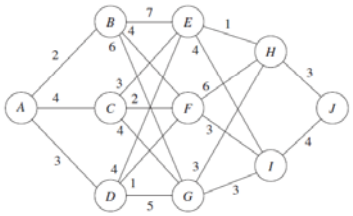
Lecture 17: Introduction to Dynamic Programming



Dynamic Programming

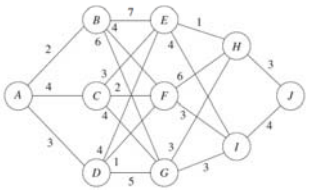
- Dynamic programming is a useful mathematical technique for making a sequence of interrelated decisions.
- It provides a systematic procedure for determining the optimal combination of decisions.

A Prototype Example

- **The stagecoach problem**
  - Mythical fortune-seeker travels West by stagecoach to join the gold rush in the mid-1900s
  - The origin and destination are fixed
    - Many options in choice of route
  - Insurance policies on stagecoach riders
    - Cost depended on perceived route safety
  - **Choose safest route by minimizing policy cost**

A Prototype Example

- Incorrect solution: choose cheapest run offered by each successive stage
  - Gives $A \rightarrow B \rightarrow F \rightarrow I \rightarrow J$ for a total cost of 13
  - There are less expensive options



Sacrificing a little on one stage may permit greater savings thereafter

A Prototype Example

- Trial-and-error solution
  - Very time consuming for large problems
- Dynamic programming solution
  - Starts with a small portion of original problem
    - Finds optimal solution for this smaller problem
  - Gradually enlarges the problem
    - Finds the current optimal solution from the preceding one, until the original problem is solved in its entirety.

A Prototype Example

- Stagecoach problem approach
  - Start when fortune-seeker is only one stagecoach ride away from the destination
  - Increase by one the number of stages remaining to complete the journey
- Problem formulation
  - Decision variables x_1, x_2, x_3, x_4
  - Route begins at *A*, proceeds through x_1, x_2, x_3, x_4, and ends at *J*

A Prototype Example

- Let $f_n(s, x_n)$ be the total cost of the overall policy for the remaining stages
  - Fortune-seeker is in state *s*, ready to start stage *n*
    - Selects x_n as the immediate destination
  - Value of c_{sxn} obtained by setting $i = s$ and $j = x_n$

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

where

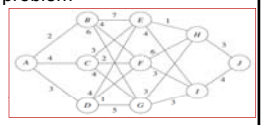
$f_n(s, x_n)$ = immediate cost (stage n) + minimum future cost (stages $n + 1$ onward)
$= c_{sx_n} + f_{n+1}^*(x_n)$.

A Prototype Example

- Immediate solution to the $n = 4$ problem

$n = 4$:

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

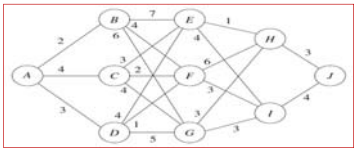


- When $n = 3$:

$n = 3$:

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

A Prototype Example

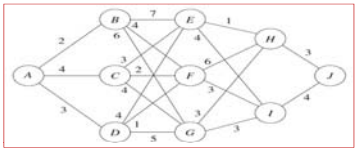


- The $n = 2$ problem

$n = 2$:

s	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

A Prototype Example



- When $n = 1$:

$n = 1$:

s	x_1: B	C	D	$f_1^*(s)$	x_1^*
A	13	11	11	11	C or D

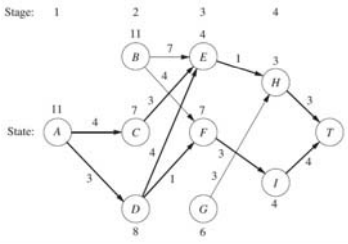
A Prototype Example

- Construct optimal solution using the four tables
  - Results for $n = 1$ problem show that fortune-seeker should choose state *C* or *D*
  - Suppose *C* is chosen
    - For $n = 2$, the result for $s = C$ is $x_2^* = E$ …
    - One optimal solution: $A \rightarrow C \rightarrow E \rightarrow H \rightarrow J$
  - Suppose *D* is chosen instead

    $A \rightarrow D \rightarrow E \rightarrow H \rightarrow J$ and $A \rightarrow D \rightarrow F \rightarrow I \rightarrow J$

A Prototype Example

- All three optimal solutions have a total cost of 11



Characteristics of Dynamic Programming Problems

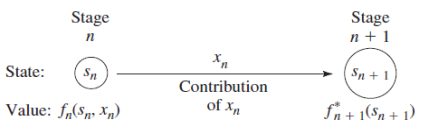
- The stagecoach problem is a literal prototype
  - Provides a physical interpretation of an abstract structure
  - A situation can be formulated as a dynamic programming problem if its basic structure is analogous to the stagecoach problem.
- Features of dynamic programming problems
  - Problem can be divided into stages with a policy decision required at each stage
  - Each stage has a number of states associated with the beginning of the stage
  - The policy decision at each stage transforms the current state into a state associated with the beginning of the next stage

Characteristics of Dynamic Programming Problems

- Features (cont'd.)
  - Solution procedure designed to find an optimal policy for the overall problem
  - Given the current state, an optimal policy for the remaining stages is independent of the policy decisions of previous stages. This is the principle of optimality for dynamic programming**.**
    (Recall Markovian Property).
  - Solution procedure begins by finding the optimal policy for the last stage
  - A recursive relationship can be defined that identifies the optimal policy for stage *n*, given the optimal policy for stage *n* + 1
  - Using the recursive relationship, the solution procedure starts at the end and works backward

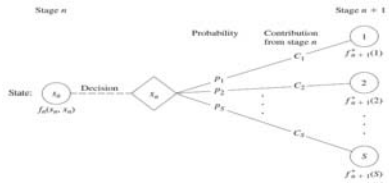
Deterministic Dynamic Programming

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



Probabilistic Dynamic Programming

- Different from deterministic dynamic programming
  - State at the next stage is *not completely determined by the state and policy decision* at the current stage.
  - Probability distribution describes what the next state will be
  - However, this probability distribution still is completely determined by the state and policy decision at the current stage.



Conclusions

- Dynamic programming
  - Useful technique for making a sequence of interrelated decisions
  - Requires forming a recursive relationship
  - Provides great computational savings for very large problems