

# DYNAMIC PROGRAMMING

**Solutions to sub-problems lead to solution of larger problems via Recurrence Relations.**

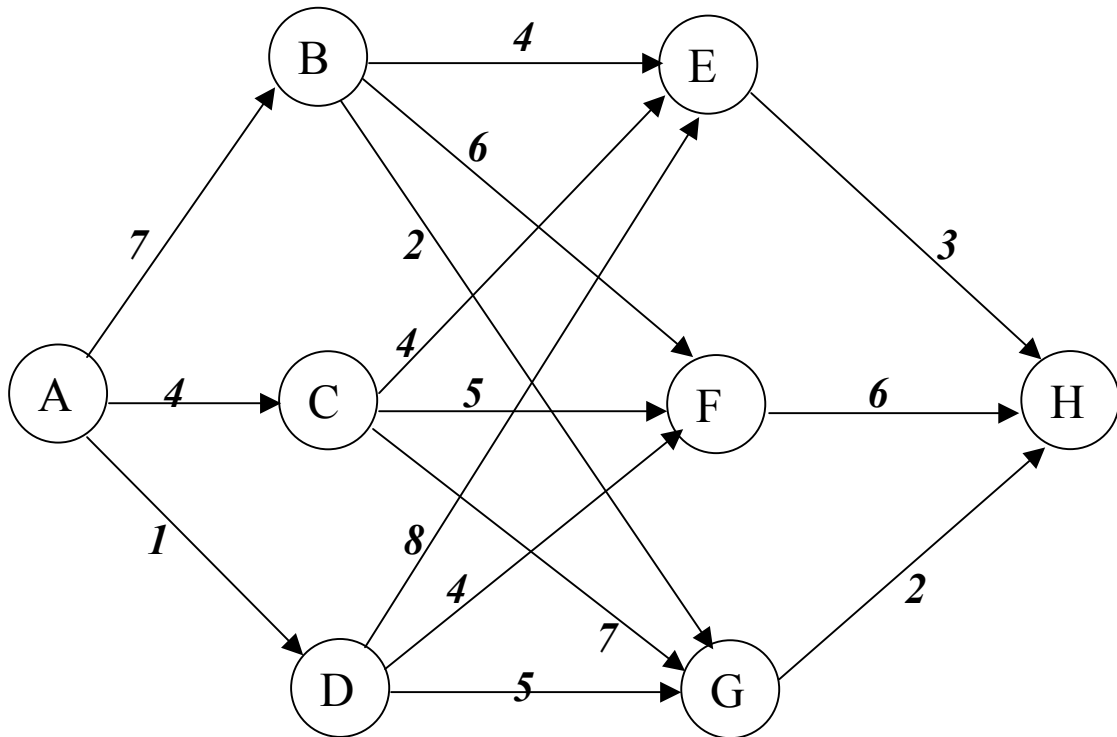
## **Definitions**

**State - Configuration of system identified by label.**

**Stage - A 'step' in a process corresponding to a transition from one state to an adjacent state.**

**Action - There is a set of actions available at each state. Dynamic programming finds the 'best' sequence of actions.**

**Example      The Stagecoach Problem**



***Stage:***

***1                      2                      3                      4***

***Stages remaining:***

***3                      2                      1                      0***

**Serial network – nodes grouped into disjoint sets.**

**e.g.       $\{(3, 1)\}$   
 $\{(2, 1), (2, 2), (2, 3)\}$       etc.**

**Actions at  $(n, i) = K_{ni} = \{1, 2, \dots, k_{ni}\}$**

**e.g. at  $(2, 1)$  actions 1, 2, 3  $\rightarrow$   
 $(2, 1) \rightarrow (1, 1), (1, 2), (1, 3)$  respectively.**

**Action has associated return –  $r(n, i, k)$   
(e.g. distance)**

**Optimal value of state –  $f(n, i)$   
(e.g. optimal distance to terminal node)**

**Recurrence relation:-**

$$f(n, i) = \underset{k \in K}{\text{Min}} \{ r(n, i, k) + f(n-1, k) \}$$

**Backward recurrence.**

## Value Iteration Algorithm

**Stage 0**     $f(0, 1) = 0$

**Stage 1**     $f(1, 1) = 3$                     } only one  
                  $f(1, 2) = 6$                     } action for  
                  $f(1, 3) = 2$                     } each state

