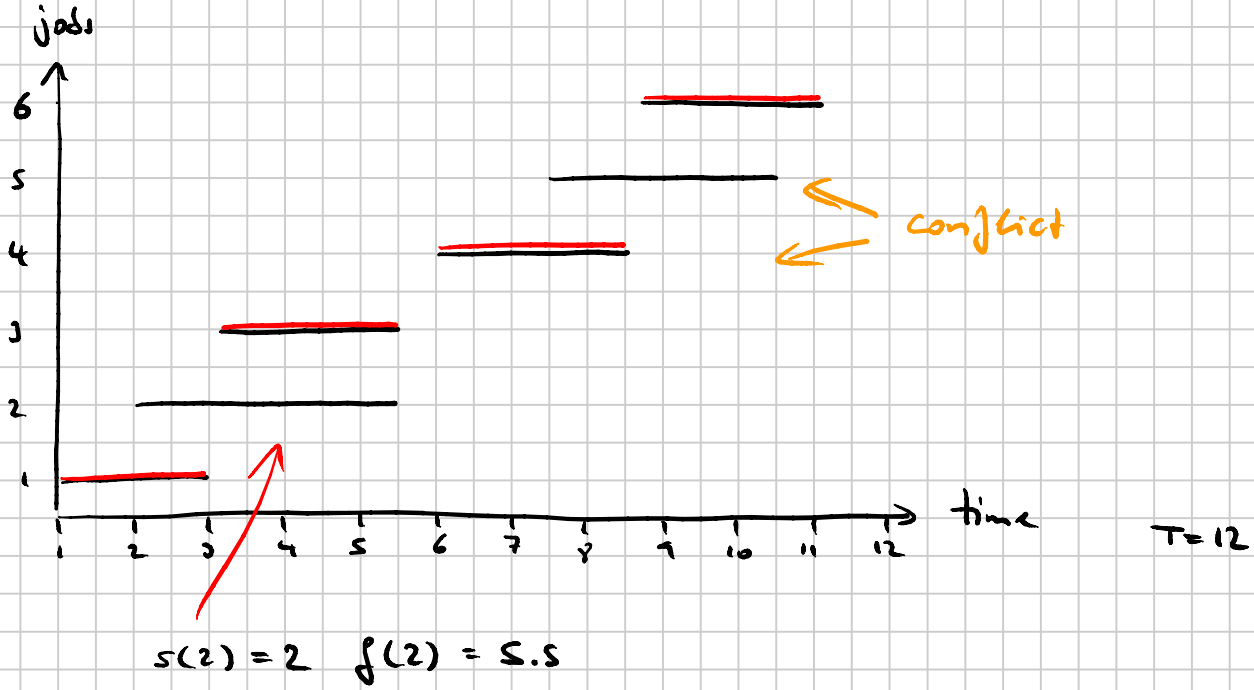
A scheduling problem

Have:
- single machine available at all times $t \in \{1, \dots, T\}$
- n jobs, job i has a
  - start time $s(i)$
  - finish time $f(i)$
- at each time t, can only execute one job!

Goal: Find subset J of jobs of largest size that can be executed on machine in $\{1, \dots, T\}$.

ex:



Can feasibly execute jobs $\{1, 3, 4, 6\}$.

Two jobs i_1 and i_2 conflict if they cannot both be scheduled, i.e., their intervals overlap.

Is there an easy way to find a max-size set of jobs that can be executed feasibly?

Integer programming certainly works but slow!

An easy algorithm:

U = set of all jobs, $J = \emptyset$

<u>while</u> $U \neq \emptyset$ <u>do</u>

- Choose job i in U with smallest finishing time.
- put i in J
- Remove all jobs i' from U that conflict with i

<u>end while</u>

Return J

Thm: This algorithm returns an optimal solution.

Pf:

Let $O \subseteq \{1,\dots,n\}$ be an optimal schedule

Let $J \subseteq \{1,\dots,n\}$ solution computed by algo.

Assume

$$O = \{j_1,\dots,j_m\} \quad J = \{i_1,\dots,i_k\}$$

Want to show: $m = k$.

Let $i_1,\dots,i_k$ be in order of addition by algo

$$\Rightarrow s(i_1) \leq f(i_1) \leq s(i_2) \leq f(i_2) \leq \dots \leq f(i_k).$$

Let's assume also that

Job j_p ends before j_q starts for all $1 \le p < q \le m$.

