

EM 605

Elements of Operations Research

Dynamic Programming



Topics

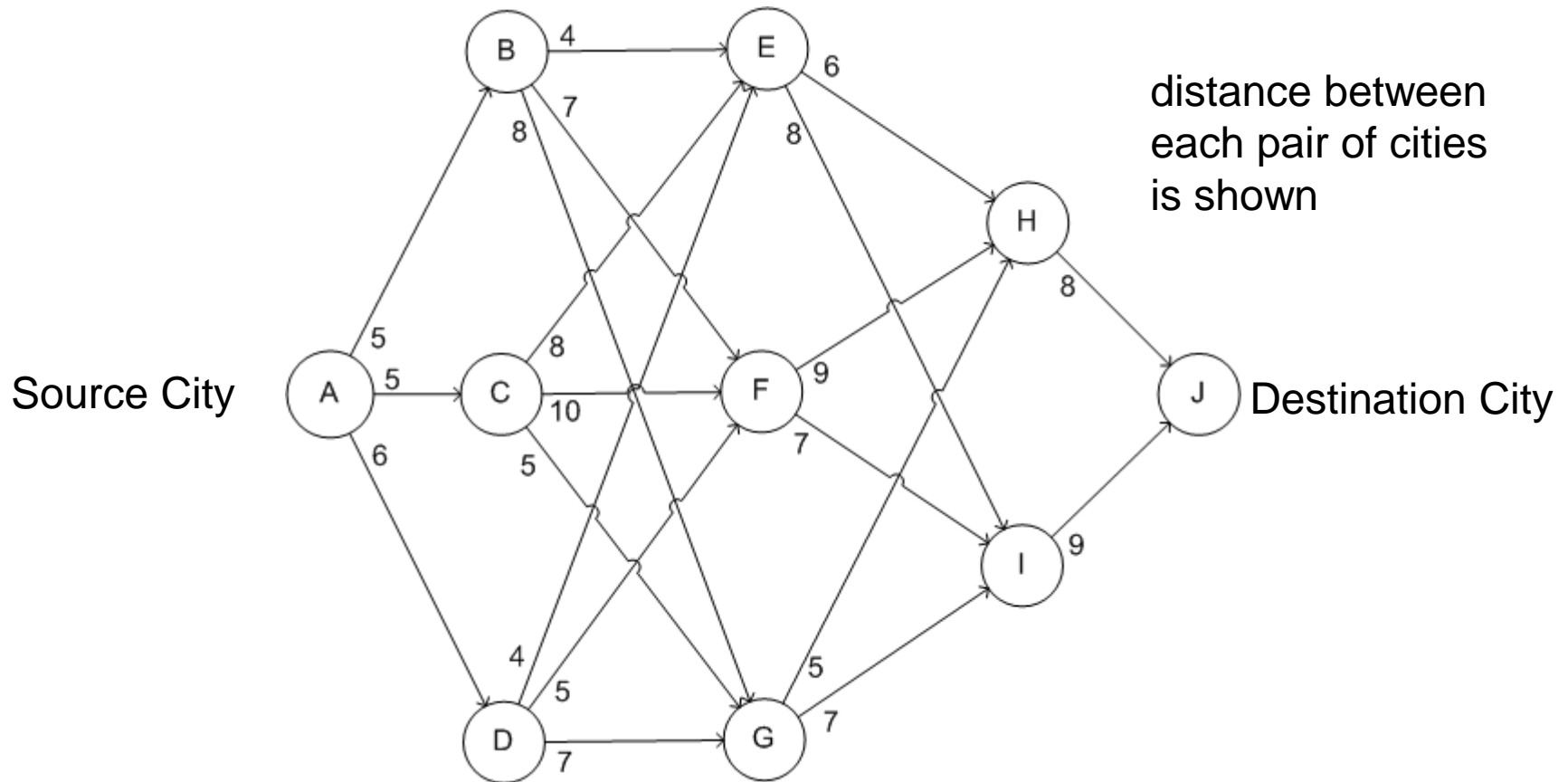
- Dynamic Programming
 - ▶ Characteristics of Dynamic Programming
 - ▶ Deterministic Dynamic Programming

Dynamic Programming

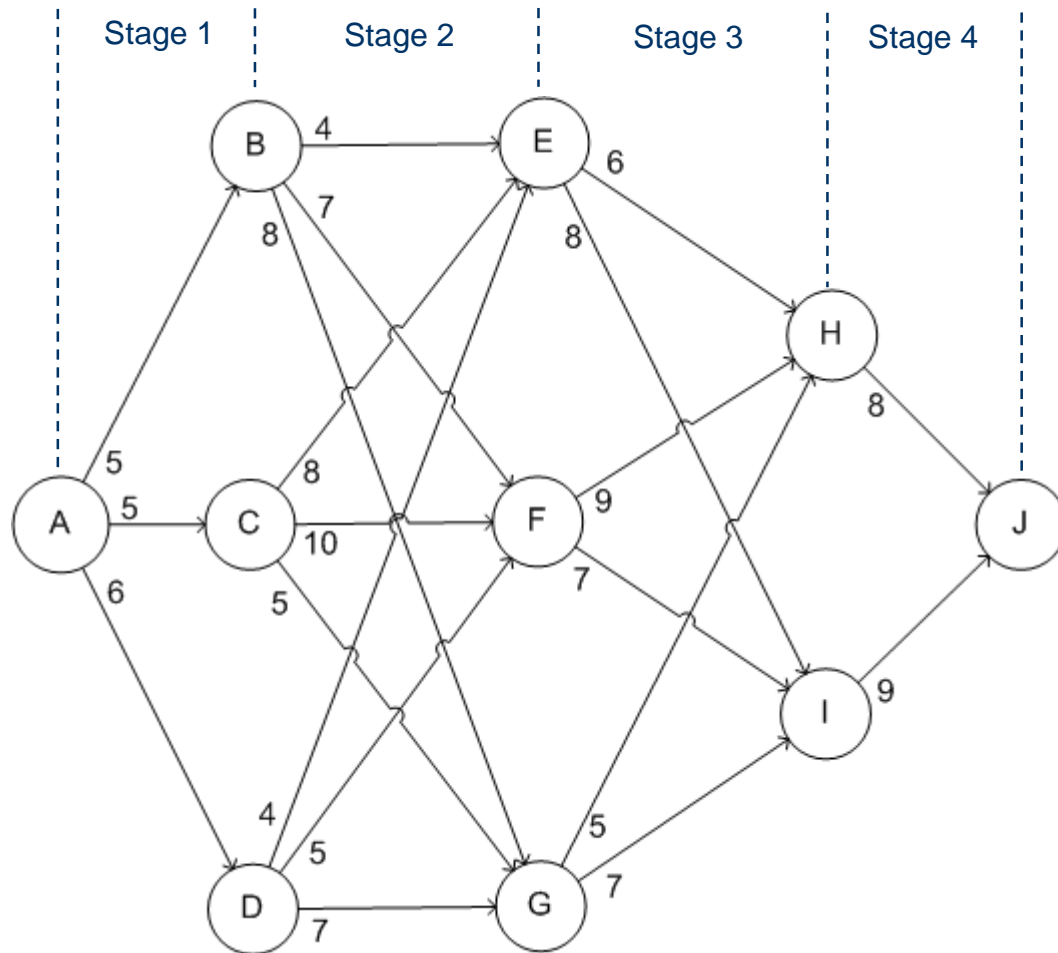
- Mathematical technique for making a sequence of inter-related decisions
- Procedure to determine the optimum combination of decisions
- It is a general approach to problem solving – there is no standard mathematical formulation
- Illustrative examples are used to discuss dynamic programming applications

Stagecoach Problem

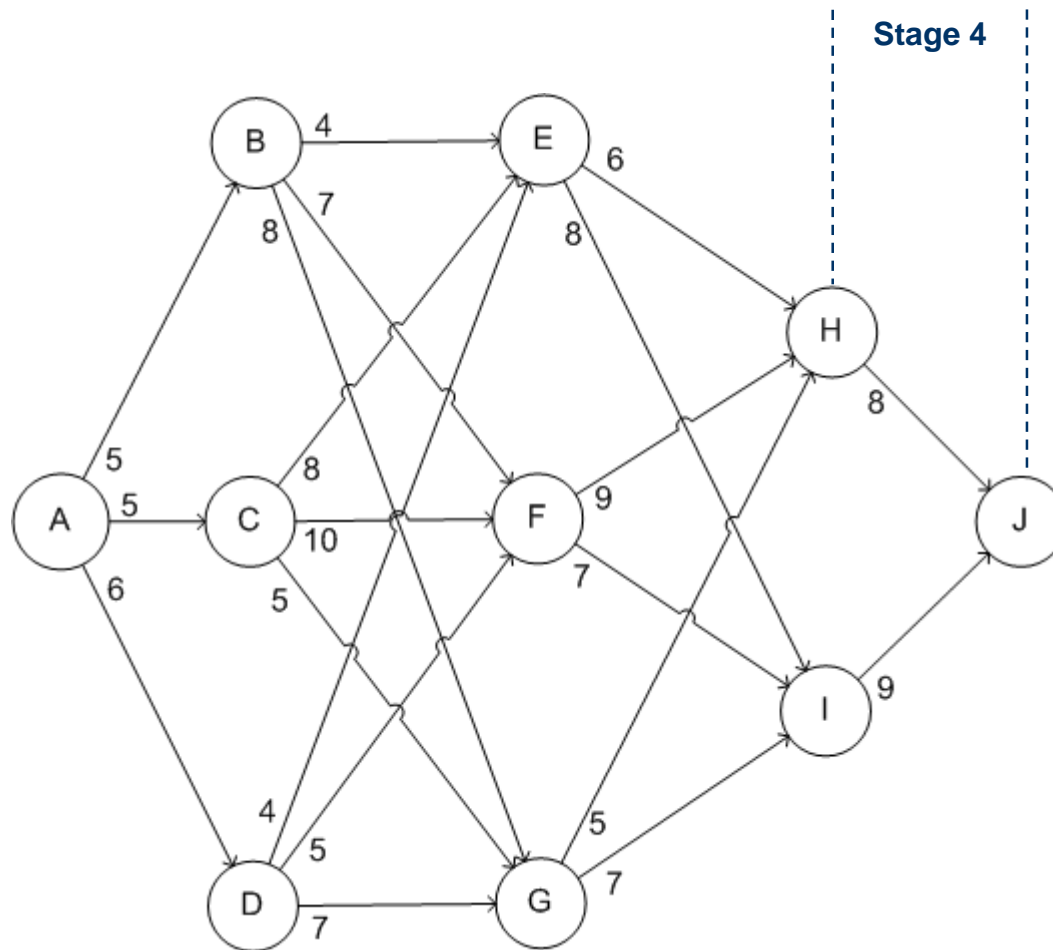
Problem: Find the shortest path from Source to Destination



