OPERATIONS RESEARCH

O R

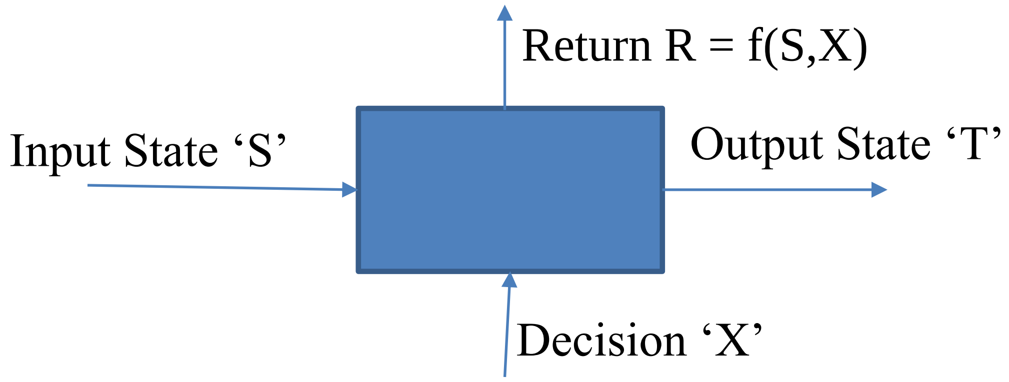
Dynamic Programming (DP)

By: -
Dr. Hakeem–Ur–Rehman
FAST, Lahore

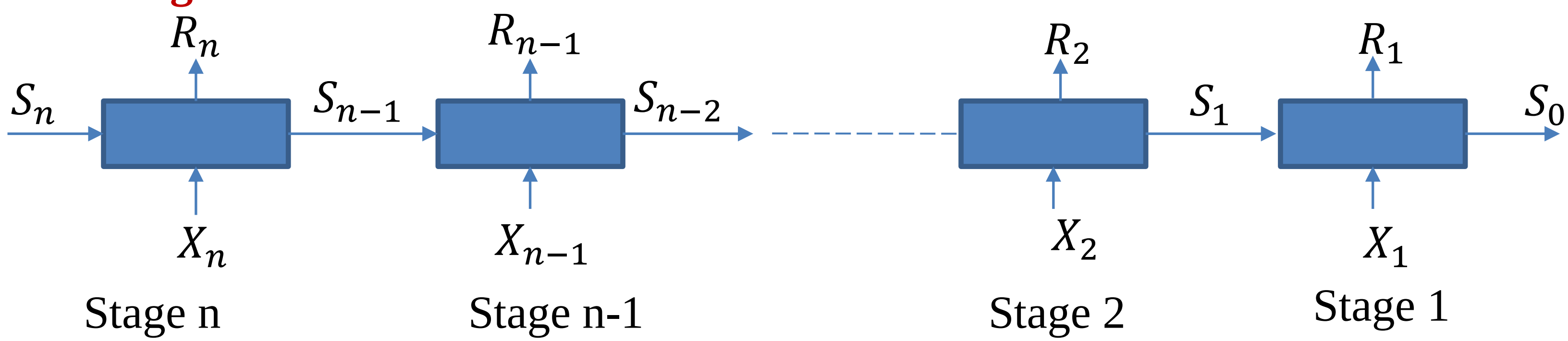
Dynamic Programming (DP): An Overview

- Developed by R. Bellman in 1950 (USA)
- DP is used when problem can be divided into "STAGES"

Single stage Decision Problem



Multi-stage Decision Problem



- Here the decisions made in a stage will affect the decisions at the subsequent stages because they are dependent on each other.
- DP also know as "Multi-Stage Optimization Problem or Recursive Optimization Problem".

Dynamic Programming (DP): An Overview…

- The problem can be divided into stages
  - **Stage:** 1, 2, …, N; Label of stage: n (n=1, 2, …, N) → Stages are NOT independent

- At the beginning of each stage: a number of states → Current **state for stage 'n': $\boldsymbol{S_n}$**

- At each stage: a decision is to be made → Decision variable for stage 'n': $\boldsymbol{x_n} \in \boldsymbol{D_n}(\boldsymbol{s_n})$
  - **A policy decision**: transforming the current state to a state of the next stage
    - Deterministic: $S_{n+1} = f(s_n, x_n)$
    - Probabilistic: $S_{n+1} = f_k(s_n, x_n)$ with probability p_k

- Dynamic Programming: Basic Principles
  - Divided and Conquer
  - Bottom up Computation

"**BACKWARD RECURSIVE APPROACH**" where the decisions are made at the end stages rather than from the beginning stages.

- Deterministic or Probabilistic Dynamic Programming

Dynamic Programming (DP): An Overview…

- **Principle of Optimality**
  - Given the current state of the system, the optimal policy (sequence of decisions) for the remaining stages is independent of the policy adopted in the previous stage
  - The principle implies that, given the state S_i of the system at a stage i, one must proceed optimally till the last stage, irrespective of how one arrived at the state S_i.

  - **Markovian Property:**
    - Knowledge of the current state of the system conveys all the information about its previous behavior necessary for determining the optimal policy henceforth

Dynamic Programming (DP): An Overview…

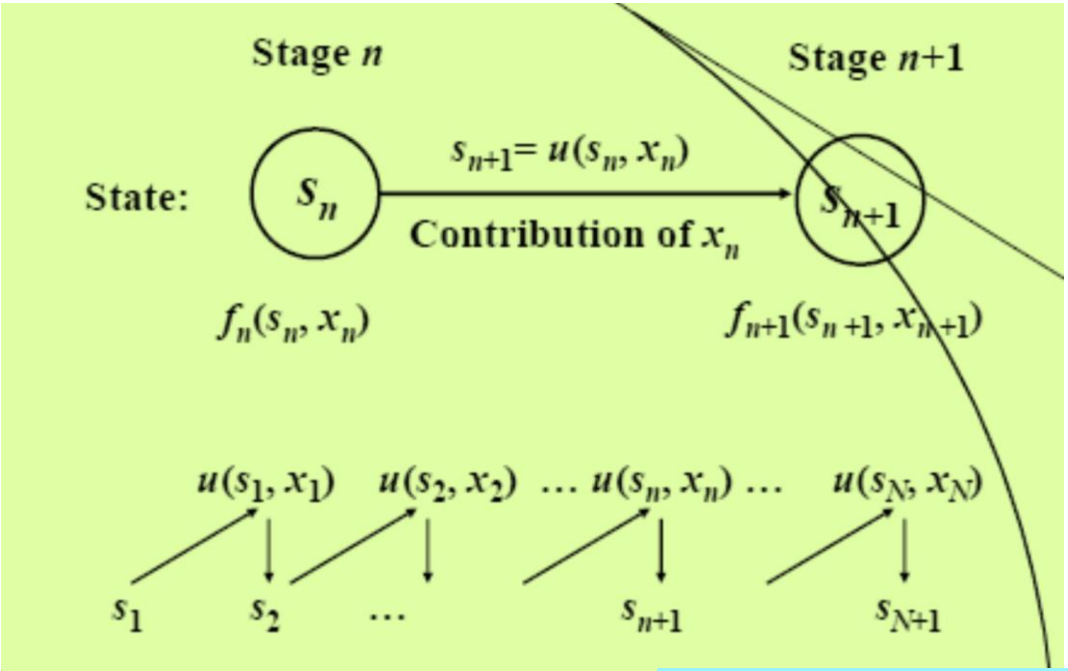
- **A recursive relationship**
  - Determining the optimal decision for stage n, given optimal decision for stage $n+1$
  - **Value functions**
    - $f_n(s_n, x_n)$: contribution of stages n, $n+1$, . . . , N to objective function if the system starts in state s_n at stage n, immediate decision is x_n, and optimal decisions are made thereafter
    - $f^*_n(s_n) = f_n(s_n, x_n^*)$
  - **Recursive relationship:**

