CE 273
Markov Decision Processes

Lecture 4

Finite Horizon MDPs

Previously on Markov Decision Processes

Consider a Tesla manufacturing plant which produces one car each day. Demand for cars, however, can occur in batches. Let $\{Y_n, n \geq 1\}$ be the sequence of iid demands on different days with a pmf

$$\alpha_k = \mathbb{P}\big[Y_n = k\big], k \in \mathbb{Z}^+$$

Let X_n represent the number of cars in the warehouse after the demand for a particular day is met. Assuming no back orders and that the production occurs before sales,

Suppose $X_n = i$. Then for $0 < j \leq i+1$, $\mathbb{P}\big[X_{n+1} = j | X_n = i, X_{n-1}, \ldots, X_0\big]$

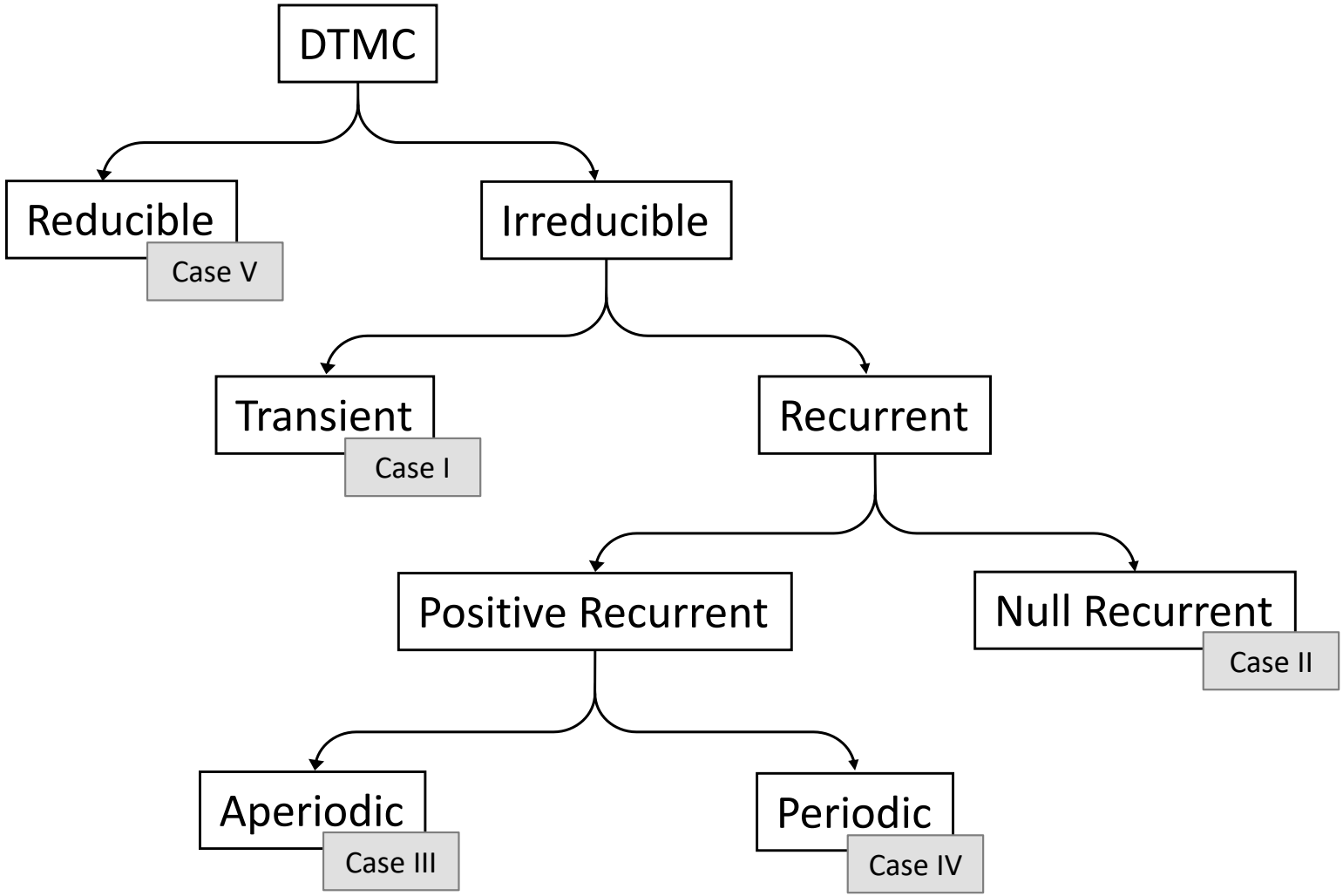
$$= \mathbb{P}\big[\max\{X_n + 1 - Y_{n+1}, 0\} = j | X_n = i, X_{n-1}, \ldots, X_0\big]$$
$$= \mathbb{P}\big[X_n + 1 - Y_{n+1} = j | X_n = i, X_{n-1}, \ldots, X_0\big] = \alpha_{i-j+1}$$

If $j = 0$, $\mathbb{P}\big[X_{n+1} = 0 | X_n = i, X_{n-1}, \ldots, X_0\big]$

$$= \mathbb{P}\big[\max\{X_n + 1 - Y_{n+1}, 0\} = 0 | X_n = i, X_{n-1}, \ldots, X_0\big]$$

$$= \mathbb{P}\big[Y_{n+1} \geq i+1 | X_n = i, X_{n-1}, \ldots, X_0\big] = \sum_{k=i+1}^{\infty} \alpha_k = \beta_i$$

Previously on Markov Decision Processes



Previously on Markov Decision Processes

Theorem (Case I)

Let $\{X_n, n \geq 0\}$ be an transient, irreducible DTMC. Then

$$\lim_{n\to\infty} p_{ij}^{(n)} = 0 \,\forall\, i, j \in S$$

Theorem (Case II)

Let $\{X_n, n \geq 0\}$ be an null recurrent, irreducible DTMC. Then

$$\lim_{n\to\infty} p_{ij}^{(n)} = 0 \,\forall\, i, j \in S$$

Previously on Markov Decision Processes

Theorem (Case III)

Let e be a column vector of ones. For an aperiodic, positive recurrent, irreducible DTMC, there exists unique $\pi_j > 0, j \in S$ such that

$$\lim_{n\to\infty} p_{ij}^{(n)} = \pi_j, \forall\, i,j \in S$$

$$\pi P = \pi \text{ (Balance Equation)}$$

$$\pi e = 1 \text{ (Normalizing Equation)}$$

Theorem (Case IV)

Let e be a column vector of ones. For a periodic, positive recurrent, irreducible DTMC, there exists unique $\pi_j > 0, j \in S$ such that

$$\lim_{n\to\infty} \frac{m_{ij}^{(n)}}{n+1} = \pi_j, \forall\, i,j \in S$$

$$\pi P = \pi$$

$$\pi e = 1$$

Previously on Markov Decision Processes

Theorem (Case V)

Let $i \in \mathcal{C}$ and $j \in C_r$.