Claim: rth job in J ends earlier than rth job in O, for $r \le k$.

Pf: induction on r

$\underline{r=1}$: true. Our algo picks job with smallest finish time first.

$\underline{r>1}$: by induction know: $f(i_{r-1}) \le f(j_{r-1})$

Look at job j_r:

does not conflict with j_{r-1}

$\Rightarrow s(j_r) \ge f(j_{r-1})$

$\Rightarrow s(j_r) \ge f(i_{r-1})$

$\ge f(i_{r-2})$

$\ge \dots \ge f(i_1)$

$\Rightarrow$ job j_r does not conflict with any job in $\{i_1, \dots, i_{r-1}\}$

$\Rightarrow$ Our algo could choose j_r in step r

Algo chooses job i_r with smallest finish time among all available jobs.

$\Rightarrow f(i_r) \le f(j_r)$ ☑

Now assume that $|O| > |J|$ for sake of contradiction.

Equiv:

$$m > k$$

Last claim implies

$$f(i_k) \leq f(j_k)$$

Feasibility:

$$s(j_{k+1}) \geq f(j_k)$$

$$\Rightarrow \quad s(j_k) \geq f(i_k)$$

Could add job j_{k+1} to J. Contradiction to algo terminating.

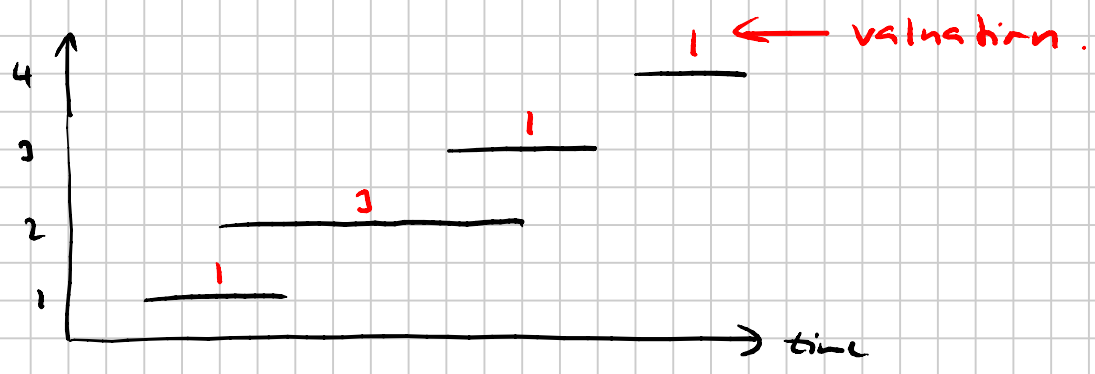
∎

Let's modify scheduling problem slightly.

Each job $i \in \{1,\dots,n\}$ also has value v_i.

Want to find set $J \subseteq \{1,\dots,n\}$ of pairwise non-conflicting jobs of largest value.

Ex:



Opt soln is $J = \{2,4\}$ of value 4.

Does previous algo work?

No. Would produce soln $J = \{1, 3, 4\}$.

What to do? Integer programming still works.

There is a smarter way still.

Let $\{1, \ldots, n\}$ be all jobs. Assume $f(1) \leq \cdots \leq f(n)$

Consider optimal solution O for a given instance.

Trivial observation: O either contains n or it does not.

For job $i \in \{1, \ldots, n\}$, let $p(i) \in \{1 \ldots n\}$ be largest s.t. $f(p(i)) \leq s(i)$.

Case 1: $n \in O$

$\Rightarrow$ none of the jobs in $p(n)+1, \ldots, n-1$ can be in O as they all conflict with n

Let $O' = O \setminus \{n\}$. Then O' is optimal solution for instance with jobs $\{1, \ldots, p(n)\}$.

Case 2: $n \notin O$

Optimal solution for jobs $\{1, \ldots, n-1\}$ is also opt. solution for instance with all jobs.

This suggests an algorithm:

O_j : optimal solution for instance with jobs $\{1 \dots j\}$

$opt_j = \sum_{i \in O_j} v(i)$ value of this solution

Define: $opt_0 = 0$ $O_0 = \emptyset$.

