

NOTE BOOK

Rokshana Ahmed

DYNAMIC PROGRAMMING

2 - SUBPROBLEMS

Weighted Interval scheduling

n - SUBPROBLEMS

"Least square fit"
SEGMENTATION

SEGMENTATION

We are given a sequence p_1, \dots, p_n "POINTS" and we aim to segment it (select $I = i_1 < i_2 < \dots < i_k = n+1$, for some $k \geq 1$; cut the sequence $(p_{i_1}, \dots, p_{i_2-1}), (p_{i_2} \dots p_{i_3-1}), \dots, (p_{i_{k-1}}, \dots, p_{i_k})$).

For $i \leq j$, the cost of the segment $(p_i, p_{i+1}, \dots, p_j)$ is c_{ij} . (In least-square-fit, $c_{ij} = a_{ij} + c$)

What is the min-cost segmentation (the cost of a segmentation is the sum of the costs of its segments)?

Let us define $\text{OPT}(j)$ to be the value of the minimum cost segmentation, for the input p_1, \dots, p_j

L: $\text{OPT}(j) = \min_{1 \leq i \leq j} (c_{ij} + \text{OPT}(i-1))$; the segment (p_i, \dots, p_j) is in an optimal solution for the input $p_1 \dots p_j$ IFF $\text{OPT}(j) = c_{ij} + \text{OPT}(i-1)$

P: If the sequence is p_1, \dots, p_j the element p_j should be the right endpoint of a segment, of the last segment.

Where does the last segment begin? It will begin at some point p_i with $1 \leq i \leq j$

The cost of the sequence $p_1 \dots p_j$ if the last segment begins at i is equal to

$$c_{ij} + \text{OPT}(i-1),$$

Since i have to pay for the segment (p_i, \dots, p_j) , and for the best segmentation of the sequence p_1, \dots, p_{i-1} . Thus, the minimum cost for the $= p_1 \dots p_j$ is:

$$OPT(j) = \min_{1 \leq i \leq j} (c_{ij} + OPT(i-1))$$

And, the segment (p_k, \dots, p_j) is part of an opt. solution for the sequence $p_1 \dots p_j$ IFF

$$c_{kj} + OPT(k-1) = \min_{1 \leq i \leq j} (c_{ij} + OPT(i-1)) \blacksquare$$

DYNAMIC PROGRAMMING

2 - SUBPROBLEMS

Weighted Interval scheduling

n - SUBPROBLEMS

"Least square fit"

} look
at
prefixes
SEGMENTATION

"ADDING A VARIABLE"

Consider the following "scheduling" problem:

- there are "n" jobs, the i th of which takes time w_i seconds (of our CPU)

Our CPU is available for W seconds.

If we schedule the generic job we are paid 1€ per second it runs.

Which subset S of jobs should we schedule to maximize our gain?

The set S of jobs is "feasible" (is a valid sol)
IFF $\sum_{i \in S} w_i \leq W$

Thus, we are asking: what is the set of jobs S satisfying $\sum_{i \in S} w_i \leq W$, that maximizes $\sum_{i \in S} w_i$?

$$w_1 = 3 \quad w_2 = 3 \quad w_3 = 5$$

$$W = 5$$



INVALID



VALID (value=3)



OPTIMAL AND VALID (value=5)

HOW TO SOLVE IT?

$$W_1 = 3 \quad W_2 = 3 \quad W_3 = 5$$

$$W = 6$$



In dynamic programming, we want to split a problem into some number of subproblems... which subproblems to use here?

For WIS/segm. we considered prefixes of the instance as subproblems.

Let us try to do the same.

Let $\text{OPT}(i)$ be the optimal gain you can achieve by using only jobs $1, 2, \dots, i$. Let O be an optimal solution.

Then,

L: IF $n \neq 0$, then $\text{OPT}(n) = \text{OPT}(n-1)$

OK but what if $n=0$?

