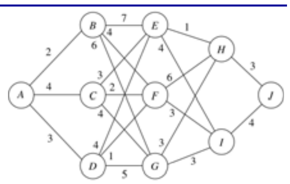
DYNAMIC PROGRAMMING

Dynamic Programming

- It is a useful mathematical technique for making a sequence of interrelated decisions.
- Systematic procedure for determining the optimal combination of decisions.
- There is no standard mathematical formulation of "the" Dynamic Programming problem.
- Knowing when to apply dynamic programming depends largely on experience with its general structure.

Prototype example



❖ **Stagecoach problem**

- Fortune seeker wants to go from Missouri (A) to California (J) in the mid-19th century.
- Journey has 4 stages.
- Cost is the life insurance of a specific route; lowest cost is equivalent to safest trip.

Costs

- Cost c_{ij} of going from state i to state j is:

	B	C	D
A	2	4	3

	E	F	G
B	7	4	6
C	3	2	4
D	4	1	5

	H	I
E	1	4
F	6	3
G	3	3

	J
H	3
I	4

➢ **Problem**: which route minimizes the total cost of the policy?

Solving the problem

- Note that greedy approach does not work.
  - Solution $A \to B \to F \to I \to J$ has total cost of 13.
  - However, e.g. $A \to D \to F$ is cheaper than $A \to B \to F$.
- Other possibility: trial-and-error. Too much effort even for this simple problem.
- **Dynamic programming** is much more efficient than exhaustive enumeration, especially for large problems.
- Starts from the last stage of the problem, and enlarges it one stage at a time.

Formulation

- Decision variables x_n (n = 1, 2, 3, 4) are the immediate destination of stage n.
  - Route is $A \to x_1 \to x_2 \to x_3 \to x_4$, where $x_4 = J$.
- Total cost of the best overall *policy* for the remaining *stages* is $f_n(s, x_n)$
  - Actual state is s, ready to start stage n, selecting x_n as the immediate destination.
- x_n^* minimizes $f_n(s, x_n)$ and $f_n^*(s, x_n)$ is the minimum value of $f_n(s, x_n)$:

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

Formulation

- where

  $f_n(s, x_n)$ = immediate cost (stage n) + minimum future cost (stages n+1 onward)

  $= c_{sx_n} + f^*_{n+1}(x_n)$

- Value of c_{sx_n} given by c_{ij} where $i = s$ (current state) and $j = x_n$ (immediate destination).

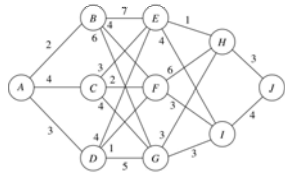
➢ **Objective**: find $f_1^*(A)$ and the corresponding route.

- Dynamic programming finds successively $f_4^*(s)$, $f_3^*(s)$, $f_2^*(s)$ and finally $f_1^*(A)$.

Solution procedure

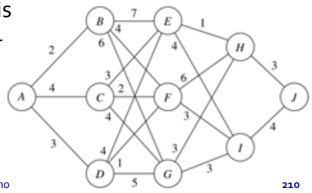
- When $n = 4$, the route is determined by its current state s (H or I) and its final destination J.
- Since $f_4^*(s) = f_4^*(s, J) = c_{sJ}$, the solution for $n = 4$:

s	$f_4^*(s)$	x_4^*
H	3	J
I	4	J



Stage $n = 3$

- Needs a few calculations. If fortune seeker is in state F, he can go to either H or I with costs $c_{F,H} = 6$ or $c_{F,I} = 3$.
- Choosing H, the minimum additional cost is $f_4^*(H) = 3$. Total cost is 6 + 3 = 9.
- Choosing I, the total cost is 3 + 4 = 7. This is smaller, and it is the optimal choice for state F.



