CompSci 260P
Spring 2023 Lecture 11:
Greedy II: Permutation Problems

Fractional Knapsack

- Decide x_i : $0 \leq x_i \leq w_i$
- Require $\sum_i x_i \leq W$
- Goal: max $\sum_i b_i(x_i/w_i)$. $= \max \sum x_i \left(\frac{b_i}{w_i}\right)$

Suppose $W = 10$ and

Item :	1	2	3	4	5
Weight:	4	8	2	6	1
Benefit:	12	32	40	30	50

b_i/w_i	3	4	20	5	50
X_i		1	2	6	1

b/c done carrying capacity

Greedy Algorithm for Fractional Knapsack

smaller i : larger $\frac{b_i}{w_i}$

- Sort by b_i/w_i
- For each item in order
  - Take all (if possible) or remaining carrying capacity

- Suppose FSOC solution ~~exists better than ours~~ some alt. is proposed
- What do we know because of that?

$i < k$ $\quad X_i < W_i$ $\quad$ and $\quad X_k > 0$

$\frac{b_i}{w_i} \geq \frac{b_k}{w_k}$

- How do we improve this "better" solution?

decrease ~~take of~~ X_k, increase X_i equally by $\min(W_i - X_i, X_k)$

Review Conversations

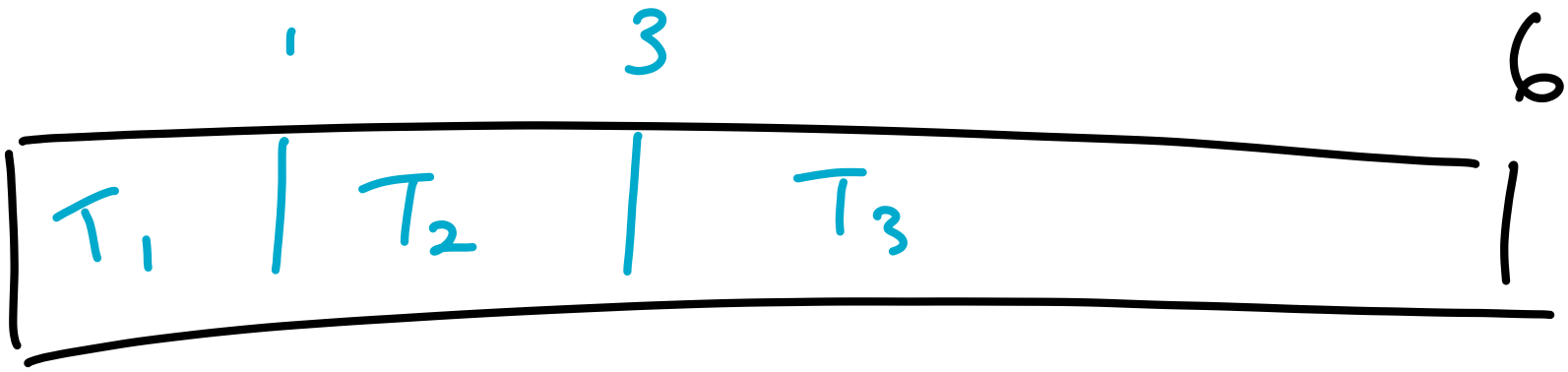
- Why don't these work for fractional knapsack?
  - Sort by weight
  - Sort by benefit
- Why don't these work for 0-1 Knapsack?
  - Sort by weight
  - Sort by benefit
  - Sort by benefit per unit weight

Scheduling with Deadlines

Example 1: What is the optimal schedule for the following input?

