DYNAMIC PROGRAMMING

WE WILL NOW DISCUSS ANOTHER ALGORITHMIC PARADIGM, DYNAMIC PROGRAMMING.
WE BEGIN WITH AN EXAMPLE.

SCHEDULING JOBS WITH DEADLINES, PROFITS AND DURATIONS

INPUT: A LIST OF JOBS $(d_1\ t_1\ g_1), \ldots, (d_n\ t_n\ g_n)$

deadline, duration, profit

A **schedule** $C = C(1), C(2), \ldots, C(n)$

$C(i)$ is the time at which job i scheduled to begin
$C(i) = -1$ means job i not scheduled

A schedule is **feasible** if

(a) $\forall i\ C(i) \geq 0 \Rightarrow C(i) + t_i \leq d_i$

each job scheduled finishes by its deadline

(b) $\forall i \neq j\ C(i) \geq 0$ and $C(j) \geq 0 \Rightarrow$ either $C(i) + t_i \leq C(j)$ or $C(j) + t_j \leq C(i)$

NO two jobs overlap

The **profit** of a feasible schedule C, $P(C) = \sum_{C(i) \geq 0} g_i$

OUTPUT: A feasible schedule C such that $P(C)$ is the maximum possible.

WE FIRST SORT JOBS SUCH THAT

$$d_1 \leq d_2 \leq \ldots \leq d_n = d$$

LEMMA

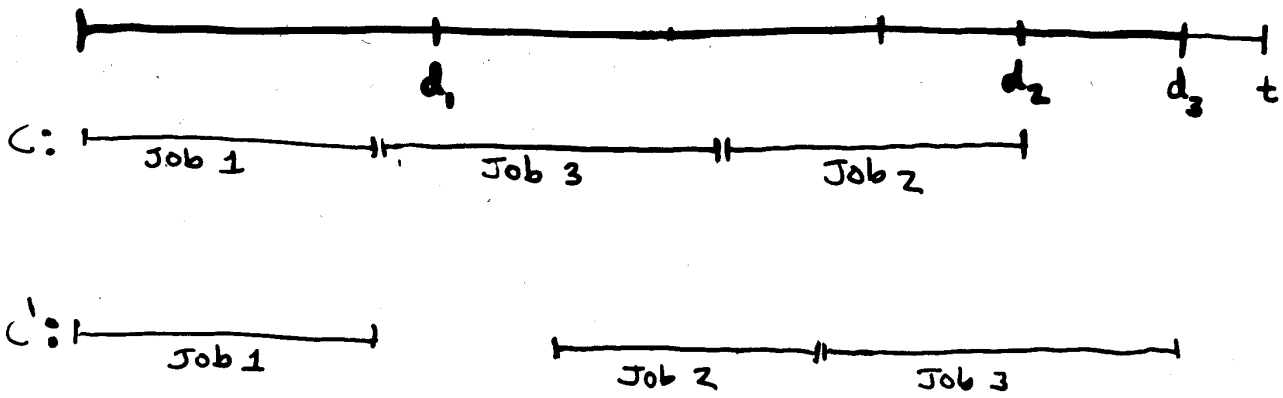
LET C BE A FEASIBLE SCHEDULE SUCH THAT AT LEAST ONE JOB IS SCHEDULED. LET i BE THE LARGEST JOB NUMBER SCHEDULED IN C.
LET EVERY JOB SCHEDULED IN C FINISH BY TIME t.
THEN THERE IS A FEASIBLE SCHEDULE C' THAT SCHEDULES SAME JOBS AS C AND SUCH THAT

$$C'(i) = \min\{t, d_i\} - t_i$$

AND SUCH THAT ALL OTHER JOBS SCHEDULED BY C' END AT OR BEFORE $\min\{t, d_i\} - t_i$.

Proof sketch

Example: Suppose jobs 1, 2, 3 scheduled by C as follows.



Claim Job i can be shifted right to end at time $\min\{t, d_i\}$. Then all jobs to right of job i can be shifted left by same amount.

DYNAMIC PROGRAMMING ALGORITHM

(1) Describe array of values we want to compute.

$\overset{*}{A}(i,t) = \max\{ P(C) \mid C$ is a feasible schedule in which only jobs from $\{1,..,i\}$ are scheduled, and all jobs scheduled finish by time $t\}$

$0 \le i \le n, \quad 0 \le t \le d$

$\overset{*}{A}(n,d)$ = desired output

(2) Give a recurrence for computing values of A one row at a time.

- $A(0,t) = 0 \qquad 0 \le t \le d$
- For $1 \le i \le n$, $0 \le t \le d$ Let $t' = \min\{d_i, t\} - t_i$
  t' is latest possible time that job i can be scheduled to end by its deadline and end by time t.

$$A(i,t) = \begin{cases} A(i-1,t) & \text{if } t' < 0 \\ \max\{ A(i-1,t),\ g_i + A(i-1,t')\} & \text{if } t' \ge 0 \end{cases}$$

Correctness

we want to show that $A(i,t) = \overset{*}{A}(i,t) \quad \forall\ 0 \le i \le n,\ 0 \le t \le d$

Clearly $\overset{*}{A}(0,t) = 0$

Showing $A(i,t) = \overset{*}{A}(i,t) \quad i > 0$

Case 1: $t' < 0$. Then cannot schedule i, so $\overset{*}{A}(i,t) = \overset{*}{A}(i-1,t) = A(i-1,t)$

Case 2 $t' \ge 0$. Best profit without scheduling job i is $\overset{*}{A}(i-1,t) = A(i-1,t)$. Best profit if we schedule i is $g_i + \overset{*}{A}(i-1,t') = g_i + A(i-1,t')$ by Lemma.