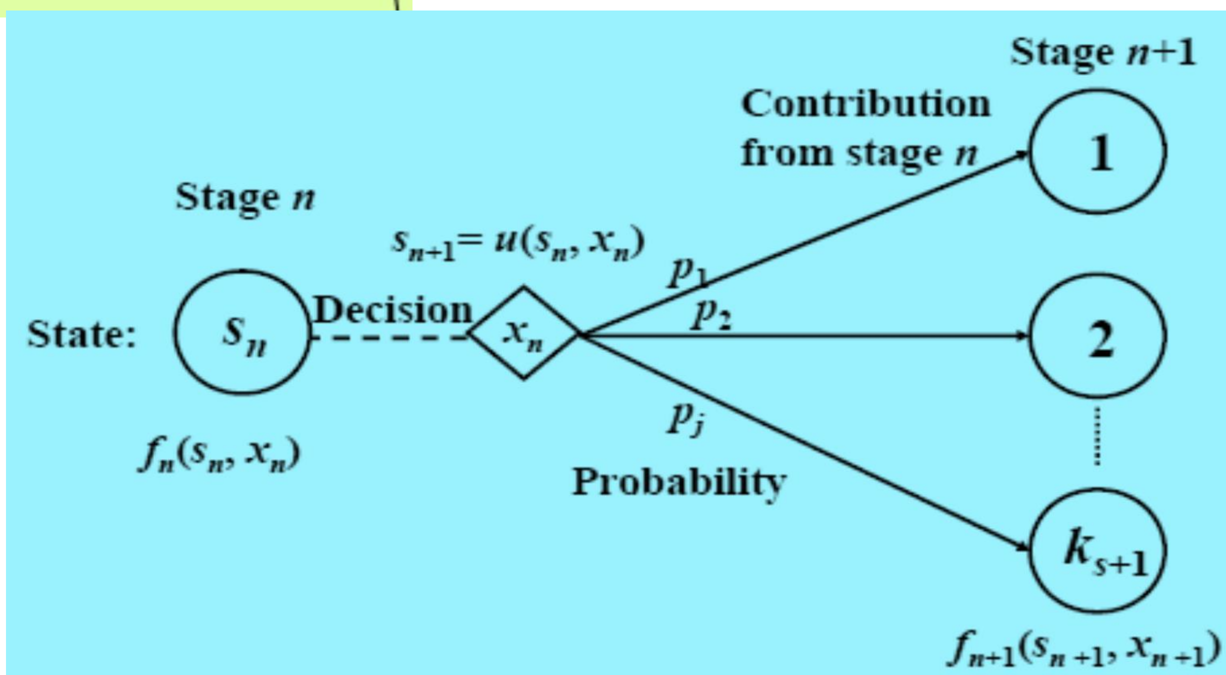
$$f^*_n(s_n) = \text{opt}_{x_n \in D_n(s_n)} \varphi[s_n, x_n, f^*_{n+1}(s_{n+1})]$$

  - **Boundary Condition**
  - **Optimal objective: $f^*_1(s_1)$**

x_n / s_n	$f_n(s_n, x_n)$	$f^*_n(s_n)$	x_n^*

Deterministic Vs Probabilistic DP

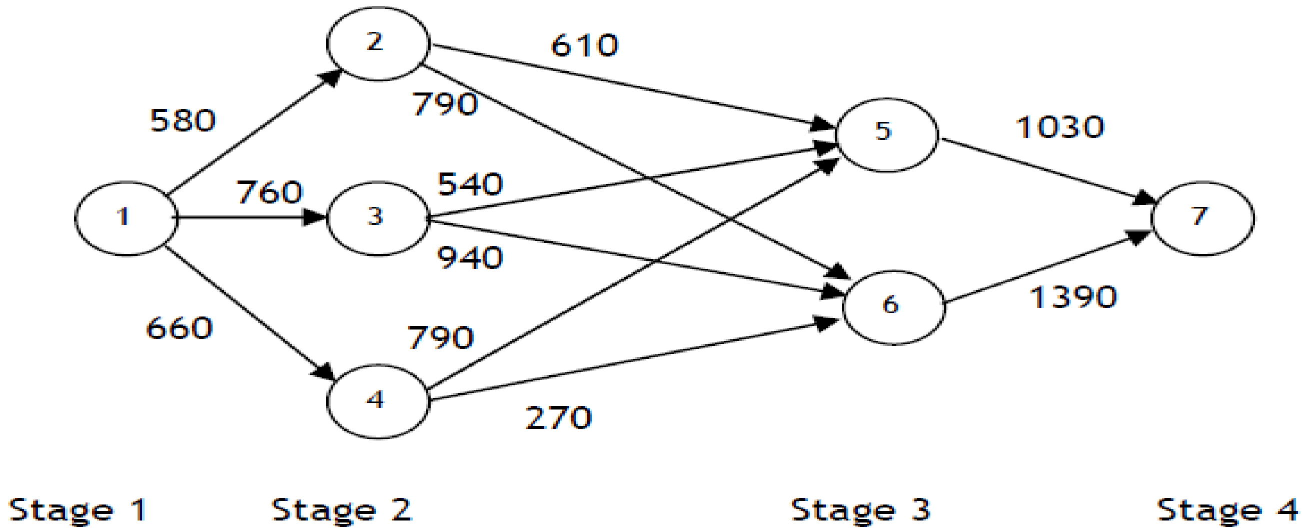




Stage Coach Problem

(Shortest Route Problem)

Consider the following example. Joe Cougar needs to travel from Nashville to Los Angeles. In order to minimize his total travel cost, he decided to spend each night at a friend's house living in each of the following cities: Kansas City, Omaha, Dallas, San Antonio and Denver. Joe knows that after one day of driving he can reach Kansas City, Omaha or Dallas and after 2 days he can arrive at San Antonio or Denver. Finally, after 3 days he can be at Los Angeles. Joe has to decide at which cities he should stay in order to minimize the total distance travelled. The corresponding network is shown on the following figure.