Penalty: 0

Time	1	2	3
Deadline	2	4	6



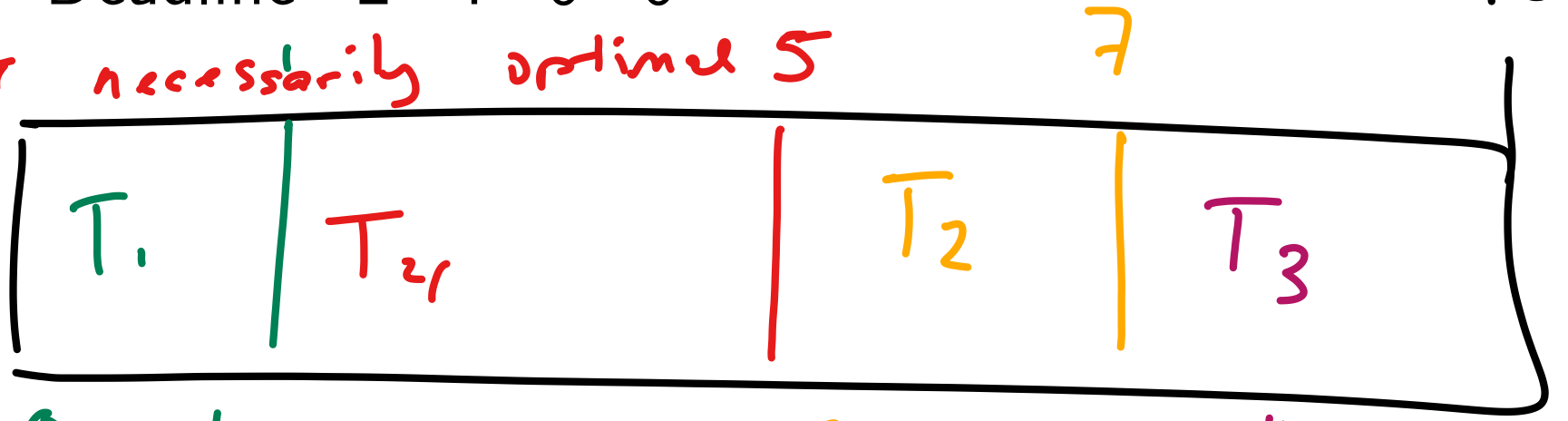
NOT TO SCALE :(

Scheduling with Deadlines

Example 2: What is the optimal schedule for the following input?

Time	1	2	3	4
Deadline	2	4	6	6

NOT necessarily optimal



0 penalty for T_1

0 penalty

3 penalty

4 penalty

Penalty(global) = 4 = max(0, 0, 3, 4)

Possible Scheduling Algorithms

Complete counterexample: include input, alg output, and better output

- Sort the jobs by increasing time t_i; schedule them in that order.