**Stage 2** At state (2, 1) there are 3 possible actions

$$r(2, 1, 1) + f(1, 1) = 4 + 3 = 7$$

$$r(2, 1, 2) + f(1, 2) = 6 + 6 = 12$$

$$r(2, 1, 3) + f(1, 3) = 2 + 2 = 4$$

at (2, 2)

$$r(2, 2, 1) + f(1, 1) = 4 + 3 = 7$$

$$r(2, 2, 2) + f(1, 2) = 5 + 6 = 11$$

$$r(2, 2, 3) + f(1, 3) = 7 + 2 = 9$$

at (2, 3)

$$r(2, 3, 1) + f(1, 1) = 8 + 3 = 11$$

$$r(2, 3, 2) + f(1, 2) = 4 + 6 = 10$$

$$r(2, 3, 3) + f(1, 3) = 5 + 2 = 7$$

### Stage 3

$$r(3, 1, 1) + f(2, 1) = 7 + 4 = 11$$

$$r(3, 1, 2) + f(2, 2) = 4 + 7 = 11$$

$$r(3, 1, 3) + f(2, 3) = 1 + 7 = 8$$

### Optimal path

$$(3, 1) \rightarrow (2, 3) \rightarrow (1, 3) \rightarrow (0, 1)$$

Total distance = 8.

### Value iteration algorithm (general)

Recurrence relation:-

$$f(n, i) = \underset{k \in K}{\text{Min / Max}} \{ r(n, i, k) + f(n-1, j) \}$$

Transition equation:

$$j = t(n, i, k)$$

Terminal values  $f(0, 1)$  must be given for all terminal states.

Recurrence relation sometimes expressed as:-

$$f(n, i) = \underset{k \in K}{\text{Min / Max}} \{ r(n, i, k) + f(n-1, t(n, i, k)) \}$$

$k \in K$

## **Bellman's Principle of Optimality**

**An optimal policy has the property that whatever the current state and decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the current decision.**

## Loading Problem

**How many items (several types) to put into a load to maximise the total value of items packed, given limited capacity.**

$$\text{Maximise } \sum_{n=1}^N V_n D_n$$

**Subject to**

$$\sum_{n=1}^N W_n D_n \leq K$$

$$D_n \geq 0, \text{ integer for all } n$$

**Where**

$V_n$  = value per unit of type  $n$

$D_n$  = number of units of type  $n$  packed

$W_n$  = weight per item of type  $n$

$K$  = capacity (weight)

## Example

Capacity = 5 tonnes. 3 item types.

<u>Type</u>	<u>Unit weight (tonnes)</u> <u>value(€)</u>	<u>Unit</u>
1	2	65
2	3	80
3	1	30

Stage → decide number of items type  $n$  to pack

State → remaining capacity  $i$

Recurrence relation:-

$$f(n, i) = \underset{k \in K}{\text{Max}} \{ r(n, i, k) + f(n-1, j) \}$$

Terminal state  $f(0, i) = 0$  for all  $i$ .

Stage 1  $k$  = number of items of type 1 to pack.

<u><math>i</math></u>	<u><math>k</math></u>	<u><math>r(n, i, k)</math></u>	<u><math>f(0, j)</math></u>	<u>Total</u>
5	2	130	0	130
4	2	130	0	130
3	1	65	0	65
2	1	65	0	65
1	0	0	0	0
0	0	0	0	0

**Stage 2  $k$  = number of items of type 2 to pack.**

$$j = i - 3k$$

<u><math>i</math></u>	<u><math>k</math></u>	<u><math>r(n, i, k)</math></u>	<u><math>f(1, j)</math></u>	<u>Total</u>
5	1	80	$f(1,2) = 65$	145
5	0	0	$f(1,5) = 130$	130
4	1	80	$f(1,1) = 0$	80
4	0	0	$f(1,4) = 130$	130
3	1	80	$f(1,0) = 0$	80
3	0	0	$f(1,3) = 65$	65
2	0	0	$f(1,2) = 65$	65
1	0	0	$f(1,1) = 0$	0
0	0	0	$f(1,0) = 0$	0

**Stage 3**  $k$  = number of items of type 3 to pack.

$$j = i - k$$

<u><math>i</math></u>	<u><math>k</math></u>	<u><math>r(n, i, k)</math></u>	<u><math>f(2, j)</math></u>	<u>Total</u>
5	5	150	$f(2,0) = 0$	150
5	4	120	$f(2,1) = 0$	120
5	3	90	$f(2,2) = 65$	155
5	2	60	$f(2,3) = 80$	140
5	1	30	$f(2,4) = 130$	160
5	0	0	$f(2,5) = 145$	145

$f(3, 5) = 160 \rightarrow$  Pack one of item type 3

$\rightarrow$  Pack 0 of item type 2

$\rightarrow$  Pack 2 of item type 1.