1. *If C_r is transient or null recurrent $d_{ij}^{(n)} \to 0$*
2. *If C_r is aperiodic and positive recurrent, $d_{ij}^{(n)} \to u_i(r)\pi_j$, where π_js are derived from limiting distribution of $P(r)^{(n)}$*
3. *If C_r is periodic and positive recurrent, $d_{ij}^{(n)}$ does not have a limit. However $\sum_{m=0}^{n} d_{ij}^{(m)}/(n+1) \to u_i(r)\pi_j$, where π_js are derived from limiting distribution of $P(r)^{(n)}$*

Previously on Markov Decision Processes

From the examples, we can see that

- $\lim_{n\to\infty} P^{(n)}$ doesn't always exist
- $\lim_{n\to\infty} \frac{M^{(n)}}{n+1}$ however always exists and equals $\lim_{n\to\infty} P^{(n)}$ when the later exists. (Why is this intuitively true?)

Case	$\lim_{n\to\infty} P^{(n)}$	$\lim_{n\to\infty} \frac{M^{(n)}}{n+1}$	Identical Rows	Row Sum $= 1$
I	✓	✓	✓	X
II	✓	✓	✓	X
III	✓	✓	✓	✓
IV	X	✓	✓	✓
V	✓	✓	X	✓

Lecture Outline

1. DTMCs with Costs and Rewards
2. Deterministic Problems
3. Finite Horizon Models
4. Dynamic Programming

DTMCs with Costs and Rewards

DTMCs with Costs and Rewards

Introduction

So far, state transitions in the DTMC did not involve any costs or rewards. We can model these and study the long-run

- Discounted costs
- Average costs

Rewards can be thought of negative costs for optimization. Assume that every time the DTMC visits state i, an expected cost of $c(i)$ $(< \infty)$ is incurred.

One can also extend this analysis to cases where costs are a function of the future state, i.e., $c(i, j)$.

We typically do not evaluate total cost because it can be unbounded. We will revisit these ideas in greater detail in the context of MDPs.

DTMCs with Costs and Rewards

Discounted Costs

Costs are usually discounted for mathematical and practical reasons. Mathematically, they guarantee that the long-run discounted costs are bounded.

Practically, a cost of c units in one future time-step is equivalent to incurring αc now, where $0 \leq \alpha < 1$. More generally, cost c at time step n is equivalent to $\alpha^n c$ now.

One interpretation of α is that it reflects the interest rate. Another grim way to look at is to assume that time is finite, and the future may not happen with probability $(1 - \alpha)$. Define the cost over the infinite horizon as

$$C = \sum_{n=0}^{\infty} \alpha^n c(X_n)$$

C is a random variable and hence let's look at the expected total discounted cost starting from state i,

$$\phi(i) = \mathbb{E}\big[C | X_0 = i\big]$$

DTMCs with Costs and Rewards

Discounted Costs

Theorem

Suppose c and ϕ represent column vectors of $c(i)$s and $\phi(i)$s. For $0 \leq \alpha < 1$,

$$\phi = (I - \alpha P)^{-1} c$$

Proof.

Define $C_1 = \sum_{n=1}^{\infty} \alpha^n c(X_n)$. Since the DTMC is time-homogeneous,

$$\mathbb{E}\big[C_1 | X_1 = j\big] = \alpha\phi(j) \,\forall\, j \in S$$

$$\therefore \mathbb{E}\big[C_1 | X_0 = i\big] = \sum_{j \in S} p_{ij} \mathbb{E}\big[C_1 | X_1 = j, X_0 = i\big] = \sum_{j \in S} p_{ij} \alpha\phi(j)$$

From the definition of ϕ,

$$\begin{aligned} \phi(i) = \mathbb{E}\big[C | X_0 = i\big] &= \mathbb{E}\big[c(X_0) + C_1 | X_0 = i\big] \\ &= c(i) + \alpha \sum_{j \in S} p_{ij} \phi(j) \end{aligned}$$

which in matrix form can be written as $\phi = c + \alpha P \phi$. ■

DTMCs with Costs and Rewards

Example

Consider the PageRank example. Suppose each website can potentially bring Google $[1 \;\; 2 \;\; 5 \;\; 3]^\mathsf{T}$ dollars of ad revenue.

$$C = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \qquad P = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1/2 & 0 & 1/2 \\ 1/3 & 0 & 1/3 & 1/3 \\ 1 & 0 & 0 & 0 \\ 1/4 & 1/4 & 1/4 & 1/4 \end{bmatrix} \end{array}$$

For $\alpha = 0.9$, What is the expected total discounted revenue?

$$c = \left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - 0.9 \begin{bmatrix} 0 & 1/2 & 0 & 1/2 \\ 1/3 & 0 & 1/3 & 1/3 \\ 1 & 0 & 0 & 0 \\ 1/4 & 1/4 & 1/4 & 1/4 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 \\ 2 \\ 5 \\ 3 \end{bmatrix} = \begin{bmatrix} 23.37 \\ 24.41 \\ 26.03 \\ 25.30 \end{bmatrix}$$

DTMCs with Costs and Rewards

Average Costs

The expected costs for the discounted case **depends on the initial state**. The advantage however is that we don't need any conditions on the DTMC and the inverse exists as long as $\alpha < 1$.

Interestingly, if we average the total expected cost without discounting, the initial state does not matter!

Define the long-run expected cost per period (average cost),

$$g(i) = \lim_{N\to\infty} \frac{1}{N+1}\mathbb{E}\left\{ \sum_{n=0}^{N} c(X_n)|X_0 = i \right\}$$

For the above limit to exist, assume that the DTMC is irreducible and positive recurrent. Since πs represent the average time spent in state i, we would expect $g(i) = g = \sum_{j\in S} \pi_j c(j)$.