Each stage *i* contains the set of cities Joe can reach at the beginning of day *i*.

DP: Important Elements

- Stage:
  - Place / position of decision making (each stage contains set of alternatives)

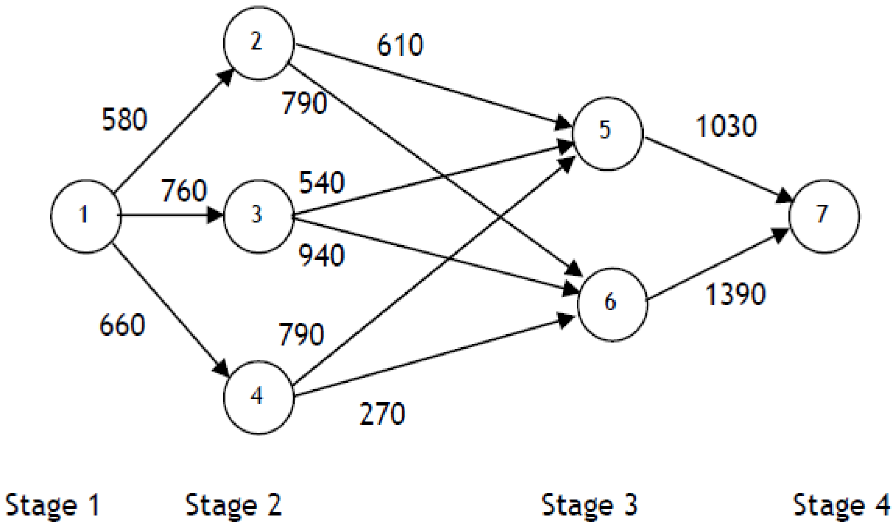
- State:
  - City at which the person is presently available

- Decision Variable:
  - Next destination

- Criterion of effectiveness:
  - Minimize the total distance

Stage Coach Problem (Shortest Route Problem)…

Forward Recursion Approach:



Suppose:

- c_{ij} = Distance between city 'i' & city 'j'
- $f_t(j)$ = Length of the shortest route from city 'i' to city 'j', given that city 'j' is at stage 't'

- **Stage – 1:** $f_1(1) = 0$

- **Stage – 2:** $f_2(2) = 580; f_2(3) = 760; f_2(4) = 660$

- **Stage – 3:**
  - 1 → 2 → 5 Distance = $c_{25} + f_2(2) = 610 + 580 = \mathbf{1190}$
  - 1 → 3 → 5 Distance = $c_{35} + f_2(3) = 540 + 760 = 1300$
  - 1 → 4 → 5 Distance = $c_{45} + f_2(4) = 790 + 660 = 1450$

  Shortest route from city '1' and city '5' is 1 → 2 → 5 with $f_3(5) = 1190$

  - 1 → 2 → 6 Distance = $c_{26} + f_2(2) = 790 + 580 = 1370$
  - 1 → 3 → 6 Distance = $c_{36} + f_2(3) = 940 + 760 = 1700$
  - 1 → 4 → 6 Distance = $c_{46} + f_2(4) = 270 + 660 = \mathbf{930}$

  Shortest route from city '1' and city '6' is 1 → 4 → 6 with $f_3(6) = 930$

- **Stage – 4:**
  - $f_4(7) = Min\,\{c_{57} + f_3(5) = 1030 + 1190 = \mathbf{2220}; c_{67} + f_3(6) = 1390 + 930 = 2320\}$

  Shortest route from city '1' and city '7' is 1 → 2 → 5 → 7 with $f_4(7) = 2220$

General Forward formulation

Let $f_k(s_k)$ be the value of an optimal sequence of decisions that takes the system from state 's_i' till state 's_k'.

The DP forward recursive formulation would be:

- $f_k(s_k) = \ Max/Min_{xk}\{g_k(s_k, x_k) + f_{k-1}(s_{k-1}\ )\}$
- Boundary Conditions: s_1 given
- Objective: find $f_{T+1}(s_{T+1})$

Applying this formulation to the shortest route example gives the following:

- $f_t(j) = \ Min_i\left\{f_{t-1}(i\ ) + c_{ij}\right\}$ Where' i' is a state at stage '$t-1$'
- Boundary Conditions: $f_1(1) = 0$
- Objective: find $f_4(7)$
  - In this case, $f_t(j)$ is the length of the shortest path from city '1' to city '7'.

Stage Coach Problem

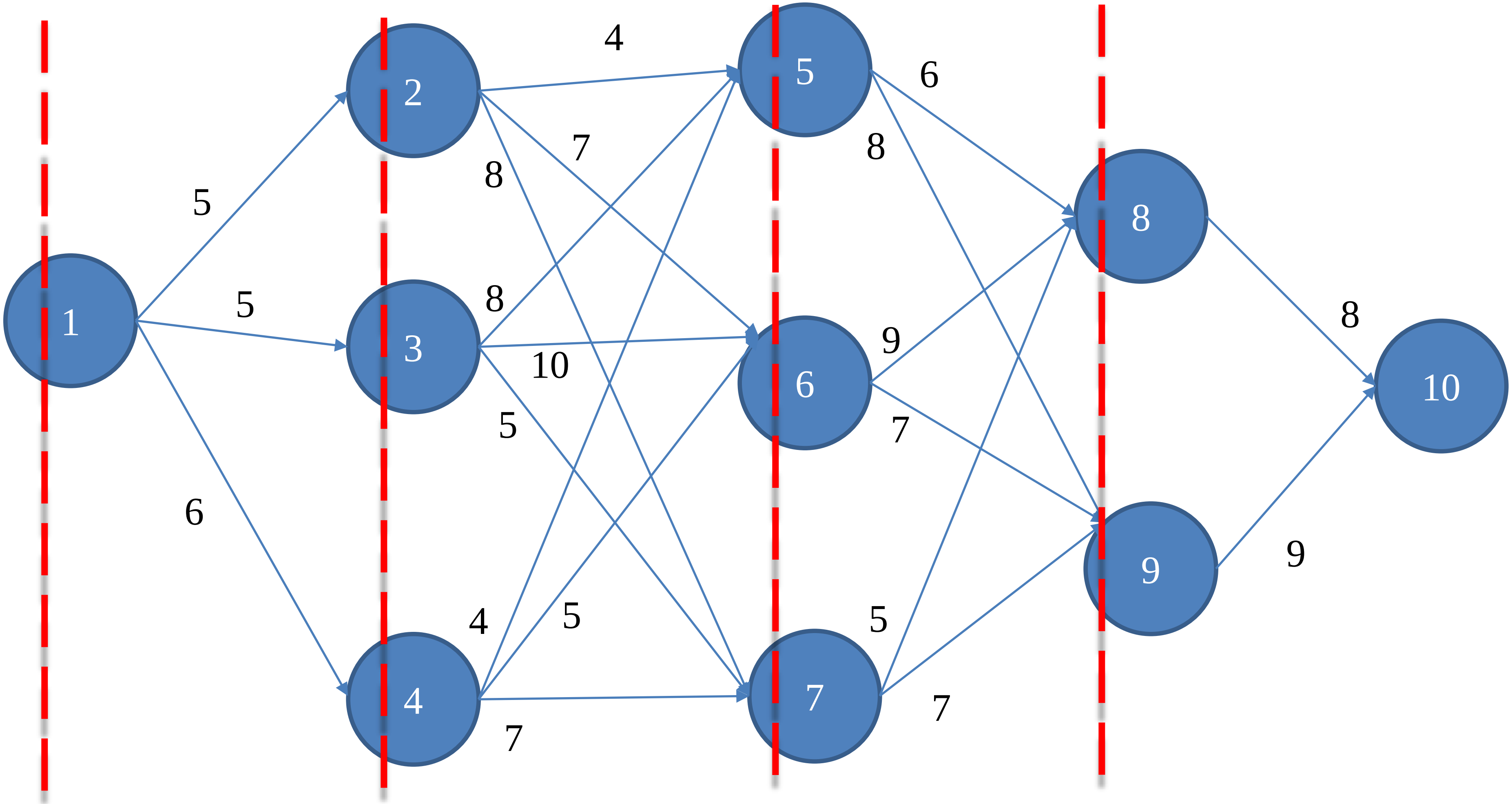
(Shortest Route Problem)