Reasoning above shows:

$$opt_n = \max \{ opt_{n-1}, opt_{p(n)} + v(n) \}$$

In general:

$$opt_j = \max \{ opt_{j-1}, opt_{p(j)} + v(j) \}$$

Notice: opt_j relies on opt_q for $q < j$.

$\Rightarrow$ Algo:

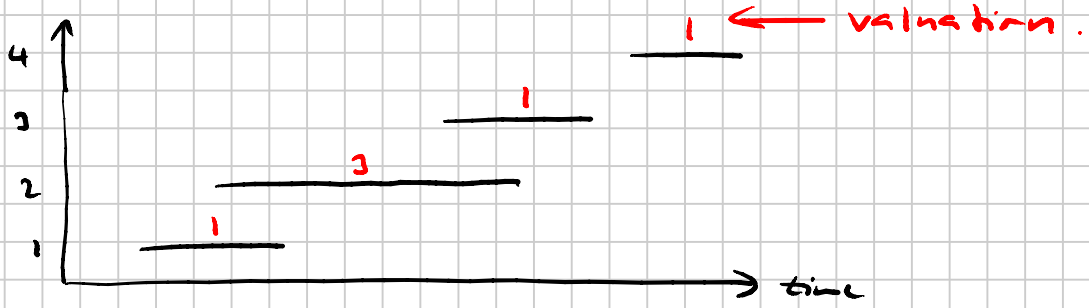
$opt_0 = 0$

for $j = 1 \dots n$ do

$opt_j = \max \{ opt_{j-1}, opt_{p(j)} + v(j) \}$

end for

Ex:



Opt soln is $J = \{2,4\}$ of value 4.

$opt_1 = 1$

$opt_2 = \max\{opt_1, opt_0 + 3\} = 3$

$opt_3 = \max\{opt_2, opt_1 + 1\} = 3$

$opt_4 = \max\{opt_3, opt_3 + 1\} = 4$

This algorithm is really fast. Basically one max-evaluation for each job.

Q: The algo gives value of opt soln. How do we find associated solution?

→ Item $n \in O$ if $opt_{p(n)} + v(n) \geq opt_{n-1}$

in general:

$j \in O_j$ if $opt_{p(j)} + v(j) \geq opt_{j-1}$.

In ex.: $O = \{4\} \cup O_3$

$= \{4\} \cup O_2$

$= \{4\} \cup \{2\}$

What features of weighted interval scheduling should we use in designing an algorithm?

Stages The process of finding a solution can be thought of in a sequential way.
Do we include job n?
Do we include job n-1?
⋮

Each job defines a stage in decision making process.

Can decompose the problem of finding a solution to problem into subproblems.

Another ex: Matchsticks

- 18 matchsticks on table
- 2 players
- Players take turns. Active player takes $i = \{1,2,3\}$ sticks from table
- Player who removes last stick looses

Q: If pl. 1 starts, does she have a winning strategy?

< play the game >

Analysis • Can think of this game in a sequential way:
Each stage corresponds to a player's turn.

Odd stages: pl. 1 moves Even stages: pl. 2 moves

- Consider stage i (i.e., ith round of game) s.t. > 1 matchsticks remain.

Q: Does the existence of a winning strategy for moving player depend on previous rounds?

→ No. Depends only on remaining matchsticks.

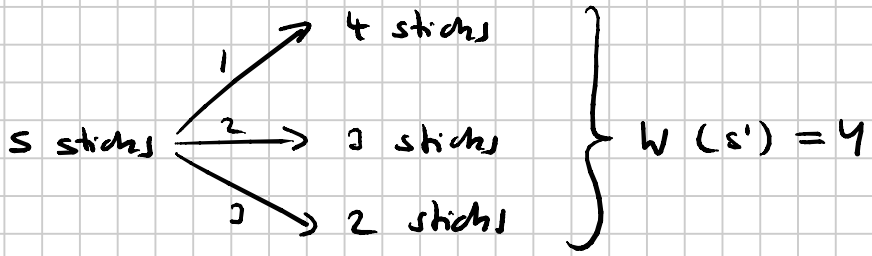
• # remaining matchsticks is often called state information.

• Define $U(s) = Y$ if moving player in stage i and state s has a winning strategy, $U(s) = N$ other.

Clearly: • $W(1) = N$ mov. pl. looses here

• $W(s) = G$ $s \in \{2,3,4\} \Rightarrow$ mov. pl. can take $(s-1)$ sticks and leave other player with loosing situation.