DTMCs with Costs and Rewards

Average Costs

Theorem

Consider an positive recurrent, irreducible DTMC with limiting occupancy distribution $\{\pi_j, j \in S\}$. *Then* $g(i) = g = \sum_{j \in S} \pi_j c(j)$

Proof.

$$
\begin{aligned}
g(i) &= \lim_{n \to \infty} \frac{1}{N+1} \sum_{j \in S} m_{ij}^{(n)} c(j) \\
&= \sum_{j \in S} \lim_{n \to \infty} \frac{m_{ij}^{(n)}}{n+1} c(j) \\
&= \sum_{j \in S} \pi_j c(j)
\end{aligned}
$$

■

Exchange of limit and summation is allowed because of the bounded convergence theorem.

DTMCs with Costs and Rewards

Example

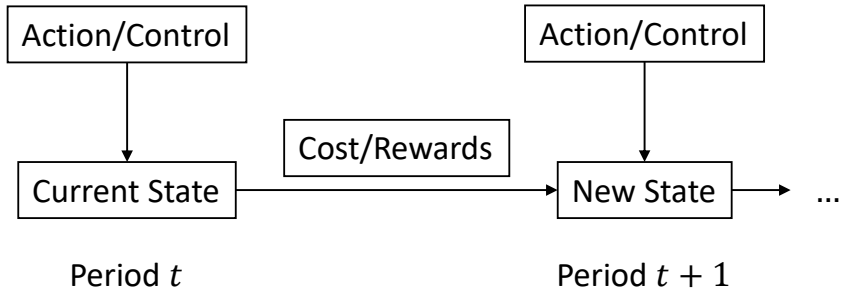
Consider the PageRank example. Using the limiting occupancy distribution calculated in Lecture 3, the average revenue per click is

$$\begin{bmatrix} 0.3077 & 0.2308 & 0.1538 & 0.3077 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 5 \\ 3 \end{bmatrix} = 2.4614$$

DTMCs with Costs and Rewards

MDPs Revisited

Let's transition into MDPs in which the decision maker can take an action/control which takes the system to another state and the process is repeated.



The end goal is to optimize some function of the sequence of cost/reward (which depends on the current state and action).

Before choosing the action, one cannot predict the future state with certainty. State transitions are usually stochastic and are a function of the actions.

Let us first look at models in which the time horizon is finite.

Lecture Outline

Deterministic Problems

Deterministic Problems

Introduction

Before looking at the stochastic case, it is helpful to look at sequential decision making in deterministic settings since the key ideas can be directly translated.

In deterministic problems, one can predict with certainty the effect of taking an action in a particular state.

Let's look at this using two examples:

- Scheduling problem
- Shortest paths

Deterministic Problems

Scheduling Problem

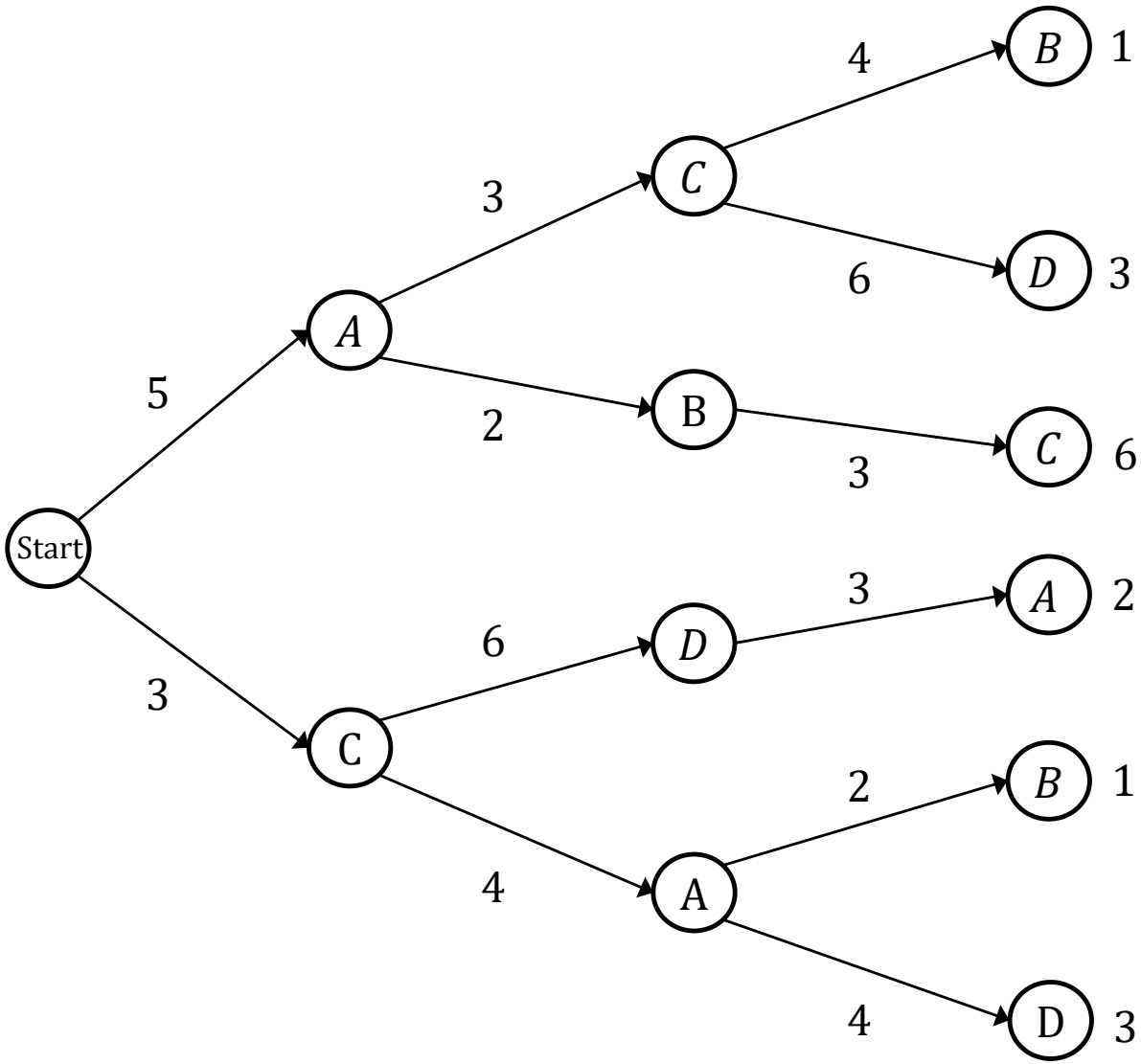
Consider the problem of sequencing operations to produce an item. Suppose producing a product requires four tasks: A, B, C, and D.

