Systems Engineering 520

Optimization Over Time Under Uncertainty

Huseyin Topaloglu
School of Operations Research
and Industrial Engineering

Lecture 17

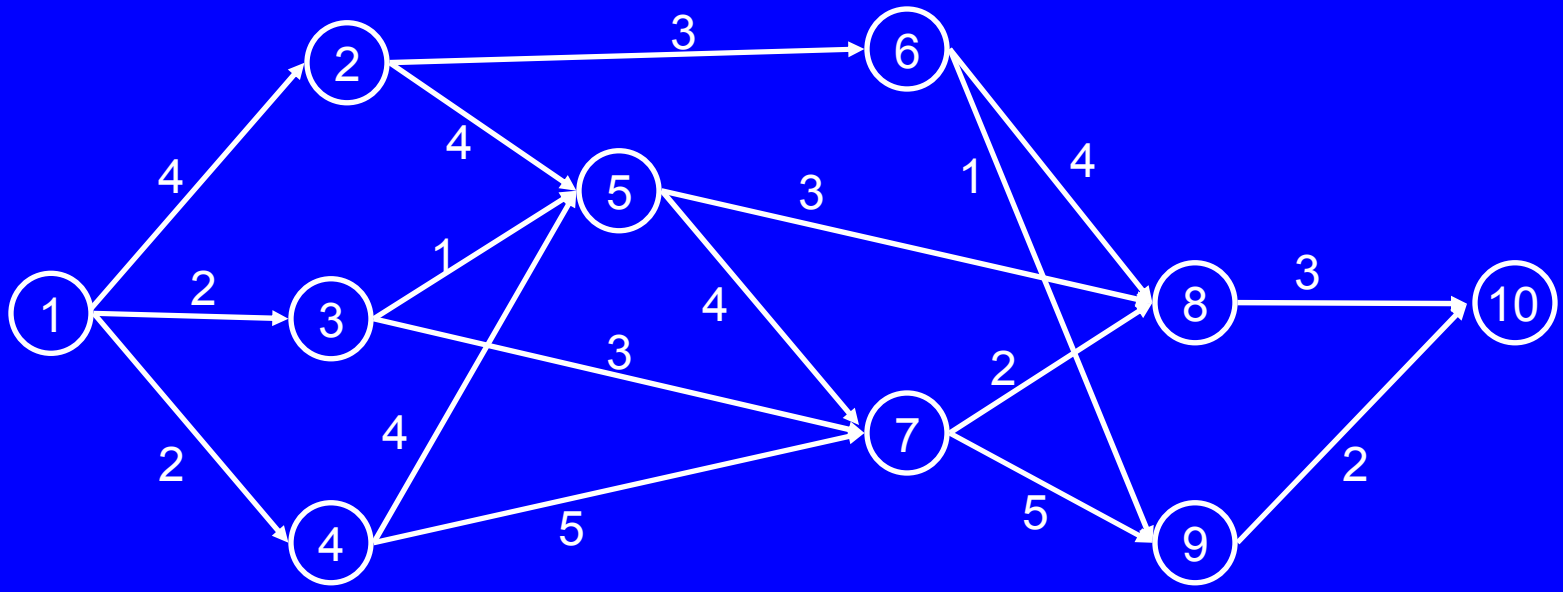
Cornell University

General Approach

- Consider a decision-making problem over the time periods t, t+1,…,T
- We are interested in the minimum total expected cost over the time periods t, t+1,…,T
- An intuitively appealing argument is

Optimal cost over time periods t,t+1,…,T = min {
Cost of action 1 + Optimal cost over time periods t+1,…,T
Cost of action 2 + Optimal cost over time periods t+1,…,T
Cost of action 3 + Optimal cost over time periods t+1,…,T
}