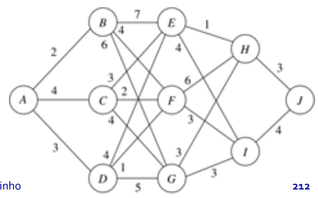
Stage $n = 3$

- Similar calculations can be made for the two possible states $s = E$ and $s = G$, resulting in the table for $n = 3$:

s \ x_3	$f_3(s, x_3) = c_{sx_3} + f_4^*(x_3)$ H	I	$f_3^*(s)$	x_3^*
E	4	8	4	H
F	9	7	7	I
G	6	7	6	H

Stage $n = 2$

- In this case, $f_2^*(s, x_2) = c_{sx_2} + f_3^*(x_2)$.
- Example for node C:
  - $x_2 = E$: $f_2^*(C, E) = c_{C,E} + f_3^*(E) = 3 + 4 = 7$ ← optimal
  - $x_2 = F$: $f_2^*(C, F) = c_{C,F} + f_3^*(F) = 2 + 7 = 9$.
  - $x_2 = G$: $f_2^*(C, G) = c_{C,G} + f_3^*(G) = 4 + 6 = 10$.



Stage $n = 2$

- Similar calculations can be made for the two possible states $s = B$ and $s = D$, resulting in the table for $n = 2$:

s \ x_2	$f_2(s, x_2) = c_{sx_2} + f_3^*(x_2)$ E	F	G	$f_2^*(s)$	x_2^*
B	11	11	12	11	E or F
C	7	9	10	7	E
D	8	8	11	8	E or F

Stage $n = 1$

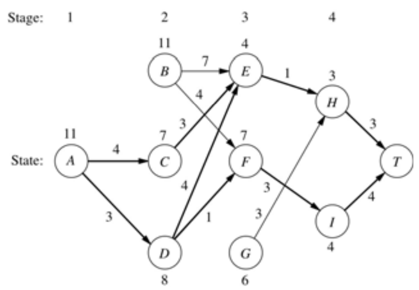
- Just one possible starting state: *A*.
  - $x_1 = B$: $f_2^*(A, B) = c_{A,B} + f_2^*(B) = 2 + 11 = 13$.
  - $x_1 = C$: $f_2^*(A, C) = c_{A,C} + f_2^*(C) = 4 + 7 = 11 \leftarrow$ optimal
  - $x_1 = D$: $f_2^*(A, D) = c_{A,D} + f_2^*(D) = 3 + 8 = 11 \leftarrow$ optimal

- Results in the table:

s \ x_1	$f_1(s, x_1) = c_{sx_1} + f_2^*(x_1)$				
	B	*C*	*D*	$f_1^*(s)$	x_1^*
A	13	11	11	11	*C* or *D*

Optimal solution

- Three optimal solutions, all with $f_1^*(A) = 11$:



Characteristics of DP

1. The problem can be divided into **stages**, with a **policy decision** required at each stage.
   - **Example**: 4 stages and life insurance policy to choose.
   - Dynamic programming problems require making a *sequence of interrelated decisions*.
2. Each stage has a number of **states** associated with the beginning of each stage.
   - **Example**: states are the possible territories where the fortune seeker could be located.
   - States are *possible conditions* in which the system might be.

Characteristics of DP

3. Policy decision *transforms the current state to a state associated with the beginning of the next stage*.
   - **Example**: fortune seeker's decision led him from his current state to the next state on his journey.
   - DP problems can be interpreted in terms of *networks*: each *node* correspond to a *state*.
   - Value assigned to each link is the *immediate contribution* to the objective function from making that policy decision.
   - In most cases, objective corresponds to finding the *shortest* or the *longest path*.

Characteristics of DP

4. The solution procedure finds an **optimal policy** for the overall problem. Finds a prescription of the optimal policy decision at each stage for *each* of the possible states.
   - **Example**: solution procedure constructed a table for each stage, *n*, that prescribed the optimal decision, x_n^*, for each possible state *s*.
   - In addition to identifying *optimal solutions*, DP provides a policy prescription of what to do under every possible circumstance (why a decision is called policy decision). This is useful for sensitivity analysis.