Assume that B can be performed only after A and D can be performed only after C.

Assume that start-up costs $s_A = 5$ and $s_C = 3$. Let c_{ij} be the cost of performing i followed by j.

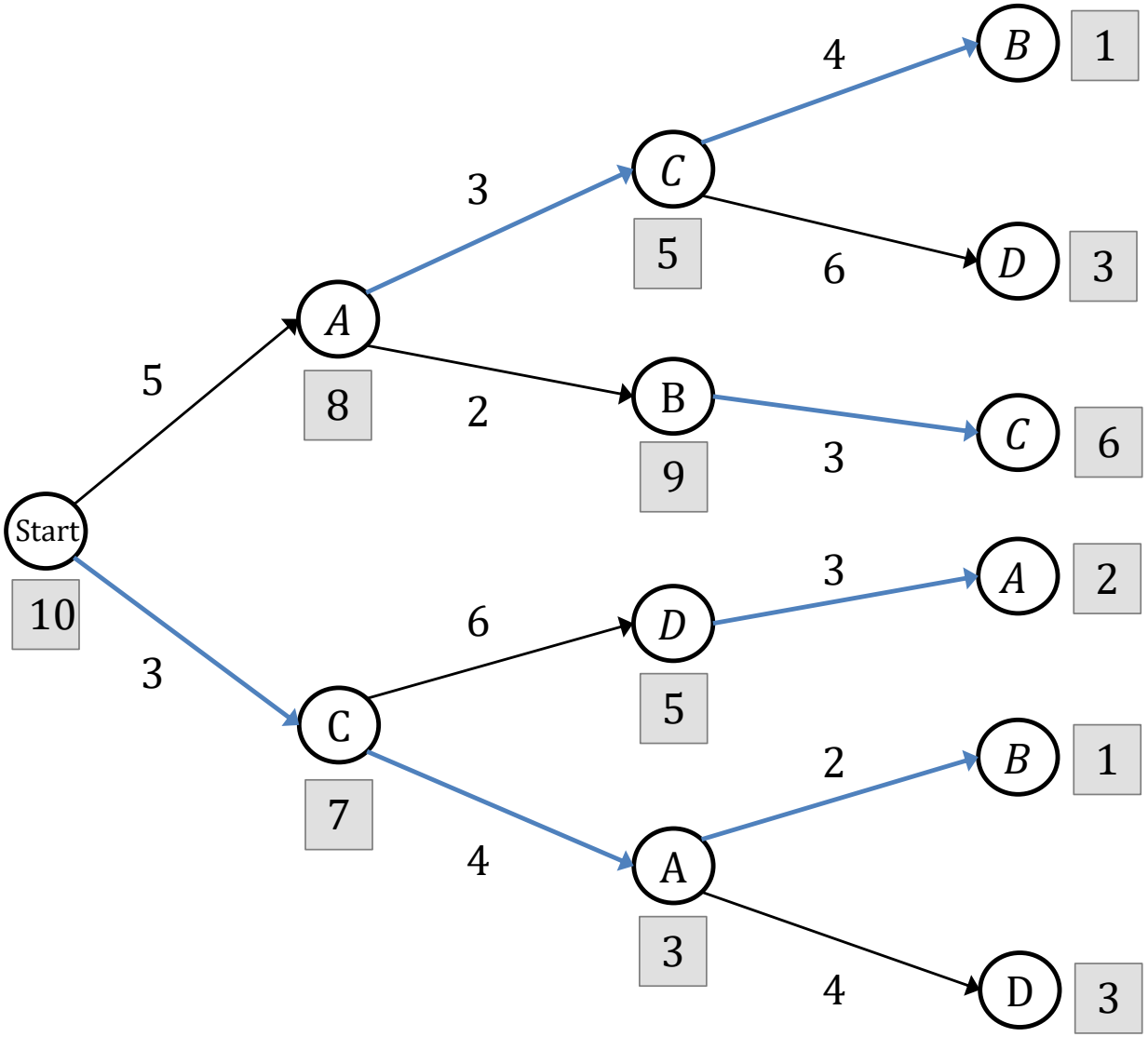
Deterministic Problems

Scheduling Problem



Deterministic Problems

Scheduling Problem



Deterministic Problems

Scheduling Problem

The idea behind the solution technique was simple. Starting from the last step $i = N - 1$,

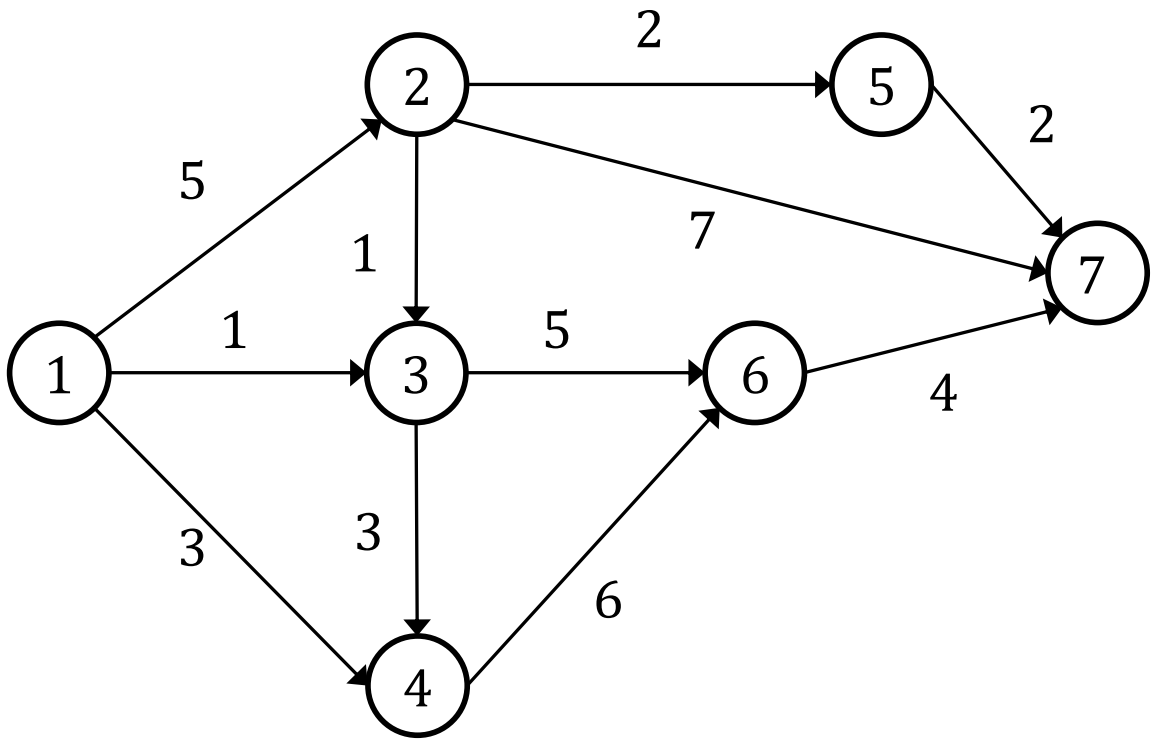
- Solve the tail subproblem from i to N and find the optimal action
- $i \leftarrow i - 1$ and repeat. The optimal actions from next time period are the same as those computed previously.