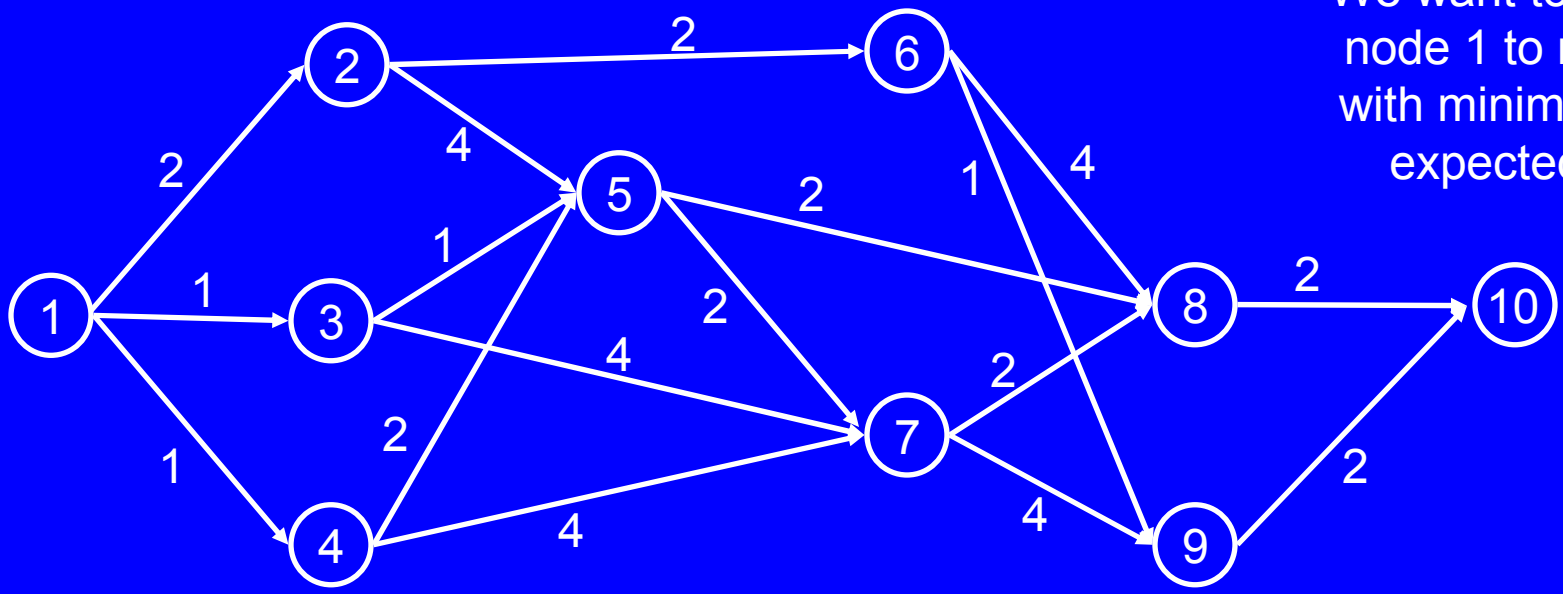
Shortest Path Problem over a Non-cyclic Network