A Person want to go from city-1 to city-10. The various possibilities and the distances are given in the following table:

ARC	Distance	ARC	Distance
1-2	5	4-6	5
1-3	5	4-7	7
1-4	6	5-8	6
2-5	4	5-9	8
2-6	7	6-8	9
2-7	8	6-9	7
3-5	8	7-8	5
3-6	10	7-9	7
3-7	5	8-10	8
4-5	4	9-10	9

Stage Coach Problem…

EXAMPLE: (Shortest Route Problem)



DP: Important Elements

- Stage:
  - Place / position of decision making (each stage contains set of alternatives)
- State:
  - City at which the person is presently available
- Decision Variable:
  - Next destination
- Criterion of effectiveness:
  - Minimize the total distance

Stage Coach Problem…

EXAMPLE: (Shortest Route Problem)

Backward Recursive Approach

$n = 1$ (One more stage to go)

- $f_1(s, x_1) = d_{s,x_1}$
- $f_1^*(s) = Minimze\ f_1(s, x_1)$
- $x_1^* = Corresponding\ value$

s	x_1	$f_1^*(s)$	x_1^*
8	10	8	10
9	10	9	10

$n = 2$ (Two more stages to go)

- $f_2(s, x_2) = d_{s,x_2} + f_1^*(x_2)$
- $f_2^*(s) = Minimze\ f_2(s, x_2)$
- $x_2^* = Corresponding\ value$

s	x_2		$f_2^*(s)$	x_2^*
	8	9		
5	6 + 8 = 14	8 + 9 = 17	14	8
6	9 + 8 = 17	7 + 9 = 16	16	9
7	5 + 8 = 13	7 + 9 = 16	13	8

$n = 3$ (Three more stages to go)

- $f_3(s, x_3) = d_{s,x_3} + f_2^*(x_3)$
- $f_3^*(s) = Minimze\ f_3(s, x_3)$
- $x_3^* = Corresponding\ value$

s	x_3			$f_3^*(s)$	x_3^*
	5	6	7		
2	4 +14 = 18	7 + 16 = 23	8 + 13 = 21	18	5
3	8 + 14 = 22	10 + 16 = 26	5 + 13 = 18	18	7
4	4 + 14 = 18	5 + 16 = 21	7 + 13 = 20	18	5

Stage Coach Problem…

EXAMPLE: (Shortest Route Problem)

Backward Recursive Approach

$n = 4$ (Four more stages to go)

- $f_4(s, x_4) = d_{s,x_4} + f_3^*(x_4)$
- $f_4^*(s) = Minimze\ f_4(s, x_4)$
- $x_4^* = Corresponding\ value$

s	x_4			$f_4^*(s)$	x_4^*
	2	3	4		
1	5 + 18 = 23	5 + 18 = 23	6 + 18 = 24	23	2, 3

Shortest Distance = 23

Paths:

- **1 – 2 – 5 – 8 – 10**
- **1 – 3 – 7 – 8 – 10**

General Recursive Relationship:

- $f_n(s, x_n) = d_{s,x_n} + f_{(n-1)}^*(x_n)$
- $f_n^*(s) = minimize f_n(s, x_n)$
- $x_n^* = Corresponding\ value\ 'x_n'$

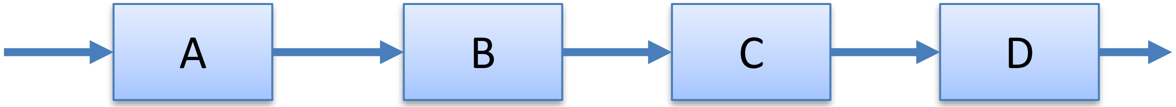
Resource allocation Problem

- **Problem description: resource allocation**
  - A resource: total number 'm' used for 'N' periods
  - If 'x_i' is allocated for period 'i', then profit is $g_i(x_i)$
  - How to allocate the resource to maximize the total profit over the 'N' periods?

- **Problem formulation**
  - Let $f_i(s_i)$ be maximal profit if the amount of s_i is allocated to the periods i, …, N
  - **Recursive relationship:** decision variable x_i
    - $f_i(s_i) = \max_{0\leq x_i \leq s_i} \{g_i(x_i) + f_{i+1}(s_i - x_i)\}$
  - **Boundary condition:** $f_N(s_N) = \max_{x_N = s_N} g_N(s_N)$
  - **Objective value:** $f_1(m)$

Reliability Problem

Consider an equipment that functions using four components connected in series:



Each component has a certain reliability of the equipment is the product of the individual reliabilities. In order to increase the reliability of the equipment, we can have some additional components in standby units for the four components are shown below:

Number of Units	Component A		Component B		Component C		Component D	
	Reliability	**Cost**	**Reliability**	**Cost**	**Reliability**	**Cost**	**Reliability**	**Cost**
1	0.6	100	0.7	80	0.7	75	0.8	60
2	0.7	130	0.75	100	0.8	80	0.85	70
3	0.8	150	0.8	120	0.9	85	0.88	75
4	0.9	170	0.85	140	0.95	90	0.90	80

We want to maximize the reliability of the system with the budget constraint of Rupees 400.