Deterministic Problems

Shortest Paths

Let's look at another example which can be solved using a similar principle.

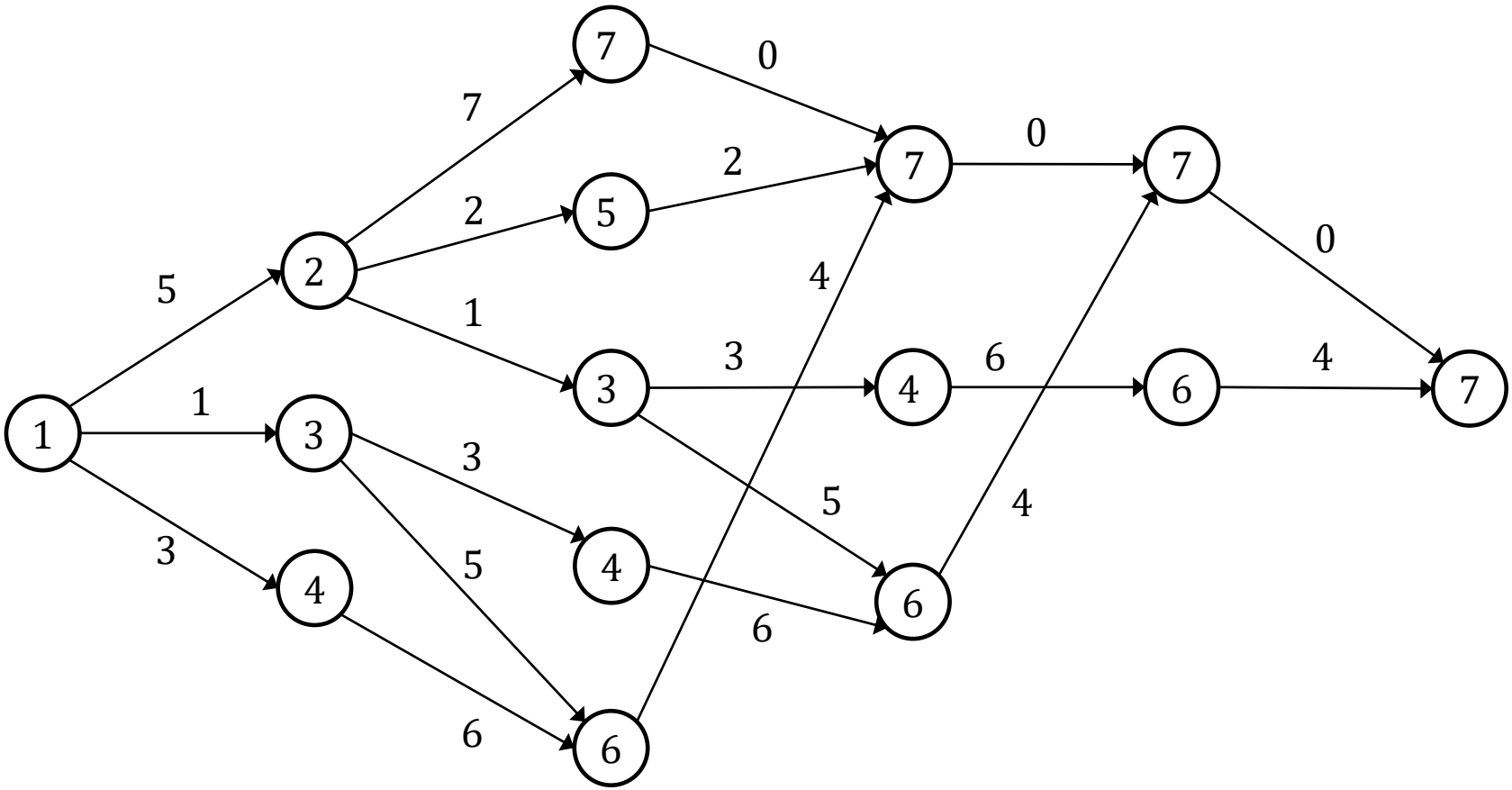
Find the shortest path in the following graph:



Deterministic Problems

Shortest Paths

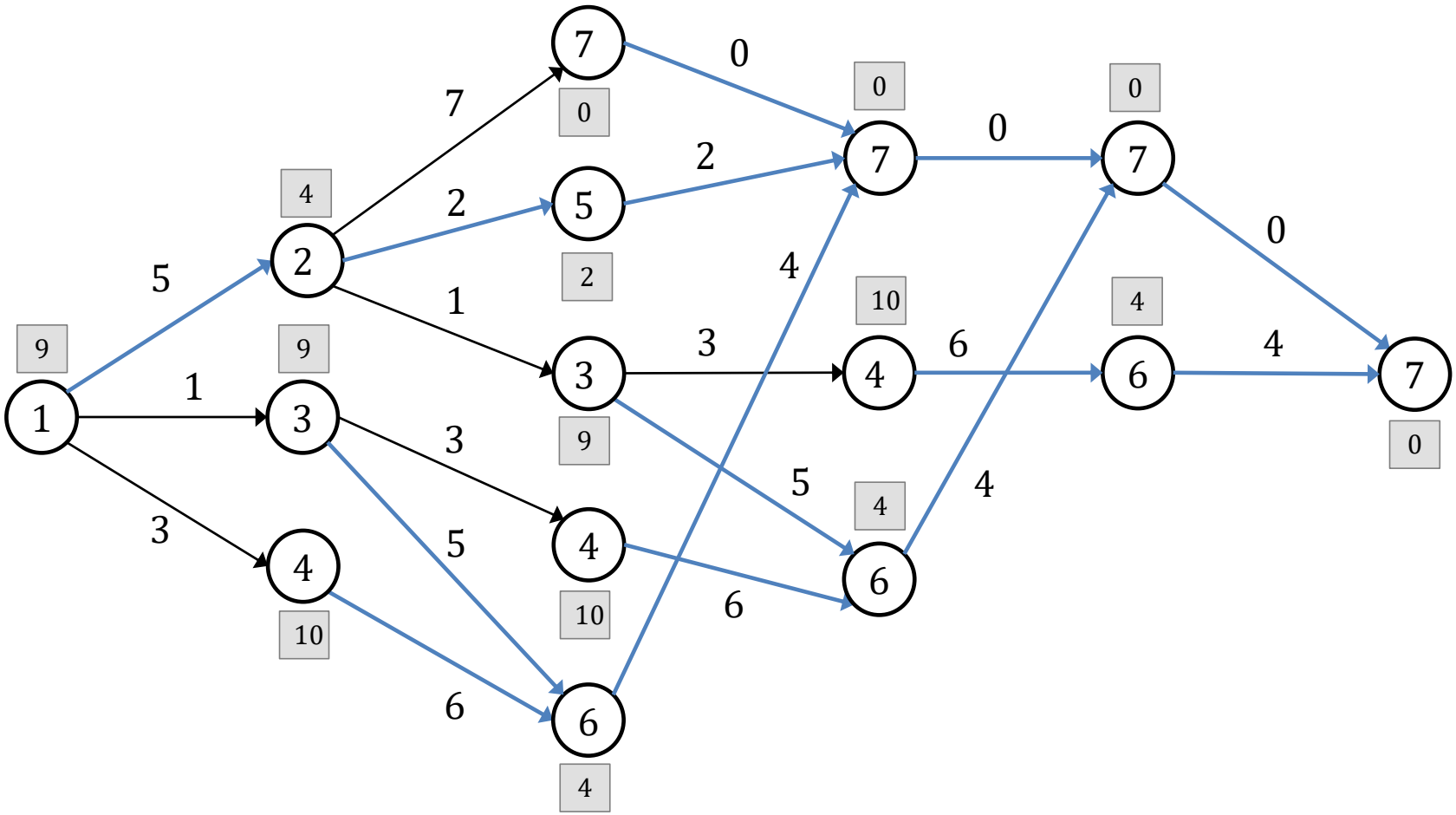
Reconstruct the graph in the following way:



We can now use the same approach over the 6 stages.

Deterministic Problems

Shortest Paths



Can you generalize this procedure? For a graph containing N nodes, create $N-1$ stages and assume a self-loop with 0 cost at the destination.

Finite Horizon Models

Finite Horizon MDPs

Introduction

For general MDPs, the result of actions is not known with certainty. Two types of notation are common:

The first, is more popular in the control theory community. The second one is ideal when the state space is countable and is similar to the notation used to study DTMCs.

Let's look at the first one. Suppose there are N time steps $k = 0, 1, 2, \ldots, N-1$. For each k, define

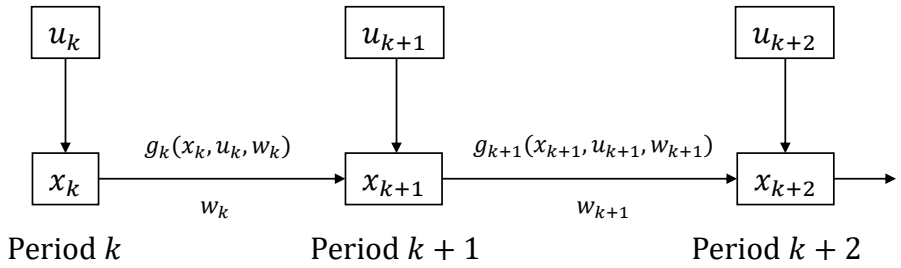
Notation	*Description*
x_k	State of the system at time k
u_k	Action/control/decision variable to be chosen at k
w_k	Disturbance, a random variable with known distribution
$f_k(x_k, u_k, w_k)$	System dynamics

The distribution of w_k may depend on x_k and u_k and is usually independent across time. We will sometimes use lower-case notation for denoting the random variables and not their realizations.

Additionally, we incur a **one-step cost** of $g_k(x_k, u_k, w_k)$ due to taking an action in a particular state. We also assume that the final state x_N results in a terminal cost of $g_N(x_N)$. One can maximize rewards by representing them as negative costs.

The total cost is

$$g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k)$$



The above cost is a random variable because $w_0, \ldots, w_{N-1}$ are random variables. Hence, we are typically interested in minimizing the expected total cost

$$\mathbb{E}\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k) \right\}$$

where the expectation is taken with respect to the joint distribution of $w_0, \ldots, w_{N-1}$.

Let w be the vector of disturbances.

Since $u_0, u_1, \ldots, u_{N-1}$ are the decision variables, one can write the optimization problem as

$$\min_{u_0,\ldots,u_{N-1}} \mathbb{E}_w \left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k) \right\}$$

This method of optimization assumes that all the decision variables are calculated at $k = 0$ and is also called **open-loop minimization**.

For example, in the context of shortest paths, if the arc travel times were stochastic, this is an example of a priori shortest path, where one chooses the entire path at the start of the trip.

However, the knowledge of the state at different stages can be used to make better decisions. Say you get information that a route is congested midway.

Alternately, instead of just searching for the optimal $u_0, u_1, \ldots, u_{N-1}$, we try to find a **sequence of functions** $\mu_0, \mu_k, \ldots, \mu_{N-1}$, where $\mu_k(x_k)$ is the optimal decision to be taken at step k.