We want to compute the shortest path from node 1 to node 10

Stochastic Shortest Path Problem over a Non-cyclic Network

We want to go from node 1 to node 10 with minimum total expected cost



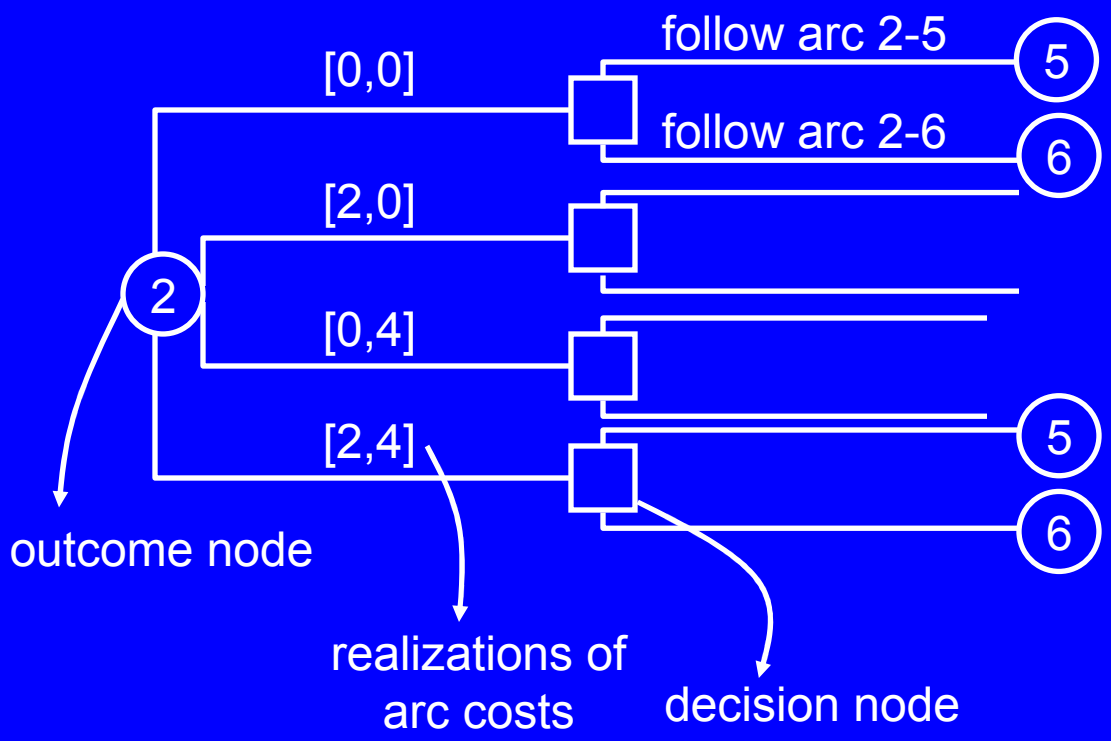
The cost of each arc is either the number indicated on the arc or 0, with equal probabilities

The costs of the arcs are independent

When we are at a particular node, we get to see the costs of the outbound arcs before we decide on the outbound arc to follow

Stochastic Shortest Path Problem over a Non-cyclic Network

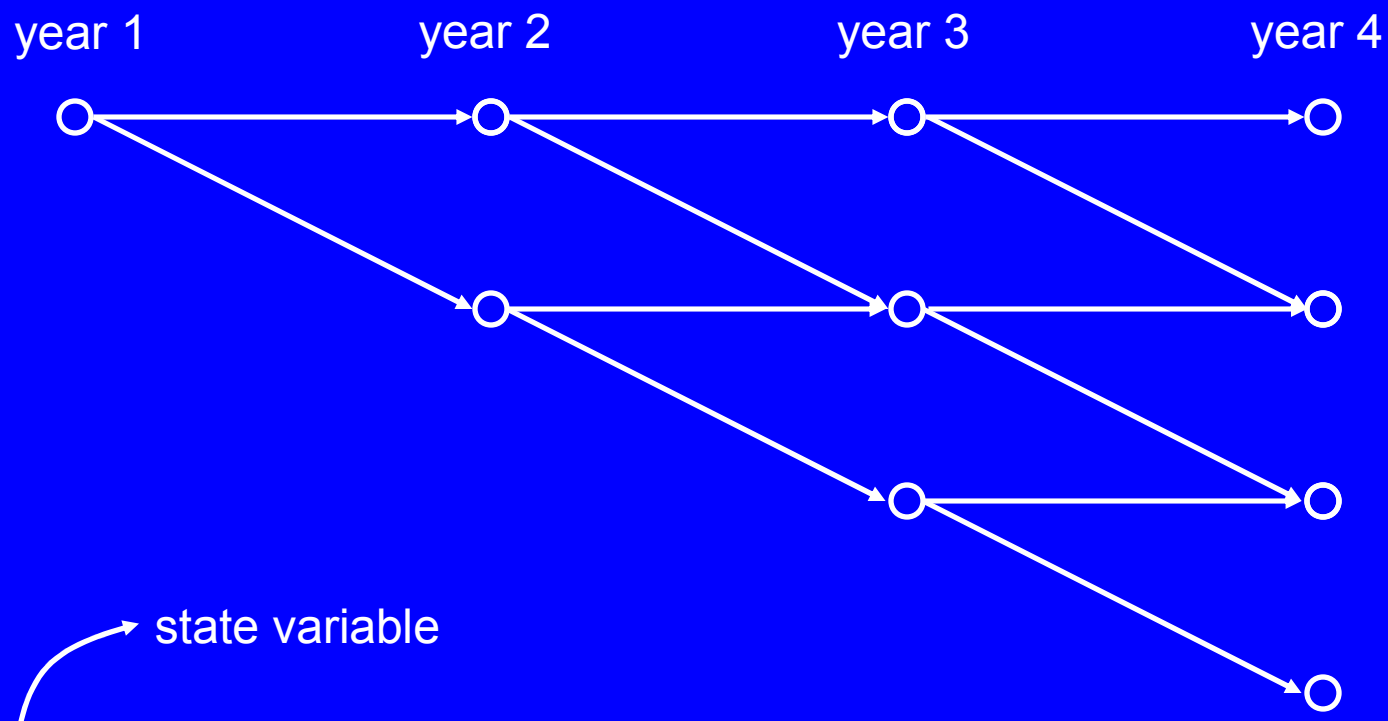
- The problem can be visualized as a decision tree