• Now assume $s > 4$. $W(s)$?



No matter what moving player does $\Rightarrow$ other player has winning strategy $\Rightarrow$ $W(s) = N$

In general, player in round i has winning strategy starting from s matches if one of

$$W(s-1),\ W(s-2),\ W(s-3)$$

$$\forall\ s>4: \quad W(s) = \begin{cases} G & : N \in \{W(s-1), W(s-2), W(s-3)\} \\ N & : \text{otherw.} \end{cases}$$

s	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
W(s)	N	G	G	G	N	G	G	G	N	G	G	G	N	G	G	G	N	G

Health Care Teams

- WHO wants to improve health-care in underdeveloped countries.
- 5 med. teams are avail for distribution to 3 different countries
- Improvement in health situation due to sending j med teams to country i measured in 1000's of addtl. person years

# teams \ Country	1	2	3
1	45	20	50
2	70	45	70
3	90	75	80
4	105	110	100
5	120	150	130

Goal: Find assignment of teams to countries that maximizes total # additional pers years.

① How can we look at this problem sequentially?

→ consider countries sequentially

in stage i: decide how many teams to send to country i

② Given that we have decided on how many teams to send to countries $1 \dots i-1$, what do we need to know in order to decide on how many teams to send to $i, i+1, \dots, 3$?

→ the # of avail. health care teams

In stage i, the current state is # of still unassigned health care teams.

- Let $v_i(N)$ be the max # of additional person years obtainable by distributing N teams over countries $i, i+1, \dots, 3$. $v_i(N)$ is independent of action in stages $j < i$!
- What are the allowable decisions in stage i and state N?
  → send $0 \le j \le N$ teams to country i
- What is the new state in stage $i+1$ after sending j teams?
  → $N - j$

Can express the optimal solution to stage i subproblem recursively:

$$v_i(N) = \max_{0 \le j \le N} \left[f_i(j) + v_{i+1}(N-j) \right]$$

This is called a Bellman equation as Bellman invented it in 1957.

Let's solve the problem:

- Keep $v_i(N)$ in a table.
- Start with $i = 3$:

$$v_3(N) = f_3(N)$$

N \ i	1	2	3
0			0
1			50
2			70
3			80
4			100
5			130

$v_2(2)$: 2 → 0, 1 → 50, 0 → 70

Now consider $i=2$. $V_i(N)$ depends on $V_{i+1}(0), \ldots, V_{i+1}(N)$.

Ex.:

$$V_2(2) = \max\{ f_2(2) + V_3(0),\ f_2(1) + V_3(1),\ f_2(0) + V_3(2) \}$$
$$= \max\{ 45 + 0,\ 20 + 50,\ 0 + 70 \}$$
$$= 70$$

N \ i	1	2	3
0		0	0
1		50	50
2		70	70
3		95	80
4		125	100
5		160	130

Now compute $V_3(N)$ $\forall N \in \{0, \ldots, 5\}$.

ex.: $V_3(4) = \max\{ 125,\ 45 + 95,\ 70 + 70,\ 90 + 50,\ 105 + 0 \}$

$= 140$

N \ i	1	2	3
0	0	0	0
1	50	50	50
2	95	70	70
3	120	95	80
4	140	125	100
5	170	160	130

Q: What is the max number of additional person years achievable with 5 teams to distribute over 3 countries?

→ $V_3(5) = 170 \cdot 1000$.

- How many teams are sent to country 1?

$$V_3(5) = 170 = \max\{0+160,\ \underline{45+125},\ 70+95,\ 90+70,\ 105+50,\ 120+0\}$$

$\Rightarrow$ one team

- How many teams are sent to country 2?

$$V_2(4) = 125 = \max\{0+100,\ 20+80,\ 45+70,\ \underline{75+50},\ 110+0\}$$

$\rightarrow$ 3 teams

- How many teams do we send to country 1?

$\Rightarrow$ remaining 1 team

Integer-nonlinear Programs

Want to solve non-linear optimization problems of the form

$$\begin{aligned} \max \quad & x_1 x_2^2 x_3^3 \\ \text{s.t.} \quad & x_1 + x_2 + x_3 \leq 4 \qquad (*) \\ & x_1, x_2, x_3 \geq 0 \text{ and integer.} \end{aligned}$$

We can use dynamic programming!