Stagecoach Problem



We'll start at Stage 4



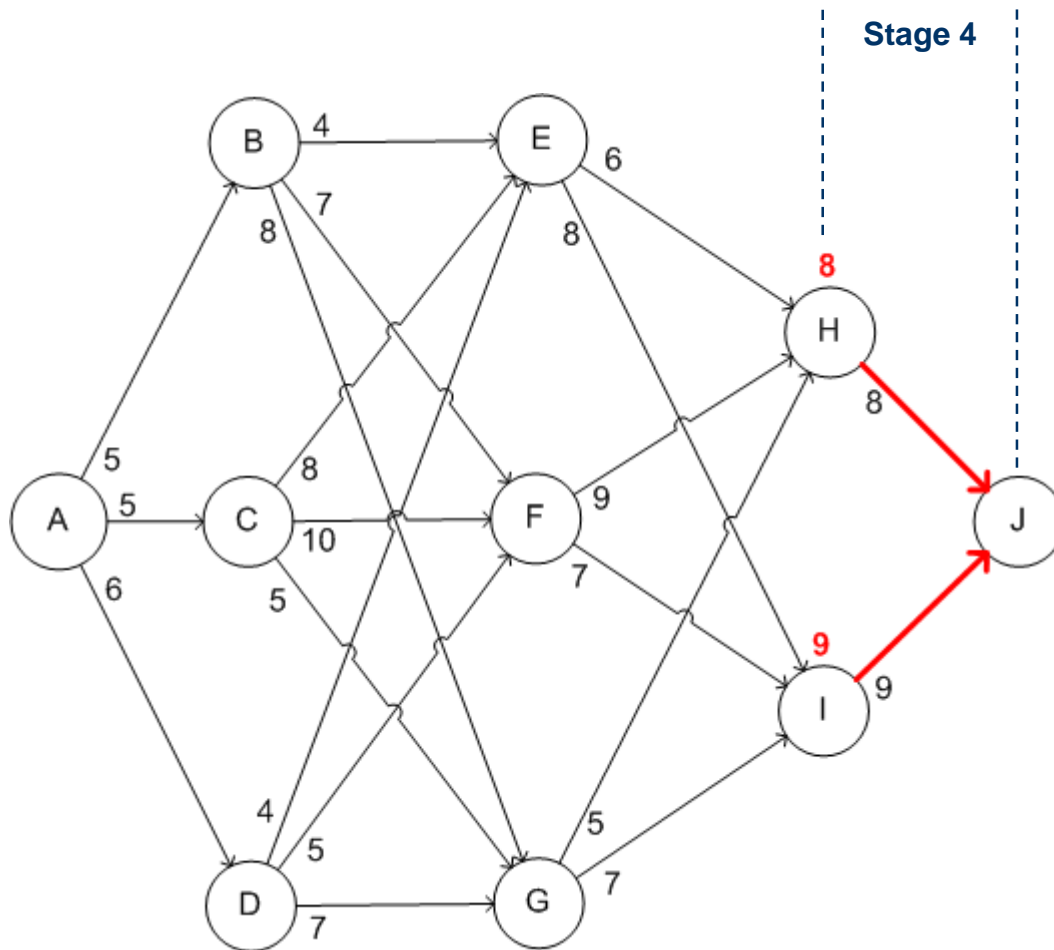
H to J

H-J 8

I to J

I-J 9

Stage 4



The best route

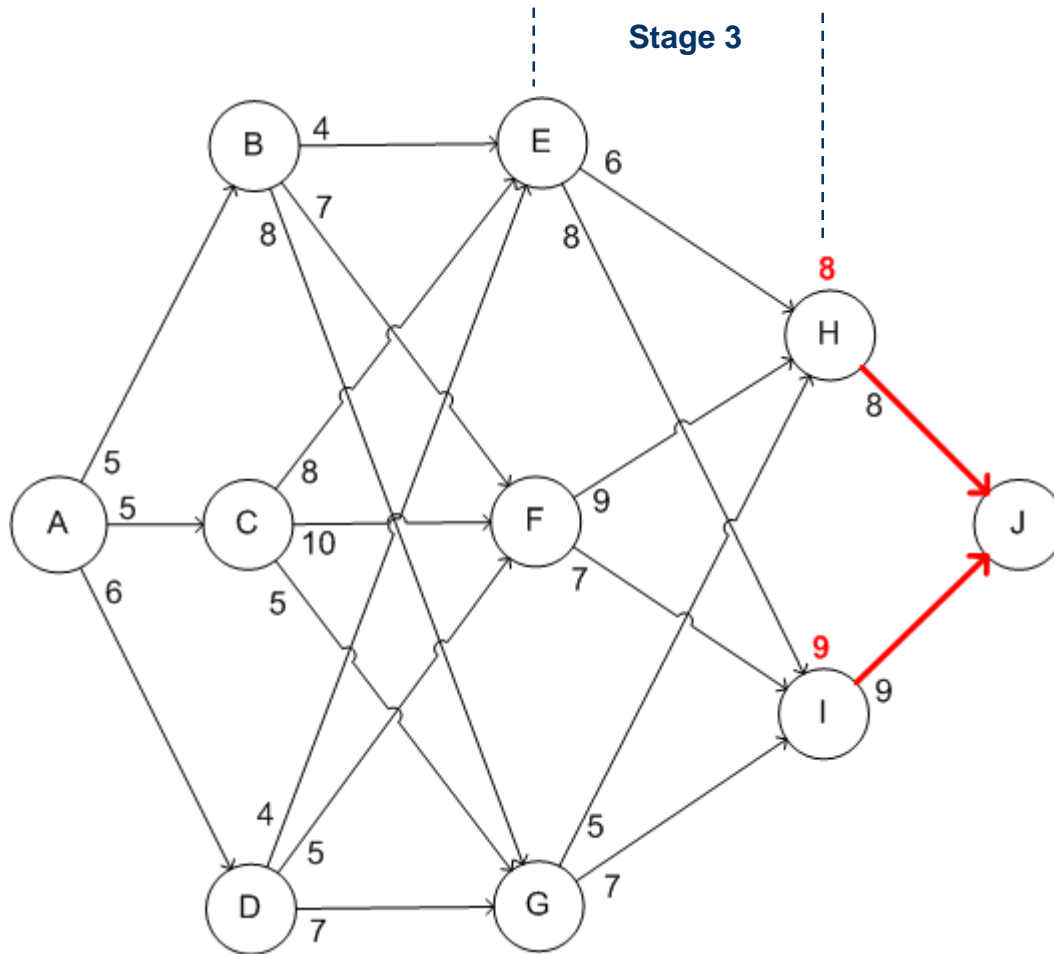
H to J

H-J 8

I to J

I-J 9

Stage 3



E to J

E-H-J $6+8=14$

E-I-J $8+9=17$

F to J

F-H-J $9+8=17$

F-I-J $7+9=16$

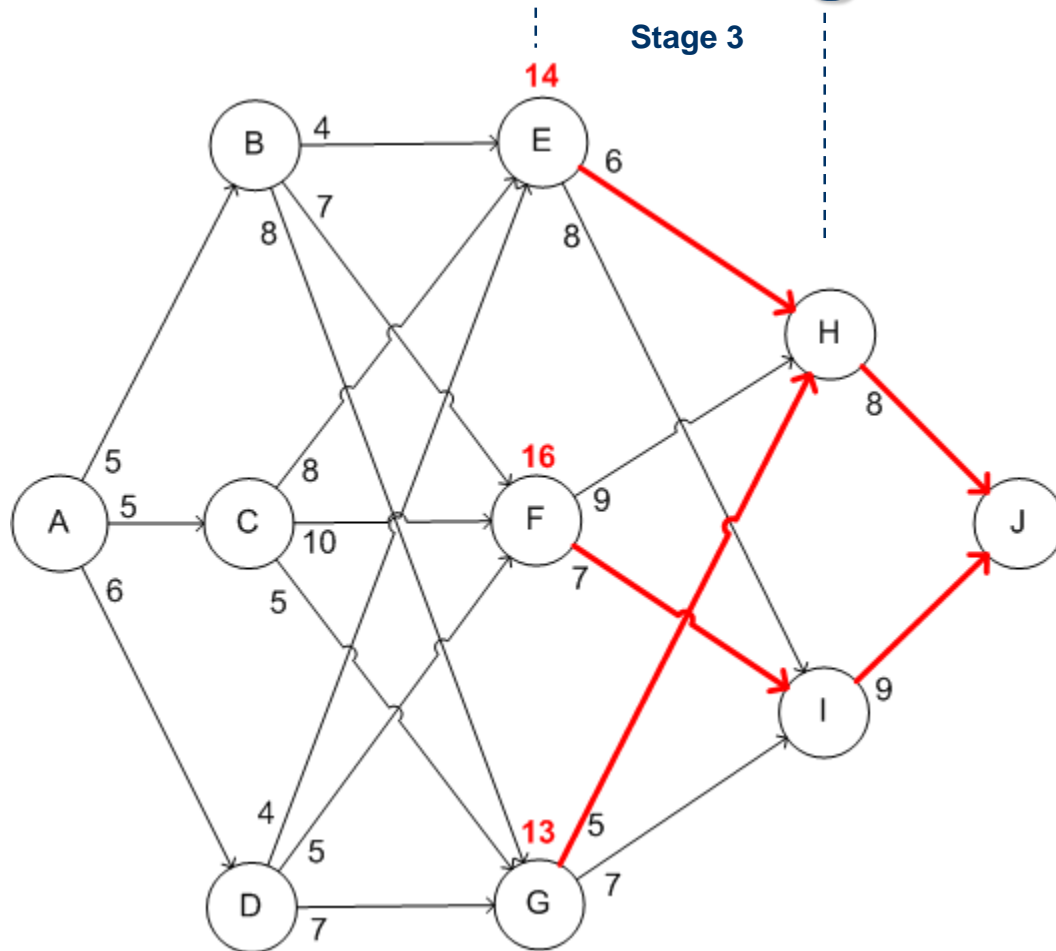
G to J

G-H-J $5+8=13$

G-I-J $7+9=16$

Stage 3

Stage 3



The best route

E to J

E-H-J $6+8=14$

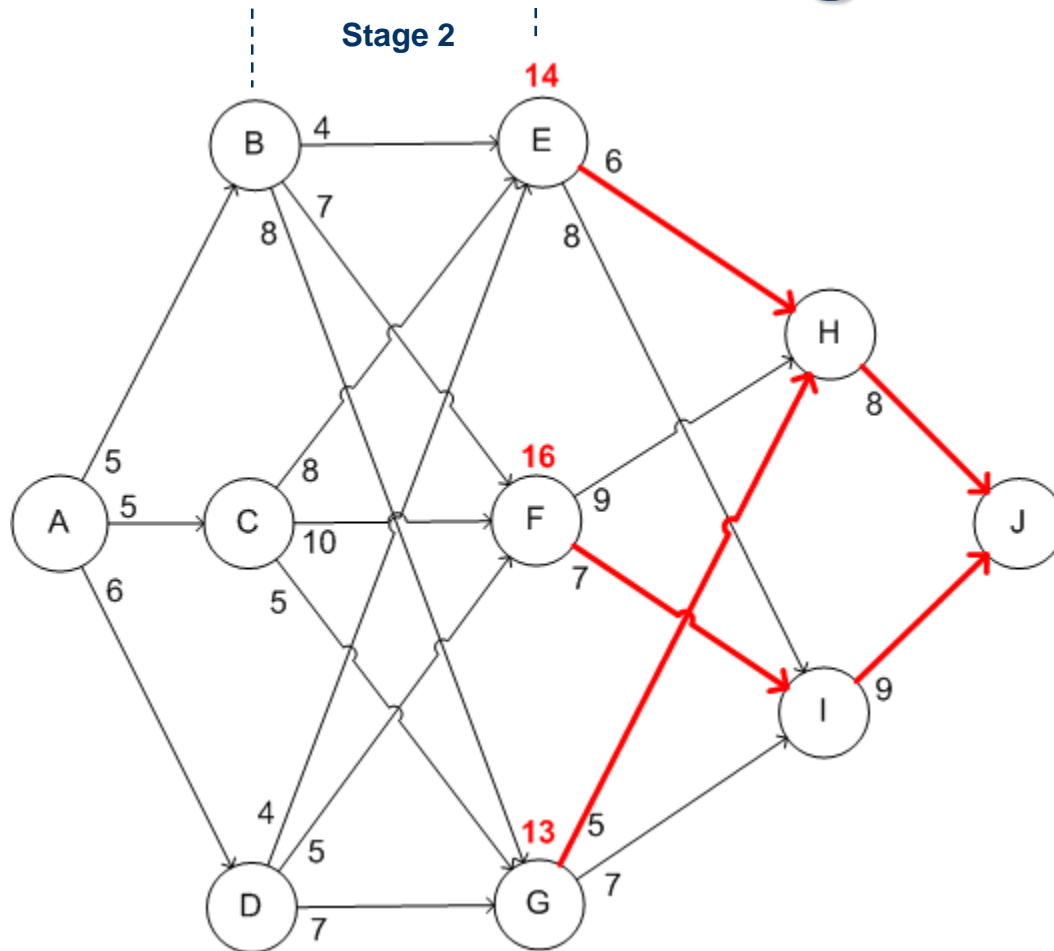
F to J

F-I-J $7+9=16$

G to J

G-H-J $5+8=13$

Stage 2



B to J

B-E-H-J $4 + 14 = 18$