Characteristics of DP

5. Given the current state, an *optimal policy for the remaining stages* is *independent* of the policy decisions adopted in *previous stages*.
   - Optimal immediate decision depends only on current state and not on how it was obtained: this is the **principle of optimality** for DP.
   - **Example**: at any state, the insurance policy is independent on how the fortune seeker got there.
   - Knowledge of the current state conveys all information necessary for determining the optimal policy henceforth (*Markovian property*). Problems lacking this property are not Dynamic Programming Problems.

Characteristics of DP

6. Solution procedure begins by *finding the optimal policy for the last stage*. Solution is usually trivial.
7. A **recursive relationship** that identifies optimal policy for stage n, given optimal policy for stage n+1, is available.
   - ❖ **Example**: recursive relationship was

$$f_n^*(s) = \min_{x_n} \left\{ c_{sx_n} + f_{n+1}^*(x_n) \right\}$$

   - ➢ Recursive relationship differs somewhat among dynamic programming problems.

Characteristics of DP

7. (cont.) Notation:

N = number of stages.

n = label for current stage ($n = 1, 2, \ldots, N$).

s_n = current *state* for stage n.

x_n = decision variable for stage n.

x_n^* = optimal value of x_n (given s_n).

$f_n(s_n, x_n)$ = contribution of stages $n, n+1, \ldots, N$ to objective function if system starts in state s_n at stage n, immediate decision is x_n, and optimal decisions are made thereafter.

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

Characteristics of DP

7. (cont.) Recursive relationship:

$$f_n^*(s_n) = \max_{x_n} \{ f_n(s_n, x_n) \} \quad \text{or} \quad f_n^*(s_n) = \min_{x_n} \{ f_n(s_n, x_n) \}$$

where $f_n(s_n, x_n)$ is written in terms of s_n, x_n, $f_{n+1}^*(s_{n+1})$, and probably some measure of the immediate contribution of x_n to the objective function.

8. Using recursive relationship, solution procedure starts at the end and moves **backward** stage by stage.
   - ➢ Stops when optimal policy starting at *initial* stage is found.
   - ➢ The optimal policy for the entire problem is found.
   - ❖ **Example**: the tables for the stages show this procedure.

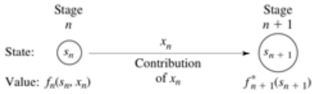
Characteristics of DP

8. (cont.) For DP problems, a table such as the following would be obtained for each stage ($n = N$, N-1, …, 1):

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

Deterministic dynamic programming

- ❑ *Deterministic problems*: the *state* at the *next stage* is *completely determined* by the *state* and *policy decision* at the *current stage*.



- ▪ **Form of the objective function:** *minimize* or *maximize* the *sum*, *product*, etc. of the contributions from the individual stages.
- ▪ **Set of states:** may be *discrete* or *continuous*, or a state *vector*. Decision variables can also be discrete or continuous.

Example: distributing medical teams

- ❑ The World Health Council has five medical teams to allocate to three underdeveloped countries.
- ❑ Measure of performance: *additional person-years of life*, i.e., *increased life expectancy* (in years) times country's population.

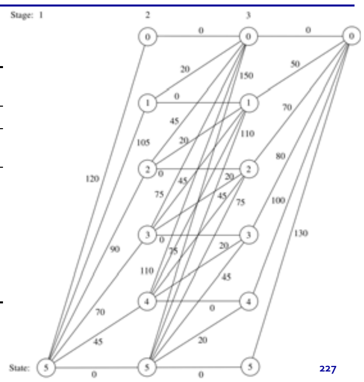
	Thousands of additional person-years of life		
	Country		
Medical teams	1	2	3
0	0	0	0
1	45	20	50
2	70	45	70
3	90	75	80
4	105	110	100
5	120	150	130

Formulation of the problem

- Problem requires three *interrelated decisions*: how many teams to allocate to the three countries (stages).
- x_n is the number of teams to allocate to stage n.
- **What are the states? What changes from one stage to another?**
- s_n = number of medical teams still available for remaining countries $(n, \ldots, 3)$.
- Thus: $s_1 = 5$, $s_2 = 5 - x_1 = s_1 - x_1$, $s_3 = s_2 - x_2$.

