

Boxes Problem

Six items to ship, item 1 > item 2 > ... > item 6.

A box ~~built~~ ^{built} to hold item 1 could instead hold item 2, etc.

Can build boxes in up to 3 sizes — we decide which sizes, and how many.

Costs of different sizes are: 25, 20, 18, 15, 11, 8. $\equiv c_n$

$x_n = \#$ boxes built of size n .

$s = \#$ sizes remaining.

$f_n(s, x_n) =$ cost of best packing policy for items $n, n+1, \dots, 6$ given s sizes left that can be built, and given that x_n boxes of size n are built.

$$f_n(s) = \underline{\hspace{15em}}$$

$$= \min_{x_n} \{ f_n(s, x_n) \}.$$

Want $f_1(3)$.

$$f_n(s, x_n) = c_n x_n + f_{n+x_n}(s-1).$$

25, 18, 13, 9, 7, 5.

$f_6(s)$: Only need to consider $s=1$.

s	$f_6(s)$	x_6^*
1	8	1

$f_5(s)$:

s	$f_5(s)$	x_5^*
1	$2 \times 11 = 22$	2
2	$11 + 8 = 19$	1

$f_4(s)$:

s	$f_4(s, x_n) =$			$f_4(s)$	x_4^*
	$x_n=1$	$x_n=2$	$x_n=3$		
1	—	—	45	45	3
2	$15 + 22 = 37$	$30 + 8 = 38$	—	37	1

$f_3(s)$:

s	$f_3(s, x_n)$				$f_3(s)$	x_3^*
	$x_n=1$	$x_n=2$	$x_n=3$	$x_n=4$		
1	—	—	—	—		
2	—	—	—	—		

$f_2(s)$:

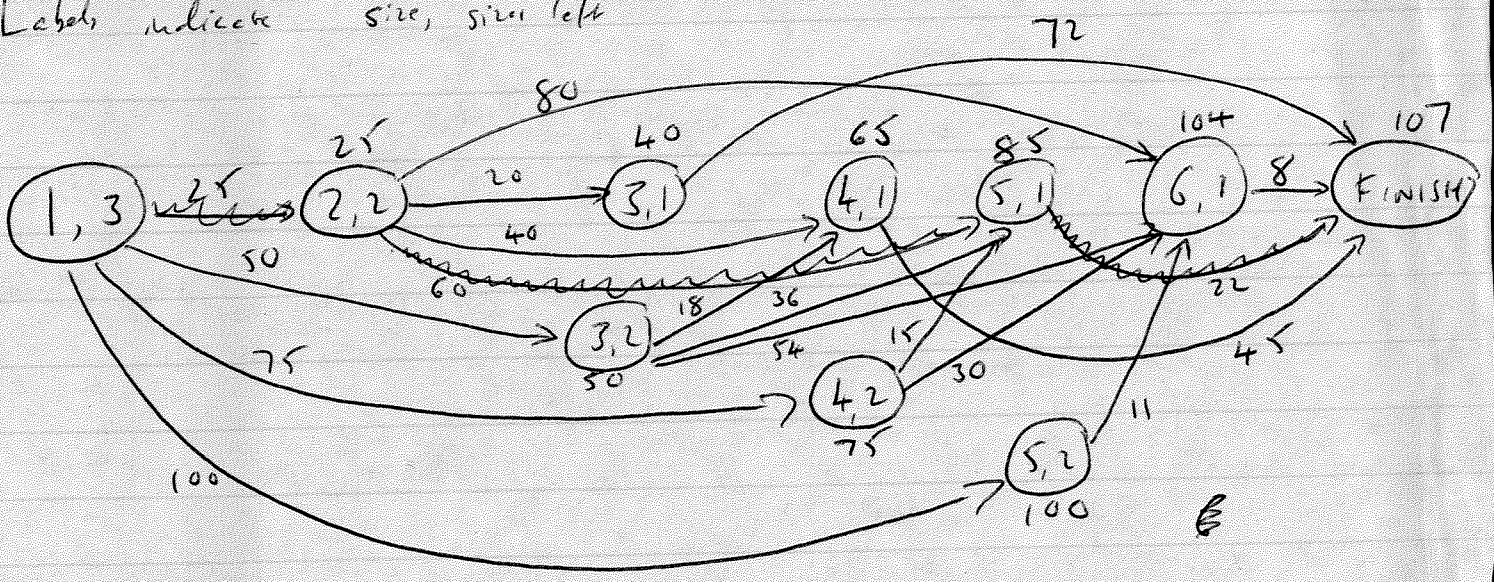
s	$f_2(s, x_n)$					$f_2(s)$	x_2^*
	$x_n=1$	$x_n=2$	$x_n=3$	$x_n=4$	$x_n=5$		
1	—						
2	—						

$f_1(s)$:

s	$x_n=1$	$x_n=2$	$x_n=3$	$x_n=4$	$x_n=5$	$f_1(s)$	x_1^*
2							

Graph for boxes problem.