B-F-I-J $7 + 16 = 23$

B-G-H-J $8 + 13 = 21$

C to J

C-E-H-J $8 + 14 = 22$

C-F-I-J $10 + 16 = 26$

C-G-H-J $5 + 13 = 18$

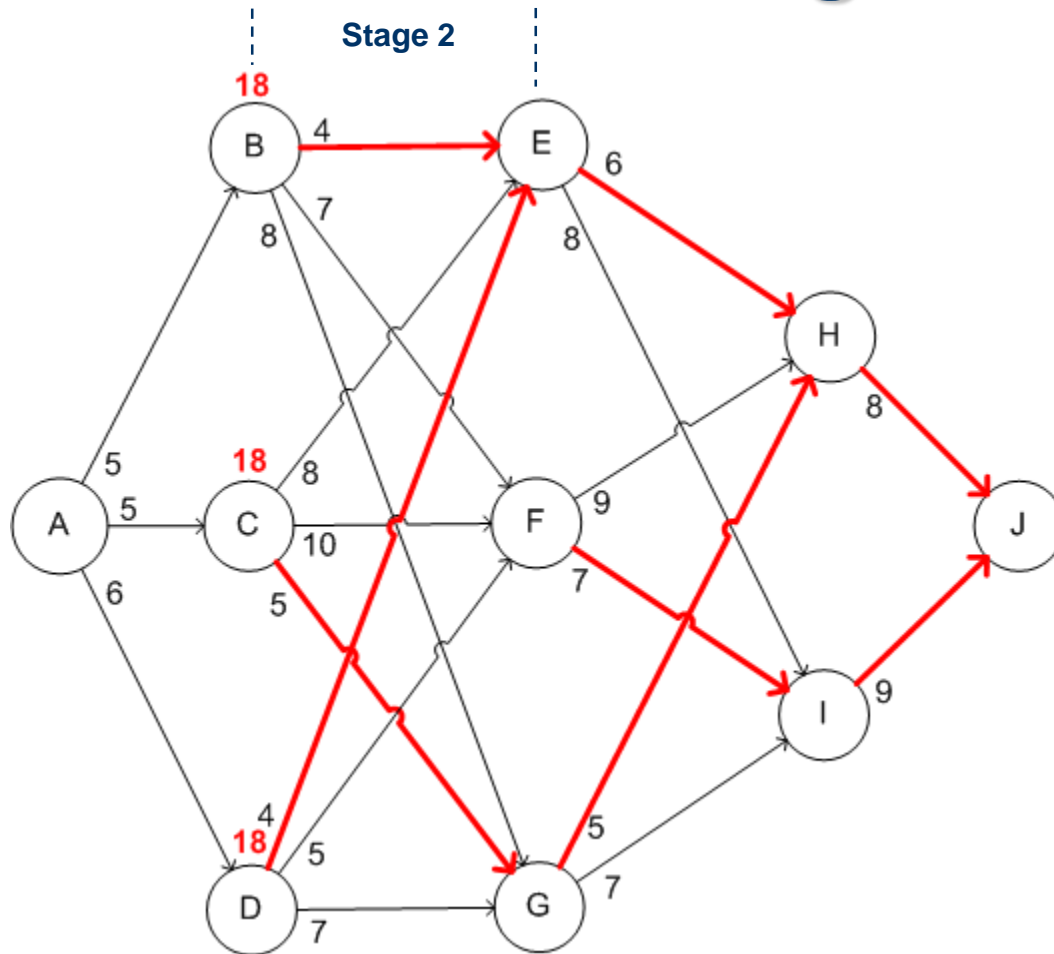
D to J

D-E-H-J $4 + 14 = 18$

D-F-I-J $5 + 16 = 21$

D-G-H-J $7 + 13 = 20$

Stage 2



The best route

B to J

B-E-H-J $4+14=18$

C to J

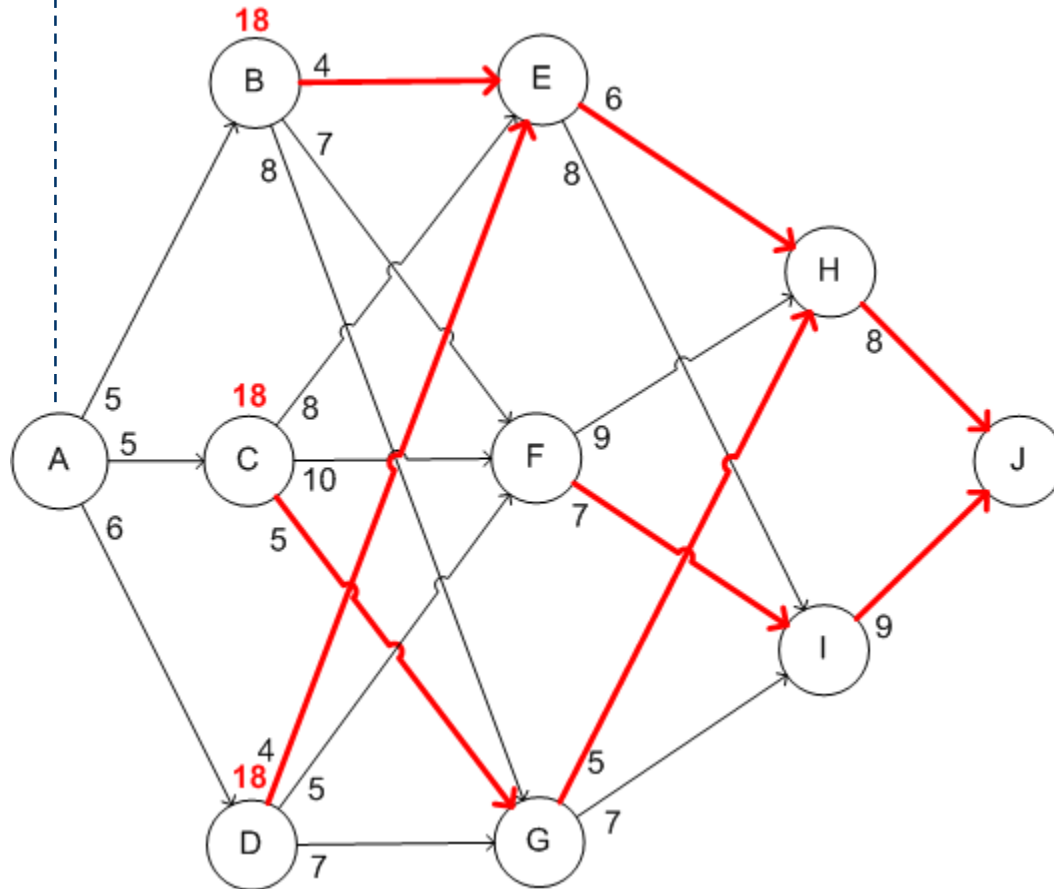
C-G-H-J $5+13=18$

D to J

D-E-H-J $4+14=18$

Stage 1

Stage 1



A to J

A-B-E-H-J

$$5 + 18 = 23$$

A-C-G-H-J

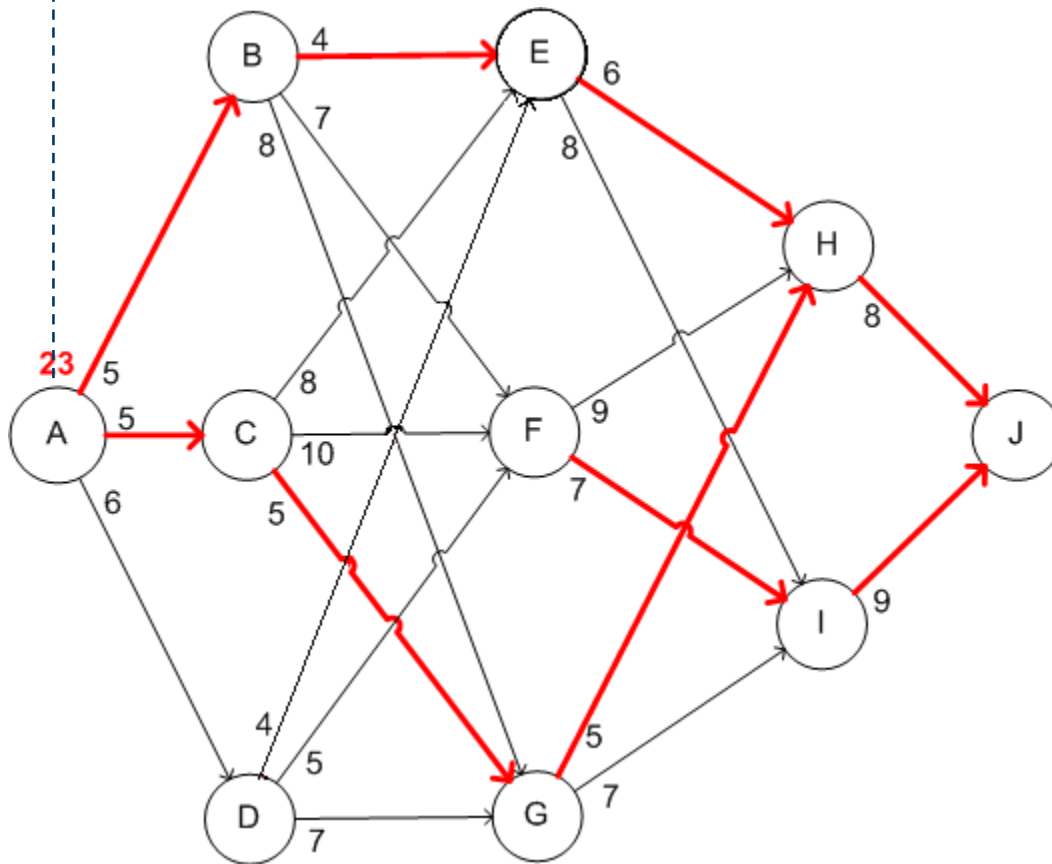
$$5 + 18 = 23$$

A-D-E-H-J

$$6 + 18 = 24$$

Stage 1

Stage 1



The best route(s)