Reliability Problem…

- Stage: Components
- State: Amount of money available
- Decision Variable: How many Units are we going to have for each one of these components
- Criterion of effectiveness: Maximize the Reliability of the System

$n = 1$ (One more stage to go)

s	x_D	$f_1^*(s)$	x_D^*
60 – 69	1	0.8	1
70 – 74	2	0.85	2
75 – 79	3	0.88	3
80 – 145	4	0.90	4

For at least one component of 'D' we must start from it's cost "60" rather than "0"

For at least one component of 'A, B & C'; then upper limit must be 400 – (100+80+75) = 145

- $f_1(s, x_D) = R_{x_D}$
- $f_1^*(s) = Maximize\ f_1(s, x_D)$
  Subject to: $cx_D \leq s$
- $x_D^* = Corresponding\ value$

Reliability Problem…

A → B → C → D

$n = 2$ (Two more stages to go)

$$f_2(s, x_C) = R_{x_C} * f_1^*(s - c_{x_c})$$

s	x_c				$f_2^*(s)$	x_c^*
	1	2	3	4		
135 – 139	.7 * .8 = 0.56	–	–	–	0.56	1
140 – 144	.7 * .8 = 0.56	.8 * .8 = 0.64	–	–	0.64	2
145 – 149	.7 * .85 = 0.595	.8 * .8 = 0.64	.9 * .8 = 0.72	–	0.72	3
150 – 154	.7 * .88 = 0.616	.8 * .85 = 0.68	.9 * .8 = 0.72	.95 * .8 = 0.76	0.76	4
155 – 159	.7 * .9 = 0.63	.8 * .88 = 0.704	.9 * .85 = 0.765	.95 * .8 = 0.76	0.765	3
160 – 164	.7 * .9 = 0.63	.8 * .9 = 0.72	.9 * .88 = 0.792	.95 * .85 = 0.8075	.8075	4
165 – 169	.7 * .9 = 0.63	.8 * .9 = 0.72	.9 * .9 = 0.81	.95 * .88 = 0.836	0.836	4
170 – 220	.7 * .9 = 0.63	.8 * .9 = 0.72	.9 * .9 = 0.81	.95 * .9 = 0.855	0.855	4

For at least one component of ‘C & D’ we must start from it’s cost “60+75=135”

For at least one component of ‘A & B’; then upper limit must be 400 – (100+80) = 220

Reliability Problem…

For at least one component of 'B,C & D' we must start from it's cost "80+60+75=215"

$n = 3$ (Three more stages to go)

$$f_3(s, x_B) = R_{x_B} * f_2^*(s - c_{x_B})$$

s	x_B				$f_3^*(s)$	x_B^*
	1	2	3	4		
215 – 219	.7 * .56 = 0.392	–	–	–	0.392	1
220 – 224	.7 * .64 = 0.448	–	–	–	0.448	1
225 – 229	.7 * .72 = 0.504	–	–	–	0.504	1
230 – 234	.7 * .76 = 0.532	–	–	–	0.532	1
235 – 239	.7 * .765 = 0.5355	.75 * .56 = 0.42	–	–	0.5355	1
240 – 244	.7 * .8075 = .5653	.75 * .64 = .48	–	–	0.5653	1
245 – 249	.7 * .836 = .5852	.75 * .72 = 0.54	–	–	0.5852	1
250 – 254	.7 * .855 = 0.5985	.75 * .76 = 0.57	–	–	0.5985	1
255 – 259	.7 * .855 = .5985	.75 * .765 = .5738	.8 * .56 = 0.448	–	0.5985	1
260 – 264	.7 * .855 = 0.5985	.75 * .8075 = .6056	.8 * .64 = 0.512	–	0.6056	2
265 – 269	.7 * .855 = 0.5985	.75 * .836 = 0.627	.8 * .72 = 0.576	–	0.627	2
270 – 274	.7 * .855 = 0.5985	.75 * .855 = .64125	.8 * .76 = 0.608	–	0.64125	2
275 – 279	.7 * .855 = 0.5985	75 * .855 = .64125	.8 * .765 = .612	.8 * .56 = .476	0.64125	2

For at least one component of 'A'; then upper limit must be 400 – 100 = 300

Reliability Problem…

$n = 3$ (Three more stages to go)

$$f_3(s, x_B) = R_{x_B} * f_2^*(s - c_{x_B})$$

s	x_B				$f_3^*(s)$	x_B^*
	1	2	3	4		
280 – 284	7 * .855 = 0.5985	75 * .855 = .64125	.8 * .8075 = .646	.85 * .64 = .544	.646	3
285 – 289	7 * .855 = 0.5985	75 * .855 = .64125	.8 * .836 = .6688	.85 * .72 = .612	.6688	3
290 – 294	7 * .855 = 0.5985	75 * .855 = .64125	.8 * .855 = 0.684	.85 * .76 = .646	.684	3
295 – 299	7 * .855 = 0.5985	75 * .855 = .64125	.8 * .855 = 0.684	.85 * .765 = .650	.684	3
300	7 * .855 = 0.5985	75 * .855 = .64125	.8 * .855 = 0.684	.85 *.8075 = .686	.686	4

$n = 4$ (Four more stages to go)

$$f_4(s, x_A) = R_{x_A} * f_3^*(s - c_{x_A})$$

s	x_A				$f_4^*(s)$	x_A^*
	1	2	3	4		
400	.6 * .686 = 0.4116	.7 * .64125 = .4489	.8 * .5985 = .4788	.9 * .532 = 0.4788	0.4788	3, 4

Alternative Optimum Solution:

$x_A = 3, x_B = 1, x_C = 4, x_D = 4$ Or $x_A = 4, x_B = 1, x_C = 4, x_D = 1$

Equipment Replacement Problem

- **An equipment: when to replace?**
  - $p_i(t)$: the profit in year 'i' generated by a machine aging 't'
  - $m_i(t)$: the maintenance (or running) cost in year 'i' for a machine aging 't'
  - $c_i(t)$: the replacement cost in year 'i' for a machine aging 't'
    - $c_i(t) = Salvage\ value\ s_i(t) - Buying\ new\ equipment\ cost\ (c)$
  - T: the age of the machine in year 1
  - Planning horizon: 'n'

Value functions

- The maximal profit from year 'i' to 'n' generated by a machine aging 't'
- The decision in year 'i': $x_i = 1\ or\ 0$ (Replace or not)

- Stage: Year
- State: Age of the equipment
- Decision Variable: Keep (K) / Replace (R)
- Criterion of effectiveness: Maximize Revenue