Fishing Problem

- Interested in maximizing the profits by fishing from a pond over three years
- At the beginning of each season, we decide whether to fish
- If we fish, then we extract 70% of the fish population, the fish will reproduce and the fish population by the beginning of the next season will the same as what it was at the beginning of the current season
- If we do not fish, then the fish population will double by the beginning of the next season
- Initial fish population is 10 tons and profit is $1/ton
- Future profits are discounted by 80%

Fishing Problem



$v_t(s)$ = total discounted profit given that we have s tons of fish in the pond at the beginning of season t

$v_t(s) = \max \{ 0.7\, s + 0.8\, v_{t+1}(s) , 0.8\, v_{t+1}(2s) \}$

Generalizing the Idea

- Dynamic programming provides a structured method to decompose a problem over multiple time periods (or multiple stages) and enables us to solve the problem by focusing on one time period at a time
- However, the time periods (or stages) do not have to represent passage of time
- Assume we have b (integer) units of space in a knapsack and we would like to decide which subset of K items to put into this knapsack
- Item k occupies a_k (integer) units of space in the knapsack
- The utility from having item k in the knapsack is c_k

Knapsack Problem

$$\max \quad \sum_{k=1}^{K} c_k \, x_k$$

$$\text{subject to} \quad \sum_{k=1}^{K} a_k \, x_k \leq b$$

$$x_k \in \{0, 1\} \quad k = 1, \ldots, K$$

- Let $v_k(s)$ be the maximum utility that we can obtain if there is s units of space in the knapsack and we can use only the first k items
- The optimal objective value to the problem is $v_K(b)$

Knapsack Problem

$$v_k(s) = \begin{cases} v_{k-1}(s) & \text{if } a_k > s \\ \max\{c_k + v_{k-1}(s - a_k), v_{k-1}(s)\} & \text{if } a_k \leq s \end{cases}$$

take item k

do not take item k

Example

$$\begin{aligned} \max \quad & 16\,x_1 + 19\,x_2 + 23\,x_3 + 28\,x_4 \\ \text{subject to} \quad & 2\,x_1 + 3\,x_3 + 4\,x_3 + 5\,x_4 \leq 7 \\ & x_k \in \{0, 1\} \quad k = 1, \ldots, 4 \end{aligned}$$