A to J

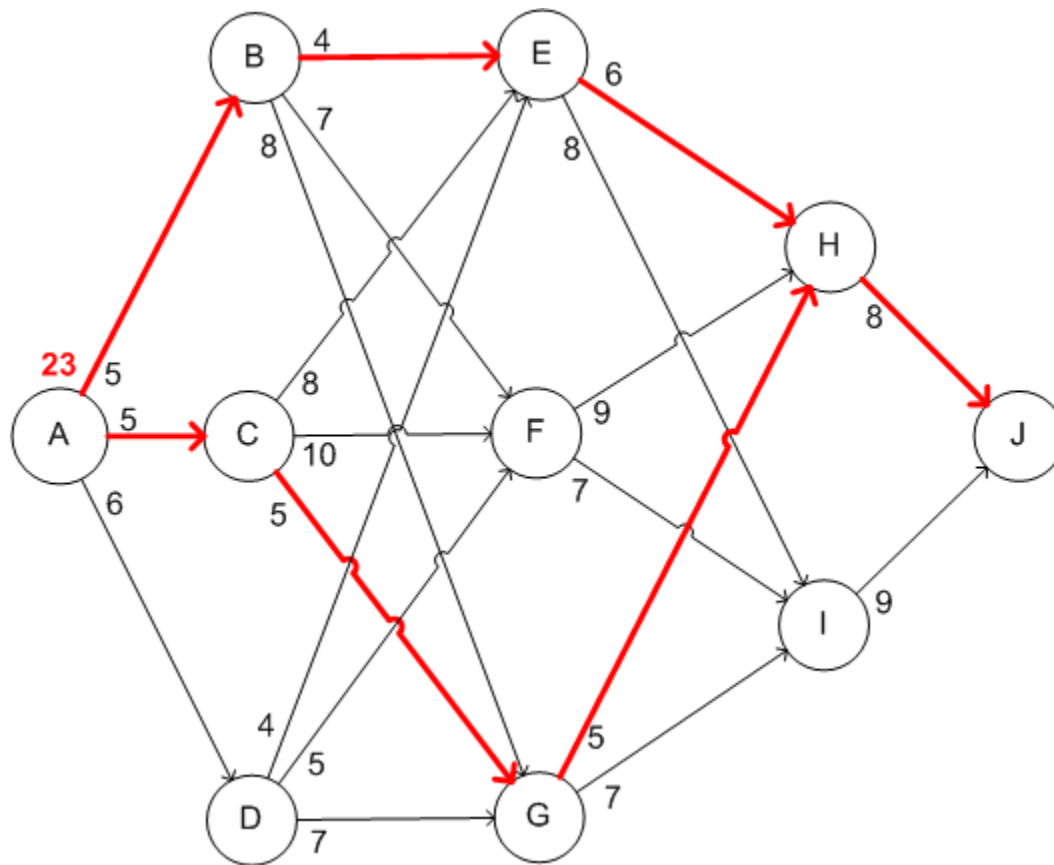
A-B-E-H-J

5+18=23

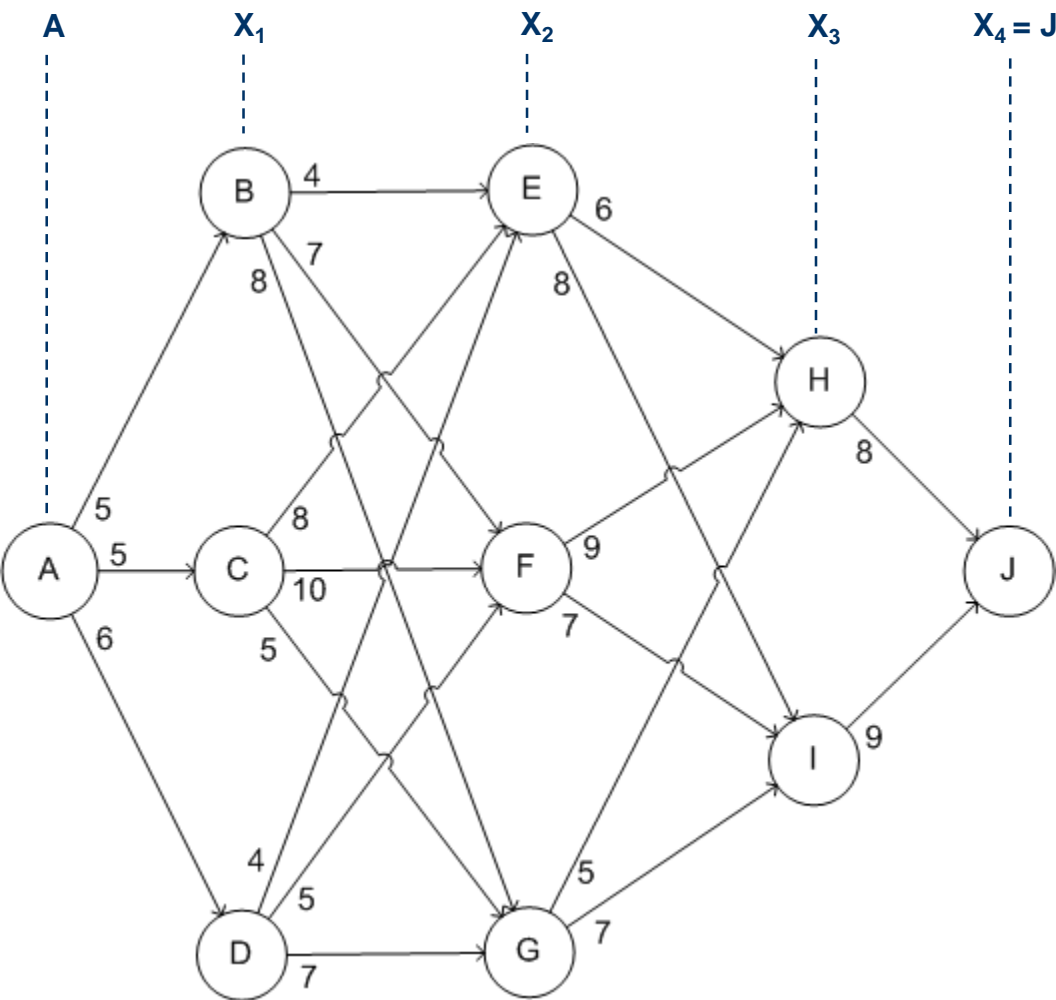
A-C-G-H-J

5+18=23

Solution to “Stagecoach Problem”



Mathematical Formulation



At any stage n:

$f_n(s_n, x_n)$ total distance for remaining stages

s_n current state

x_n selection made

Let's set $f_n^*(s_n)$ to be the “best” selection

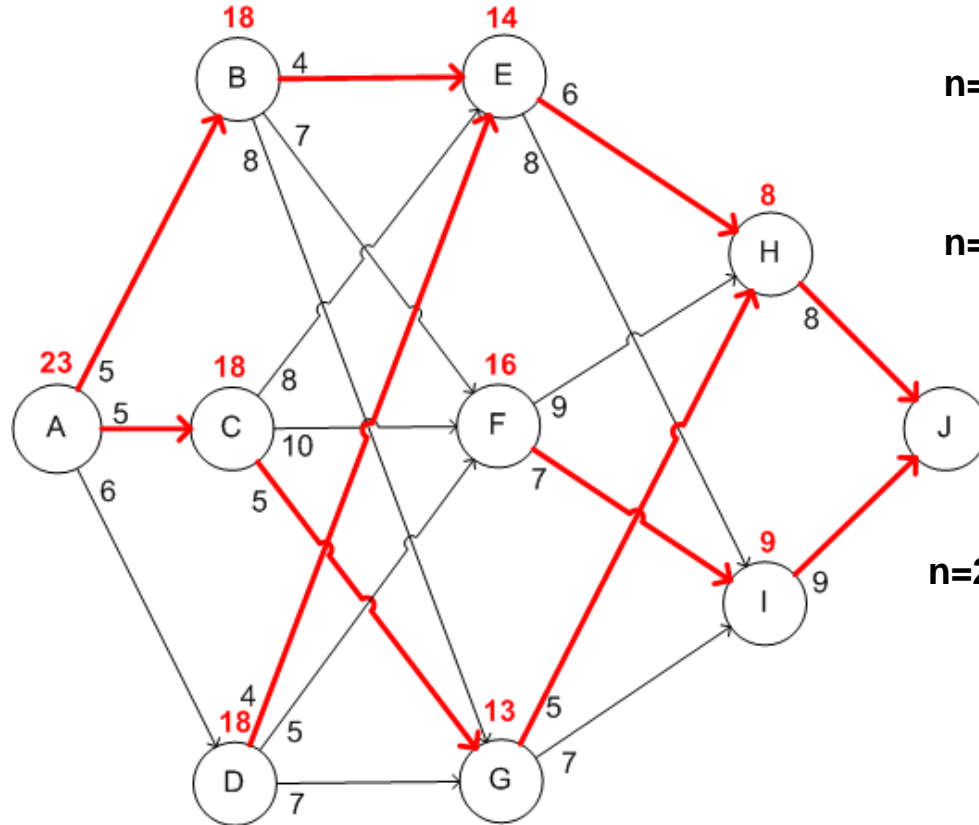
$$f_n^*(s_n) = \min f_n(s_n, x_n) = f_n(s_n, x_n^*)$$

x_n^* best selection

$f_n(s_n, x_n)$ = immediate distance (in stage n) +
minimum future distance
(stages n+1 onwards)

$$f_n(s_n, x_n) = s_n x_n + f_{n+1}^*(x_n)$$

Solution Procedure



$$f_n(s_n, x_n) = s_n x_n + f_{n+1}^*(x_n)$$

$$f_n^*(s_n) = \min(f_n(s_n, x_n))$$

$n=4$

s_4	$f_4^*(s_4)$	x_4^*
H	8	J
I	9	J

$n=3$

s_3	$f_3(s_3, x_3) = s_3 x_3 + f_4^*(x_3)$		$f_3^*(s_3)$	x_3^*
	H	I		
E	6+8=14	8+9=17	14	H
F	9+8=17	7+9=16	16	I
G	5+8=13	7+9=16	13	H

$n=2$

s_2	$f_2(s_2, x_2) = s_2 x_2 + f_3^*(x_2)$			$f_2^*(s_2)$	x_2^*
	E	F	G		
B	4+14=18	7+16=23	8+13=21	18	E
C	8+14=22	10+16=26	5+13=18	18	G
D	4+14=18	5+16=21	7+13=20	18	E

$n=1$

s_1	$f_1(s_1, x_1) = s_1 x_1 + f_2^*(x_1)$			$f_1^*(s_1)$	x_1^*
	B	C	D		
A	5+18=23	5+18=23	6+18=24	23	B, C

Solution Procedure

Finally, although the construction of the tables was *backwards*, we will now READ it *forwards*...

Shortest Path:

A-B-E-H-J or A-C-G-H-J

Shortest Distance: 23

n=1

s ₁	f ₁ (s ₁ ,x ₁)=ds ₁ x ₁ + f [*] ₂ (x ₁)			f [*] ₁ (s ₁)	x [*] ₁
	B	C	D		
A	5+18=23	5+18=23	6+18=24	23	B,C

n=2

s ₂	f ₂ (s ₂ ,x ₂)=ds ₂ x ₂ + f [*] ₃ (x ₂)			f [*] ₂ (s ₂)	x [*] ₂
	E	F	G		
B	4+14=18	7+16=23	8+13=21	18	E
C	8+14=22	10+16=26	5+13=18	18	G
D	4+14=18	5+16=21	7+13=20	18	E

n=3

s ₃	f ₃ (s ₃ ,x ₃)=ds ₃ x ₃ +f [*] ₄ (x ₃)		f [*] ₃ (s ₃)	x [*] ₃
	H	I		
E	6+8=14	8+9=17	14	H
F	9+8=17	7+9=16	16	I
G	5+8=13	7+9=16	13	H

n=4

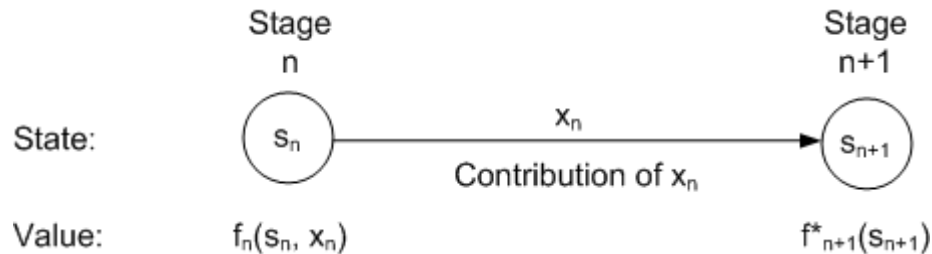
s ₄	f [*] ₄ (s ₄)	x [*] ₄
H	8	J
I	9	J