Equipment Replacement Problem…

EXAMPLE

Age	0	1	2	3	4	5	6	7	8
$p_i(t)$	200	180	160	150	130	120	100	90	80
$m_i(t)$	10	15	20	27	35	45	55	65	75
$s_i(t)$	800	700	650	600	550	500	500	400	400

Recursive Relationship for <u>Keeping the machine</u> (i.e. Equipment):

$$f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$$

Recursive Relationship for <u>Replacing the machine</u> (i.e. Equipment):

$$f_n(i,R) = c_i(t) + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$$

$$f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$$

Equipment Replacement Problem…

Age (i)	0	1	2	3	4	5	6	7	8
$p_i(t)$	200	180	160	150	130	120	100	90	80
$m_i(t)$	10	15	20	27	35	45	55	65	75
$s_i(t)$	800	700	650	600	550	500	500	400	400

Keep:

- $\boldsymbol{i} = \mathbf{0}$: $f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$ = 200 – 10 + 0 = 190
- $\boldsymbol{i} = \mathbf{1}$: $f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$ = 180 – 15 + 0 = 165
- $\boldsymbol{i} = \mathbf{2}$: $f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$ = 160 – 20 + 0 = 140

Replace:

- $\boldsymbol{i} = \mathbf{0}$: $f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$ = 800 – 800 +200 – 10 + 0 = 190
- $\boldsymbol{i} = \mathbf{1}$: $f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$ = 700 – 800 +200 – 10 + 0 = 90
- $\boldsymbol{i} = \mathbf{2}$: $f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$ = 650 – 800 +200 – 10 + 0 = 40

n = 1 (One more stage to go)

Age ‘*i*’	0	1	2	3	4	5	6	7	8
Keep	190	165	140	123	95	75	45	25	5
Replace	190	90	40	−10	−60	−110	−110	−210	−210

Equipment Replacement Problem…

Age 'i'	0	1	2	3	4	5	6	7	8
Keep	**190**	**165**	**140**	123	95	75	45	25	5
Replace	**190**	90	40	−10	−60	−110	−110	−210	−210

Age 'i'	0	1	2	3	4	5	6	7	8
Keep	190	**165**	**140**	**123**	95	75	45	25	5
Replace	190	90	40	−10	−60	−110	−110	−210	−210

Keep:

- $\boldsymbol{i=0}$: $f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$ = **190** + **165** = 355
- $\boldsymbol{i=1}$: $f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$ = **165** + **140** = 305
- $\boldsymbol{i=2}$: $f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$ = **140** + **123** = 263

Replace:

- $\boldsymbol{i=0}$: $f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$ = **190** + **165** = 355
- $\boldsymbol{i=1}$: $f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$ = 90 + **165** = 255
- $\boldsymbol{i=2}$: $f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$ = 40 + **165** = 205

n = 2 (Two more stages to go)

Age 'i'	0	1	2	3	4	5	6	7
Keep	355	305	263	218	170	120	70	30
Replace	355	255	205	155	105	55	55	−45

Equipment Replacement Problem…

Age 'i'	0	1	2	3	4	5	6	7	8
Keep	**190**	**165**	**140**	123	95	75	45	25	5
Replace	**190**	90	40	−10	−60	−110	−110	−210	−210

Age 'i'	0	1	2	3	4	5	6	7
Keep	355	**305**	**263**	**218**	170	120	70	30
Replace	355	255	205	155	105	55	55	−45

Keep:

- $\boldsymbol{i=0}$: $f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$ = **190** + **305** = 495
- $\boldsymbol{i=1}$: $f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$ = **165** + **263** = 428
- $\boldsymbol{i=2}$: $f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$ = **140** + **218** = 358

Replace:

- $\boldsymbol{i=0}$: $f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$ = **190** + **305** = 495
- $\boldsymbol{i=1}$: $f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$ = 90 + **305** = 395
- $\boldsymbol{i=2}$: $f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$ = 40 + **305** = 345

n = 3 (Three more stages to go)

Age 'i'	0	1	2	3	4	5	6
Keep	495	428	358	293	215	145	75
Replace	495	395	345	295	245	195	195

Equipment Replacement Problem…

Age 'i'	0	1	2	3	4	5	6	7	8
Keep	**190**	**165**	**140**	123	95	75	45	25	5
Replace	**190**	90	40	−10	−60	−110	−110	−210	−210

Age 'i'	0	1	2	3	4	5	6
Keep	495	**428**	**358**	293	215	145	75
Replace	495	395	345	**295**	245	195	195

Keep:

- $\boldsymbol{i=0}$: $f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$ = **190** + **428** = 618
- $\boldsymbol{i=1}$: $f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$ = **165** + **358** = 523
- $\boldsymbol{i=2}$: $f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$ = **140** + **295** = 435

Replace:

- $\boldsymbol{i=0}$: $f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$ = **190** + **428** = 618
- $\boldsymbol{i=1}$: $f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$ = 90 + **428** = 518
- $\boldsymbol{i=2}$: $f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$ = 40 + **428** = 468

n = 4 (Four more stages to go)

Age 'i'	0	1	2	3	4	5
Keep	618	523	435	368	290	270
Replace	618	518	468	418	368	318

Equipment Replacement Problem…

Age 'i'	0	1	2	3	4	5	6	7	8
Keep	**190**	**165**	**140**	123	95	75	45	25	5
Replace	**190**	90	40	−10	−60	−110	−110	−210	−210

Age 'i'	0	1	2	3	4	5
Keep	618	**523**	435	368	290	270
Replace	618	518	**468**	**418**	368	318

Keep:

- $\boldsymbol{i = 0}$: $f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$ = **190** + **523** = 713
- $\boldsymbol{i = 1}$: $f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$ = **165** + **468** = 633
- $\boldsymbol{i = 2}$: $f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$ = **140** + **418** = 558

Replace:

- $\boldsymbol{i = 0}$: $f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$ = **190** + **523** = 713
- $\boldsymbol{i = 1}$: $f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$ = 90 + **523** = 613
- $\boldsymbol{i = 2}$: $f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$ = 40 + **523** = 563

n = 5 (Five more stages to go)

Age 'i'	0	1	2	3	4
Keep	713	633	558	491	413
Replace	713	613	563	513	463

Equipment Replacement Problem…

Age 'i'	0	1	2	3	4	5	6	7	8
Keep	**190**	**165**	**140**	123	95	75	45	25	5
Replace	**190**	90	40	−10	−60	−110	−110	−210	−210

Age 'i'	0	1	2	3	4
Keep	713	**633**	558	491	413
Replace	713	613	**563**	**513**	463

Keep:

- $\boldsymbol{i = 0}$: $f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$ = **190** + **633** = 823
- $\boldsymbol{i = 1}$: $f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$ = **165** + **563** = 728
- $\boldsymbol{i = 2}$: $f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$ = **140** + **513** = 653