Time:	1	2	3	1040
dl:	1050	1050	1050	1090

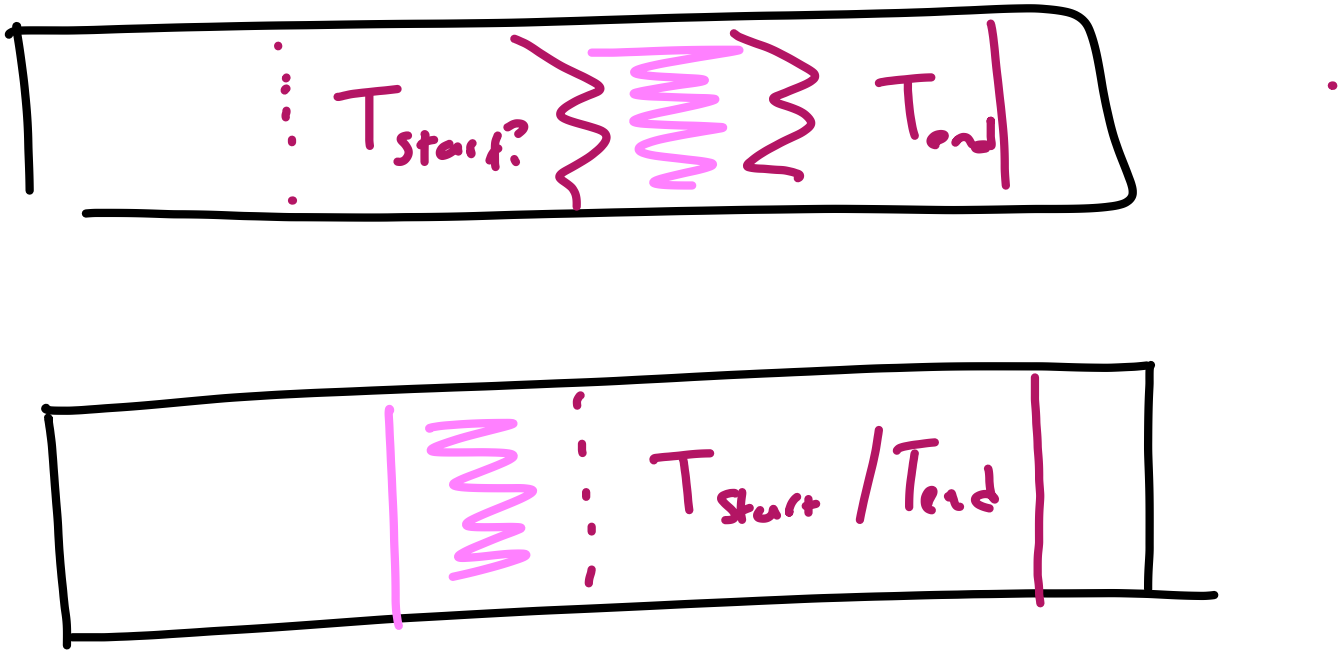
- Sort the jobs by $d_i - t_i$; schedule them in that order.

"slack"

1	2	1	4
1	2	3	4

Can we break up tasks?

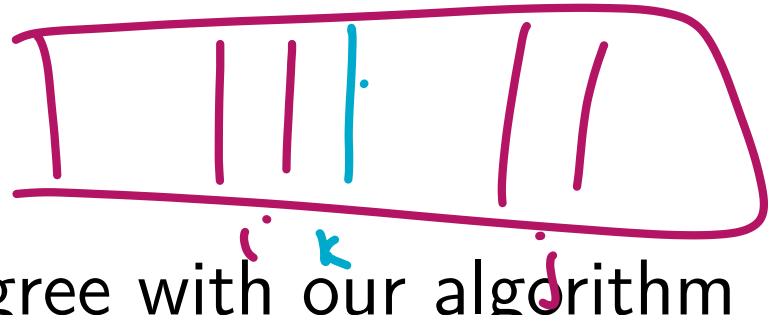
Is it beneficial to break up tasks? Why or why not?



Proof: Lemma 1

When deciding start times, don't leave any gaps; $s_{i+1} = s_i + t_i$.

Proof: Lemma 2



Any schedule that doesn't agree with our algorithm has at least one pair of *consecutive* intervals $i, i+1$ that are *inverted* relative to our order.

Any diff $\longrightarrow \exists i, j \quad i < j$

but $A[i] > A[j]$

if $j = i+1$, done

else $k = i+1$. i, k inverted? idk

if yes, done

else // $A[i] < A[k]$

$A[i] > A[j]$

We can now finish the proof

Our Algorithm: Sort the jobs by earliest deadline first

Claim: Any schedule with an inversion can be modified to be more like our algorithm's output without making it worse.

i, j inverted, $j = i+1$. but $d_i \geq d_j$

Swap i, j in alt. order

before

$f_i = s_i + t_i$

$f_j = s_i + t_i + t_j$

post swap

$f_i' = s_i + t_j + t_i = f_j$

no worse if $d_i \geq d_j$

$f_j' = s_i + t_j < f_j$

Proof of Correctness

- We proved this:
  Claim: Any schedule with an inversion can be modified to be more like our algorithm's output without making it worse.

- What does the full proof look like?