States to be considered

Medical teams	Thousands of additional person-years of life — Country 1	2	3
0	0	0	0
1	45	20	50
2	70	45	70
3	90	75	80
4	105	110	100
5	120	150	130



Overall problem

- $p_i(x_i)$: *measure of performance* from allocating x_i medical teams to country i.

$$\text{Maximize} \quad \sum_{i=1}^{3} p_i(x_i),$$

subject to

$$\sum_{i=1}^{3} x_i = 5,$$

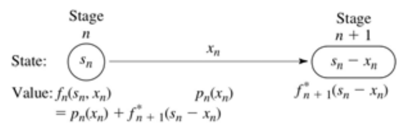
and x_i are nonnegative integers.

Policy

- *Recursive relationship* relating functions:

$$f_n^*(s_n) = \max_{x_n=0,1,\ldots,s_n} \left\{ p_n(x_n) + f_{n+1}^*(s_n - x_n) \right\}, \quad \text{for } n = 1,2$$

$$f_3^*(s_3) = \max_{x_n=0,1,\ldots,s_3} p_3(x_3)$$



Solution procedure, stage n = 3

- For last stage n = 3, values of $p_3(x_3)$ are the last column of table. Here, $x_3^* = s_3$ and $f_3^*(s_3) = p_3(s_3)$.

Medical teams	Thousands of additional person-years of life — Country 1	2	3
0	0	0	0
1	45	20	50
2	70	45	70
3	90	75	80
4	105	110	100
5	120	150	130

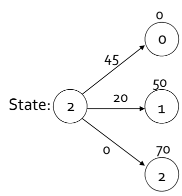
n = 3:

s_3	$f_3^*(s_3)$	x_3^*
0	0	0
1	50	1
2	70	2
3	80	3
4	100	4
5	130	5

Stage n = 2

- Here, finding x_2^* requires calculating $f_2(s_2, x_2)$ for the values of x_2 = 0, 1, …, s_2. Example for s_2 = 2:

Medical teams	Thousands of additional person-years of life — Country 1	2	3
0	0	0	0
1	45	20	50
2	70	45	70
3	90	75	80
4	105	110	100
5	120	150	130



Stage $n = 2$

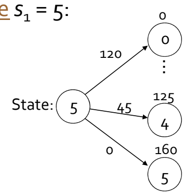
- Similar calculations can be made for the other values of s_2:

$n = 2$:

s_2 \ x_2	$f_2(s_2, x_2) = p_2(x_2) + f_3^*(s_2 - x_2)$: 0	1	2	3	4	5	$f_2^*(s_2)$	x_2^*
0	0						0	0
1	50	20					50	0
2	70	70	45				70	0 or 1
3	80	90	95	75			95	2
4	100	100	115	125	110		125	3
5	130	120	125	145	160	150	160	4

Stage $n = 1$

- Only state is the starting state $s_1 = 5$:

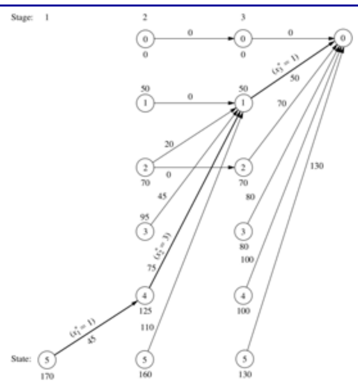


Medical teams	Thousands of additional person-years of life — Country: 1	2	3
0	0	0	0
1	45	20	50
2	70	45	70
3	90	75	80
4	105	110	100
5	120	150	130