Can think of process of finding a solution sequentially:

First decide on value of x_1, then x_2 and finally x_3.

Stage i: decide on value of x_i

Suppose we have chosen x_1, what do we have to know in order to decide on values for x_2 and x_3?

→ Only the slack in constraint $(*)$!

Initial slack is 4.

State: slack after choices so far

Ex: suppose we pick $x_1 = 2$

⇒ left hand side of $(*)$ is 2

⇒ slack is 2

What are the possible choices for x_2?

Need

$$x_2 + x_3 \leq 2 \quad \Rightarrow \quad x_2 \leq 2$$

$\Rightarrow$ can choose $x_2 \in \{0,1,2\}$

$$V_i(s) = \max \prod_{j=i}^{3} x_j^j$$

$$\text{s.t.} \quad \sum_{j=1}^{3} x_j \leq s$$

$$x_j \geq 0, \text{ integer}$$

$$\Rightarrow V_i(s) = \max_{0 \leq j \leq s} j^j \cdot V_{i+1}(s-j)$$

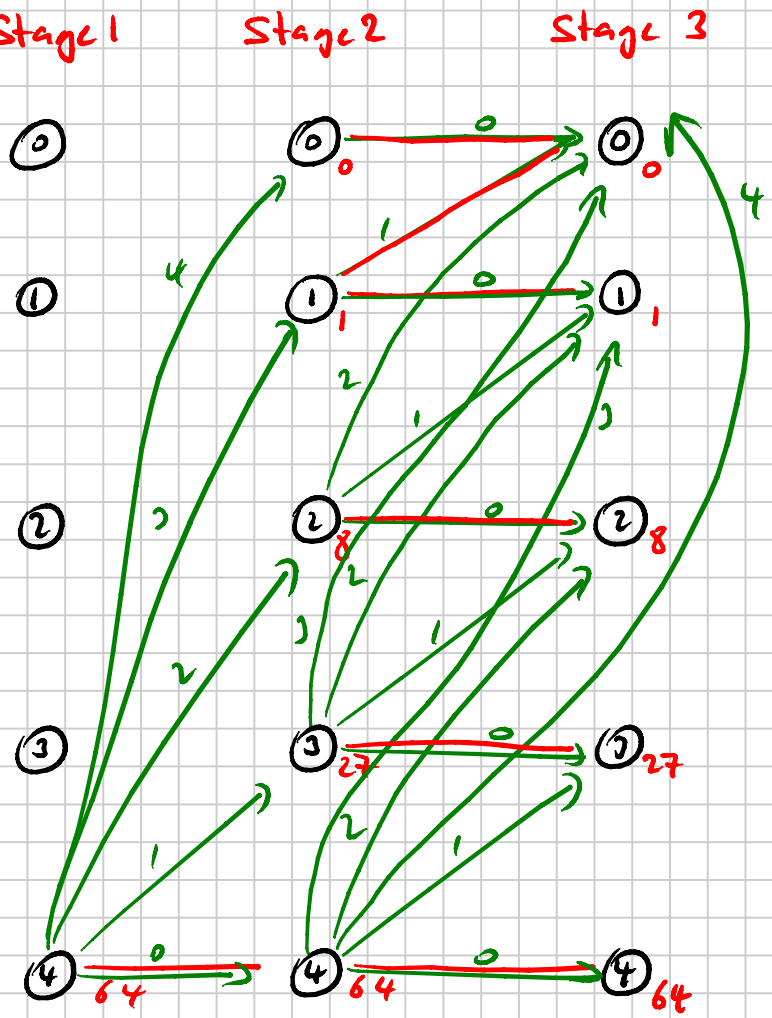
Solving the DP: go backwards!

$$V_3(s) = s^3 \qquad \forall\, 0 \leq s \leq 4$$

Then compute $V_2(s)$ $\forall\, 0 \leq s \leq 4$ – it depends only on $V_3(s)$!

Finally compute $V_1(s)$.

Convenient way of depicting solution: State transition diagram.


Stage 1
Stage 2
Stage 3
0
1
2
3
4
0
1
2
3
4
0
1
2
3
4
0
0
0
1
1
2
8
3
27
4
64
0
1
2
8
3
27
4
64
0
1
2
8
3
27
4
64
4
1
2
3
2
1
2
3
1
2
1