Largest Interval Sum

- We are given a sequence of integers $X_1, X_2, \ldots, X_n$
- We want a subsequence of consecutive integers $X_i, \ldots, X_j$ such that the sum $X_i + \ldots + X_j$ is as large as possible
- Solution can be found by trying n (n-1) possibilities
- Can we do something faster?
- Let v_k be the largest possible sum obtained by starting from somewhere in the sequence and going all the way to the k-th integer
- We have the recursion

$$v_k = \max\{ X_k , X_k + v_{k-1} \}$$

Optimized Matrix Multiplication

- We are given matrices $A_1, A_2, \ldots, A_n$
- Size of matrix A_i is P_i x P_{i+1}
- We want to find in what order we should multiply these matrices so that the number of scalar multiplications is minimized
- If A and B are matrices of size p x q and q x r, then computing AB requires pqr scalar multiplications
- Consider computing ABC with A (10 x 100), B (100 x 5) and C (5 x 50)
- If we go with (AB)C, then the number of multiplications is…
- If we go with A(BC), then the number of multiplications is…

Optimized Matrix Multiplication

- If we have n matrices, then the number of possible ways to multiply n matrices get large very fast as n gets large
- How can we find the best way of multiplying the matrices that requires the least number of scalar multiplications?
- Let M[i,j] be the least number of scalar multiplications required to multiply the matrices A_i, $A_{j+1},\ldots,A_j$
- We have the recursion

$$M[i,j] = \min_{i \le k < j} M[i,k] + M[k+1,j] + P_i P_{k+1} P_j$$

with M[i,i] = 0

Optimized Matrix Multiplication

- Consider the case where n = 4 and A_1 (10 x 2), A_2 (2 x 3), A_3 (3 x 5), A_4 (5 x 4)

Finding Longest Common Substring

- Consider two sequences X and Y, say

  X = A L G O R I T M

  Y = P A R A C H U T E

- RT is a common substring in X and Y
- What is the longest common substring in X and Y?
- Let v(n,m) be the longest common substring when we look at the first n elements of sequence X and the first m elements of sequence Y

Finding Longest Common Substring

- We have the recursion

$$v(n,m) = \max\Big\{v(n-1,m), v(n,m-1),$$
$$v(n-1,m-1) + \mathbf{1}(n\text{-th character of } X$$
$$\text{matches } m\text{-th character of } Y)\Big\}$$

Finding Longest Common Substring

	λ	p	a	r	a	c	h	u	t	e
λ	0	0	0	0	0	0	0	0	0	0
a	0	0	1	1	1	1	1	1	1	1
l	0	0	1	1	1	1	1	1	1	1
g	0	0	1	1	1	1	1	1	1	1
o	0	0	1	1	1	1	1	1	1	1
r	0	0	1	2	2	2	2	2	2	2
i	0	0	1	2	2	2	2	2	2	2
t	0	0	1	2	2	2	2	2	3	3
h	0	0	1	2	2	2	3	3	3	3
m	0	0	1	2	2	2	3	3	3	3

values of v(n,m)

A More Conventional Example

- We have C seats on an airplane
- There are n customer classes indexed by j=1,…,n in their order of arrival
- If we sell a seat to customer class j, then we make a revenue of r_j
- The demand from customer class j is random
- D_j is the random variable representing the demand from customer class j
- What is the optimal policy to sell the seats to the arriving customers in order to maximize the total expected revenue from n customer classes?

A More Conventional Example

- Let $v_j(x,D_j)$ be the optimal total expected revenue that can be obtained from customer classes j+1,…,n if we have x seats available when we face the demand D_j from customer class j
- We have the recursion

$$v_j(x, D_j) = \max_{0 \le u \le \min\{D_j, x\}} \left\{ r_j\, u + \sum_{d_{j+1}=0}^{\infty} \mathbb{P}\{D_{j+1} = d_{j+1}\}\, v_{j+1}(x-u, d_{j+1}) \right\}$$

A More Conventional Example

- Consider the case where n=3, r_1= 4, r_2 = 5, r_3 = 10

d	$P\{D_1=d\}$	$P\{D_2=d\}$	$P\{D_3=d\}$
0	2/7	1/5	1/9
1	1/7	1/5	2/9
2	2/7	1/5	3/9
3	1/7	1/5	2/9
4	1/7	1/5	1/9