Replace:

- $\boldsymbol{i = 0}$: $f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$ = **190** + **633** = 823
- $\boldsymbol{i = 1}$: $f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$ = 90 + **633** = 723
- $\boldsymbol{i = 2}$: $f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$ = 40 + **633** = 673

n = 6 (Six more stages to go)

Age 'i'	0	1	2	3
Keep	823	728	653	586
Replace	823	723	673	623

Equipment Replacement Problem…

Age 'i'	0	1	2	3	4	5	6	7	8
Keep	**190**	**165**	**140**	123	95	75	45	25	5
Replace	**190**	90	40	−10	−60	−110	−110	−210	−210

Age 'i'	0	1	2	3
Keep	823	**728**	653	586
Replace	823	723	**673**	**623**

Keep:

- $\boldsymbol{i = 0}$: $f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$ = **190** + **728** = 918
- $\boldsymbol{i = 1}$: $f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$ = **165** + **673** = 838
- $\boldsymbol{i = 2}$: $f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$ = **140** + **623** = 763

Replace:

- $\boldsymbol{i = 0}$: $f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$ = **190** + **728** = 918
- $\boldsymbol{i = 1}$: $f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$ = 90 + **728** = 818
- $\boldsymbol{i = 2}$: $f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$ = 40 + **728** = 768

n = 7 (Seven more stages to go)

Age 'i'	0	1	2
Keep	918	838	763
Replace	918	818	768

Equipment Replacement Problem…

Age 'i'	0	1	2	3	4	5	6	7	8
Keep	**190**	**165**	140	123	95	75	45	25	5
Replace	**190**	90	40	−10	−60	−110	−110	−210	−210

Age 'i'	0	1	2
Keep	918	**838**	763
Replace	918	818	**768**

Keep:

- $\boldsymbol{i = 0}$: $f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$ = **190** + **838** = 1028
- $\boldsymbol{i = 1}$: $f_n(i,k) = p_i(t) - m_i(t) + f^*_{(n-1)}(i+1)$ = **165** + **768** = 933

Replace:

- $\boldsymbol{i = 0}$: $f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$ = **190** + **838** = 1028
- $\boldsymbol{i = 1}$: $f_n(i,R) = s_i(t) - c + p_i(0) - m_i(0) + f^*_{(n-1)}(1)$ = 90 + **838** = 928

n = 8 (Eight more stages to go)

Age 'i'	0	1
Keep	1028	933
Replace	1028	928

Equipment Replacement Problem…

- Decision is **Keep** → Look at '$i + 1$' in the next year (Previous row)

- Decision is **Replace** → Look at '$i = 1$' in the previous row.

- For Example:
    - (when n = 3 (i.e. 3 more years to go) & i = 1 (i.e. Age of Equipment = 1)
        - At n = 3 & i = 1 → Keep, Keep, Keep
    - (when n = 7 (i.e. 7 more years to go) & i = 2 (i.e. Age of Equipment = 2)
        - At n = 7 & i = 2 → Replace, Keep, Replace, Keep, Keep, Keep, Keep

DP: Continuous Variables

$$Minimize\ z = x_1 + x_2 + x_3$$

Subject to:

$$x_1 x_2 x_3 = 27$$

$$x_1, x_2, x_3 \geq 0$$

- Stage: Each Variable
- State: Resource available for allocation
- Decision Variable: Values of x_1, x_2, x_3
- Criterion of effectiveness: Maximize 'z'

$Minimize\ z = x_1 + x_2 + x_3$
Subject to:

$$x_1 x_2 x_3 = 27$$
$$x_1, x_2, x_3 \geq 0$$

DP: Continuous Variables

n = 1 (One stage to go)

$f_1(s_1, x_3) = x_3$

$f_1^*(s_1) = Minimize\ \ x_3$

Subject to: $x_3 = s_1$ & $x_3 \geq 0$

$\boldsymbol{x_3^* = s_1 \rightarrow f_1^*(s_1) = s_1}$

$x_3^* = s_1 = 3$

n = 2 (Two stages to go)

$f_2(s_2, x_2) = x_2 + f_1^*(s_1) = x_2 + f_1^*(\frac{s_2}{x_2})$

$f_2^*(s_2) = Minimize\ \ x_2 + \frac{s_2}{x_2}$

Subject to: $x_2 \geq 0$

$\frac{df_2}{dx_2} = 0 \rightarrow 1 + \frac{s_2}{x_2^2} = 0 \rightarrow x_2 = +\sqrt{s_2}$

$\boldsymbol{x_2^* = \sqrt{s_2} \rightarrow f_2^*(s_2)} = \sqrt{s_2} + \frac{s_2}{\sqrt{s_2}} = \boldsymbol{2\sqrt{s_2}}$

$x_2^* = \sqrt{s_2} = \sqrt{9} = 3$

$\frac{d^2 f_2}{dx_2^2} > 0\ Minimize$

n = 3 (Three stages to go)

$f_3(27, x_1) = x_1 + f_2^*(s_2) = x_1 + f_2^*(\frac{27}{x_1})$

$f_3^*(27) = Minimize\ \ x_1 + 2\sqrt{\frac{27}{x_1}}$

$\frac{df_3}{dx_1} = 0 \rightarrow 1 + 2\sqrt{27}(-\frac{1}{2}\ x_1^{-\frac{3}{2}}) = 0 \rightarrow x_1^* = 3$

$f_3^*(27) = 3 + 2\sqrt{\frac{27}{3}} = 9 \rightarrow s_2 = 9$

$\frac{d^2 f_3}{dx_1^2} > 0\ Minimize$

DP: Integer Programming – Knapsack Problem

- The Model

$$Max. \sum_{j=1}^{n} v_j x_j$$

Subject to:

$$\sum_{j=1}^{n} w_j x_j \leq W$$

$$x_j: Non-negative\ integers\ (j = 1,2,\dots,n)$$

Example:

$$Max.\ \ 7y_1 + 8y_2 + 4y_3 + 9y_4$$

Subject to:

$$3y_1 + 2y_2 + y_3 + 2y_4 \leq 15$$

$$y_j \geq 0\ \&\ Integer$$

While solving these problems we have to simplify the problem in such a way that there is at least one variable with a coefficient of '+1' in the constraint.
So, here Variable 'y_3' satisfies the condition & we solve for this variable first always.