Correctness continued

more formally we want to show

$$A^*(i,t) = \max\{A^*(i-1,t),\ g_i + A^*(i-1,t')\},$$

where $t' = \min\{t, d_i\} - t_i \geq 0$

Prove $A^*(i,t) \geq \max\{A^*(i-1,t),\ g_i + A^*(i-1,t')\}$ (a)

$A^*(i,t) \leq \max\{A^*(i-1,t),\ g_i + A^*(i-1,t')\}$ (b)

(a) show $A^*(i,t) \geq A^*(i-1,t)$ ✓

and $A^*(i,t) \geq g_i + A^*(i-1,t')$ ✓

(b) show either $A^*(i,t) \leq A^*(i-1,t)$ or $A^*(i,t) \leq g_i + A^*(i-1,t')$

Let C be feasible schedule in which only jobs from $\{1..i\}$ are scheduled, all jobs finish by time t, and C has maximum profit.

Case 1 C does not schedule job i

Then C schedules only jobs from $\{1,..,i-1\}$ so profit of C equals $A^*(i-1,t)$. Thus $A^*(i,t) \leq A^*(i-1,t)$.

Case 2 C schedules job i.

By Lemma, there exists feasible C' where
C' schedules same jobs as C,
C' schedules i beginning at time t'
and all other jobs scheduled end by t'.

$\therefore P(C') = P(C) = A^*(i,t)$

Let C'' be same as C', but C'' doesn't schedule job i.

Then $P(C'') = A^*(i,t) - g_i$

$\therefore A^*(i,t) - g_i \leq A^*(i-1,t') \Rightarrow A^*(i,t) \leq A^*(i-1,t') + g_i$

③ HIGH LEVEL PROGRAM FOR COMPUTING A

```
FOR  t ∈ {0, .., d}
    B[0, t] ← 0
END FOR

FOR i = 1 .. n
    FOR EACH  t ∈ {0, .., d}
        t' ← min{t, d_i} − t_i
        IF t' < 0 THEN  B[i, t] ← B[i−1, t]
        ELSE  B[i, t] ← max{ B[i−1, t] , g_i + B[i−1, t'] }
        END IF
    END FOR
END FOR
```

④ COMPUTE OPTIMAL SOLUTION

```
PRINTOPT (i, t)
    IF i = 0  RETURN  END IF
    IF B[i, t] = B[i−1, t] THEN
        PRINTOPT (i−1, t)
    ELSE
        t' ← min{t, d_i} − t_i
        PRINTOPT (i−1, t')
        PUT "SCHEDULE JOB" i "AT TIME" t
    END IF
END PRINTOPT

PRINTOPT (n, d)
```

RUNTIME ANALYSIS

1. SORTING $O(n \log n)$
2. COMPUTING B FROM STEP 3 $O(nd)$
3. COMPUTING PRINTOPT $O(n)$

$\therefore \quad O(n \log n + nd)$

NOTE

THIS IS NOT REALLY A POLYTIME ALGORITHM SINCE deadlines given in binary notation.

So input size is $\approx O(n(\log d + \log g))$

(d: max deadline, g: max profit)

$$\approx O(n \log d)$$

DYNAMIC PROGRAMMING IN A NUTSHELL

THERE ARE 4 STEPS TO MOST DP ALGORITHMS.

* **STEP 1** DESCRIBE AN ARRAY OF VALUES THAT YOU WANT TO COMPUTE. (DESCRIBE WHAT IT IS THAT YOU WANT TO COMPUTE)
  SAY HOW CERTAIN ELEMENTS IN ARRAY COMPUTE OPTIMAL VALUE FOR ORIGINAL PROBLEM.

* **STEP 2** GIVE A RECURRENCE TO COMPUTE ARRAY VALUES.
  PROVE CORRECTNESS (VALUE GIVEN BY RECURRENCE EQUALS WHAT WE WANTED TO COMPUTE, AS DESCRIBED IN STEP 1.)

STEP 3 GIVE ALGORITHM FOR COMPUTING ARRAY VALUES, USING RECURRENCE. RUN-TIME ANALYSIS.

STEP 4 SHOW HOW TO COMPUTE OPTIMAL SOLUTION TO ORIGINAL PROBLEM FROM ARRAY VALUES. RUN-TIME ANALYSIS

* USUALLY THE HARD PART

GENERAL KNAPSACK PROBLEM

INPUT Weights (nonnegative) $w_1, w_2, \ldots, w_n$
and a capacity $C \geq 0$
and profits $g_1, g_2, \ldots, g_n$ (non-negative)

For $S \subseteq \{1, 2, \ldots, n\}$ LET $P(S) = \sum_{i \in S} g_i$, $K(S) = \sum_{i \in S} w_i$

S is feasible if $K(S) \leq C$.

OUTPUT Feasible $S \subseteq \{1, \ldots, n\}$ so that $P(S)$ is as large as possible.

Note: Simple Knapsack is special case of general Knapsack where $g_i = w_i$ $\forall i$.

General Knapsack is a special case of Scheduling with deadlines, profits, and durations where all deadlines are same:

$$C \qquad \Longrightarrow \qquad d_1 = d_2 = \ldots = d_n = C$$
$$w_1, \ldots, w_n \qquad\qquad t_1 = w_1,\ t_2 = w_2, \ldots, t_n = w_n$$
$$g_1, \ldots, g_n \qquad\qquad g_1, \ldots, g_n$$

DP algorithm has runtime $O(nC)$