Characteristics of Dynamic Programming Problems

1. The problem can be divided into **stages**, with a **decision** required at each stage
2. Each stage has a number of states (or possible conditions) associated with the beginning of the stage
3. The effect of the decision is to transform the current state to a state associated with the beginning of the next stage
4. The solution procedure is designed to find an optimal decision for the overall problem
5. Given the current state, the optimal decision for the remaining stages is **independent of the decision adopted in the previous stages** - this is the ***principal of optimality*** for dynamic programming
6. The solution procedure begins by finding the optimal decision for the last stage
7. A recursive relationship that identifies the optimal decision for stage n , given the optimal decision for stage $n+1$ is available
8. When we use this recursive relationship, the solution procedure starts at the end and moves backward stage by stage – each time finding the optimal decision for that stage – until it finds the optimal decision at the initial stage

Deterministic Dynamic Programming

- State at the next stage is completely determined by the state and decision at the current stage
- Basic structure for deterministic dynamic programming:



- Deterministic dynamic programming can be categorized by the:
 - ▶ form of the objective function (minimize sum, maximize sum, minimize product, ...)
 - ▶ nature of the set of states (discrete variables, continuous variables)

Example of Resource Allocation

- Distributing Scientists to Research Teams
- We're going to add two scientists to existing teams on a project - we'd like to allocate them so that we reduce the probability of failure for the entire project

New Scientists Assigned to Team	Probability of Failure		
	Team 1	Team 2	Team 3
0	0.40	0.60	0.80
1	0.20	0.40	0.50
2	0.15	0.20	0.30