## Production and Inventory Planning

<b>Orders for product:</b>	<b>January</b>	<b>2</b>
	<b>February</b>	<b>5</b>
	<b>March</b>	<b>3</b>
	<b>April</b>	<b>4</b>

**Production capacity = 5 units per month**

**Production cost = €50 (fixed) + €20 per unit**

**Holding cost = €4 per unit per month**

**Storage capacity = 4 units**

**Zero inventory start and finish.**

**Formulation:**

$$\text{Minimise } \sum_{n=1}^N R_n$$

**Subject to**

$$D_n \leq K \quad \text{for all } n = 1 \dots N$$

$$S_n + D_n - q_n \leq W \quad \text{for all } n = 1 \dots N$$

$$S_n + D_n \geq q_n \quad \text{for all } n = 1 \dots N$$

$$D_n \geq 0, \text{ integer} \quad \text{for all } n = 1 \dots N$$

**Where**

$R_n$  = total cost in month  $n$

$D_n$  = number of units made in month  $n$

$S_n$  = starting inventory in month  $n$

$W$  = storage capacity

$K$  = production capacity

$q_n$  = demand in month  $n$

**Stage** → month

**State** → inventory at start of month

**Action** → number of units produced

$f(0, i) = 0$  for all  $i$

**Stage 1 (April)**

<u><math>i</math></u>	<u><math>k</math></u>	<u><math>r(n, i, k)</math></u>	<u><math>f(0, j)</math></u>	<u>Total</u>
4	0	0	0	0
3	1	70	0	70
2	2	90	0	90
1	3	110	0	110
0	4	130	0	130

**Stage 2 (March)**

$$j = i + k - 3$$

---

$$r(n, i, k)$$

<u><i>i</i></u>	<u><i>k</i></u>	<u>Prod'n</u> <u>cost</u>	<u>Holding</u> <u>cost</u>	<u><i>f</i>(<i>L</i>, <i>i</i>)</u>	<u>Total</u>
4	3	110	16	0	126
	2	90	12	70	172
	1	70	8	90	168
	0	0	4	110	114
3	4	130	16	0	146
	3	110	12	70	192
	2	90	8	90	188
	1	70	4	110	184
	0	0	0	130	130
2	5	150	16	0	166
	4	130	12	70	212
	3	110	8	90	208
	2	90	4	110	204
	1	70	0	130	200
1	5	150	12	70	232
	4	130	8	90	228
	3	110	4	110	224
	2	90	0	130	220
0	5	150	8	90	248
	4	130	4	110	244
	3	110	0	130	240

**Stage 3 (February)       $j = i + k - 5$**

<i>i</i>	<i>k</i>	<u><math>r(n, i, k)</math></u>		<u><math>f(2, j)</math></u>	<u>Total</u>
		<u>Prod'n cost</u>	<u>Holding cost</u>		
3	5	150	12	130	292
	4	130	8	166	304
	3	110	4	220	334
	2	90	0	240	330
2	5	150	8	166	324
	4	130	4	220	354
	3	110	0	240	350
1	5	150	4	220	374
	4	130	0	240	370
0	5	150	0	240	390

**Stage 4 (January)       $j = i + k - 2$**

<i>i</i>	<i>k</i>	<u><math>r(n, i, k)</math></u>		<u><math>f(3, j)</math></u>	<u>Total</u>
		<u>Prod'n cost</u>	<u>Holding cost</u>		
0	5	150	12	292	454
	4	130	8	324	462
	3	110	4	370	484
	2	90	0	390	480