$n = 1$:

s_1 \ x_1	$f_1(s_1, x_1) = p_1(x_1) + f_2^*(s_1 - x_1)$: 0	1	2	3	4	5	$f_1^*(s_1)$	x_1^*
5	160	170	165	160	155	120	170	1

Optimal policy decision



Distribution of effort problem

- One kind of *resource* is allocated to a number of *activities*. **Objective**: how to distribute the effort (resource) among the activities most effectively.
- DP involves only one (or few) resources, while LP can deal with thousands of resources.
- The assumptions of LP: *proportionality*, *divisibility* and *certainty* can be violated by DP. Only *additivity* (or analogous for *product* of terms) is necessary because of the *principle of optimality*.
  - World Health Council problem violates proportionality and divisibility (***WHY?***)

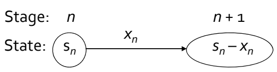
Formulation of distribution of effort

Stage n = activity n ($n = 1, 2, \ldots, N$).

x_n = amount of resource allocated to activity n.

State s_n = amount of resource still available for allocation to remaining activities ($n, \ldots, N$).

- When system starts at stage n in state s_n, choice x_n results in the next state at stage $n + 1$ being $s_{n+1} = s_n - x_n$:



Example

- Distributing scientists to research teams
  - 3 teams are solving engineering problem to safely fly people to Mars.
  - 2 extra scientists reduce the probability of failure.

New scientists	Probability of failure — Team: 1	2	3
0	0.40	0.60	0.80
1	0.20	0.40	0.50
2	0.15	0.20	0.30

Continuous dynamic programming

- Previous examples had a **discrete** state variable s_n, at each stage.
- They all have been **reversible**; the solution procedure could have moved backward or forward stage by stage.
- Next example is *continuous*. As s_n can take any values in certain intervals, the solutions $f_n^*(s_n)$ and x_n^* must be expressed as *functions* of s_n.
- Stages in the next example will correspond to *time periods*, so the solution *must* proceed backwards.

Example: scheduling jobs

- The company Local Job Shop needs to schedule employment jobs due to seasonal fluctuations.
  - Machine operators are difficult to hire and costly to train.
  - Peak season payroll should not be maintained afterwards.
  - Overtime work on a regular basis should be avoided.
- Minimum requirements in near future:

Season	Spring	Summer	Autumn	Winter	Spring
Requirements	255	220	240	200	255

Example: scheduling jobs

- Employment above level in the table costs $2,000 per person per season.
- Total cost of changing level of employment from one season to the other is $200 times the square of the difference in employment levels.
- Fractional levels are possible due to part-time employees.

Formulation

- From data, maximum employment should be 255 (spring). It is necessary to find the level of employment for other seasons. *Seasons* are *stages*.
- One cycle of four seasons, where stage 1 is summer and stage 4 is spring (known employment).
- x_n = employment level for stage n (n =1,2,3,4); x_4=255
- r_n = minimum employment requirement for stage n: r_1=220, r_2=240, r_3=200, r_4=255. Thus:

$$r_n \le x_n \le 255$$

Formulation

- Cost for stage $n = 200(x_n - x_{n-1})^2 + 2000(x_n - r_n)$
- State s_n: employment in the preceding season x_{n-1}

$$s_n = x_{n-1} \qquad (\text{n=1: } s_1 = x_0 = x_4 = 255)$$

Problem:

Choose x_1, x_2 and x_3 as to

$$\text{minimize} \quad \sum_{i=1}^{4}\left[2000(x_i - x_{i-1})^2 + 200(x_i - r_i)\right],$$

$$\text{subject to} \quad r_i \le x_i \le 255, \quad \text{for } i = 1,2,3,4$$

Data

Choose x_1, x_2 and x_3 as to

$$\text{minimize} \quad \sum_{i=1}^{4}\left[2000(x_i - x_{i-1})^2 + 200(x_i - r_i)\right],$$

$$\text{subject to} \quad r_i \le x_i \le 255, \quad \text{for } i = 1,2,3,4$$

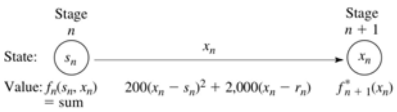
n	r_n	Feasible x_n	Possible $s_n = x_{n-1}$	Cost
1	220	$220 \le x_1 \le 255$	$s_1 = 255$	$200(x_1 - 255)^2 + 2000(x_1 - 220)$
2	240	$240 \le x_2 \le 255$	$220 \le s_2 \le 255$	$200(x_2 - x_1)^2 + 2000(x_2 - 240)$
3	200	$200 \le x_3 \le 255$	$240 \le s_3 \le 255$	$200(x_3 - x_2)^2 + 2000(x_3 - 200)$
4	255	$x_4 = 255$	$200 \le s_4 \le 255$	$200(255 - x_3)^2$

Formulation

- Recursive relationship:

$$f_n^*(s_n) = \min_{r_n \le x_n \le 255} \left\{ 200(x_n - s_n)^2 + 2000(x_n - r_n) + f_{n+1}^*(x_n) \right\}$$

- Basic structure of the problem:



Solution procedure

n	r_n	Feasible x_n	Possible $s_n = x_{n-1}$	Cost
1	220	$220 \le x_1 \le 255$	$s_1 = 255$	$200(x_1 - 255)^2 + 2000(x_1 - 220)$
2	240	$240 \le x_2 \le 255$	$220 \le s_2 \le 255$	$200(x_2 - x_1)^2 + 2000(x_2 - 240)$
3	200	$200 \le x_3 \le 255$	$240 \le s_3 \le 255$	$200(x_3 - x_2)^2 + 2000(x_3 - 200)$
4	255	$x_4 = 255$	$200 \le s_4 \le 255$	$200(255 - x_3)^2$

- Stage 4: the solution is known to be $x_4^* = 255$.

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

Solution procedure

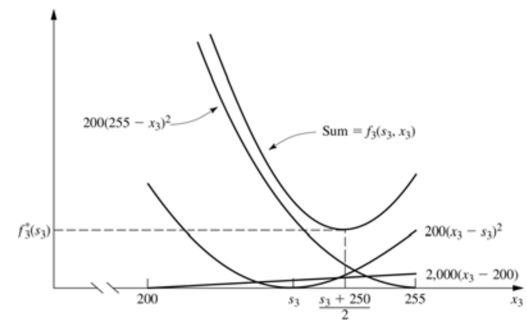
n	r_n	Feasible x_n	Possible $s_n = x_{n-1}$	Cost
1	220	$220 \le x_1 \le 255$	$s_1 = 255$	$200(x_1 - 255)^2 + 2000(x_1 - 220)$
2	240	$240 \le x_2 \le 255$	$220 \le s_2 \le 255$	$200(x_2 - x_1)^2 + 2000(x_2 - 240)$
3	200	$200 \le x_3 \le 255$	$240 \le s_3 \le 255$	$200(x_3 - x_2)^2 + 2000(x_3 - 200)$
4	255	$x_4 = 255$	$200 \le s_4 \le 255$	$200(255 - x_3)^2$

- Stage 3: $240 \le s_3 \le 255$:

$$f_3^*(s_3) = \min_{200 \le x_3 \le 255} \left\{ 200(x_3 - s_3)^2 + 2000(x_3 - 200) + f_4^*(x_3) \right\}$$

$$= \min_{200 \le x_3 \le 255} \left\{ 200(x_3 - s_3)^2 + 2000(x_3 - 200) + 200(255 - x_3)^2 \right\}$$

Graphical solution for $f_3^*(x_3)$



Calculus solution for $f_3^*(x_3)$

- Using *calculus*:

$$\frac{\partial}{\partial x_3} f_3(s_3, x_3) = 400(x_3 - s_3) + 2000 - 400(255 - x_3)$$

$$= 400(2x_3 - s_3 - 250) = 0$$

$$x_3^* = \frac{s_3 + 250}{2}$$

Guarantees minimum?