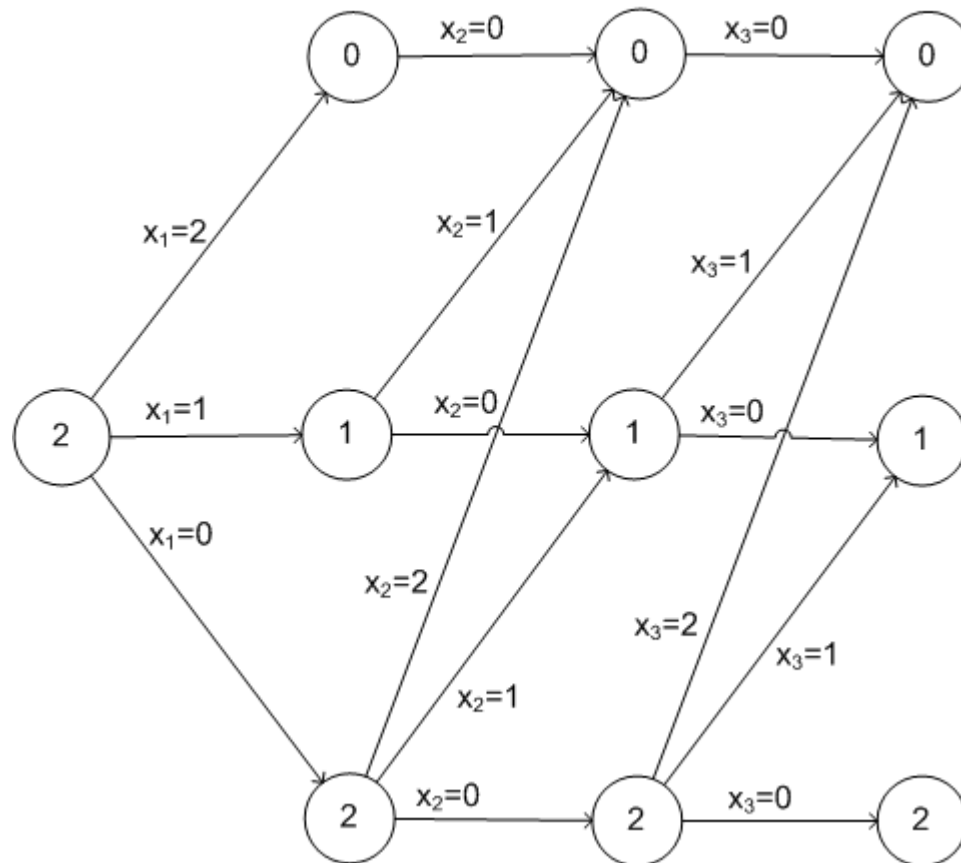
- Let x_1 , x_2 , x_3 be the number of new scientists to be distributed to each team i , where $i = 1, 2, 3$
- Objective is to minimize $p_1(x_1)p_2(x_2)p_3(x_3)$ where $p_i(x_i)$ is the probability of failure of each team

Example of Resource Allocation

Assigned to Team 1

Assigned to Team 2

Assigned to Team 3



Recursive Function

$$f_n(s_n, x_n) = p_n x_n [f_{n+1}^*(s_n - x_n)]$$

$$f_n^*(s_n) = \min(f_n(s_n, x_n))$$

We'll consider the stages to be our TEAMS

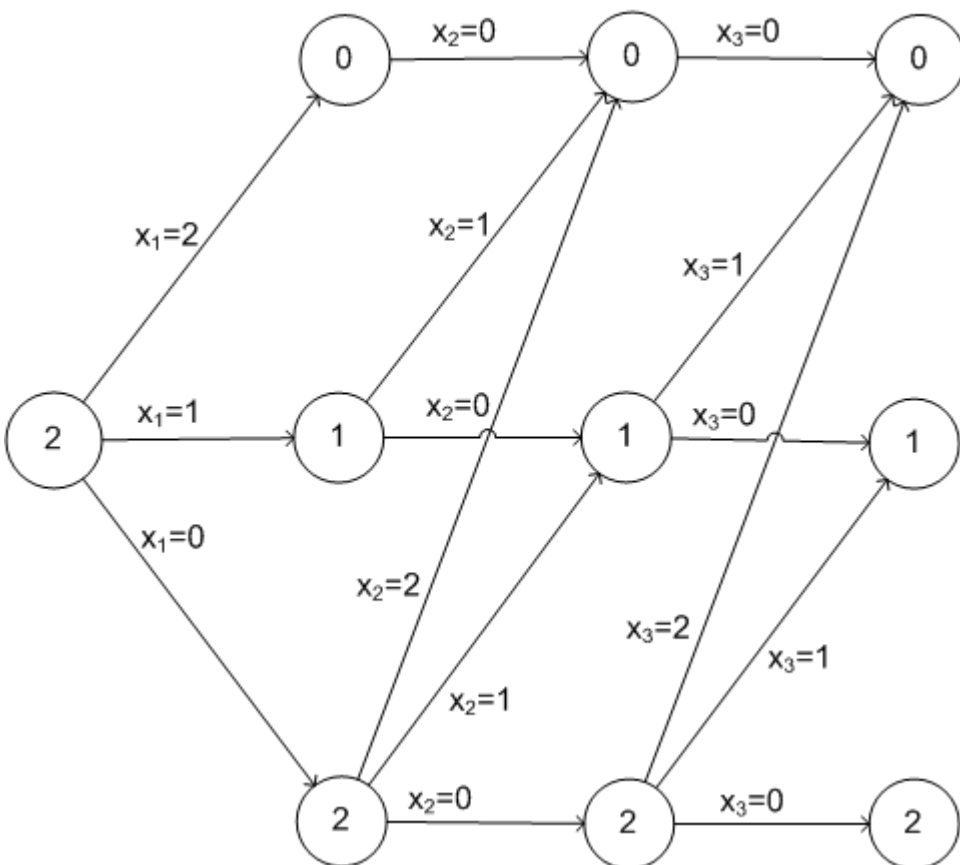
Constraint

$$x_1 + x_2 + x_3 \leq 2$$

Some textbooks consider the constraint to be an equality function.

I prefer to think of it as an inequality, and my thinking is reflected in the figure on this page and the tables in the next page

Assigned to Team 1 Assigned to Team 2 Assigned to Team 3



$$f_n(s_n, x_n) = p_n x_n f_{n+1}^*(s_n - x_n)$$

New Scientists	Probability of Failure		
	Team 1	Team 2	Team 3
0	0.40	0.60	0.80
1	0.20	0.40	0.50
2	0.15	0.20	0.30

n=3

s ₃	f ₃ (s ₃ , x ₃) = p ₃ x ₃			f* ₃ (s ₃)	x* ₃
	0	1	2		
0	0.80	-	-	0.80	0
1	0.80	0.50	-	0.50	1
2	0.80	0.50	0.30	0.30	2

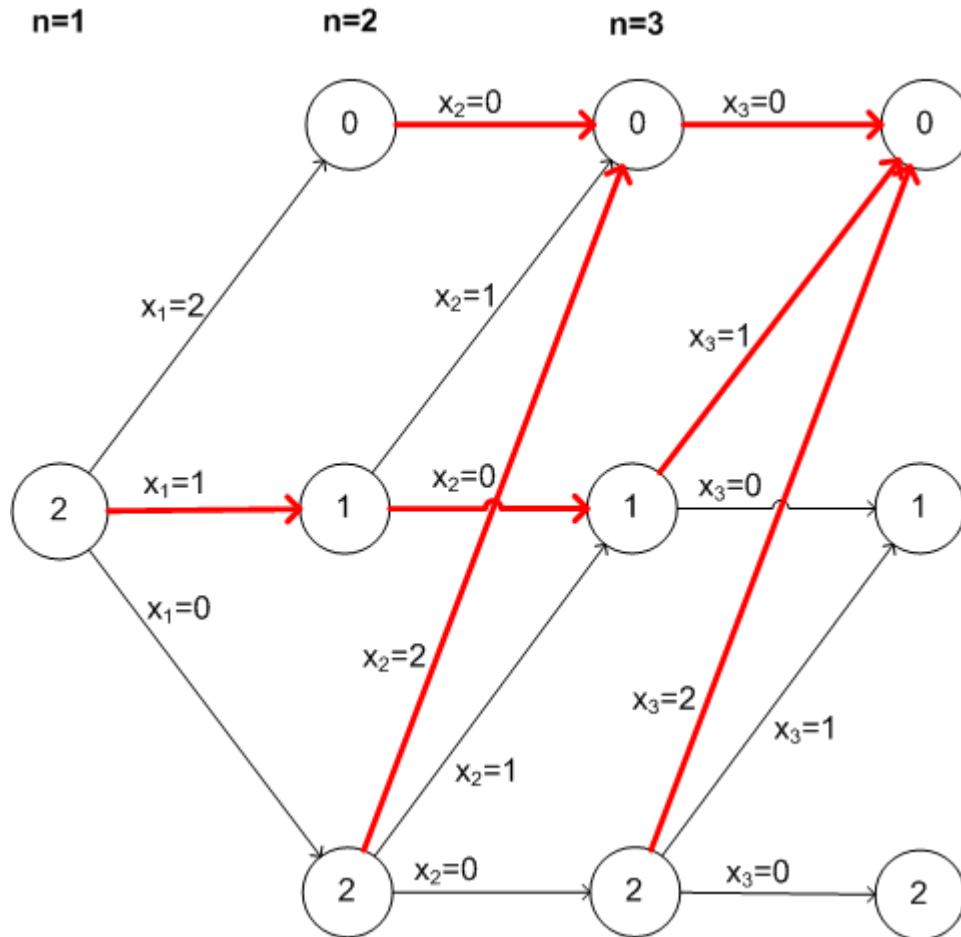
n=2

s ₂	f ₂ (s ₂ , x ₂) = p ₂ x ₂ [f* ₃ (s ₂ -x ₂)]			f* ₂ (s ₂)	x* ₂
	0	1	2		
0	0.6*0.8 =0.48	-	-	0.48	0
1	0.6*0.5 =0.3	0.4*0.8 =0.32	-	0.30	0
2	0.6*0.3 =0.18	0.4*0.5 =0.20	0.2*0.8 =0.16	0.16	2

n=1

s ₁	f ₁ (s ₁ , x ₁) = p ₁ x ₁ [f* ₂ (s ₁ -x ₁)]			f* ₁ (s ₁)	x* ₁
	0	1	2		
2	0.4*0.16 =0.064	0.2*0.3 =0.06	0.15*0.48 =0.072	0.06	1

