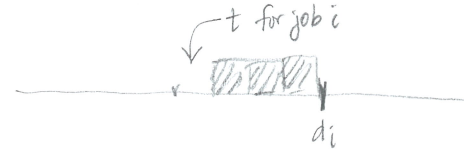
$\Rightarrow$ profit of S_J' = profit of S_I'

$\therefore$ schedule S_J yields optimal (i.e. maximum) profit. ∎

Determining feasibility:

FeasOracle(J): for each $i \in J$, schedule i at t, the latest possible free slot $t \leq \min(\overset{|J|}{n}, d_i)$.

Lemma: J is feasible iff FeasOracle(J) returns a feasible solution.



Proof: $\Leftarrow$: trivial.

$\Rightarrow$: suppose J is feasible.

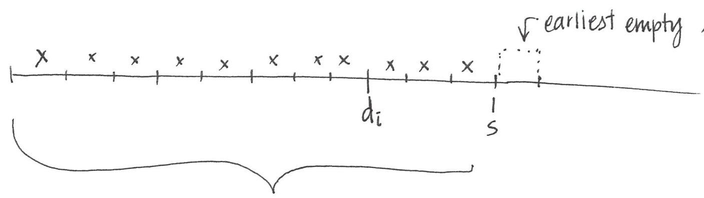
then $\exists$ a feas. sch.

then $\exists$ a feas. sch. scheduling all jobs in first $|J|$ timeslots } "left-shifted sch."

- can always move jobs earlier.

Suppose FeasOracle(J) does not return a feasible sol'n,

i.e. $\exists$ job $i \in J$ s.t. FeasOracle was unable to add it before $\min(|J|, d_i)$



since slot s is empty, all $(s-1)$ jobs scheduled here have deadline $\leq s-1$

$\therefore$ J has at least s jobs, each of whose deadline is $< s$.

By Pigeonhole Prin., J cannot be feasible. ∎

How to implement Greedy Scheduler with FeasOracle: note that FeasOracle doesn't specify order in which jobs of J are added to schedule. We will choose to add them in same order of non-increasing g_j so that we don't have to rebuild the sch. from scratch with each oracle call.

Greedy Scheduler details

Total time:
$O(n \log n) + O(n^2) = \underline{O(n^2)}$.

- Sort jobs s.t. $g_1 \geq g_2 \geq \dots \geq g_n$ (compute d_{max} along the way)
- for each $t \leftarrow 1$ to $\min(n, d_{max})$ do
  - $S[t] \leftarrow$ NIL (schedule)
  - free$[t] \leftarrow t$ (latest free slot earlier than or equal to t)
- for each job $j \leftarrow 1$ to n do
  - $m \leftarrow$ free$[\min(n, d_j)]$ (get latest free slot $\leq \min(n, d_j)$)
  - if $m > 0$ do
    - $S[m] \leftarrow j$ (schedule j there)
    - $m' \leftarrow m$
    - while $S[m'] \neq$ NIL do
      - free$[m'] \leftarrow$ free$[m-1]$
      - $m' \leftarrow m' + 1$

    (update free[·])