When we see the state x_k in stage k, we use the above functions to take an action $u_k = \mu_k(x_k)$. The sequence of functions is called a **policy**, and it is a complete contingent plan of action.

We denote policies using $\pi = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$. This style of optimization is also called **closed-loop optimization**.

In other words, our decision variables are not real numbers (or vectors of reals) but functions!

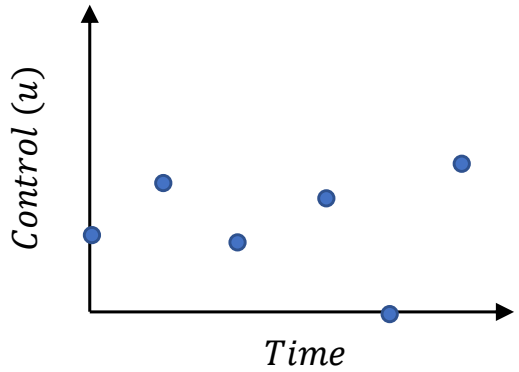


Figure: Open-loop Policy

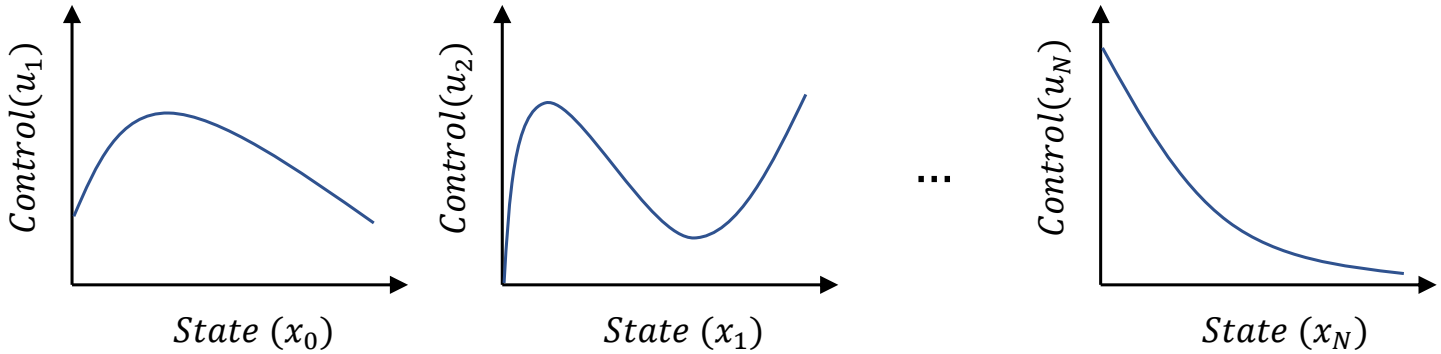


Figure: Closed-loop Polciy

Finite Horizon MDPs

Introduction

We denote the set of all states at time k using S_k. The actions at k can depend on the time step and the state (think of chess).

Let $U_k(x_k)$ be the set of actions available at time step k and at state x_k. We say that a policy π is **admissible** if $\mu_k(x_k) \in U_k(x_k) \, \forall \, x_k \in S_k, k \in \{0, 1, \ldots, N-1\}$. In other words, $\mu_k(x_k) : S_k \to U_k(x_k)$. Let Π be the set of all admissible policies.

Think of π as the decision variable and Π as the feasible region. The objective for a given π is

$$J_\pi(x_0) = \mathbb{E}\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right\}$$

where expected is taken over w and states evolve according to $x_{k+1} = f_k(x_k, u_k, w_k)$. The goal is to find π^* that minimizes the above cost.

$$J^*(x_0) = J_{\pi^*}(x_0) = \min_{\pi \in \Pi} J_\pi(x_0)$$

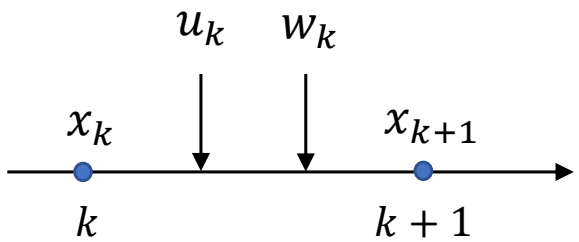
$J^*(x_0)$ and is called the **optimal value or cost function**. Note that it is a function of the initial state just like DTMC with costs.

Finite Horizon MDPs

Example

Consider a variant of the inventory example we saw earlier. Assume that

Notation	*Description*
x_k	No. of cars at the beginning of period k
u_k	Cars ordered at the begining of period k (delivered instantaneously)
w_k	Demand for cars during kth period with known distribution



Assume that back orders are allowed. The state transitions can be expressed as

$$x_{k+1} = f_k(x_k, u_k, w_k) = x_k + u_k - w_k$$

Suppose the one-step cost $g_k(x_k, u_k, w_k)$ is $r(x_k) + cu_k$. It is not necessary that g depends on all three arguments.

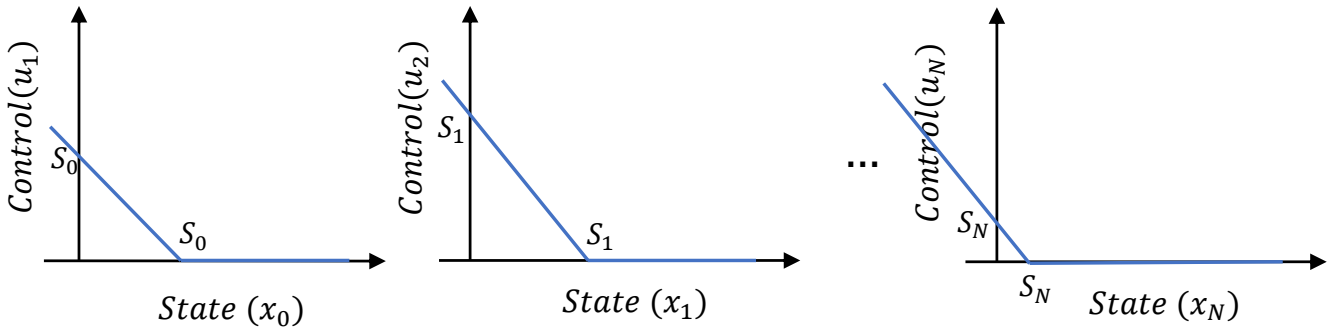
The first term is the penalty for holding or falling short and the second term is the cost of ordering u_k units. Additionally, let $R(x_N)$ denote the terminal cost.