**Optimal schedule:-**

**January produce 5 → February produce 5  
→ March produce 0 → April produce 4.**

### **Discounted Returns**

**Discount factor =  $1 / (1 + r) = b$**

**$r > 0$                       Therefore  $0 \leq b \leq 1$**

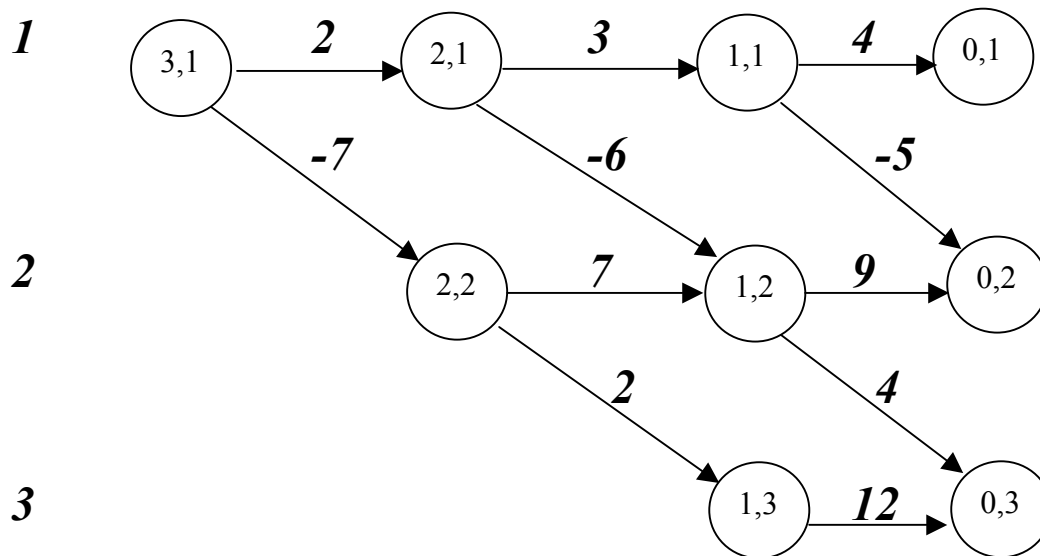
### **Capital Investment Problem**

- **Company has a plant, capacity level 1**
  - **Possible actions in year 1:-**
    - **maintain level 1 capacity (net return = 2)**
    - **expand to level 2 capacity (net return = -7)**
- ..... and so on.**

### Three year plan:-

**Note: maximum capacity = level 3. No reduction possible.**

**Capacity**  
**Level,  $i$**



***Yrs. 3 2 1 0***  
***remaining***

### Value of plant at end of three years:-

**Level 1 = 0 (  $f(0, 1) = 0$  )**

**Level 2 = 4 (  $f(0, 2) = 4$  )**

**Level 3 = 8 (  $f(0, 3) = 8$  )**

**Interest rate = 33.33% p.a. i.e.  $b = 1 / 1.33 = 0.75$**

**Objective is to maximise present value.**

**Recurrence relation:-**

$$f(n, i) = \underset{k \in K}{\text{Max}} \{ r(n, i, k) + b.f(n-1, j) \}$$

**$k = 1 \rightarrow$  maintain current capacity**

**$k = 2 \rightarrow$  expand to next level**

$$j = i + k - 1$$

$$f(0, 1) = 0 \quad f(0, 2) = 4 \quad f(0, 3) = 8$$

**At node (1, 1):**

$$r(1, 1, 1) + 0.75 f(0, 1) = 4 + 0.75 * 0 = 4 *$$

$$r(1, 1, 2) + 0.75 f(0, 2) = -5 + 0.75 * 4 = -2$$

$$f(1, 1) = 4$$

**and so on.**

<u>Stage (n)</u>	<u>State (i)</u>	<u>Action (k)</u>	<u>Value</u>
1	1	1	$4 + 0.75 * 0 = 4$
1	1	2	$-5 + 0.75 * 4 = -2$
1	2	1	$9 + 0.75 * 4 = 12$
1	2	2	$4 + 0.75 * 8 = 10$
1	3	1	$12 + 0.75 * 8 = 18$
2	1	1	$3 + 0.75 * 4 = 6$
2	1	2	$-6 + 0.75 * 12 = 3$
2	2	1	$7 + 0.75 * 12 = 16$
2	2	2	$2 + 0.75 * 18 = 15$
3	1	1	$2 + 0.75 * 6 = 6.5$
3	1	2	$-7 + 0.75 * 16 = 5$