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

Stage 2

- Solved in a similar fashion, with

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

$$= 200(x_2 - s_2)^2 + 2000(x_2 - 240)$$

$$+ 50(250 - x_2)^2 + 50(260 - x_2)^2 + 1000(x_2 - 150)$$

for $220 \le s_2 \le 255$ (possible values) and $240 \le x_2 \le 255$ (feasible values).

- Solving $\partial/\partial x_2[f_2(s_2, x_2)] = 0$, yields: $x_2 = \dfrac{2s_2 + 240}{3}$

Stage 2

- *The solution has to be feasible for $220 \le s_2 \le 255$ (i.e., $240 \le x_2 \le 255$ for $220 \le s_2 \le 255$)!*

  $x_2^* = \dfrac{2s_2 + 240}{3}$ only feasible for $240 \le s_2 \le 255$.

- Need to solve for feasible value of x_2 that minimizes $f_2(s_2, x_2)$ when $220 \le s_2 < 240$.
- For $s_2 < 240$, $\dfrac{\partial}{\partial x_2} f_2(s_2, x_2) > 0$ for $240 \le x_2 \le 255$

  so $x_2^* = 240$.

Why?

Stage 2 and Stage 1

s_2	$f_2^*(s_2)$	x_2^*
$220 \le s_2 \le 240$	$200(240 - s_2)^2 + 115000$	240
$240 \le s_2 \le 255$	$200/9[(240 - s_2)^2 + (255 - s_2)^2 (270 - s_2)^2] + 2000(s_2 - 195)$	$(2s_2 + 240)/3$

- Stage 1: procedure is similar.

s_1	$f_1^*(s_1)$	x_1^*
255	185000	247.5

- **Solution:**
  - $x_1^* = 247.5$, $x_2^* = 245$, $x_3^* = 247.5$, $x_4^* = 255$
  - Total cost of \$185,000

How?

Deterministic continuous problem

- Consider the following nonlinear programming problem:

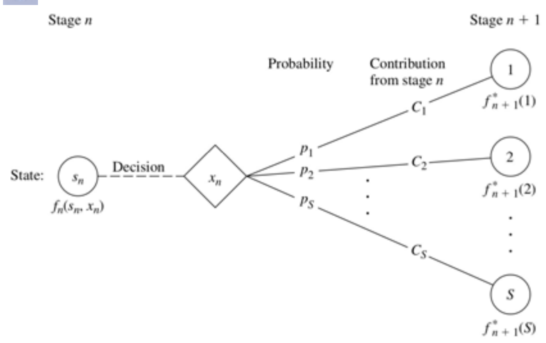
  Maximize $Z = x_1^2 x_2$,
  subject to $x_1^2 + x_2 \le 2$.
  (There are no nonnegativity constraints.)

- Use dynamic programming to solve this problem.

Probabilistic dynamic programming

- State at next stage is *not* completely determined by state and policy decision at current stage.
- There is a *probability distribution* for determining the next state, see figure.
  - S = number of possible states at stage $n + 1$.
  - system goes to i ($i = 1,2,\ldots,S$) with probability p_i given state s_n and decision x_n at stage n.
  - C_i = contribution of stage n to objective function.
- If figure is expanded to all possible states and decisions at all stages, it is a **decision tree**.