The Problem is rewritten as:

$$Max.\ \ 7x_1 + 8x_2 + 9x_3 + 4x_4$$

Subject to:

$$3x_1 + 2x_2 + 2x_3 + x_4 \leq 15$$

$$x_j \geq 0\ \&\ Integer$$

Stage: Each Variable
State: Resource available for allocation
Decision Variable: Values of x_j
Criterion of effectiveness: Maximize 'z'

DP: Integer Programming – Knapsack Problem…

The Problem is rewritten as:

$$Max.\ \ 7x_1 + 8x_2 + 9x_3 + 4x_4$$

Subject to:

$$3x_1 + 2x_2 + 2x_3 + x_4 \leq 15$$

$$x_j \geq 0 \ \&\ Integer$$

$n\ = 1$ (One more stage to go)

$f_1(s_1, x_4) = 4x_4$

$f_1^*(s_1) = Maximize\ 4x_4$

Subject to: $x_4 \leq s_1$ & x_4 Integer

Assuming that 's_1' is a non–negative integer

$\boldsymbol{f_1^*(s_1) = 4s_1}$ & $\boldsymbol{x_4^* = s_1}$

$n\ = 2$ (Two more stages to go)

$f_2(s_2, x_3) = 9x_3 + f_1^*(s_1)$

$f_2^*(s_2) = Maximize\ \ 9x_3 + f_1^*(s_2 - 2x_3)$

Subject to: $2x_3 \leq s_2$ & x_3 Integer

$\boldsymbol{f_2^*(s_2)} = Maximize\ \ 9x_3 + 4(s_2 - 2x_3) = \boldsymbol{Max.\ 4s_2 + x_3}$

Assuming that 's_2' is a non–negative integer

$\boldsymbol{x_3^*} = \left\lfloor \frac{s_2}{2} \right\rfloor$ & $\boldsymbol{f_2^*(s_2) = 4s_2 + \left\lfloor \frac{s_2}{2} \right\rfloor}$

$n\ = 3$ (Three more stages to go)

$f_3(s_3, x_2) = 8x_2 + f_2^*(s_2)$

$f_3^*(s_3) = Maximize\ \ 8x_2 + f_2^*(s_3 - 2x_2)$

Subject to: $2x_2 \leq s_3$ & x_2 Integer

$\boldsymbol{f_3^*(s_3)} = Maximize\ \ 8x_2 + 4(s_3 - 2x_2) + \left\lfloor \frac{(s_3 - 2x_2)}{2} \right\rfloor = \boldsymbol{Max.\ 4s_3 + \left\lfloor \frac{(s_3 - 2x_2)}{2} \right\rfloor}$

Assuming that 's_3' is a non–negative integer; the maximum occurs at

$\boldsymbol{x_2^* = 0}$ & $\boldsymbol{f_3^*(s_3) = 4s_3 + \left\lfloor \frac{s_3}{2} \right\rfloor}$

DP: Integer Programming – Knapsack Problem…

$\boldsymbol{n} = \mathbf{4}$ **(Four more stages to go)**

$f_4(15, x_1) = 7x_1 + f_3^*(s_3)$

$f_4^*(15) = Maximize\ \ 7x_1 + f_1^*(15 - 3x_1)$

Subject to: $3x_1 \leq 15$ & x_1 Integer

$$f_4^*(15) = Maximize\ \ 7x_1 + 4(15 - 3x_1) + \left\lfloor \frac{(15 - 3x_1)}{2} \right\rfloor$$

'x_1' can takes values 0 , 1, 2, 3, 4, & 5; we evaluate $f_4^*(15)$ for each of these values:

- At $x_1 = 0$; $f_4^*(15) = 0 + 60 + 7 = 67$
- At $x_1 = 1$; $f_4^*(15) = 7 + 48 + 6 = 61$
- At $x_1 = 2$; $f_4^*(15) = 14 + 36 + 4 = 54$
- At $x_1 = 3$; $f_4^*(15) = 21 + 24 + 3 = 48$
- At $x_1 = 4$; $f_4^*(15) = 28 + 12 + 1 = 41$
- At $x_1 = 5$; $f_4^*(15) = 35 + 0 + 0 = 35$

$Z = 67$; at the optimum values of:

- $x_1^* = 0 \rightarrow s_3 = 15$
- $x_2^* = 0 \rightarrow s_2 = 15$
- $x_3^* = 7 \rightarrow s_1 = 1$
- $x_4^* = 1$

$\boldsymbol{x_2^* = 0}$

$\boldsymbol{x_3^* = \left\lfloor \frac{s_2}{2} \right\rfloor}$

$\boldsymbol{x_4^* = s_1}$

The Solution to the original Problem is: $y_1^* = 0$; $y_2^* = 0$; $y_3^* = 1$; $y_4^* = 7$ & $Z = 67$

DP: Linear Programming

Let us solve a linear programming using Dynamic Programming.

$Max.\ \ 6x_1 + 5x_2$

Subject to:

$$x_1 + x_2 \leq 5$$
$$3x_1 + 2x_2 \leq 12$$
$$x_j \geq 0$$

Here; we have two resources & Hence we have two state variables. We call them u & v respectively

Stage: Each Variable

State: Resource available for allocation (u & v)

Decision Variable: Values of x_j

Criterion of effectiveness: Maximize 'z'

DP: Linear Programming

$Max.\ \ 6x_1 + 5x_2$

Subject to:

$$x_1 + x_2 \le 5$$
$$3x_1 + 2x_2 \le 12$$
$$x_j \ge 0$$

n = 1 (One more stage to go)

$f_1(u_1, v_1, x_2) = 5x_2$

$f_1^*(u_1, v_1) = Maximize\ \ 5x_2$

Subject to: $0 \le x_2 \le u$; $0 \le 2x_2 \le v$

We have

$x_2^* = minimum\ (u_1, \frac{v_1}{2})$ &

$f_1^*(u_1, v_1) = 5\ minimum\ (u_1, \frac{v_1}{2})$

n = 2 (Two more stages to go)

$f_2(5,12, x_1) = 6x_1 + f_1^*(5 - x_1, 12 - 3x_1)$

$f_1^*(5, 12) = Maximize\ \ 6x_1 + 5\ minimum\ (5 - x_1, \frac{12-3x_1}{2})$

Subject to: $0 \le x_1 \le 5$; $0 \le 3x_1 \le 12$

$5 - x_1 = \frac{12-3x_1}{2}$; at $x_1 = 2$

$f_1^*(5, 12) = Maximize\ \ 6x_1 + 5(5 - x_1)\ \ 0 \le x_1 \le 2$

$f_1^*(5, 12) = Maximize\ \ 6x_1 + 5(\frac{12-3x_1}{2})\ \ 2 \le x_1 \le 4$

- At $x_1 = 0$; Z=25
- **At $x_1 = 2$; Z=27**
- At $x_1 = 4$; Z=24

The Optimum Solution is

$$x_1^* = 2\ \&\ x_2^* = minimum\left(u_1, \frac{v_1}{2}\right) = minimum\left(5 - x_1, \frac{12 - 3x_1}{2}\right) = minimum\left(3, \frac{6}{2}\right) = 3$$

QUESTIONS