Example of Resource Allocation



Solution: $x_1=1, x_2=0, x_3=1$

n=1

s_1	$f_1(s_1, x_1) = p_1 x_1 f_2^*(s_1 - x_1)$			$f_1^*(s_1)$	x_1^*
	0	1	2		
2	$0.4 \cdot 0.16 = 0.064$	$0.2 \cdot 0.3 = 0.06$	$0.15 \cdot 0.48 = 0.072$	0.06	1

n=2

s_2	$f_2(s_2, x_2) = p_2 x_2 f_3^*(s_2 - x_2)$			$f_2^*(s_2)$	x_2^*
	0	1	2		
0	$0.6 \cdot 0.8 = 0.48$	-	-	0.48	0
1	$0.6 \cdot 0.5 = 0.3$	$0.4 \cdot 0.8 = 0.32$	-	0.30	0
2	$0.6 \cdot 0.3 = 0.18$	$0.4 \cdot 0.5 = 0.20$	$0.2 \cdot 0.8 = 0.16$	0.16	2

n=3

s_3	$f_3(s_3, x_3) = p_3 x_3$			$f_3^*(s_3)$	x_3^*
	0	1	2		
0	0.80	-	-	0.80	0
1	0.80	0.50	-	0.50	1
2	0.80	0.50	0.30	0.30	2

Example of Continuous Variable

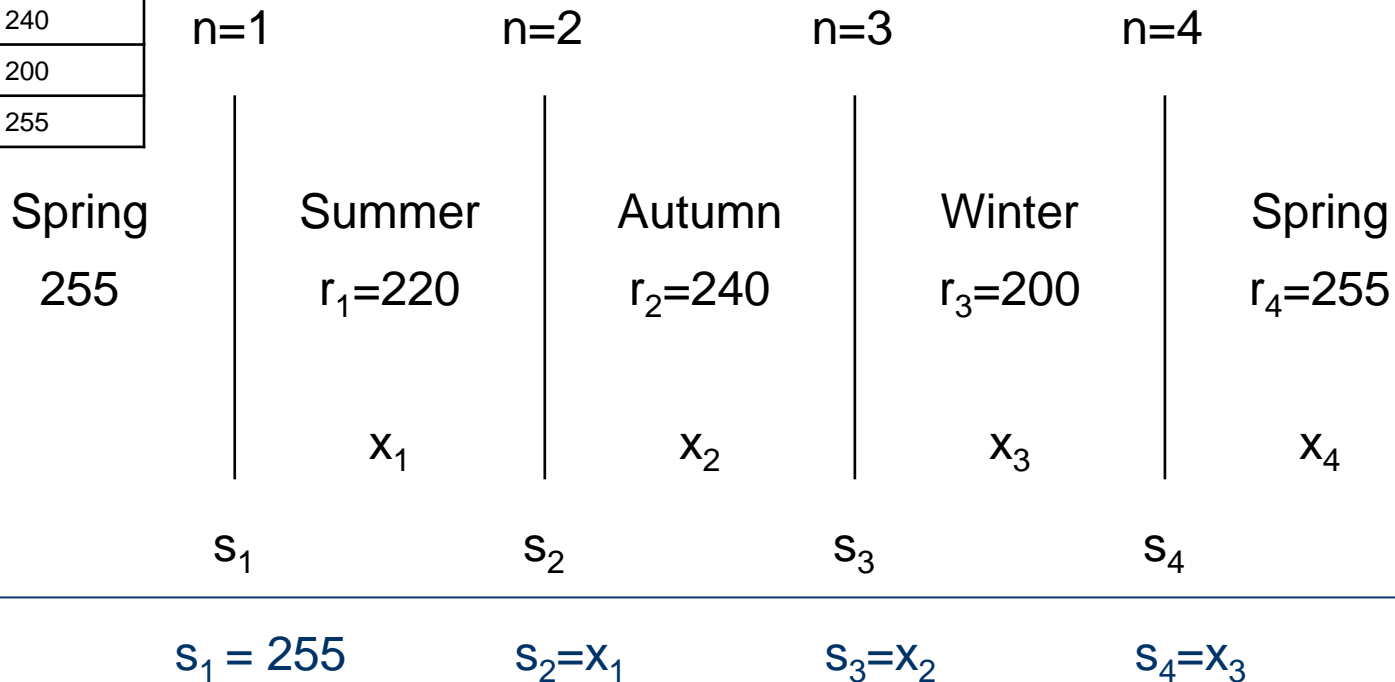
Season	Minimum Requirement
Spring	255
Summer	220
Autumn	240
Winter	200
Spring	255

- Scheduling Employment Levels
- Minimum Employment requirements
- Cost of extra people = \$2000/person per season
- Cost of hiring / firing = $\$200 * \text{the square of the difference in \# of people between 2 consecutive seasons}$

Season	Minimum Requirement
Spring	255
Summer	220
Autumn	240
Winter	200
Spring	255

Cost of extra people = \$2000 per person per season

Cost of hiring / firing = \$200 * the square of the difference in number of people between two consecutive seasons



$$220 \leq x_1 \leq 255 \quad 240 \leq x_2 \leq 255 \quad 200 \leq x_3 \leq 255 \quad x_4 = 255$$

$$\text{Cost for stage } n = 200 (x_n - x_{n-1})^2 + 2000 (x_n - r_n)$$

Continuous Variable

Stage 4 (n=4)

$$\text{Cost for stage 4} = 200 (x_4 - x_3)^2 + 2000 (x_4 - r_4)$$

$$x_4 = r_4 = 255$$

$$s_4 = x_3; \quad 200 \leq s_4 \leq 255$$

$$f_4(s_4, 255) = 200 (255 - s_4)^2$$

$$f_4^*(s_4) = \min (f_4(s_4, 255))$$

s_4	$f_4^*(s_4)$	x_4^*
$200 \leq s_4 \leq 255$	$200(255 - s_4)^2$	255

Continuous Variable

Stage 3 (n=3)

$$r_3 = 200; \quad s_3 = x_2; \quad 240 \leq s_3 \leq 255$$

$$f_3(s_3, x_3) = 200 (x_3 - s_3)^2 + 2000 (x_3 - r_3) + f_4^*(x_3)$$

$$f_3(s_3, x_3) = 200 (x_3 - s_3)^2 + 2000 (x_3 - 200) + 200 (255 - x_3)^2$$

$$f_3^*(s_3) = \min (f_3(s_3, x_3))$$

$$\partial / \partial x_3 (f_3(s_3, x_3)) = 400 (2x_3 - s_3 - 250) = 0$$

$$x_3^* = (s_3 + 250) / 2$$

s_3	$f_3^*(s_3)$	x_3^*
$240 \leq s_3 \leq 255$	$50(250 - s_3)^2 + 50(260 - s_3)^2 + 1000(s_3 - 150)$	$(s_3 + 250) / 2$

Continuous Variable

Stage 2 (n=2)

$$r_2 = 240; s_2 = x_1; 220 \leq s_2 \leq 255$$

$$f_2(s_2, x_2) = 200(x_2 - s_2)^2 + 2000(x_2 - r_2) + f_3^*(x_2)$$

$$f_2(s_2, x_2) = 200(x_2 - s_2)^2 + 2000(x_2 - 240) + 50(250 - x_2)^2 + 50(260 - x_2)^2 + 1000(x_2 - 150)$$

$$f_2^*(s_2) = \min(f_2(s_2, x_2))$$

Setting $\partial / \partial x_2 (f_2(s_2, x_2)) = 0$, we get $x_2 = (2s_2 + 240) / 3$

But this is valid only if $240 \leq x_2 \leq 255$; but $220 \leq s_2 \leq 255$

Therefore, if $220 \leq s_2 \leq 240$, we set $x_2 = 240$

s_2	$f_2^*(s_2)$	x_2^*
$220 \leq s_2 \leq 240$	$200(240 - s_2)^2 + 115,000$	240
$240 \leq s_2 \leq 255$	$200/9[(240-s_2)^2 + (255-s_2)^2 + (270-s_2)^2] + 2000(s_2 - 195)$	$(2s_2 + 240)/3$

Continuous Variable

Stage 1 (n=1)

$$r_1 = 220; \quad s_1 = 255$$

$$f_1(s_1, x_1) = 200 (x_1 - s_1)^2 + 2000 (x_1 - r_1) + f_2^*(x_1)$$

We have different expressions for $f_2^*(x_1)$

for the ranges $220 \leq x_1 \leq 240$ and $240 \leq x_1 \leq 255$

$$f_1^*(s_1) = \min (f_1(s_1, x_1))$$

Setting $\partial / \partial x_2 (f_1(s_1, x_1)) = 0$, we get $x_1 = 247.5$ from Range 2

Range 1 gives a limiting value $x_1 = 240$

s_2	$f_2^*(s_2)$	x_2^*
255	185,000	247.5

Continuous Variable

Solution

	n=1	n=2	n=3	n=4
Spring 255	Summer $r_1=220$	Autumn $r_2=240$	Winter $r_3=200$	Spring $r_4=255$
	$x^*_1=247.5$	$x^*_2=245$	$x^*_3=247.5$	$x^*_4=255$
	s_1	s_2	s_3	s_4