Basic structure



Probabilistic dynamic programming

- Relation between $f_n(s_n, x_n)$ and $f^*_{n+1}(s_{n+1})$ depends upon form of overall objective function.
- **Example**: *minimize* the *expected sum* of the contributions from individual stages.
  - $f_n(s_n, x_n)$ is the minimum expected sum from stage n onward, given state s_n and policy decision x_n at stage n:

$$f_n(s_n, x_n) = \sum_{i=1}^{S} p_i \left[C_i + f^*_{n+1}(i) \right]$$

with

$$f^*_{n+1}(i) = \min_{x_{n+1}} f_{n+1}(i, x_{n+1})$$

Example: determining reject allowances

- The Hit-and-Miss Manufacturing Company received an order to supply 1 item of a particular type.
  - Customer requires specified stringent quality requirements.
  - Manufacturer has to produce more than one to achieve one acceptable. Number of *extra* items is the *reject allowance*.
  - Probability of *acceptable* or *defective* is ½.
  - Number of acceptable items in a lot of size L has a *binomial distribution*: probability of not acceptable items is $(1/2)^L$.
  - Setup cost = \$300, cost per item = \$100. Maximum production runs = 3. Cost of no acceptable item after 3 runs = \$1,600.

Formulation

- **Objective**: determine policy regarding lot size (1+reject allowance) for required production run(s) that minimizes total expected cost.

- Stage n = production run n (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.
  - At stage 1, state $s_1 = 1$.

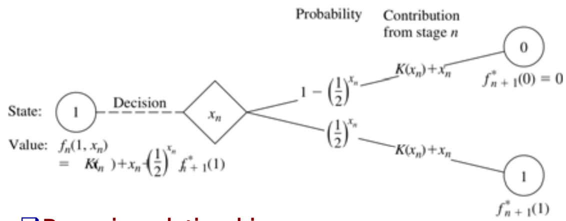
Formulation

- $f_n(s_n, x_n)$ = total expected cost for stages $n,\ldots,3$ and optimal decisions are:
  - $f_n^*(s_n) = \min\limits_{x_n=0,1,\ldots} f_n(s_n, x_n)$
  - $f_n^*(0) = 0$.
- Monetary unit is \$100. Contribution to cost from stage n is $[K(x_n) + x_n]$, with

$$K(x_n) = \begin{cases} 0, & \text{if } x_n = 0 \\ 3, & \text{if } x_n > 0 \end{cases}$$

  - Note that $f_4^*(1) = 16$.

Basic structure of the problem



- **Recursive relationship:**

$$f_n^*(1) = \min_{x_n=0,1,2,\ldots} \left\{ K(x_n) + x_n + 0.5^{x_n} f_{n+1}^*(1) \right\}$$

for $n = 1,2,3$

Solution procedure

$n = 3$:

s_3 \ x_3	$f_3(1,x_3) = K(x_3) + x_3 + (1/2)^{x_3}16$: 0	1	2	3	4	5	$f_3^*(s_3)$	x_3^*
0	0						0	0
1	16	12	9	8	8	8.5	8	3 or 4

$n = 2$:

s_2 \ x_2	$f_2(1,x_2) = K(x_2) + x_2 + (1/2)^{x_2} f_3^*(1)$: 0	1	2	3	4	$f_2^*(s_2)$	x_2^*
0	0					0	0
1	8	8	7	7	7.5	7	2 or 3

$n = 1$:

s_1 \ x_1	$f_1(1,x_1) = K(x_1) + x_1 + (1/2)^{x_1} f_2^*(1)$: 0	1	2	3	4	$f_1^*(s_1)$	x_1^*
1	7	7.5	6.75	6.875	7.44	6.75	2

Optimal solution?

Probabilistic problem

- An enterprising young statistician believes that she has developed a system for winning a popular Las Vegas game. Her colleagues do not believe that her system works, so they have made a large bet with her that if she starts with three chips, she will not have at least five chips after three plays of the game. Each play of the game involves betting any desired number of available chips and then either winning or losing this number of chips. The statistician believes that her system will give her a probability of 2/3 of winning a given play of the game.
- Assuming the statistician is correct, use dynamic programming to determine her optimal policy regarding how many chips to bet (if any) at each of the three plays of the game. The decision at each play should take into account the results of earlier plays. The objective is to maximize the probability of winning her bet with her colleagues.