Labels indicate size, size left



Want shortest path

Optimal value: 107.

Optimal soln:

1 box of size 1	:	25	
3 boxes of size 2	:	+ 60	
2 boxes of size 5	:	+ 22	
		107.	

Forward recursive relations

Let $g_n(s)$ = cost of best route for stages $1, \dots, n$ given that we end up in state s (after the n th stage)

Then $g_1(\text{Calais}) =$
 $g_2(\text{Brussels}) =$

Want to find $g_5(\text{Athens})$

So let $g_n(s, x_n)$ = cost of best route for stages $1, \dots, n$ given that we end up in state s , and that we came from x_n on the n th stage.

Then $g_n(s, x_n) = c(x_n, s) + g_{n-1}(x_n)$

so $g_n(s) = \text{minimum } \{ g_n(s, x_n) \}$ over all possible x_n 's.

Note

Solving a problem by dynamic programming provides much more information than just the optimal solution to the original problem.

Ex: In shortest path problem:

If instead we wanted shortest path from Frankfurt to Berlin, could read off the value of f_3 (Frankfurt), and then trace the optimal route.

In ~~optimal~~^{budget} allocation problem:

If decided not to build any schools, and if had \$80m instead of \$100m, could read off $f_2(8)$ to find best policy.

Continuous - State dynamic programming problems

Consider

$$\max x_1^2 + x_2$$

$$x_1 + x_2 \leq 2.$$

$$x_1 \geq 0, x_2 \geq 0.$$

At stage n decide on amount x_n to be allocated to n^{th} variable

The state s is amount not yet allocated.

Then $f_n(s)$ = best value for variables $n, \dots, 2$ given that s is the amount that remains to be allocated.

$$f_n(s, x_n) = \frac{\quad}{\quad} \text{ given that the } n^{\text{th}} \text{ variable is allocated } x_n.$$

Note that $0 \leq s \leq 2$. (infinite & possible values.)

$$\text{Then } f_2(s, x_2) = x_2$$

$$\text{So } f_2(s) = s$$

$$\text{Also, } f_1(s, x_1) = x_1^2 + f_2(s - x_1) = x_1^2 + s - x_1$$

Now, $s = 2$.

$$\therefore f_1(2, x_1) = x_1^2 + 2 - x_1 = (x_1 - \frac{1}{2})^2 + \frac{7}{4}.$$

Maximized at one of endpoints, i.e., either $x_1^* = 0$ or $x_1^* = 2$.

Find values:

$$f_1(2, 0) = 2, \quad f_1(2, 2) = 4.$$

$$\therefore x_1^* = 2. \quad \Rightarrow \quad x_2^* = 0$$

Problems with several state variables

Eg. Have S crates of apples
 R crates of oranges.

Have chain of supermarkets $1, \dots, N$.

~~Let~~ Each store can realize a certain profit $P_n(x_n, y_n)$
if it has x_n crates of apples and y_n crates of oranges.
(~~Too~~ Too many crates and can't sell them all;
too few and customers don't notice them)

Then problem is to

$$\text{maximize } \sum P_n(x_n, y_n)$$

$$\text{s.t. } \sum x_n = S$$

$$\sum y_n = R$$

$$x_n, y_n \geq 0 \text{ integer } n=1, \dots, N$$

Formulate as a DP:

At stage n , decide how many crates of oranges and how many crates of apples to give to store n .

State is (s, r)

crates apples remaining

crates oranges remaining.

Note: State is given by 2 variables

Go to get recursive relations:

$f_n(s, r, x_n, y_n)$ = cost of best allocation policy for stores n, \dots, N , given s crates of apples and r crates of oranges left to be allocated, and given that store n gets x_n crates of apples and y_n crates of oranges.

$$f_n(s, r) = \underline{\hspace{10cm}}$$

Then $f_n(s, r, x_n, y_n) = P(x_n, y_n) + f_{n+1}(s - x_n, r - y_n)$

and $f_n(s, r) = \max \{ f_n(s, r, x_n, y_n) \}$ over all possible x_n, y_n .