Finite Horizon MDPs

Example

An admissible 'parameterized' policy could be to reorder as soon the stock falls below S_k to bring it to S_k.

$$\mu_k(x_k) = \begin{cases} S_k - x_k & \text{if } x_k < S_k \\ 0 & \text{otherwise} \end{cases}$$



The goal is to find a policy π that optimizes

$$J_\pi(x_0) = \mathbb{E}\left\{ R(x_N) + \sum_{k=0}^{N-1} \left(r(x_k) + c\mu_k(x_k) \right) \right\}$$

The state could be a continuous variable if we are dealing with other kinds of commodities (say petrol).

Finite Horizon MDPs

Note on Notation

Different communities (OR, Economics, CS, and EE) use different notation for describing an MDP.

You'll sometimes find s and a used for state and actions and V for value functions in other texts and papers.

Lecture Outline

Dynamic Programming

Dynamic Programming

Optimality Conditions

We can extend the ideas from deterministic settings to general finite horizon MDPs.

Proposition (Principle of Optimality)

Let $\pi^* = \{\mu_0^*, \mu_1^*, \ldots, \mu_{N-1}^*\}$ be an optimal policy. Consider the subproblem in which we are at x_i and seek the minimum cost-to-go from i to N.

$$\mathbb{E}\left\{ g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right\}$$

The truncated policy $\{\mu_i^*, \mu_{i+1}^*, \ldots, \mu_{N-1}^*\}$ is optimal for this subproblem.

The proof is straightforward. If the truncated policy was not optimal, we can swap it with the optimal one and improve π^*. But this contradicts the assumption that π^* is optimal.

Dynamic Programming

Algorithm

Theorem (DP Algorithm)

The optimal cost $J^*(x_0)$ equals $J_0(x_0)$ which solves the following system of equations

$$J_N(x_N) = g_N(x_N) \,\forall\, x_N \in S_N$$

$$J_k(x_k) = \min_{u_k \in U_k(x_k)} \mathbb{E}_{w_k} \left\{ g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k)) \right\}$$

$$\forall\, x_k \in S_k \; k = N-1, \ldots, 1, 0$$

Further, if $u_k^* = \mu_k^*(x_k)$ minimizes the RHS of the above expression then $\pi^* = \{\mu_0^*, \mu_1^*, \ldots, \mu_{N-1}^*\}$ is optimal.

According to the theorem, at a given state x_k, select an action that minimizes the one-stage cost and the cost-to-go from the next step at the new state x_{k+1}.

From the previous proposition, we know that J_{k+1} is optimal if we were to consider the problem from $k+1$ to N. A formal proof can be easily established using backward induction.

Dynamic Programming

Complexity

The DP algorithm is more efficient than a brute force computation for finite state spaces. To see why, assume that there are n states and m actions in each state at each time step.

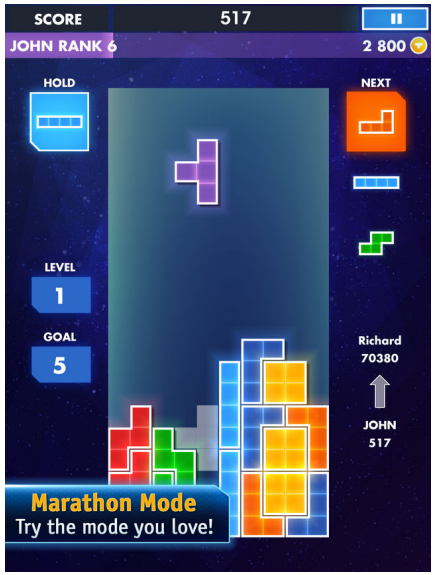
Then a total of nmN expectation calculations must be performed. Each expectation calculation roughly requires $O(n)$ calculations.

On the other hand, if we were to evaluate all policies we would require $(m^n)^N$ calculations.

Dynamic Programming

Uncontrollable State Components

In some problems, a part of the state may be unaffected by the choice of control. For example, consider the game of tetris.



The state has two components: the board configuration (which can be represented using 0s and 1s) and the shape of the falling block.

The actions include selecting the horizontal position and rotations.

However, the actions do not affect the shape of the next falling block. They only influence the board configuration.

Dynamic Programming

Uncontrollable State Components

In such cases, we can write the state as (x_k, y_k) where x_k is affected by u_k and y_k is not. Let p_i represent the pmf of y_k. In such cases, the DP algorithm can be simplified as

$$\hat{J}_k(x_k) = \sum_{i=1}^{m} p_i J_k(x_k, i)$$

$$\hat{J}_k(x_k) = \sum_{i=1}^{m} p_i \min_{u_k \in U_k(x_k)} \mathbb{E}_{w_k} \left\{ g_k(x_k, y_k, u_k, w_k) + \hat{J}_{k+1}(f_k(x_k, y_k, u_k, w_k)) | y_k = i \right\}$$

In the case of Tetris, x_k is the board configuration and y_k is the shape of the block. There is no exogenous disturbance and the action uniquely determines the new state. Hence, we can write

$$J_k(x_k) = \sum_{i=1}^{m} p_i \min_{u_k \in U_k(x_k)} \left\{ g_k(x_k, i, u_k) + J_{k+1}(f_k(x_k, i, u_k)) \right\}$$

f_k represents the new board position and g_k could be the number of rows cleared.

Dynamic Programming

Countable State Notation

When states are countable, we can simplify the notation. Suppose indices i and j represent the states. Let

$$p_{ij}(u,k) = \mathbb{P}\left[x_{k+1} = j | x_k = i, u_k = u\right]$$

How is this different from the transition probabilities of DTMCs?

The DP algorithm can be written as

$$J_k(i) = \min_{u_k \in U_k(i)} \left\{ g_k(i,u_k) + \sum_{j \in S_{k+1}} p_{ij}(u_k,k) J_{k+1}(j) \right\}$$

where $g_k(i,u_k)$ is expected cost of choosing u_k in state i. If it depends on the disturbance, one can treat it as $g_k(i,u_k,j)$ and move it inside the summation.

Your Moment of Zen