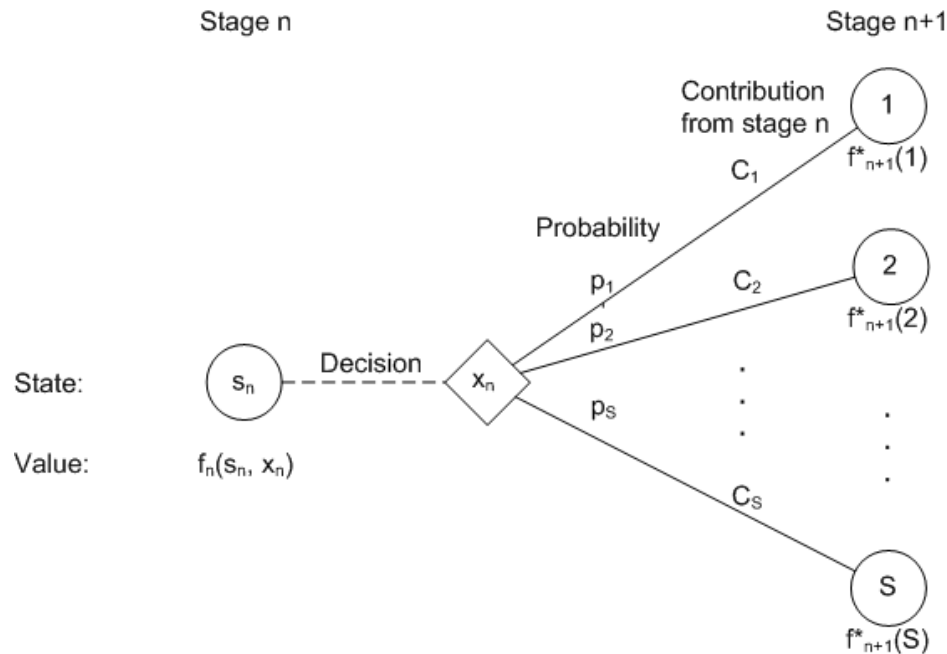
Total cost = \$185,000

Probabilistic Dynamic Programming

- The state at the next stage is not completely determined by the state and decision at the current stage
- There is a probability distribution for what the next state will be
- The relationship between $f_n(s_n, x_n)$ and $f_{n+1}^*(s_{n+1})$ is more complicated than that for deterministic dynamic programming

Probabilistic Dynamic Programming

- Basic structure for deterministic dynamic programming:



- When the above figure includes all possible states and decisions at all stages, it is called a decision tree



Probabilistic Dynamic Programming

- The Hit-n-Miss Manufacturing Company has received an order to supply one item of a particular style
- The customer has stringent quality requirements, which typically results in the manufacturer having to produce more than one item to get one that is acceptable
- The number of extra items produced in a production run is called the reject allowance
- Including a reject allowance is a common practice, especially in custom orders

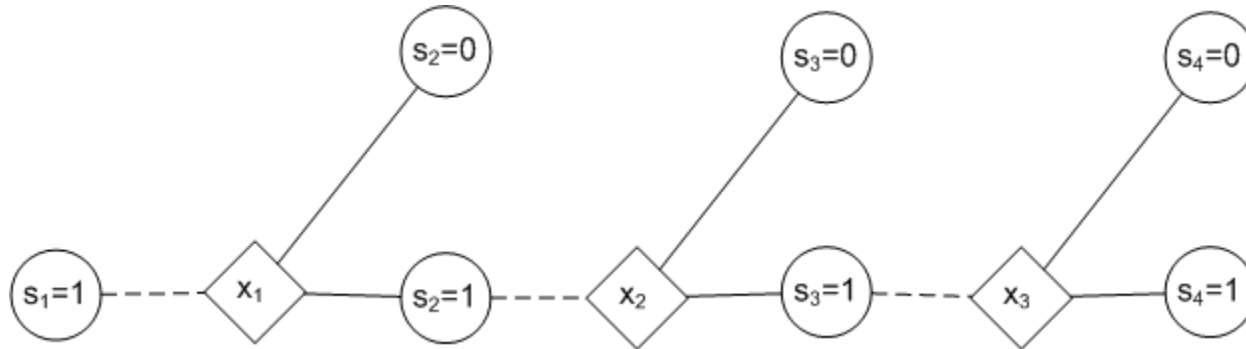
Probabilistic Dynamic Programming

- Probability of $\frac{1}{2}$ each for an acceptable or defective item
 - The number of acceptable items produced in a lot of size L will have a binomial distribution
- Up to three production runs possible; \$300 setup cost per run; \$100 marginal production cost per item
- Excess items are worthless
- If there are no acceptable items produced at the end of the third run, there is a loss/penalty cost of \$1600 to the manufacturer

Probabilistic Dynamic Programming

- What's our objective here?
 - To determine the lot size L (1 + reject allowance) for the required production run(s) that minimizes total expected cost for the manufacturer
- Formulation:
 - Stage n = production run ($n = 1, 2, 3$)
 - X_n = lot size for stage n
 - State S_n = number of acceptable items still needed (1 or 0) at the beginning of stage n

Probabilistic Dynamic Programming

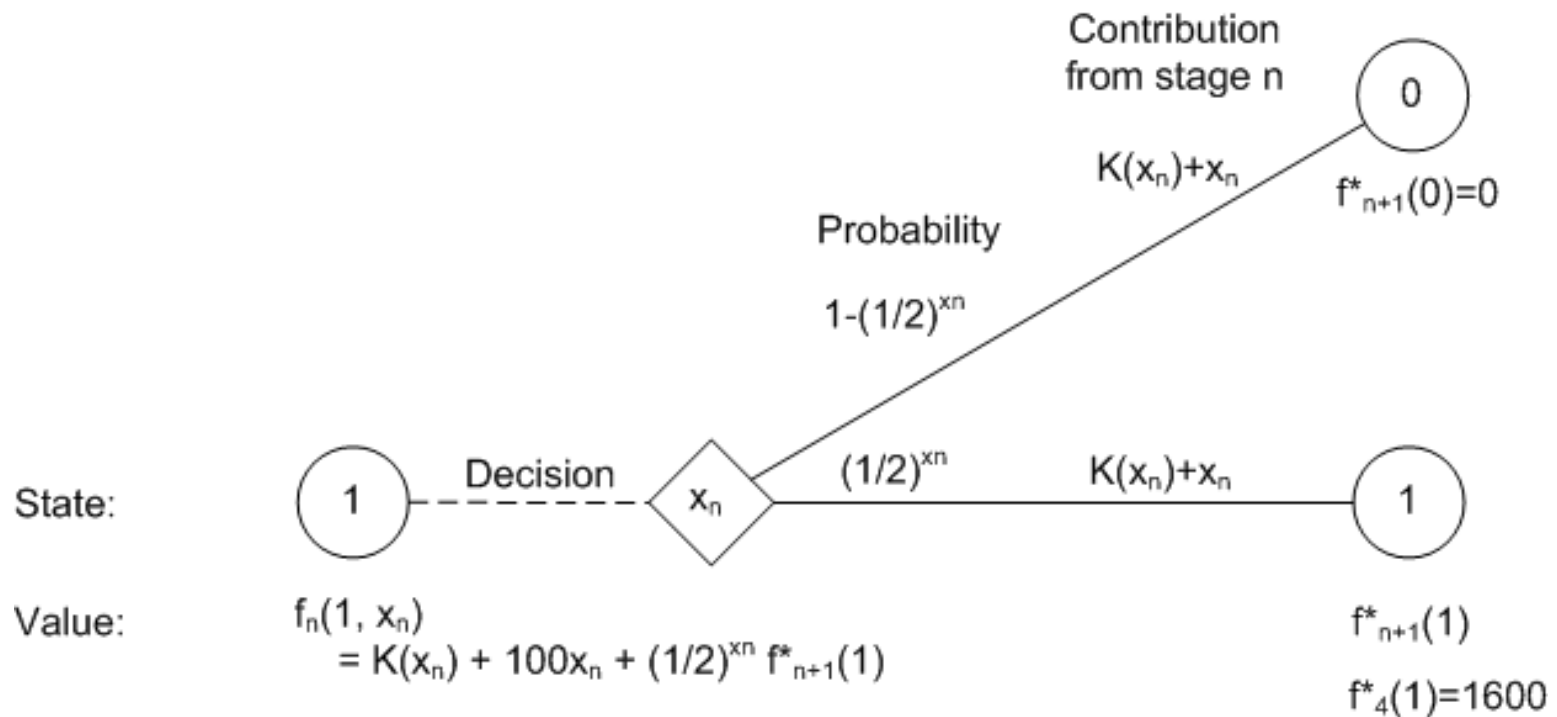


s_n : number of acceptable items required at the start of stage n

Set up cost: $K(x_n) = 0$ if $x_n = 0$ and 300 if $x_n > 0$

Once an acceptable item is produced, there is no need for any further production

Probabilistic Dynamic Programming



Probabilistic Dynamic Programming

n=3

s_3	$f_3(1, x_3) = K(x_3) + 100x_3 + 1600(1/2)^{x_3}$						$f^*_3(s_3)$	x^*_3
	0	1	2	3	4	5		
0	0						0	0
1	1600	1200	900	800	800	850	800	3,4

n=2

s_2	$f_2(1, x_2) = K(x_2) + 100x_2 + (1/2)^{x_2} f^*_3(1)$					$f^*_2(s_2)$	x^*_2
	0	1	2	3	4		
0	0					0	0
1	800	800	700	700	750	700	2,3

n=1

s_1	$f_1(1, x_1) = K(x_1) + 100x_1 + (1/2)^{x_1} f^*_2(1)$					$f^*_1(s_1)$	x^*_1
	0	1	2	3	4		
1	700	750	675	687.5	743.75	675	2

Total expected cost: \$675

Optimal decision: produce 2 items in stage 1;
if no item is acceptable, produce 2 or 3 items in stage 2;
if no item is acceptable, produce 3 or 4 items in stage 3