IF $n=0$, then the remaining jobs will only have $W - w_n$ seconds available.



The amount of time available strictly depends on the length of the job we schedule.

In particular, we cannot obtain $\text{OPT}(n)$ in terms of just $\text{OPT}(1), \text{OPT}(2), \dots, \text{OPT}(n-1)$.

We need to solve the following subproblems?

- Given V and i , what is the optimal value if i have time V and the jobs $1, 2, \dots, i$.

("Adding a variable")
↳ we came up with
the idea of using V

Assume that W is an integer and, also, that w_1, \dots, w_n are also integers.

Then,

$$\text{OPT}(i, w) = \max_{\substack{S \subseteq \{1, \dots, i\} \\ (\sum_{j \in S} w_j \leq w)}} \sum_{j \in S} w_j$$

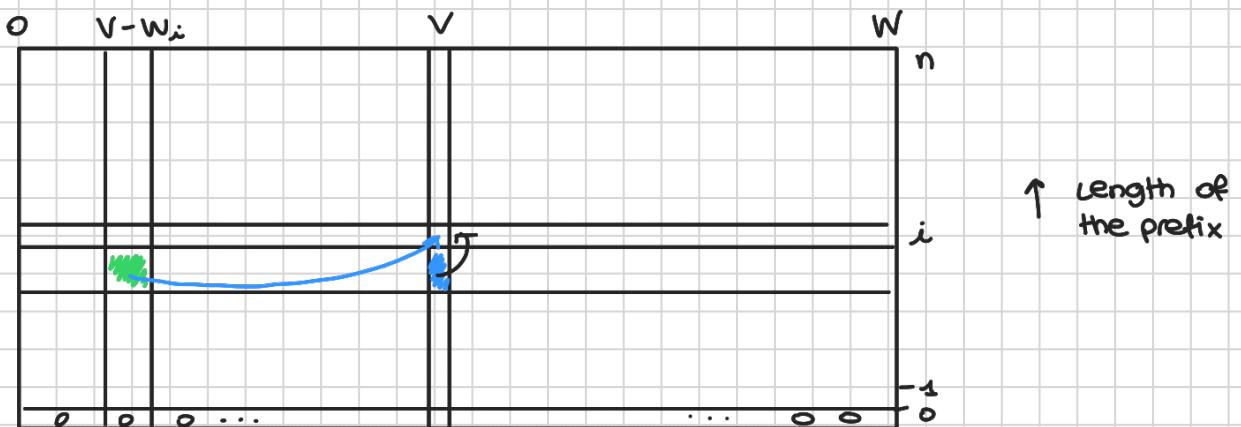
Using these "enlarged" set of subproblems, we can solve the original problem:

- If $n < 0$, then $\text{OPT}(n, w) = \text{OPT}(n-1, w)$
- If $n \geq 0$, then $\text{OPT}(n, w) = w_n + \text{OPT}(n-1, w-w_n)$,
(IF $w_n > w$, then $\text{OPT}(n, w) = \text{OPT}(n-1, w)$)

T1: If $w < w_i$ then $\text{OPT}(i, w) = \text{OPT}(i-1, w)$

Otherwise;

$$\text{OPT}(i, w) = \max_{\substack{\rightarrow \\ \text{the avail}}} (\text{OPT}(i-1, w), w_i + \text{OPT}(i-1, w-w_i)).$$



SUBSET-SUM (n, w):

INITIALIZE THE ARRAY $M[0 \dots n][0 \dots w]$ $O(n \cdot w)$

LET $M[0][v] = 0 \quad \forall v = 0, 1, \dots, w$ $O(w)$

$O(nw)$

{ FOR $i = 1 \dots n$
 FOR $w = 0 \dots w$
 use the recurrence of T1 to set the value
 of $M[i][w]$
 RETURN $M[n, w]$

$$W = \underbrace{100 \dots 0}_{\text{BINARY } n} = 2^{n-1}$$