Ex: 3 stores, 5 crates of apples, 4 crates of oranges

Store 1:

		r oranges		
		0	1	
s apples	0	0	3	⚡
	1	2	4	
	2			

Store 3:

		r oranges		
		0	1	
s apples	0	0	2	⚡
	1	2	5	
	2			

Store 1:

		r	
		0	1
s	0	0	2
	1	3	4

Optimal allocation to store 3 is to use all remaining crates.

$$f_2(s, r) = P_3(s, r)$$

For store 2: $f_2(s, r, x_n, y_n) = P_2(x_n, y_n) + P_3(s - x_n, r - y_n) \mid f_1(s, r) \mid x_1^*, y_1^*$

s, r	$x_n=0, y_n=0$	$x_n=0, y_n=1$	$x_n=1, y_n=0$	$x_n=1, y_n=1$	---	x_1^*, y_1^*
0,0 0,0	0 + 0	—	—	—	0	0,0
0,1 0,1	0 + $f_3(0,1)$	3 + $f_3(0,0)$	—	—	3	0,1
1,0 1,0	0 + $f_3(1,0)$	—	2 + $f_3(0,0)$	—	2	0,0 or 1,0
1,1 1,1	0 + $f_3(1,1)$	3 + $f_3(1,0)$	2 + $f_3(0,1)$	4 + $f_3(0,0)$	5	0,0 or 0,1.

For store 1: $f_1(s, r, x_n, y_n) = P_1(x_n, y_n) + P_2(s - x_n, r - y_n) \mid f_1(1,1) \mid x_1^*, y_1^*$

s, r	$x_n=0, y_n=0$	$x_n=0, y_n=1$	$x_n=1, y_n=0$	$x_n=1, y_n=1$	$f_1(1,1)$	x_1^*, y_1^*
1,1	0 + 5	2 + 2	3 + 3	4 + 0	6	1,0.

So best policy is to give

1 crate of apples to store 1

1 crate of oranges to store 2

0 to store 3.

Curse of dimensionality.

Conceptually, little difference between one-state^{variable} problems and two-state^{variable} problems. Computationally, major difference, because number of ~~at~~ states can grow exponentially, so large number of decisions need to be considered.

Eg, if need to distribute 9 cases of apples (and no oranges), could only be in one of 10 states.

If distribute 9 cases of apples and 9 cases of oranges, then number of possible states is $10 \times 10 = 100$. For each of these states, would need to consider all decisions where $x_n \leq 5$ and $y_n \leq 5$. (i.e., approx 5000 decisions $= 5 \times 10^3$).

If distribute in addition 9 cases of grapefruit, would have $10 \times 10 \times 10 = 1000$ possible states, and ~~the~~ loads of possible decisions ($\sim 5 \times 10^6$).

Referred to as curse of dimensionality.

An example of the forward recursive equations:

Consider knapsack problem:

$$\max 2x_1 + 5x_2 + 3x_3 + 2x_4$$

$$\text{s.t. } x_1 + 2x_2 + 4x_3 + 5x_4 \leq 8$$

$$x_i \geq 0, \text{ integer.}$$

$g_n(s)$ = ~~value~~ ^{profit} of best way to allocate s units among projects $1, \dots, n$

$g_n(s, x_n) =$ ~~value~~ ^{profit} of best way to allocate s units, given that we ~~set~~ allocate x_n to project n .

Want $g_4(15)$.

$$g_n(s) = \max \{g_n(s, x_n)\}$$

$$g_n(s, x_n) = c_n x_n + g_{n-1}(s - a_n x_n)$$

$g_1(s) = 2s$ for any integer s , $0 \leq s \leq 15$ (take $x_1 = s$)

$g_2(s)$:

s	$g_2(s, x_2) = 3x_2 + g_1(s - 2x_2)$					$g_2(s)$	x_2^*
	$x_2=0$	$x_2=1$	$x_2=2$	$x_2=3$	$x_2=4$		
0	0	—	—	—	—	0	0
1	2	—	—	—	—	2	0
2	4	5	—	—	—	5	1
3	6	7	—	—	—	7	1
4	8	9	10	—	—	10	2
5	10	11	12	—	—	12	2
6	12	13	14	15	—	15	3
7	14	15	16	17	—	17	3
8	16	17	18	19	20	20	4

$g_3(s)$:

s	$g_3(s, x_3) = 3x_3 + g_2(s - 4x_3)$			$g_3(s)$	x_3^*
	$x_3 = 0$	$x_3 = 1$	$x_3 = 2$		
0	0	—	—	0	0
1	2	—	—	2	0
2	5	—	—	5	0
3	7	—	—	7	0
4	10	3	—	10	0
5	12	5	—	12	0
6	15	7	—	15	0
7	17	10	—	17	0
8	20	12	6	20	0

$g_4(s)$: It may be best to take $s < 8$.
 So calculate all $g_4(s)$, and take max.

s	$g_4(s, x_4) = 2x_4 + g_3(s - 5x_4)$		$g_4(s)$	x_4^*
	$x_4 = 0$	$x_4 = 1$		
0	0	—	0	0
1	2	—	2	0
2	5	—	5	0
3	7	—	7	0
4	10	—	10	0
5	12	2	12	0
6	15	4	15	0
7	17	7	17	0
8	20	9	20	0

So best value is 20.

Given by $x_4 = 0, x_3 = 0, x_2 = 4, x_1 = 0$.